The image shows a stack of white, lined papers with three binder holes on the left side. The papers are slightly offset, creating a sense of depth. The top paper has the title 'Hand written complete C# Notes' written in a bold, black, sans-serif font. The background is a solid blue color.

Hand written complete C# Notes

Santosh Kumar Mishra

SDE @Microsoft

Introduction to .NET -Advantage of Java & .NET -

- Platform independency
- Security
- Automatic memory management

.NET FRAMEWORK -

1. NGWS (Next Generation Windows Services/System)
2. CLI (Common Language Infrastructure)
3. ISO & ECMA (European Computer Manufacture Association)
ECMA Standardization is only for computer product.
4. Open Specification - Everyone can develop.

Development of .NET Framework -

- Microsoft has started the development of .NET Framework in late 90's originally under the name NGWS.
- To develop the framework, first a set of specifications have been prepared known as CLI specification.
- CLI specifications are open specifications standardized under ISO & ECMA giving a chance for developing the framework by anyone.
- The CLI specification talks about four important things:-
 - CLS (Common Language Specification)
 - CTS (Common Type System)
 - BCL (Base Class Library)
 - VES (Virtual Execution System) or, CLR (Common Language Runtime)

CLS - (Common Language Specification) -

It is set of base rules all the .NET language has to adopt to interoperate with each other, most importantly after compilation of any .NET language program they should give the same output code i.e., CIL code as following -

2

C#.Net → CSC → CIL code

VB.Net → VBC → CIL code

J#.Net → JSC → CIL code

F#.Net → FSC → CIL Code

CompilersCTS - Common Type System):-

According to this all languages of .NET has to adopt uniform data structure ie, similar data types must be same in size under all language of .NET, so that if any two language wants to communicate with each other size problem with data ~~size~~ types will not come.

Note:-

- CLS & CTS are foundation for language interoperability of .NET languages.
- Every language of .NET is for attracting programmer like COBOL.NET for COBOL programmer.

<u>C#.NET</u>	<u>VB.NET</u>
int	integer
bool	Boolean
float	Single
↑	↑

Data types are mismatched. How interoperability work? .. Before compilation reusability is not possible. It is only possible after compilation. Because after compilation, there will be no error. Any data types that is used in any .NET language once after compilation get converted into IL types as following:-

C#.Net → Compiled → CIL Code

C#.Net	{	int	Int 32	}	VB.NET
data		float	single		data
types		bool	Boolean		types

VB.NET → Compiled → CIL Code

Integer

Single

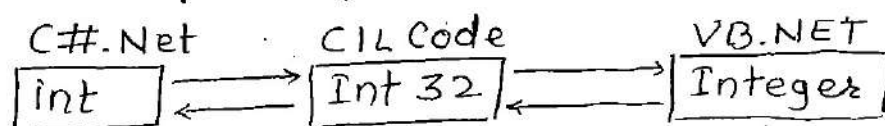
Boolean

Int 32

single

Boolean

- At most of the .NET language's are extension to some existing language like COBOL.NET extension to COBOL, VB.NET extension to VB etc... so the data types name will be different from language to language, but even if names are different, similar data types will be uniform in size.
- So in IL format all data types will be same. When we want to consume the code of a language from other .NET languages the data type of first language are first converted to IL type and then presented to the second language as its understandable data types as following :-



→ C# to VB.NET

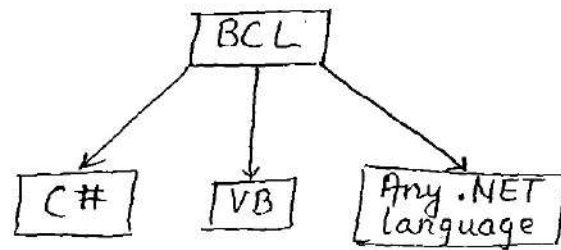
← VB.NET to C#

- Library is nothing but a set of reusable functionality.
- C/C++ library are called header files.
- Java library are called Packages.
- .NET Library are called Base Class Library

BCL - (Base Class Library) -

- A library is a set of reusable functionalities where each & every programming language has built in library like header files in C/C++, packages in Java etc.
- Same as the above, .NET languages are provided with built in libraries known as BCL. The speciality of these

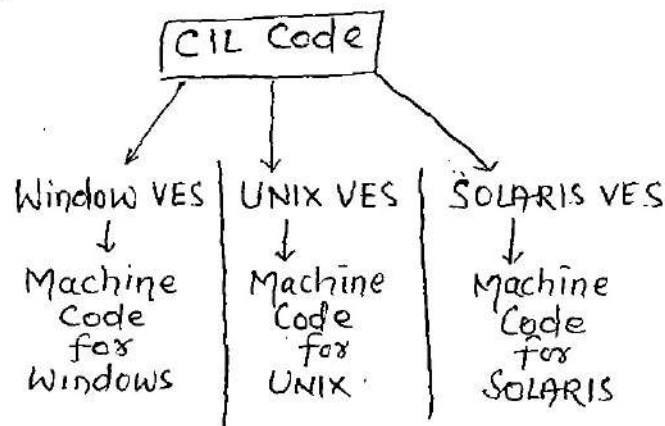
Library is that they can be consumed under any .NET language.



Note:- BCL are best example of language interoperability because those library are developed in C# language & can be consumed from any .NET language.

VES/CLR-

- All .NET languages once after compilation will generate the same type of IL code known as CIL code & this CIL is what we install on the client machines for execution.
- To run CIL code on a machine first the machine should be installed with .NET Framework software and inside of the Framework we have CLR or VES that converts CIL code into machine code according to the OS and the microprocessor



- The framework software is available separately for each OS because Framework is not platform independent.

Note:-

CLR or VES were responsible for providing portability or platform independency to .NET applications.

Adopting the above CLI specification Microsoft has implemented the .NET framework for Window OS only but not for any other OS, whereas CLI specification

are open, so third party vendors like MONO came forward⁵ & implemented framework of .NET for few other OS like Unix, Linux, Sun Solaris, Apple Mac etc.

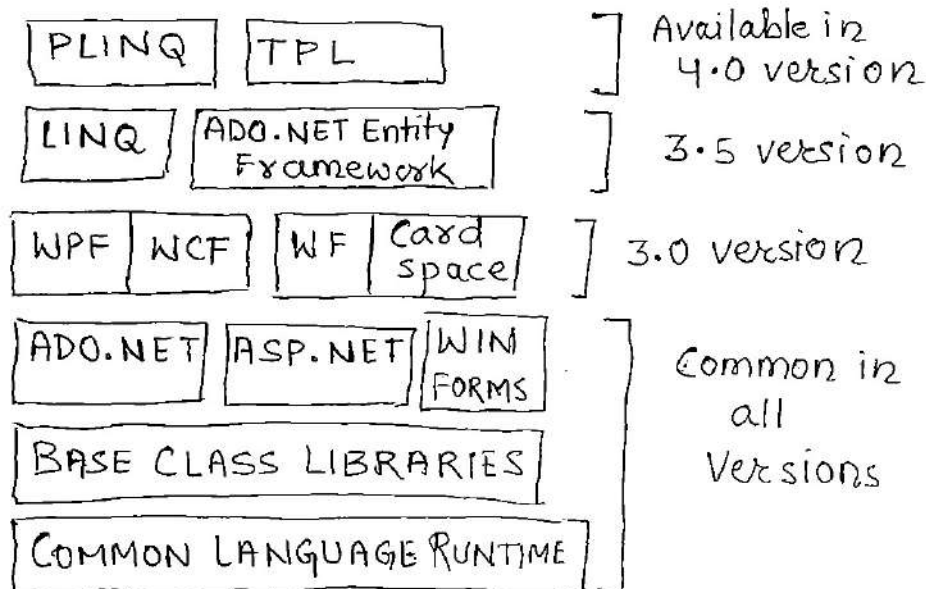
.NET FRAMEWORK VERSIONS-

2000 →	.NET FRAMEWORK	1.0 Beta
2002 →	"	1.0 RTM
2003 →	"	1.1 RTM
2005 →	"	2.0 RTM
2006 →	"	3.0 RTM
2007 →	"	3.5 RTM
2008 →	"	4.0 RTM
2012 →	"	4.5 RTM

RTM → Release to Manufacturer

The development of .NET has been started in late 90's and the first version of the framework has been launched in the year 2000 as 1.0 Beta (Trial Version) & officially it is launched into the market as 1.0 RTM in 2000.

.NET FRAMEWORK ARCHITECTURE-



TPL → Task Parallel Library

PLINQ → Parallel language Integrated Query

5
LINQ → Language Integrated Query
WPF → Windows Presentation Foundation
WCF → Windows Communication Foundation
WF/WWF → Windows Workflow Foundation
ADO → ActiveX Data Objects
ASP → Active Server Pages

CLR or VES -

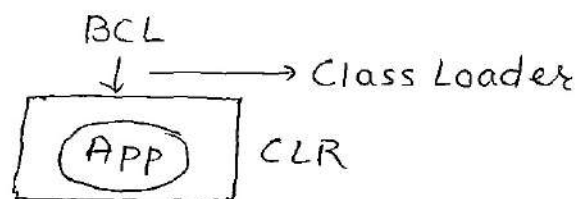
It is execution engine of .NET Framework, where all .NET applications are running under supervision of CLR & it provides various features (benefits) to applications like:-

- Security
- Platform Independency (Portability)
- Automatic Memory Management
- Runtime error handling

CLR internally contains the following things in it :-

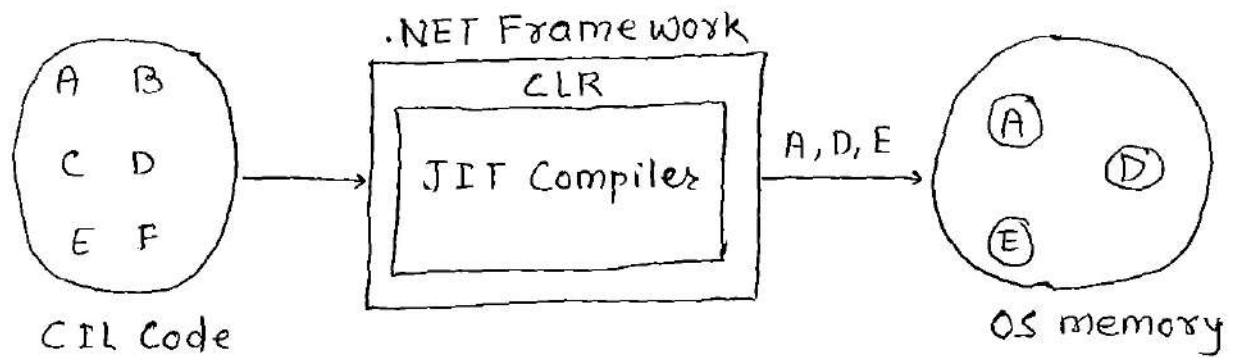
1. Security Manager
2. Class Loader
3. JIT (Just in Time) Compiler
4. Garbage Collector
5. Exception Manager

Security Manager is responsible for taking care of the security of the application ie, it will not allow application to interact directly with the OS or OS to interact with application.



Class Loader is responsible for loading required libraries that are consumed under the application from base class libraries in the runtime for the execution of application.

JIT compiler is responsible for the conversion of CIL code into machine code adopting a process known as "Conversion gradually during program execution".



- Garbage Collector is responsible for automatic memory management, where memory management is a ~~pre~~ process of allocation & deallocation of memory that is required for a program. Memory Management is of 2 types -
 1. Manual / Explicit Memory Management
 2. Automatic / Implicit Memory Management
- In the first case programmers are responsible for allocation and deallocation of memory that is required for a program's execution, whereas in the second case garbage collector will take care of allocation & deallocation process of memory.
- Garbage Collector will allocate the memory for object when & where they are required & also deallocates or re-claims the memory of those objects once they become unused under the program. Unused objects of a program are treated as garbage & deallocated immediately.

Line 1:

....

Line 10: String str = "Hello"; (Allocates Memory)

....

Line 100: WriteLine(str);

Line 101: str = null; (String marked as ^{unused} ~~unread~~)

..... (From here any time string may be deallocated)....

Line 1000:

Note:-

Garbage Collector was designed by John McCarthy around the year 1959 to overcome the problem with Manual Memory Management.

Exception Manager is responsible for taking care of the runtime errors which occurs under the program.

CONCERNS & CRITICISMS RELATED TO .NET -

- ① Managed applications which runs directly under the .NET Framework's CLR or Java's JVM consumes more system resources for execution, which when compared with unmanaged application (Machine code); However some application have proven to perform better in their managed version only when compared with their unmanaged version.
- ② Byte Code of Java & CIL code of .NET languages can be easily reverse engineered into source code when compared with Machine Code (which can't be reverse engineered), however to restrict reverse engineering of Byte code & CIL code under the current version of Java & .Net we are provided with tools known as obfuscation Tools.
- ③ Whenever garbage collector comes into picture for reclaiming the memory of unused objects, it will suspend the execution of program & resumes them after its work is completed so here there are chance of application execution being delayed (but not more than few millisec).
- ④ To run a .NET application on any machine it is must to install the .NET Framework of same version to which the application is developed on that machine. It is same in the case of Java's application also.

Source Code Converted to Machine Code

Machine Code —————→ Can't be reversed engineered.

C Sharp Programming Language —

- It is object oriented & Platform Independent programming language developed by Microsoft as part of the .NET initiative and approved as a standard by ECMA & ISO.
- Ander's Hejlsberg leads development of the language, which has procedural, object oriented syntax based on C++ and includes influences from several other programming languages most importantly Delphi & Java with a particular emphasis on simplification.

History of the language —

- During development of .NET, the class libraries were originally written in a language called Simple Managed C (SMC) & later the language had been renamed C#.
- C#'s principal designer & lead architect Ander's Hejlsberg has previously involved with the design of Visual J++, Borland Delphi, Turbo Pascal languages.
- In interviews & technical papers, he has stated that flaws in most major programming language like C++, Java, Delphi & Smalltalk drove the design of C# programming language.

Design Goals of C# :-

The ECMA Standards list these design goals of C# :-

- It is intended to be a simple, modern, general programming language (General purpose & Object Oriented).

- The language includes strong type checking, array bound checking, code portability & Automatic memory management. Programmers portability is very important in software industry, so especially for those already familiar with C & C++, C# will be the best choice.
- Support for internationalization (like language changes)
- C# is suitable for writing application to distributed, hosted & embedded system.

Features of C# 2.0 —

- Partial Classes
- Generics or parameterized types
- Static classes
- Anonymous delegates
- The accessibility of property accessors can be set independently.
- Nullable Value types
- Coalesce Operator (??) that returns the first of its operands which is not null or null, if no such operand exists.

Feature of C# 3.0 —

- Language integrated Query
- Object initializers & Collection initializers
- Anonymous types
- Implicitly-typed array & variables
- Lambda expressions
- Automatic methods
- Partial methods

Feature of C# 4.0 -

- Dynamic programming & Lookups
- Named & Optional parameters
- Covariance & Contra-variance
- Indexed Properties
- Com specific interop features

Feature of C# 5.0 -

- Support for parameterized constructor in generics
- Support for weak delegates or weak events
- Better treatment for null.
- Smart "Case" Support.
- Extension properties

Writing Programs in various programming approaches -

① Procedural Programming Approach :-

Ex - C language

Here a program is a collection of members like variables & functions and the members we define in program, if we want to be executed should be called explicitly from main function, because it is entry point of any program.

- Collection of members

void main() ← entry point

{

call the members from here for execution

}

Note -

The drawback of procedural programming language is lack of security & reusability for the code.

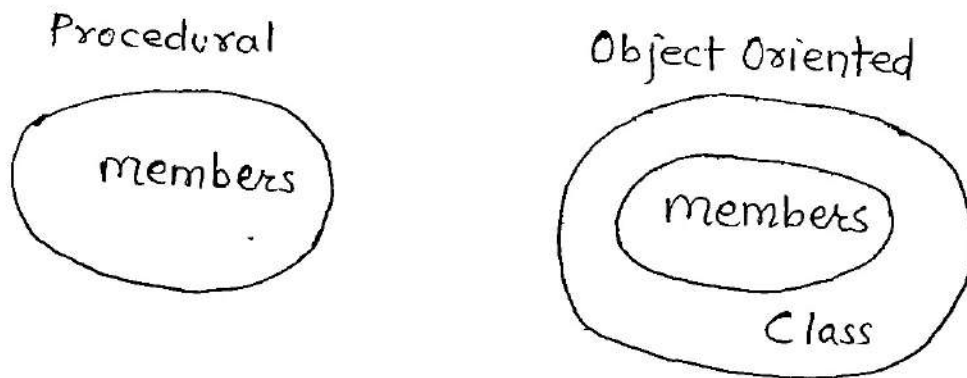
② Object Oriented Approach -

12

In 70's, to overcome the drawbacks of procedural languages, Objected Oriented Languages are introduced. Which provides security & reusability of code.

Ex- C++, Java, C# etc.

An object oriented programming is also a collection of member like variables & functions only, but in object oriented language, these members are wrapped under special container known as class, that provides security for the content of the class.



Class -

It is a user defined datatype much like structure in our procedural languages.

Note -

A structure under procedural language can contain only variables in it, whereas a class can contain variable & function also.

A copy of simple type like int, float, char etc is called as variables.

Ex- `int x = 100 ;`

A copy of complex type like class or structure is called as object.

Q. How to consume members of a class or a structure?

- A class or structure is user defined data type, so if we want to consume the members of a class or structure we need to first create object of them.

Q. What is an object & why do created?

A datatype can never be consumed directly, if at all we want to consume a datatype, we need to create a copy of that data type.

```
int = 100;    // Invalid
```

```
int x = 100;  // valid
```

In above case x is copy of type `int` for which memory is allocated for storing the value whereas it will not have any memory allocation. It only contain information like how much memory has to be allocated, when memory has to be allocated & when memory has to be allocated etc. Any datatype predefined or user defined can never be consumed directly. To consume them, it is must to create a copy of them, so if we define a structure or a class, they are considered as new data types (user defined) & to consume them we need to create a copy of them.

```
class Example
```

```
{
```

```
- Collection of members
```

```
};
```

```
void main() ← entry point
```

```
{
```

```
- Create object of class
```

```
- Using the object call members of class
```

```
}
```


19
C++ suffers from a criticism that it is not fully object-oriented. C++ is the first object oriented language to come into existence. Main function is outside the class in above example (violating the rule), because encapsulation is not followed.

Description -

Because as per the rule of object oriented programming, every member of the program should be defined inside of the class (encapsulation), whereas in C++, we never define the main() inside the class. It gets defined outside of the class because if it is defined inside of the class, it will become member of the class & if it is member of the class, it should be called by using object of the class. But the object is created inside of main() only. So until & unless object is created, main() cannot be executed & until and unless main() executes, object cannot be created.

Object Oriented Programming in Java -

When Sun Microsystems introduced Java, the designers of Java have taken this as a challenge i.e., Java language should not suffer from criticism that it is not fully object oriented. So to overcome the drawback ~~the~~ with C++ of main(), they divided the members of a class into 2 categories -

- Static Members
- Non Static members

A static member of a class does not require object of the class for initialization or ~~installation~~ execution. Whereas non static member of class requires object of the class both for initialization & execution.

- If we want the main method inside of the class, it must be in a position to execute without object of the class. So to make it executable we need to define

it as static.

- So that object of the class will not be required for starting the execution from there & will become the entry point of program.
- Now under the main() we can create object of the class & call non static members present in the class.

Class example

```
{
  collection of members (static & non static)
  static void main() ← entry point
{
  Create object of class
  Using the object call of non-static member of class
}
}
```

⇒ Programming approach in C# :-

C# language follows the guidelines of Java in writing a program, so a C# program also looks exactly same as a Java program with main() inside the class declared as static.

Note:-

In Java or C# language if a class is defined only with the method, object of that class is not required for execution of that class.

SYSTEM REQUIREMENTS -

- Windows 8 or Window 7 (Home Premium or above)
- Minimum 1GB RAM required
- SQL Server 2012 or 2008
- Visual Studio.NET 2012 or 2010
- Oracle
- MS Office

Syntax To Define A Class In C# -

```
[<modifiers>] class <Name>
```

```
{
```

```
-Members
```

```
}
```

[] → Optional

< > → Any

- Modifiers are special keywords which can be used on a class optionally like public, static, abstract, sealed etc.
- C# is a case sensitive language. So while writing the code we need to adopt the following rules :-
 - Keywords must be used in lower case
 - While consuming the libraries, we need to adopt proper case while referring to class or its members.
 - While defining a user defined class and namespaces, we can adopt any casing style as per our requirement but it will be better if we also adopt proper case.
 - Class is a keyword to tell that we are defining a class. Very much similar to struct keyword we used for defining a structure.
 - Name of class can be anything because it is user defined, but make sure that class name always starts with an alphabet.

Note :-

The file in which we write C# program should be saved with .cs extension & it will be better if the filename is same as your class name.

SYNTAX TO DEFINE MAIN METHOD -

```
Static void / int Main ([string[] args])
{
    - Statements
}
```

- Main method must be declared as a static using the static modifier if at all we want to make it as entry point.
- Main method can be either non value returning or can return a value also but of type int only.
- Main is the name of the method & must be in proper case only. Main ✓
- If required Main method can be passed with the parameters but of type string array only.

GetType is a predefined method which written the type of a given variable or objects.

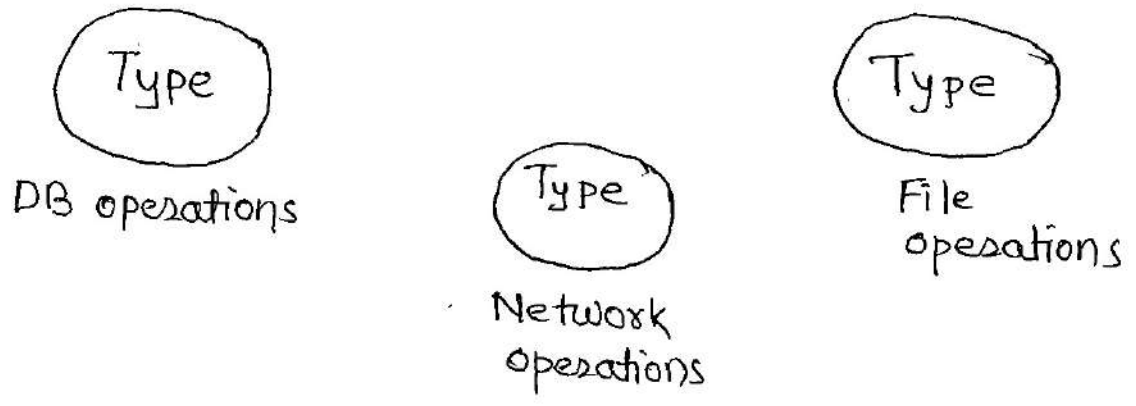
System. Console. WriteLine :-

- Console is a predefined class under the libraries which provide a set of member using which we can perform I/O operation on the standard I/O devices.
- The class contain a set of method like write and WriteLine to perform output operation and readline to perform input operation.

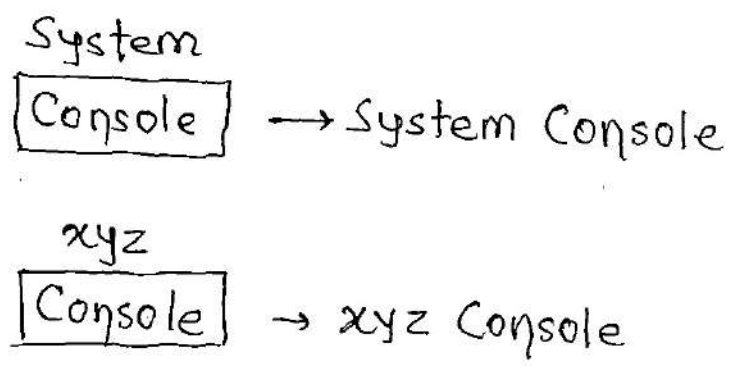
System :-

It is a namespace where namespace is a logical container of types (Class, Structure, interface, Enumeration).

- Namespaces are used for grouping of related types as well as to overcome the problem with naming collision i.e, types which defined to perform similar operation or boot under one namespace for better organising.



- If at all we have multiple type with the same name → they can be differentiated by putting under separate namespace then refer to them with the help of namespace name.



Importing A Namespace —

- If at all a type is defined under a namespace whenever we want to consume the type we need to refer to it with its namespace prefix but referring to the type with the namespace always

will be increasing volume of code, to overcome this problem we have an option of importing a namespace.

ie, on the top of the class we specify the namespace name with the help of using statement. all the types of that namespace can be directly consumed in that program without a namespace prefix again.

Syntax:-

Using <namespace>

- We can import any number of namespace on the top of the program.

Note:-

Namespace name type name and member of type name will always be in proper case under the libraries.

Program Example -

```

    Using System ;    // Importing a namespace
    Class import Demo
    {
    Static void Main()
    {
    Console.WriteLine ("importing a namespace");
    }
    }
    
```

Program:-

Adding two numbers through input from user.

```

    Using System;
    Class Addnums
    {
    
```



```

{
    Static void Main()
    {
        int x, y, z;
        Console.Write ("Enter value for x : ");
        String s1 = Console.ReadLine();
        x = int.parse(s1);
        Console.Write ("Enter value for y : ");
        String s2 = Console.ReadLine();
        y = int.parse(s2);
        z = x + y;
        Console.WriteLine ("Sum of {0} & {1} is {2} ", x, y, z);
    }
}

```

output -

```

Enter the value for x : 100
Enter the value for y : 200
Sum of x & y : 300

```

Readline() method of Console class is used for taking the input from the users in run time, When we use Readline() statement the cursor wait at command prompt for the user enter the value & once the user enters the value & clicks on the enter key, Readline() statement read the value and brings it into the program as string.

Because return type of the method Readline is string.

- Whatever type of value we give as an input to the program will come into the program as string only and if we want to use that value in its appropriate type format, we need to connect the string value by calling Parse() method.

Ex- String s1 = "100";
 int i = int.parse(s1);
 String s2 = "3.14";
 float f = float.parse(s2);

Parse() :-

Parse() method converts a string into a type in which the method is called.

Note -

We cannot use this method to perform an invalid type conversion i.e., ~~the~~ the value being converted should be compatible for being converted.

Ex- String S = "10 A B 01";
 int i = int.parse(s);
 we can use this method like this
 int i = int.Parse(Console.ReadLine());

Implicitly Typed Variables -

It is a new feature that has been added in C# 3.0 which allows you to declare a variable using the var keyword where the type of variable is identified depending on value assigned to it.

Ex- `Var i = 100; // int type`
 `Var s = "false"; // String type`
 `var d = 5.415; // double type`

Limitation While using Implicitly typed Variable—

- We can declare them ^{under} only a block of code but cannot declare on the top of the class.
- Variable declared by `var` keyword must be initialized the time of statement.

ex- `Var x; // Invalid`

Writing a program in C# :-

We can write C# programs either under Visual Studio IDE (Integrated Development Environment) or any text editor like a Notepad also.

Writing a program using Notepad :-

Open Notepad & Write the following code in it :-

```

Class Example
{
    static void Main()
    {
        System.Console.WriteLine("My first C# Program");
    }
}

```

Save the program as `Example.cs` in desired Location.

ex- `C:\Csharp7`

Compiling the program -

We need to compile the above program by using C# compiler manually from Visual Studio Command prompt.

To open Visual Studio command prompt.

Start Menu → All programs → Microsoft Visual Studio → Visual Studio Tools → Developer command prompt or Visual Studio command prompt.

Click on it to open the Visual Studio Command Prompt & By default it will be pointing to the location where Visual software is installed on our machine, Now change to our folder where we save the program.

C:\Csharp7

We compile the C# program using C# compiler as following :-

CSC < File Name >

C:\Csharp7 > CSC Example.cs

Once the program is compiled successfully, it generates an output file Example.exe or else display the error list over there.

Executing the program -

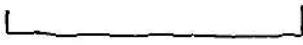
We can execute a program directly from the command prompt of the compilation by running .exe as following :-

C:\Csharp7 > Example

Data types -

• Integer Types -

<u>C# types</u>	<u>CIL types</u>	<u>Capacity/Size</u>
byte	System.Byte	0 - 255
short	System.Int16	-32768 - 32767
int	System.Int32	$-2^{31} - 2^{31} - 1$
long	System.Int64	$-2^{63} - 2^{63} - 1$
Sbyte	System.SByte	-128 - 127
ushort	System.UInt16	0 - 65535
uint	System.UInt32	0 - $2^{32} - 1$
ulong	System.UInt64	0 - $2^{64} - 1$


 CIL types

Sbyte → Signed byte

uint → Unsigned byte

• Decimal Types -

Float	System.Single	4 bytes
double	System.Double	8 bytes
decimal	System.Decimal	16 bytes

• Boolean Types -

bool	System.Boolean	True or false
------	----------------	---------------

• Character Types -

Char	System.Char	2 bytes
String	System.String	

• Base Type -

object	System.Object	
--------	---------------	--

The size of the Char type has been increased to 2 bytes for providing the support for Unicode characters ie, Language other than english.

Note -

English Language characters have a numeric representation known as ASCII. In the same way numeric representation for other language characters is known as Unicode, where ASCII consume 1 byte & Unicode consumes 2 byte memory location.

Ex -

Char Ch = 'A'

↳ ASCII → 1 byte

Char Ch = 'अ'

↳ Unicode → 2 byte

String is variable length type where the size of string depends upon the value we assign to it.

Object type is the parent of all the other datatype which is capable of storing any type of value in it & moreover it is also a variable length type.

Categories of datatypes -

Data types are divided into 2 types -

1. Value Types
2. Reference Types

1. Value Types -

Value Type stores the data on stack which is a place where data stores in fixed length. Types which are fixed in length like int, float, char etc. get stored

Declaring Variables —

[<modifiers>] [Const] [readonly] <type> <var> [=value] [...]

int x;

int y = 100;

String s1, s2 = "Hello", s3;

public bool flag = true;

Const float Pi = 3.14f;

double d = 12.34;

readonly decimal d = 567.8975m;

Default scope for members of a class in C# is private, when variables are declared on the top of a class & under any static block, initializing those variables is only optional, if not initialized all numeric variables will be initialized with 0, boolean variable with false, String & Object variables with NULL.

Numeric variables → int, double

Note:-

If variable are declared under any non static block like a function or method initializing those variable is mandatory at the time of declaration.

Every decimal value is by default treated as double, so to consider it as a float, we need to suffix the value with character f and with character m to treat it as a decimal.

```
Class varDemo
{
    Static void Main()
    {
        int x=100;
        System.Console.WriteLine(x);
        System.Console.WriteLine(x.GetType());
        float f=3.14f;
        System.Console.WriteLine(f);
        System.Console.WriteLine(f.GetType());
        decimal de=123.456m;
        System.Console.WriteLine(de);
        System.Console.WriteLine(de.GetType());
    }
}
```

Output-

```
100
System.Int32
3.14
System.Single
123.456
System.Decimal
```

GetType() is a predefined method which returns the type of given variable and object.

Nullable Value Types —

Before C# 2.0, we cannot store null under a value type if at all a value is unknown for a value type variable but it is possible to store null value in a reference type variable. String s = null; // valid int x = null; // Invalid

To overcome the above problem in C# 2.0, we are provided with the nullable value type additionally to the existing value type (Non-nullable) which gives the flexibility of storing null value under a value type.

```
int? x = null; // Invalid
double? d = null; // Valid
double d = null; // Invalid
```

To specify the value type is nullable, we need to specify
suffix the type name with the question mark (?)

Boxing and Unboxing —

If at all a value type is assigned to your reference type variable and stored on the managed ~~heap~~ heap, we call it as boxing.

```
int x = 100;
object obj = x; // Boxing
```

If a value type which is converted into reference type is converted back to value type, we call it as unboxing, but ^{while} performing unboxing explicit conversion is required.

Value type → Reference type → Value type

```
int y = Convert.ToInt32(obj); // Unboxing
```

Note —

- A direct reference type can never be converted to value type.

Operators in C# -

30

It is a special symbol which perform some action when used between two operands.

Arithmetic

+, -, *, /, %

Assignment

=, +=, -=, *=, /=, % =

Comparison

==, !=, <, <=, >, >=, is, as, like

Concatenation

+

Increment & Decrement

++, --

Logical

&&, ||, ^

Conditional Statements :-

A block of code that is executed basing on a condition is known as a conditional statements.

They are divided into 2 category -

- 1> Conditional branching
- 2> Conditional Looping

1> Conditional Branching -

These statements allow you to branch your code depending on whether certain condition were met or not. C# has 2 constructs for branching the code, the if ~~statements~~ ~~to~~ statements allows you to test whether a specific

condition is met & the switch case statement allows you to compare an expression with a number of different values.

Syntax -

```

    if (<condition>)
        <stmts>;
    else if (<condition>)
        <stmts>;
    - - - - -
    else
        <stmts>;

```

Switch Case -

```

Switch (<expression>)
{
    Case <value>;
        <stmts>;
        break;

```

- <multiple Case blocks> -

```

    default:
        <stmts>;
        break;

```

```

}

```

Note -

In traditional C language using a break statement after a case block is only optional whereas in C# language it is mandatory which should be used after even after default also.

Ex-

Using System ;

Class Switch Demo

{

Static void Main()

{

Console. Write ("Enter a student no. (1-3) :");

int Sno = int.parse (Console. Readline ());

Switch (Sno)

{

Case 1 :

Console. Writeline ("student 1");

break ;

Case 2 :

Console. Write line ("student 2");

break ;

Case 3 :

Console. Writeline ("student 3");

break ;

default :

Console. Writeline ("Student does not Exist ");

break

}

}

}

Conditional Loops—

C# provides four different loops that allows you to execute a block of code repeatedly until a certain condition is met, those are :-

- For Loop
- While Loop
- do. while Loop
- For each Loop

Every loop requires 3 things in common —

- 1> Initialization which sets the starting point of the loop.
- 2> Condition which decides the ending point of the loop.
- 3> Iteration which takes you to the next level of the cycle, either in a forward & backward direction.

```
for (initializer ; Condition ; iteration)
    <stmts>;
```

```
for (int x=1; x<=100; x++)
    Console.WriteLine(x);
```

```
While (Condition)
```

```
    <stmts>;
```

```
    int x=1;
```

```
    While (x<=100)
```

```
    {
```

```
        Console.WriteLine(x);
```

```
        x++;
```

```
    }
```

```

do
{
<stmts>
} while (condition);

int x=1;
do
{
Control.WriteLine(x);
x++;
} while (x <= 100);

```

For loop and while loop have the minimum number of execution as 0, because in both these two cases first condition is verified and if at all the condition is satisfied only execution starts, whereas in case of a do while loop minimum number of execution will be 1, because hereafter completion of one execution then only it checks with the condition to proceed for the next executes.

For each Loop —

This is specially designed for accessing the values from an array & a collection [Stack, Queue, Linked List]

```

for each (type var in coll | array)
{
<stmts>;
}

```

Jump Statement:-

It's a statement which transfers the execution from one line to the other. C# has a number of statements that allow you to jump to another line in a program.

Those are:-

- 1> goto
- 2> Break
- 3> Continue
- 4> Return

goto -

It allows you to jump directly to another specified line in a program indicated by a label which is an identifier followed by a colon.

```
goto xxx:
```

```
Console.WriteLine("Hello"); // This is unreachable
```

```
xxx:
```

```
Console.WriteLine("Goto Called");
```

break -

It is used to exit from a case in a switch statement and also used to exit from any conditional loop statements which will switch control to the statement immediately after end of the loop.

```
for (int i = 1; i <= 100; i++)
```

```
{
```

```
    Console.WriteLine(i);
```

```
    if (i == 50)
```

```
        break;
```

```
}
```

Continue -

This statement is mostly under a for loop and whenever it executes the control directly jumps to the iteration part of the loop without executing next line of code.

```
for (int i=0 ; i<=100 ; i++)  
{  
    if ( i==7 || i==77 )  
        Continue ;  
    Console.WriteLine(i);  
}  
Console.WriteLine("End of the loop");
```

Return -

This jump statement is used for jumping out of a function and method which is in execution and while jumping out it is capable of carrying a value out of the function or method from where it is jumping.

```
using System ;  
class Tables  
{  
    static void Main()  
    {  
        Console.Write("Enter an integer value ");  
        int x = Int.Parse ( Console.ReadLine () );  
        if ( x==0 )  
            return ; // jumps to end of the method.
```

```
for (int i=0; i<=10; i++)
```

37

```
Console.WriteLine("{0} * {1} = {2}", x, i, x * i);
```

```
} // End of method
```

```
}
```

Arrays -

It is a set of similar type values that are stored in a sequential order. C# supports 3 different types of arrays:-

1> One-Dimensional

2> Two-Dimensional

3> Jagged Array

- In the first case, data is arranged in the form of a row whereas the second & third cases, data is arranged in the form of rows & columns.

- We access the values of an array using index positions where the array index starts at 0, that means first item of an array will be stored at 0th position and the position of last item of an array will be total no. of item - 1.

- In C#, arrays can be declared as fixed length & dynamic. Fixed length array can store a predefined no. of times while size of dynamic arrays increases as we add new items to the array.

1-D Array -

```
<type> [ ] <name> = new <type> [size];
```

```
int [ ] arr = new int [5];
```

```
int [ ] arr;
```

```
arr = new int [5];
```



```
int [] arr = { <list of values > } ;
```

Note:-

In Java and .NET languages an array gets initialized either with new keyword or ~~assignment~~ ^{argument} of values.

Ex-

```
Using System ;
```

```
Class SD Array
```

```
{
```

```
Static void Main()
```

```
{
```

```
int [] arr = new int [6] ;
```

```
int a = 0 ;
```

```
// Accessing array values using for loop
```

```
for (int i = 0 ; i < 6 ; i++)
```

```
Console.WriteLine (arr[i] + " ");
```

```
Console.WriteLine ();
```

```
// Assigning values to an array using for loop
```

```
for (int i = 0 ; i < 6 ; i++)
```

```
{
```

```
    a++ = 10 ;
```

```
    arr[i] = a ;
```

```
}
```

```
// Accessing array values using foreach loop
```

```
foreach (int i in arr)
```

```
    Console.WriteLine (i + " ");
```

```
}
```


```
}
```

For each loop -

This is specially designed for accessing the values from an array or a collection. When we use a for each loop for accessing the values of an array or collection, we only require to handover the array & collection to the loop, which does not require any initialization, condition or iteration. The loop starts its execution by providing access to each and every value in the array & collection starting from the first upto the last value in a sequential order.

Difference between For Loop & Foreach loop to access array values -

- ① In the case of for loop, the loop variable refers to the index of an array whereas in case of a foreach loop, the loop variable refers to the values of the array.
- ② In the case of for loop, whatever types of values the array ~~contains~~ contains loop variable will be int only because it is referring to int x . In case of a foreach loop the datatype of the loop variable will be same as the type of values inside the array.


 foreach (int i in iarr)
 foreach (String s in sarr)
 foreach (float f in farr)

- ③ For loop can be used both for accessing and assigning values to an array whereas foreach loop can be used only for accessing.

Array Class -

40

It is a predefined class under σ base class libraries defined inside system namespace, provides a set of members using which we can perform manipulation on an array.

Array :

- Sort (<array>)
- Reverse (<array>)
- Copy (src, dest, n)
- GetLength (int)
- Length // Property

Ex-

Using Systems;

Class SDArray2

{

Static void Main()

{

int [] arr = { 17, 93, 4, 59, 27, 64, 36, 96, 3, 1, 9 }

for (int i = 0; i < arr.Length; i++)

Console.WriteLine (arr[i] + " ");

Console.WriteLine ();

→ Array.Sort (arr);

foreach (int i in arr)

Console.Write (arr[i] + " ");

Console.WriteLine ();

→ Array.Reverse (arr);

foreach (int i in arr)

Console.Write (arr[i] + " ");

Console.WriteLine ();

int [] arr = new int [10];

```

- Array.Copy(arr, brr, 5);
  for (int i in arr)
    Console.WriteLine(arr[i] + " ");
  Console.WriteLine();
}
}

```

Implicitly Typed Arrays -

Just like we can declare variable by using Var keyword, we can also declare array by using the Var keyword.

```
var arr = new[] { 10, 20, 30, 40, 50 };
```

Two dimensional Arrays -

```
<type> [,] <name> = new <type> [rows, cols];
```

```
int [,] arr = new int [3, 4];
or,
```

```
int [,] arr;
```

```
arr = new int [2, 3];
or,
```

```
int [,] arr = { list of values };
```

Ex -

```
Using System;
```

```
Class TD Array
```

```
{
```

```
Static void Main()
```

```
{
```

```
int [,] arr = new int [4, 5];
```

int a = 0;

42

// Printing values by using for each loop

foreach (int i in arr)

Console.Write(i + " ");

Console.WriteLine("\n");

// Assigning values by using nested for loop

for (int i = 0; i < arr.GetLength(0); i++)

{

for (int j = 0; j < arr.GetLength(1); j++)

{

a++ = 5;

arr[i, j] = a;

}

}

// Printing values by using nested for loop

for (int i = 0; i < arr.GetLength(0); i++)

{

for (int j = 0; j < arr.GetLength(1); j++)

Console.Write(arr[i, j] + " ");

Console.WriteLine();

}

}

Assigning Values to 2-D Array at the time of declaration -

int [,] arr = {

{ 11, 12, 13, 14 }

{ 21, 22, 23, 24 }

{ 31, 32, 33, 34 }

};

Implicit typed 2-D Array -

43

```
var arr = {  
    new[] { 11, 12, 13, 14 }  
    new[] { 21, 22, 23, 24 }  
    new[] { 31, 32, 33, 34 }  
};
```

Tagged Arrays -

These are also 2-D array which will store the data in the form of rows & columns. But in 2-D array, all the rows will have equal number of columns whereas in a Tagged array the column size differs from row to row.

It is also called as array of arrays, because here each row is a single dimensional array where multiple single dimensional arrays with different sizes are combined together to form a new array.

	0	1	2	3	4	5	6	7	
0	1	2	3	4	5				→ arr[0]
1	1	2	3	4	5	6			→ arr[1]
2	1	2	3	4	5	6	7	8	→ arr[2]
3	1	2	3	4					→ arr[3]

Syntax for Tagged Array -

<type>[][] <name> = new <type> [rows][];

int [][] arr = new int [3][];

or,

int [][] arr = { List of values };

44

To declare a jagged array in the initial declaration, we can only specify the number of rows we want in array.

Ex-

```
int [][] arr = new int [4][ ];
```

After specifying the number of rows now pointing to each row we need to specify the number of columns to that row.

```
int [][] arr = new int [4][ ];
```

```
arr[0] = new int [5];
```

```
arr[1] = new int [6];
```

```
arr[2] = new int [7];
```

```
arr[3] = new int [8];
```

In 2-D,

```
[0,0]
```

In Jagged,

```
[0][0]
```

Example-

```
using System;
```

```
class Jagged Demo
```

```
{
```

```
Static void Main()
```

```
{
```

```
int [][] arr = new int [4][ ];
```

```
arr[0] = new int [5];
```

```
arr[1] = new int [6];
```

```
arr[2] = new int [7];
```

```
arr[3] = new int [8];
```

```
// Printing values by using a nested for loop
```

```
for (int i=0 ; i<arr.GetLength(0) ; i++)
```

```

    {
        for (int j=0 ; j<arr[i].Length ; j++)
            Console.WriteLine(arr[i][j] + " ");
    }
}

```

// Assigning values by using nested for loop

```

for (int i=0 ; i<arr.GetLength(0) ; i++)
{
    for (int j=0 ; j<arr[i].Length ; j++)
        arr[i][j] = j++ ;
}

```

// printing values by using a for each loop in for loop

```

Console.WriteLine();
for (int i=0 ; i<arr.GetLength(0) ; i++)
{
    foreach (int x in arr[i])
        Console.WriteLine(x + " ");
}
}

```

Assigning jagged array at the time of declaration -

```

int[][] arr = { new int[4] { 11, 12, 13, 14 },
                new int[5] { 21, 22, 23, 24, 25 },
                new int[3] { 31, 32, 33 }
            };

```

Implicitly typed jagged arrays -

46

```
var arr = {  
    new[] { 11, 12, 13, 14 },  
    new[] { 21, 22, 23, 24, 25 },  
    new[] { 31, 32, 33 }  
};
```

Command Line Parameters -

This is an approach used in Console applications for supplying input to a program from the command prompt while executing the program.

```
using System;  
class Params  
{  
    static void Main (string args[] args )  
    {  
        foreach (string str in args )  
            Console.WriteLine (str);  
    }  
}
```

After compiling the program & executing, we can supply input values to the program from cmd as following:-

C:\CSharp\params 10 Hello 3.14 true A

10
Hello
3.14
true
A

Working With Visual Studio. NET —

It is an IDE used for developing .NET applications with any .NET language like C#, VB etc., as well as we can develop any kind of application like Console, Windows, web etc.

Versions of Visual Studio. NET —

VS	(Framework 1.0)
VS 2003	(Framework 1.1)
VS 2005	(Framework 2.0)
VS 2008	(Framework 3.5)
VS 2010	(Framework 4.0)
VS 2012	(Framework 4.5)

- To open VS, go to Start Menu → program → MS Visual Studio & Click on it to open.
- ☐ - Application developed under VS are known as projects, where each project is collection of items. To create a project either click on "New Project" option or go to file menu and Select New → Project, which opens the "New Project" window.
- Under New project Window, we need to specify following details :—
 1. In the LHS, Choose the language in which we want to develop the application.
Ex - Visual C#
 2. In the middle, Choose the type of application we want to develop by selecting a project template.
Ex - Console Application.

3. In the bottom, specify a name to the project.

Ex- First Project

4. Below project name, specify the location where to save the project.

Ex- C:\Csharp7

- Click on "OK" button, which creates the project with a default class program under the file Program.cs
- When classes are defined in VS by defined those classes will be defined under a namespace, whose name is same as project name ie, First project in our case, from now each & every class of the project comes within the same namespace only.

Note :-

As discussed earlier a namespace is logical container of types.

- Now under main method of the program class, write the following code:-

```
Console.WriteLine("My First Project");  
Console.ReadLine();
```

- To run the class, either press "F5" or "Ctrl + F5" or start debugging button on the top of the studio which will save, compile & executes the program.

Adding new items to the project :-

- Under VS we find a window in the RHS, known as Solution Explorer used for organising the complete

application, which allows us to view, add & delete items under the projects. 49

Note:-

If solution explorer is not available in RHS, go to View Menu & Select Solution Explorer.

- To add a new class under project, open Solution Explorer right click on project, Select Add and Choose New item, which opens "Add new item" window in that select "Class" template; Specify a name to it in the bottom and click on add button, which adds the class under project.

Ex - Class 1.cs

Note:-

The new class added, also comes under the same namespace i.e., First Project.

- Now under the new class write the following code:-

```
Static void Main()  
{  
    Console.WriteLine("Second Class");  
    Console.ReadLine();  
}
```

- To run the above class open solution explorer, right click on the project, select properties, which opens project property windows, under it we find an option "Startup Object", which list all the classes of the project that contains valid main method in them, choose your class & run.

Object Oriented Programming -

50

- Encapsulation (Hiding the data) → Security
 - Abstraction (Hiding of complexity)
 - Inheritance (Reusability)
 - Polymorphism
- A language is called object oriented, if it satisfies all the above four principle features.
Behaving in different ways depending on input (like exam result) is polymorphism.
- It is an approach used in application development which comes into existence in 70's to overcome the problems in procedural languages like security & reusability.

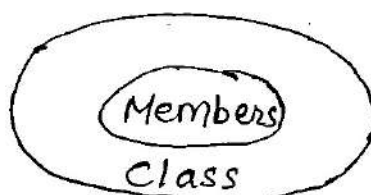
Encapsulation -

This is all about hiding of data of a program by enclosing it or wrapping it under wrapper or container i.e., class, which provides basic security for the members in the program.

Procedural



Object Oriented

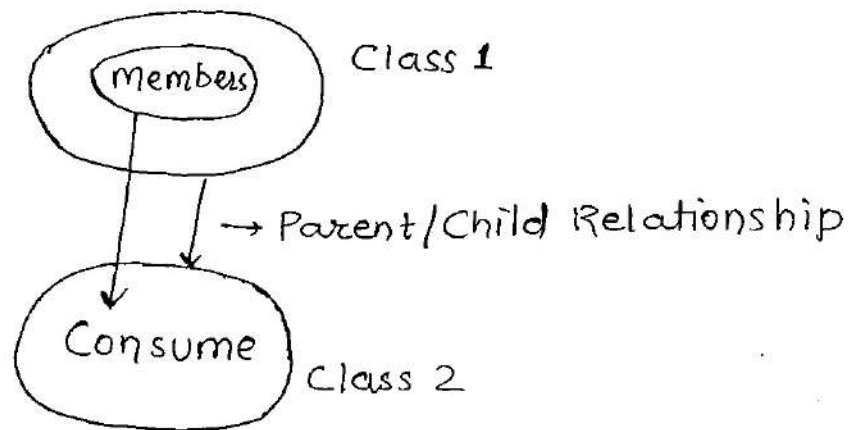


Abstraction -

This is all about hiding of complexity behind the programs and exposing with the set of interface to consume the functionalities of the program.

Inheritance -

It is a concept of acquiring the parents property by the children for reusing them, same as this, the members of a class in a object oriented programming can be consumed in a new class by establishing parent/Child relationship between the class.



Polymorphism -

It is an approach of behaving in the different ways according to input recieved. Whenever the input changes, the behaviour or output also changes accordingly.

Subprograms -

It is a named block of code which can be called with its name for executing of the code defined in it. These subprogram or refer with different name in different languages in approaches like functions & C/C++ languages, methods under Java/.NET Language, whereas store procedure under database.

Method :-

A method is an action that has to be performed.

```

[<modifier>] void / type <Name> ([Parameter definition])
{
    -stmts
}
    
```

Modifiers-

Modifiers are ~~key~~ special keyword that can be used optionally on a method like public, static, virtual, override, abstract, sealed etc.

Void-

It is a non value returning.

Type-

It's perform an action & give us result that action.

Note-

A non value returning method just perform an action, but after performing the action will not give or bring the result of the action into the program.

Ex- WriteLine Method

When perform action of the display output of the monitor, whereas a value returning method perform an action & also brings result of the action back into the program.

Ex- ReadLine Method

Which performs an action of reading the input at the command prompt & brings back value into the program.

Name-

It refers to the name of the method using which we call the method, which is user defined.

Parameter-

It is used to specify or define any parameters to the method whereas passing parameter is only optional & if we want to pass any parameters to a method we can pass them as following -

[ref/out] <type> <var> [=value] [...]

* Parameter makes an action more dynamic.

Note-

The return type & parameter types of a method need not only a predefined type like int, float, char, string, object etc. It can be also be a user defined type also like a user defined class or a user defined structure.

Defining Methods-

If we want to define any methods in an object oriented programming language as per the rule of encapsulation, they must be defined inside of the class only. The method we define under a class if we want to be executed must be called explicitly for execution.

Calling a method for execution-

A method can be defined either as static or non-static only. If the method is static we can directly call method using "Class Name" For example: Console.WriteLine, Console.ReadLine etc. Whereas if a method is defined as non static, it is must & should we need to call the method using object of the class because after creating the method only the memory required for execution get allocated.

Creating of the object of a class-

<class> <obj> = new <class> [<List of values>]

program p = new program(); // P is object

or

program p; // P is variable

p = new program(); // P is a object

A variable of a class will not have any memory allocation where memory is allocated only to object of the class & object of a class can be created only with the case of new keyword.

Where to we create the object of a class —

54

The object of a class we created either within the same class or under another class also. If we want to create the object within the same class it can be created under any static block of the class.

Note —

Generally objects are created under main method because it is also static block & moreover it is the entry point of our program.

Open Visual Studio → Go to File Menu → Select New Project
→ Choose language as C# → Project as Console application
→ Name the project as OOPs project & Under the default class program, write following code :-

Class Program

{

// Method without any input & return value

public void test1() // Static method

{

int m = 5;

for (int i = 1; i <= 10; i++)

Console.WriteLine("{0} * {1} = {2}", m, i, m * i);

}

// Method with input & without any return value

public void Test2(int m, int n) // Dynamic method

{

for (int i = 1; i <= n; i++)

Console.WriteLine("{0} * {1} = {2}", m, i, m * i);

}

// Method with return value & without any input

```
Public string Test 3()
{
    String str = "Hello World";
    str = str.ToUpper();
    return str;
}
```

// Method with input & return value also

```
Public string Test 4(String str) // Dynamic Method
{
    String = str.ToUpper();
    return str;
}

Static void Main (String [] args)
{
    Program p = new program()
    p.Test 1();
    p.Test 2 (6, 12);
    String s1 = p.Test 3();
    Console.WriteLine (s1);
    String s2 = p.Test 4 ("Hello, how are you");
    Console.WriteLine (s2);
    Console.ReadLine ();
}
}
```

How to consume class from another class —

We can consume a class from any other class and call its members if required, which can be done in 2 different ways —

- ① By creating the object of the class, we want to consume under a new class we can call members of that class.
- ② By using inheritance also, we can consume the member of the class from another class.

Consuming the members by creating an object —

To test this process, add a new class in the project naming it as `testprogram.cs`, write the following code:-

```

Class Test Program
{
    Static void Main()
    {
        Program p = new Program();
        p.test1();
        p.test2(7, 6);
        Console.WriteLine(p.Test3());
        Console.WriteLine(p.Test4("Test Program"));
        Console.ReadLine();
    }
}

```


Parameters -

A parameter of a method is defined for making the methods or actions more dynamic. Parameters of a method are of two types:-

- ① Input Parameters
- ② Output Parameters

Input Parameters are used for sending a value into the method for execution, Whereas output parameters are used for sending the result out of the method after execution.

Input → Method → Output

By default every parameter of a method is input parameters & if we want to define a parameter as output, we need to prefix the parameter either with a `ref` / `out` keywords.

Ex-

```
public void Test (int x, ref int y)
```

or,

```
public void Test (int x, out int y)
```

Note -

Return types of a method can also send results of a method after execution, but the main difference between return type and output parameter is : A return type can bring only a single result after execution of method, Whereas output parameter are capable of bringing multiple results after execution of the method.

Passing Default Values to Parameters -

While defining parameter to a method it is possible to provide default values for those parameters. If default values are given to a parameter while defining the method, while calling the method is only optional to supply

value to that parameter. If that parameter is not supplied⁵⁸ with a value it automatically its default value for execution.

Note-

This feature was added in C# 4.0. If we want to supply default values to parameter those parameter must be in the end of parameters list.

Add a Class Param.cs

```
Class Params  
{
```

```
// Method with default values to the parameters (4,0)
```

```
Public void AddNums(int x, int y=50, int z=25)
```

```
{
```

```
Console.WriteLine(x+y+z);
```

```
}
```

```
// Methods with both input & output parameter
```

```
Public void main(int a, int b, ref int c, ref int d)
```

```
{
```

```
c = a + b;
```

```
d = a * b;
```

```
}
```

```
Static void Main()
```

```
{
```

```
Params p = new Params();
```

```
// Calling method with default values to parameters
```

```
p.AddNums(100);
```

```
p.AddNums(100, 100);
```

```
p.AddNums(100, z: 100);
```

```
p.AddNums(100, 100, 100);
```

// Calling methods with output parameters

```

int x=0, y=0;
P.Math1(100, 50, ref x, ref y);
Console.WriteLine(x + " " + y);

int m, n;
P.Math = (50, 25, out m, out n);
Console.WriteLine(m + " " + n);
Console.ReadLine();
}
}

```

Execution of methods with output parameters —

The execution of a method with input/output parameters will be starting as following :-

```

void math1( int a , int b , ref int c , ref int d )
{
    c = a + b;
    d = a * b;
}

```

Main method:
 int x=0, int y=0;
 P.Math1(100, 50, ref x, ref y)

- In the above case, the parameters a & b of the method were input parameters in the sense they will store the values that are sent for the method to execute whereas the parameters c and d are input parameters in the sense they will store address of variables that are sent to method to execute because they are pointer variables (implicit).
- When the method body is executed the results that are assigned to parameter c & d are internally redirected to the variables x, y to which c, d are pointing.

66

3

5

22

 $\{$

3

3

2

3

Ex -

↓

Calling constructor

Types of Constructor -

Constructor are of 2 types :-

- ① Parameterless Constructor
- ② Parameterized Constructor

- A constructor without any parameter is a parameterless constructor, where if it is defined with parameter, it is a parameterized constructor.

Note -

- Parameterless constructor can be defined explicitly by the programmer or will be defined implicitly provided there is no constructor in the class defined explicitly. Whereas, parameterized constructor can only be defined explicitly, but can never be defined implicitly.
- If a constructor is parameterized, values to the parameter has to be sent while creating the object of the class because we call the constructor at that time only.

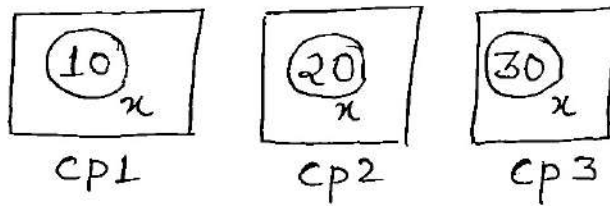
Add a class Conparam.cs & write the following code:-

```

class Conparam
{
    int x;
    public Conparam(int a)
    {
        x = a;
    }
    public void display()
    {
        Console.WriteLine("Value of x is : {0}", x);
    }
    static void main()
    {
        Conparam cp1 = new Conparam(10);
        Conparam cp2 = new Conparam(20);
        cp1.display(); cp2.display();
        Console.ReadLine();
    }
}

```

- In the above case, we are initializing the variable x of the class under the constructor. The advantage of initializing it under the constructor is each object of the class that is created can have different values for execution.



Why do we define Constructor explicitly when there are implicit Constructor-

- Each & every class will be requiring some initialization value to execute the values that are required for a class to execute, will be sent to the class through the constructor only because we static accessing member of class only after object creating & to create object we call the constructor so if we send to value to a class through the constructor making use of those values the class can start execution.
- A parameterless constructor can not be used for passing values to the class so we define a parameterized constructor explicitly & then we pass values to the class for execution while creating the object.
- So whenever we create object of a class, we have a chance of sending a different value to the class for execution.

Add a class math.cs & write the following code:-

```

Class Math
{
    int x, y;                                // Class Variable
    Public Math (int x, int y)               // Block Variable
    {
        this.x = x;
        this.y = y;
    }
    Public void Add()
    {

```

```

Console.WriteLine (x+y);
}
Public void Sub ()
{
    Console.WriteLine (x-y);
}
Public void mul ()
{
    Console.WriteLine (x*y);
}
Public void Div ()
{
    Console.WriteLine (x/y);
}
}

```

The above class contains 4 methods on it which has to be executed by some set of values that are required for execution or sent to constructor of the class so that whenever it is created we can send it values for the class to execute & then call the methods of the class.

To test this, add a new class Testmath.cs :-

```

Class Testmath ()
{
    Static void Main ()
    {
        Math m = new Math (100,50);
        m.add(); m.sub();
        m.mul(); m.div();
        Console.ReadLine();
    }
}

```


Static Modifier :-

70

The members of a class are divided into two categories -

- ① Static member
- ② Non static member

- The member of a class which does not required object of the class for initialization or execution are known as Static members.
- The member which requires the object for initialization and execution are known as non static members.

Static Variables Vs Non Static Variables -

- ① A variable that is declared by using a static modifier or a variable that is declared inside of any static block were considered as static variables, whereas rest of the other are non static only.

Ex- `int x = 100 ; // non static`

`Static int x = 50 ; // static`

`Static void Main()`

`{`

`int x = 100 ; // static`

`}`

- ② A static variable get initialized immediately once the execution of a class starts. Whereas a non static variable gets initialized only after creating object of the class as well as each time the object of the class is created.

- ③ A static variable get initialized one & only one time in the life cycle of the class whereas a non static variable is initialized either 0 or n time, based on no. of object is created.

- ④ When we are accessing member of a class access them by using name of the class whereas non static members by using object of the class.

Constant Variable -

A variable whose values cannot be modified once after declaration is known as a constant variable and must be declare by using the keyword "const"

Note -

- It is mandatory to initialize a Constant variable at the time of its declaration.
- The behaviour of a constant variable is same as the behaviour of static variable i.e, maintains one and only one copy in the life cycle of class & initialize immediately once the execution of class start.
(Object not required)
- The only difference between a static and constant variable is that the static variable can be modified but constant variable can not.

Read Only Variable -

- A variable whose values cannot be modified once after initialization is known as read only variable & must be declared using "readonly" keyword.
- A read only variable need not to be initialized at the time of its declaration like a constant. It can also be initialized under a constructor but once after initialization, it is not possible to modify the value.
- The behaviour of a read only variable is a similar to the behaviour of a non static variable i.e, maintain a separate copy for each copy that is created. The only difference between two is : non static variable can be modified but not read only.

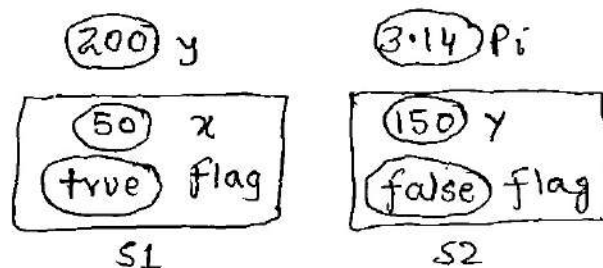
- 72
- A constant variable is a fixed value for the complete class whereas a readonly variable is a fixed value specific to one object of the class.

Add a class Statvars.cs & write the following -

```

Class StatVars
{
    int x;
    Static int y = 200;
    Const float Pi = 3.14f;
    readonly bool flag;
    Public Statvars(int x, bool flag)
    {
        this.x = x;
        this.flag = flag;
    }
    Static void Main()
    {
        Console.WriteLine(Statvars.y);
        Console.WriteLine(Statvars.Pi);
        Statvars s1 = new Statvars(50, true);
        Statvars s2 = new Statvars(100, false);
        Console.WriteLine(s1.x + " " + s2.x);
        Console.WriteLine(s1.flag + " " + s2.flag);
        Console.ReadLine();
    }
}

```



Static Method Vs. Non Static Method -

- A method that is declared by using static modifier is a static method & rest of other are non static only.
- While defining a static method under the method, if we want to consume only non static members we can consume them only by using object of the class whereas

Under non static method, we can consume static method directly without any restriction.

Rule -

- ① Non static member of a class can be consumed on a static block only by using object of the class.
- ② Static to static :- Can be consumed directly or ^{with} class name also.
- ③ Static to non static :- Can be consumed directly or class name also.
- ④ Non static to Non Static :- Can be consumed directly or by using 'this' keyword.

Ex- Class Statmets

```

{
    int x = 100;
    static int y = 200;
    static void Main() add()
    {
        Statmets obj = new statmets();
        Console.WriteLine(obj.x + y);
    }
    static void Main()
    {
        Statmets.Add()
        Console.ReadLine();
    }
}

```

Static Constructor Vs. Non Static Constructor -

- If constructor is explicitly define by using the static modifier we call it as a static constructor rest of the other non static only.
- Static constructor is first block of code which executes under a class whereas a non static constructor get executed only after creating the object of class as well as each time, the object of class is created.
- In the life cycle of the class a static constructor get executed one & only one line whereas a non static constructor gets executed each & every time the object is created the min count is '0' (n= object are create) max count is 'n'.
- Static constructor cannot be parameterised, it must be parameterless only because static constructor is first block of code that execute under a class and more over implicitly called to even if parameterized there is no chance to sending a value.

Ex-

```

Class Statcon
{
    static Statcon()
    {
        Console.WriteLine("Static constructor is called");
    }
    Public Statcon()
    {
        Console.WriteLine("Non-static constructor is called");
    }
    static void Main()
    {

```

```
Console.WriteLine("Main method is called");
```

```
StatCon s1 = new StatCon();
```

```
StatCon s2 = new StatCon();
```

```
Console.ReadLine();
```

```
}
```

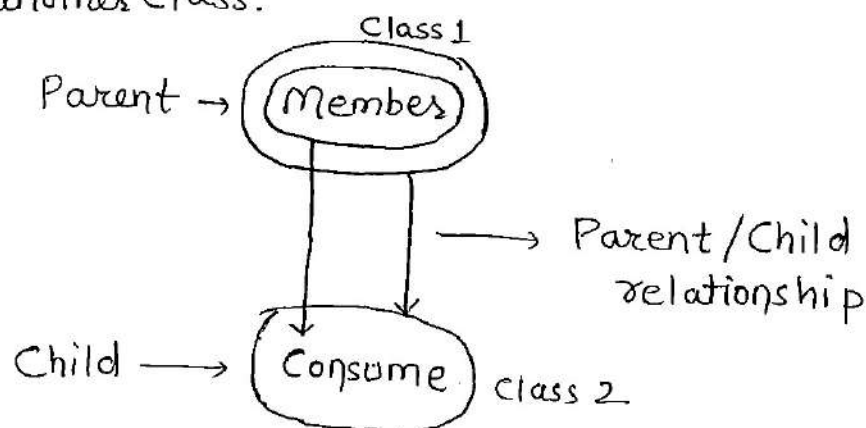
```
}
```

Static Class —

- A class that declared by using a static modifier is a Static Class & it can contain only static member in it.
- We cannot create object of a static class because it contain only static member & more over of the class is not required for static class to execute.

Inheritance —

- Acquiring properties of parent by their children is known as inheritance which provides reusability.
- We can use same principle in an object oriented language & establish Parent-child relationship between classes, so that the member that are defined in one class can be consumed under another class.



Syntax —

```
[<modifier>] Class <C.C name> : <PC name>
```

eg. Class class 1
 { members }

```
Class Class 2 : Class 1
```

```
{ Consume members of Parent here }
```

```
Class Class 1
{
    Public Class 1()
    {
        Console.WriteLine("Class 1.Constructor");
    }
    Public void Test 1()
    {
        Console.WriteLine("First Method");
    }
    Public void Test 2()
    {
        Console.WriteLine("Second Method");
    }
}
```

Class 2. CS -

```
Class Class 2 : Class 1
{
    Public Class 2()
    {
        Console.WriteLine("Class 2 Constructor");
    }
    Public void Test 3()
    {
        Console.WriteLine("Third Method");
    }
    Static void Main()
    {
        Class 2 c = new Class 2();
        c.Test 1();
        c.Test 2();
        c.Test 3();
        Console.ReadLine();
    }
}
```


In inheritance private members of Class cannot be consumed by their child classes.

Rules & Regulation that has to be followed by working with inheritance

① In inheritance the constructor of the parent class must be accessible to its child class, otherwise the inheritance will not possible.

- The reason why ^{it} should be accessible is when we create the child object first it goes and calls the parent classes constructor so that parent classes variable will be initialized & we can consume them under the child class.

Note-

The reason why a child class internally calls its parent class constructor is to initialize variable of parent class & that we can consume them under child class.

② In inheritance child classes can consumes parent classes members but a parent class can never consume any member of child class that are 'Purely defined in the child class'.

- To test this rewrite the code under child class main() as following :-

```
Class L p = new Class 1();
P.Test1(); P.Test2();
// Can't call Test3() by using object of Class 2 (Parent)
// P.Test3();
Console.ReadLine();
```

③ Just like the object of a class can be assigned to variable of same class to make it as a reference it can also be assigned a variable of its parents to make it as reference. So that the reference starts consuming memory of object assigned to it, but now also using that we control access child pure classes members.

- To test this rewrite the code under main() of class 2 :-

Class 2 c = new Class2(c);

78

Class 1 p = c;

P.Test 1();

P.Test 2();

// Can't call test by using reference of class 1 (parent) because pure child classes members are not accessible to parents reference also even if they are called by using child class object.

// P.Test 3(); can't be called now also

Console.ReadLine();

Note-

A parent classes object can never be assigned to a child classes variable.

A parent classes reference i.e, created by using child classes object can be converted back into a child class reference if required by performing an explicit conversion which can be done in two ways :-

// Creating parents reference by using child's object

Class 2 c = new Class 2();

Class 1 p = c;

// Converting parents reference back into child's reference :

Class obj = (Class 2) P;

or,

Class obj = P as class 2;

Object is first base class which is define in library. Object is default parent class.

Each & each class in .NET Language, contain 4 method in Common those are :-

① equals

② gethash code

③ get type

④ ToString

Inherited from the object class because the Class object is a default parent class of all the classes & members of this class can be consumed from any where.

Note-

Whenever a class being compile the compiler first verifies whether the class is inheriting from other class or not. If inheriting ~~number~~ no issue or else will be inherited from object class.

- To test the process of consuming object class members from any where rewrite the code under main method of class 2 as following :-

```
Object obj = new object();
Console.WriteLine(obj.GetType());
Class1 p = new Class1();
Console.WriteLine(p.GetType());
Class2 c = new Class2();
Console.WriteLine(c.GetType());
Console.ReadLine();
```

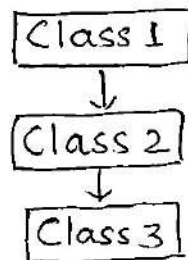
Types of Inheritance-

It is concept ~~of~~ which tells the number of parent classes a child class can have. As per the standards of object oriented programming, there are only two types of inheritance :-

- ① Single Inheritance
- ② Multiple Inheritance

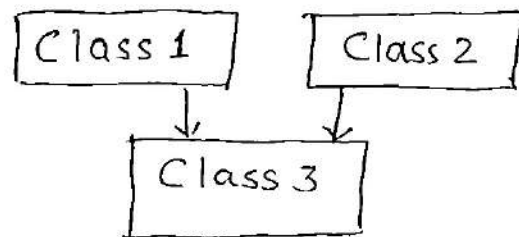
If a class is having one immediate parent class to it, we call it as single inheritance, whereas if a class has more than one immediate parent class to it, we called it as multiple inheritance.

Single Inheritance

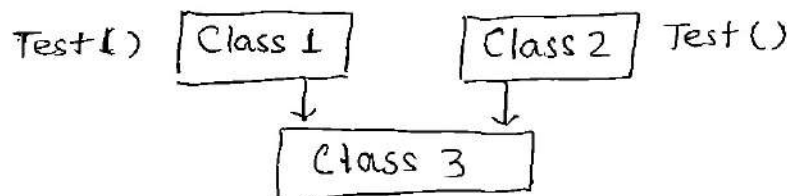


Multiple Inheritance

80



④ Under .Net & Java languages, we don't have support for multiple inheritance through classes, we are only provided with single inheritance because multiple ~~inheritance~~ inheritance suffers with the problem of ambiguity. Problem of ambiguity is, chances of having a member with the same name under multiple parent classes from where a child class is inheriting.



Note:-

In traditional C++ language, we are provided with both single & multiple inheritances also, because C++ is the first object oriented language that came into existence & at that point this problem was not anticipated.

⑤ In our first rule of inheritance, we have learned whenever the child class object is created, it implicitly calls its parent class constructor, but implicit calling of parent class constructor from child class is possible only if parent class constructor is parameterized, child class cannot call the constructor implicitly because the parent class constructor requires value for execution & child class does not know what value is suitable for the parent class to execute.

- To resolve this problem, now it is responsibility of us as a programmer to explicitly call the parent's constructor from child class & pass required values for execution. To call the parent's constructor from child class use "base" keyword.

Note-

- Using base keyword we can refer any non static member of parent class from child class.
- We need to use base keyword & call the parent class constructor in child class from the child classes constructor.
- To test this process, do the following :-
Open the class Class 1 & rewrite the ~~code~~ constructor as following :-

```
public class 1 (int x)
{
    Console.WriteLine("Class 1 Constructor. " + x);
}
```

Note-

- After making the above modification, if we run the child class, we will get an error stating that Class 1 does not contain any constructor that 0 parameters (so implicit calling is not possible).
- Now to fix the problem, rewrite the constructor of class 2 :-

```
public class 2 (int p, int c) : base (p)
{
    Console.WriteLine("Class 2 Constructor" + c);
}
```
- Now in Main() of class 2 rewrite the code as following :-

```
Class 2 c = new Class 2 (10, 20);
Console.ReadLine();
```

Note-

In the above case, in the two values we are sending for the class to execute the first value ~~is~~ is sent to class 1 for execution and second value is used by Class 2 for execution.

How to make use of inheritance in application development ? ⁸²

Generally when we develop an application, we will be following a process as following :-

- ① Identify the entity that associated in the application.
- ② Identify the attribute that are associated with the application.
- ③ Now separate the attribute of each entity in a hierarchical order without having any duplicates.
- ④ Convert those entity into classes.

- Suppose we have developing an application for school the entities in the attribute of entity will be as following :-

Student

id
name
address
phone
Class
Fees
Marks
Grade

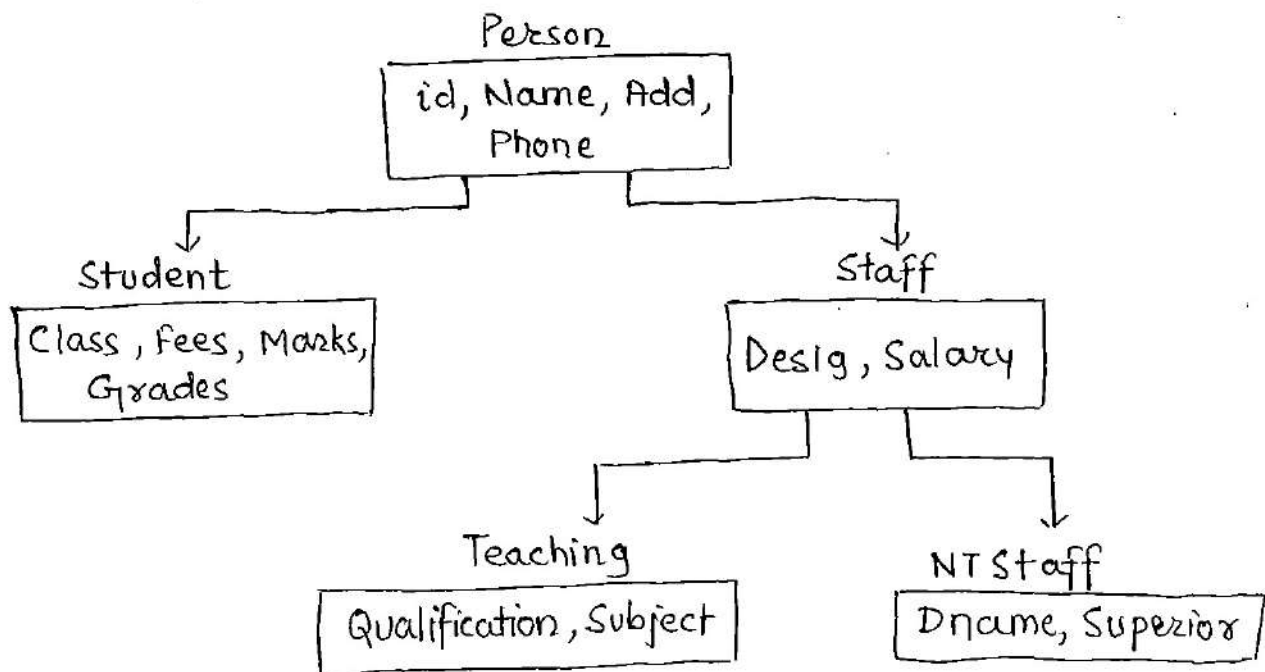
TStaff

id
name
address
phone
Designation
Salary
Qualification
Subject

NTStaff

id
name
address
phone
designation
Salary
Dname
Supervisor

- Now separate the attribute of that entity based on the hierarchy as following :-



Now define the class representing the entity as following:-

```

public class person
{
    id, name, add, phone
}
public class student : Person
{
    Class, Fees, Marks, Grade
}
public class Staff : Person
{
    Designation, Salary
}
public class Teaching : Staff
{
    Qualification, Subject
}
public class NTStaff : Staff
{
    Dname, Superior
}

```

Polymorphism -

- Behaving in different ways depending upon input receive is known as polymorphism. i.e, Whenever input changes automatically the output or behaviour also changes.
- Polymorphism can be implemented in object oriented programming language using 3 different approach -

- ① Overloading
- ② Overriding
- ③ Hiding/Shadowing

- Overloading is again of 3 types -

- Method Overloading
- Constructor overloading
- Operator overloading

- Add a class LoadCon.cs & write the following:-

86

```
Class LoadCon
{
    int x;
    public LoadCon() // Default Constructor
    {
        this.x = 10;
    }
    public LoadCon(int x)
    {
        this.x = x;
    }
    public void Display()
    {
        Console.WriteLine("Values of x is: {0}", x);
    }
    static void Main()
    {
        LoadCon c1 = new LoadCon();
        LoadCon c2 = new LoadCon(20);
        c1.display();
        c2.display();
        Console.ReadLine();
    }
}
```

- Generally we overload constructor of a class so that variables of that class can be initialized either with default values with the help of parameterless constructor or can be initialized with the given values that are passed as parameters to the constructors while creating the object.

Note -

Parameterless constructors are also known as default constructors, because they will initialize variable of a class with the default value.

Inheritance Based Overloading -

A method is defined under a parent class also be overloaded under the child class & if they are overloaded in this process, we called it as inheritance based overloading.

Ex-

Class 1

public void Test ()

Class 2 : Class 1

public void Test (int x)

Note-

A child class to overload its parent class method under it, doesnot requires any permission from its parent.

Method Overriding -

If at all a parent classes method is redefined or reimplemented under the child class, exactly with the same name & signature, we call it as method overriding.

Difference between method overloading and method overriding -

Method Overloading

- It is an approach of defining multiple method with same name & different signatures.
- Method can be overloaded either within a class or under parent child classes.
- To overload parent classes method under child class, child class does not require any permission from parent.
- This is all about defining multiple behaviour to method.

Method Overriding

- It is an approach of defining multiple methods with same name & same signature.
- Methods can be overridden only between parent & child class but never can be performed within a class.
- To override parent method under child class, child class requires an explicit permission from parent.
- This is all about changing the behaviour of method.

How to override a parent classes method under child class?

- To override parent classes method under child class, first the method in parent class should be declared by using virtual modifier. Declaring the method as virtual is marking the method as overridable.
- Methods that are declared as virtual under a parent class can now be overridden under the child class by using the override modifier.

Class 1

```
public virtual void Show() //overridable
```

Class 2 : Class 1

```
public override void show() //overriding
```

- In overriding parent class, defines a method as virtual and gives it to child class to consume that method. So the child class now consume the method as it is or override that method as per the requirement of the child class, so overriding parent classes virtual method under a child is only optional. ⇒

Add a class LoadParent.cs & write the following code :-

```
Class LoadParent
{
    public void Test()
    {
        Console.WriteLine("Parent's Test Method.");
    }
    public virtual void show // overridable
    {
        Console.WriteLine("Parent's Show method.");
    }
    public void display()
    {
        Console.WriteLine("Parent's Display Method.");
    }
}
```

Add a class LoadChild.cs & write the following :-

```
class LoadChild : LoadParent
{
    // Overloading parent's test method under child
    public void Test (int x)
    {
        Console.WriteLine ("Child's Test Method.");
    }
    static void Main()
    {
        LoadChild c = new LoadChild();
        c.Test(); // Call's Parent test method
        c.Test(10); // Call's Child's test method
        c.Show(); // Call's Parent's test method
        c.Display(); // Call's Parent's display method
        Console.ReadLine();
    }
}
```

- In above case, we are overloading the test method of parent class under child class & then calling both the two method by using child classes by object.
- The show() of the parent class is declared as virtual under the parent class so that the child class can override the method, but the child class did not override the method. When show() is called using child class object, it will execute parent classes method, whereas if at all the child class is overriding the method, they call the show() under child class will not execute parent class method but will execute child classes method.
- To test this first run the child class once & watch the output of show method, where we will notice the parent class method giving the output. Now add a new method under the child class LoadChild as following :-

11 Overriding parent's show method under child

```
public override void show()
{
    Console.WriteLine("Child's Show Method");
}
```

- Now reexecute the child class again & watch the output of show method where in this case, child class method get executed but not parent class.

Method Hiding / Shadowing -

This is another approach that is used for reimplementing a parent class method under child class. A parent classes method can be reimplemented under its child class using 2 different approaches :-

1. Overriding
2. Hiding / Shadowing

In the first case we reimplement the parent class method that are declared as virtual under the child class by using override modifier whereas in second case. We reimplement the parent classes method under child class even if they are not declared as virtual i.e., reimplementation being performing without parent classes permission.

Class 1

public void display()

Class 2 : Class 1

public new void display()

Note:-

Using the new keyword while reimplementing the method in child is only optional or else compiler issue a warning message asking to use the new keyword with hiding was intentional.

- To test this first, run the child class Load Child &

- watch the output of the display method where we will be noticing the display method in parent method being called.
- Now add a new method in the child class LoadChild as :-

```
// Hiding or shadowing parents display method under child
public new void display()
{
    Console.WriteLine("Child Display Method");
}
```

- Now again run the child class LoadChild & watch the output of display method. Now child class method gets executed but not parent classes method.

In the above two we are performing the following :-

```
LoadParent
public void Test()
public virtual void show()
public void Display()
```

```
Load Child : Load Parent
public void Test(int x) // overloading
public override void show() // overriding
public new void Display // Hiding or shadowing
```

Calling Parent Classes Methods from child Class after reimplementing in child class —

After reimplementing parent classes methods under child class, object of the child class calls its own methods but not its parent class method, whereas if we want to still consume or call the parent classes methods from child class, it can be done in two different ways :-

- ① By creating parent classes object under child class, we can call parent classes method from child class.

- To ~~the~~ test this rewrite the code under child classes Main() as following:

Load Parent p = new LoadParent();

92

p.show(); // Calls Parent's show method

p.display(); // Calls Parent's display Method

LoadChild c = new LoadChild();

c.show(); // Calls Child's show method

c.display(); // Calls child's display method

Console.~~Read~~ReadLine();

② By using the base keyword, we can call parent classes method from child class, but this and base keyword cannot be used under static block.

- To test this process, now add two new methods in Child's class as following:

```
public void Pshow()  
{  
    base.show();  
}  
public void PDisplay()  
{  
    base.display();  
}
```

The above two methods acts as a interface for calling parent classes methods from child class directly by using child class object. To check that rewrite the code under child class main method as following :-

LoadChild c = new LoadChild();

c.Pshow(); // Call's parents show method

c.PDisplay(); // Calls Parents display method

c.show(); // Calls child's show method

c.Display(); // Calls child's display method

Console.ReadLine();

Note-

In the 3rd rule of inheritance, we have discussed that a parent class reference even if created by using child class object, cannot access the child class member but we have an assumption for this rule in overriding i.e., Parents reference can call child classes overridden members because overridden ~~metho~~ members, they are first defined in parent class & then reimplemented under child class with the permission of parent class, whereas now also using that reference any ~~pub~~ pure child class member cannot be called.

To test this rewrite the code under child main method as :-

```
LoadChild c = new LoadChild();
LoadParent p = c;
p.show(); // Calls child's show method
p.Display(); // Calls Parent's display method.
Console.ReadLine();
```

Sealed Class & Sealed Method -

- A class that is defined by using the sealed modifier is a sealed class, where a sealed class cannot be inherited by any other class.

```
sealed class class 1
{ - Define members }
```

```
class class 2 : class 1 // Invalid
```

Note-

Even if a sealed class cannot be inherited, we can still the members of that class by creating its object.

For example-

String is a sealed class in our base class library string anywhere by creating its object.

- A method that is defined in a parent class, if cannot be overridden under the child class, we call it as a sealed method. By default every method is a sealed method,

because we can never override any parent classes method⁹⁴ under child class without the method under child class without the method declared as virtual in parent class.

- If at all a method under a class is declared as virtual, any child class of a linear hierarchy has a right to override the method.

Class 1

public virtual void show()

Class 2 : Class 1

public override void show()

Class 3 : Class 2

public override void show() // valid

Note-

- Even if class 2 is not overriding the method then also class 3 override the method.
- If at all a virtual method of a parent class has to be restricted with further overriding of the method, so that the next child classes cannot override the method, we need to use sealed modifier on the method while overriding it.

Class 1

public virtual void show()

Class 2 : Class 1

public sealed override void show()

Class 3 : Class 1

public override void show() // Invalid

OPERATOR OVERLOADING -

- This is something very much similar to the concept of method overloading which allows to define multiple behaviours to an operator.
- In method overloading the behaviour of method changes according to the type of parameter being passed to the method.

95

, whereas in the operator overloading the behaviour of an operator changes according to the operands types between which we use the operator.

For eg-

'+' is an overloaded operator which can be used both for addition as well as concatenation also. It works as a addition operator when used between numeric operands and works as a concatenation operator when used between string operands or string & numeric operands.

- We can also add new behaviour to any existing operators if required by defining a operator method as following.
- Operator method must be defined as static only.

Syntax-

```
[<modifiers>] static <type> Operator<opt> (<operand type def">)  
{  
    - stmts -  
}
```

- <type> implies the type of the result we are expecting when the operator is used between two operands.
- <Operand type definition> in the sense the type of operands between which we want to use the operator.
- operator - is the name of the method that cannot be changed.
- <opt> - implies the operator we wanted to overload.
- '+' is already an overloaded operator under the base class library that is defined with various overload as following -
public static int operator + (int i1, int i2)
public static string operator + (string s1, string s2)
public static string operator + (string s, int i)

Class Matrix

{

// Declaring attributes for a 2*2 matrix

public int a, b, c, d;

public Matrix(int a, int b, int c, int d)

{

// initializing the attributes

this.a = a; this.b = b; this.c = c; this.d = d;

{

// overloading + operator for using it between operands
of type matrix

public static Matrix Operator + (Matrix M1, Matrix M2)

{

Matrix obj = new Matrix (m1.a + m2.a;

+ m1.b + m2.b;

+ m1.c + m2.c;

+ m1.d + m2.d;

return obj;

{

// overloading - operator for using it, between operands
of type matrix

public static matrix operator - (Matrix 1, matrix 2)

{

Matrix obj = new matrix (M1.a - M2.a, M1.b - M2.b,
M1.c - M2.c, M1.d - M2.d)

return obj;

{

// overloading the ToString() method inherited from
object class ~~class~~ to get the desired output

public override string ToString()

{

String s = a + " " + b + "\n" + c + " " + d + "\n";

return s;

```

}
}

```

Add a class TestMatrix.cs & write the following code

```

Class TestMatrix

```

```

{

```

```

    Static void Main()

```

```

    {

```

// Creating different matrix object with different values

```

        matrix m1 = new matrix (1,2,3,4,5);

```

```

        matrix m2 = new matrix(3,4,5,6);

```

```

        matrix m3 = new matrix (5,6,7,8);

```

// Performing arithmetic operation on matrix

```

        matrix m4 = m1 + m2;

```

```

        matrix m5 = m3 + m4;

```

```

        matrix m6 = m3 - m2;

```

// Printing the matrix value in 2*2 matrix format

```

        Console.WriteLine(m1);

```

```

        Console.WriteLine (m2);

```

```

        Console.WriteLine (m3);

```

```

        Console.WriteLine (m4);

```

```

        Console.WriteLine (m5);

```

```

        Console.WriteLine (m6);

```

```

        Console.ReadLine();

```

```

    }

```

```

}

```

- Whenever we try to print the object of any class using write or write methods it will not print the values associated with that object it will only name of the class to which the object belongs.

- When we use the object under write or writeLine method for printing internally on that ~~project~~ object to String method

of the class inherited from object class gets called & that method identifies the name of the class to which the object belong & returns it which is captured by the write or writeline method & printed back. 98

- The testing method that is inherited from object class is internally declared as virtual, it can be overridden under any class because every class is a child class of object class, so in the above case we are overriding the ToString() method under matrix class so that it return values of the matrix in matrix form without returning the class name.

Static and Dynamic Polymorphism -

Polymorphism is divided into 2 types :-

- ① Static / Compile time / Early binding
 - ② Dynamic / Runtime / Late binding
- In case of overloading methods whenever we call a method for that method call based on parameter it identifies which method should be executed and binds the method calls its method definition and that method executed in runtime.
 - In the second case that is dynamic polymorphism, if overloading or hiding are used, then the program for those methods when compiler will identify for a particular called which method used to executed in runtime & executes that method based on the hierarchy choosing all the different reimplemented if the method.

Abstract Methods & Abstract Classes -

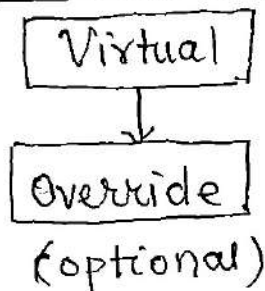
- A method without method body is known as a Abstract method. It contains only declaration of the method. If we want to define a abstract method we need to use abstract modifier on that method.
- The class under which we define abstract method is called as an abstract class and we need to use an abstract modifier on the class also while defining it.

Ex-

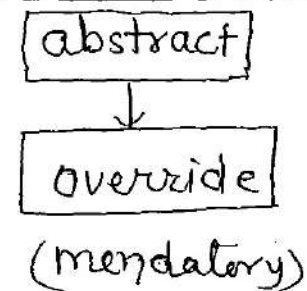
```
abstract class Math
{
    public abstract void add(int x, int y);
}
```

- A concept of abstract method is near similar to the concept of method overriding. In method overriding if a parent class contains any virtual methods in it, those methods can be reimplemented under the child class by using the override modifier, whereas in abstract methods. if a parent class contains any abstract methods in it, those methods must be implemented under the child class using override modifier.

Method Overriding



Abstract Methods



- An abstract class can contain both abstract & non-abstract member in it. If a child class of an abstract class wants to consume any non abstract methods of its parent should implement all abstract method of its parent.

Abstract Classes -

- Non abstract methods
- Abstract methods

Child Class of Abstract Class: -

160

- Implement all the abstract methods of parent
- These only we can consume non abstract methods of parent

Note -

An abstract class is never usable to itself, because we can't create the object of an abstract class. The members of an abstract class can be consumed only from the child class of the abstract class.

Add a class AbsParent, write a following code -

```
abstract class AbsParent
{
    public void add (int x, int y)
    {
        Console.WriteLine (x+y);
    }
    public void sub (int x, int y)
    {
        Console.WriteLine (x-y);
    }
    public abstract void Mul (int x, int y);
    public abstract void Div (int x, int y);
}
```

Add a class AbsChild.cs & write the following code:-

```
class AbsChild : AbsParent
{
    public override void mul (int x, int y)
    {
        Console.WriteLine (x * y);
    }
    public override void Div (int x, int y)
    {
        Console.WriteLine (x/y);
    }
    static void Main ()
    {

```

```

AbsChild c = new AbsChild();
c.Add(100, 50);
c.Sub(54, 64);
c.Mul(10, 24);
c.Div(56, 4);
Console.ReadLine();
}
}

```

Note:-

As we discussed about the object of a abstract class cannot be created but still it is possible to create a reference of the abstract class by using its child class object & with the help of that reference we can access or call the non abstract member of itself as well as abstract members of itself that are implemented in child class.

To test this rewrite the code under main method child class AbsChild as following -

```

AbsParent c = new AbsChild();
AbsParent p = c;
p.Add(50, 100);
p.Sub(54, 24);
p.Mul(10, 24);
p.Div(56, 4);
Console.ReadLine();

```

What is the need of Abstract Classes & Abstract Methods?

The concept of abstract class can be taken as an extension to the concept of inheritance i.e., in inheritance we have discussed that whenever we are developing an application, first we will identify the entities that are associated with the application & then the attributes of those entities.

Suppose in an application, we need the entities like

102

rectangle, circle, cone, triangle etc. First we need to identify attributes & members we want under the entities (class)

<u>Rectangle</u>	<u>Circle</u>	<u>Cone</u>	<u>Triangle</u>
- Width	- Radius	- Radius	- Width
- Height	- PI	- Height	- Height
- GetArea()	- GetArea()	- PI	- GetArea()
- GetPerimeter()	- GetPerimeter()	- GetArea()	- GetPerimeter()
		- GetPerimeter()	

- In the above case, the 4 entities have attributes in common and also requires the two methods GetArea() and GetPerimeter() in common. As we discussed earlier, To avoid duplication, we can define common members under a parent class.

- In the above context we can define parent class for the 4 class & declare the common attributes width, height, radius and PI under it and consume them under child class but when coming to the method GetArea() and GetPerimeter(), even if these method are common in all 4 class, implement of the method change from class to class. So we cannot implement the methods under the parent class, which must be implemented under child class only.

- As we required, GetArea() & GetPerimeter() in all the child class but with different implemented, it will be better if we define the methods as abstract under Parent class & then implement under all the child classes. If at all the method are declared abstract in the parent.

We have 2 advantage:-

① There is a guarantee, in every child class these method will parent.

② There is a guarantee that in all the child classes

the method will have same name & same signature.

- Implementation of the classes will be as following -

Add a class Figure.cs & write the following code:-

```
public abstract class Figure
{
    public double width, Height, Radius;
    public const float Pi = 3.14f;
    public abstract double GetArea();
    public abstract double GetPerimeter();
}
```

- The above class Figure is now going to be a parent class for all the figures we want like Rectangle, Circle, Cone, etc. So all the child Class can make use of the variables (attributes) directly under them as well as must implement the methods GetArea() and GetPerimeter().

- In every child Class of figure, we need to do following:-

- ① Define a constructor for initializing the attributes associated with that entity.
- ② Implement the getArea() and getPerimeter() methods according to that entity.

Add a class Rectangle.cs & Write the following code -

```
public class Rectangle : Figure
{
    public Rectangle(double width, double Height)
    {
```

// Here using this or base keywords will be same

this.width = width;

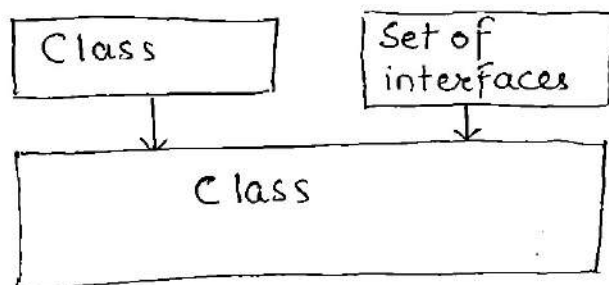
base.Height = Height;

```
}
```

```
public override double GetArea()
```

```
{
```

```
    return width * Height;
```



Inheritance is divided into two category :-

- ① Implementation Inheritance
- ② Interface Inheritance

- In the first case, a class inherits from another class to consume the members of its parent, whereas in second case, a class inherit from an interface for implementing the members of its parent interface.
- Implementation inheritance is single whereas interface inheritance is multiple both under .NET & Java. But in C++ both are multiple.

Note-

In the case of implementation inheritance, a parent class provides members to its child class for consumption whereas in the case of interface inheritance, the parent interface is not providing any members to the child for consumption. It is only requesting the child class to implement its members. So interface inheritance doesnot provide any reusability.

Syntax-

```

[<modifiers>] interface <Name>
{
    - Abstract member declaration
}
  
```

- We cannot declare any variable under interface.
- Every member of interface is by default abstract. so we do not require to declare them as abstract again explicitly.

- The default scope for the members of the interface is public where it is private in the case of the ~~class~~ class.
- If required an interface can inherit from another interface but cannot inherit from a class.

Add a interface item under the project choosing it from the add new item window & name it as Inter1.cs, write the following code in it :-

```
interface Inter1
{
    void Add(int x, int y);
    void Sub(int x, int y);
}
}
```

Add another interface Inter2.cs & write the following :-

```
interface Inter2
{
    void Mul(int x, int y);
    void Div(int x, int y);
}
}
```

Now to implement the four abstract methods that are defined under both the two interfaces, add a class name it as ImplClass.cs & write the following :-

```
Class ImplClass : Inter1, Inter2
{
    public void Add(int x, int y)
    {
        Console.WriteLine(x+y);
    }
    public void Sub(int x, int y)
    {
        Console.WriteLine(x-y);
    }
}
```



```

    }
    public void Mul (int x, int y)
    {
        Console.WriteLine (x * y);
    }
    public void Div (int x, int y)
    {
        Console.WriteLine (x / y);
    }
    static void main ()
    {
        ImplClass c = new ImplClass ();
        c.Add (123, 567);
        c.Sub (98, 23);
        c.Mul (21, 34);
        c.Div (78, 3);
        Console.ReadLine();
    }
}

```

Note-

We cannot create the object of the interface, it is still possible to create a reference of the interface by using its child classes object and with that reference we can call all the method that are declared in the interface & implemented under child. To test this rewrite the code under main method of above class as following:-

```

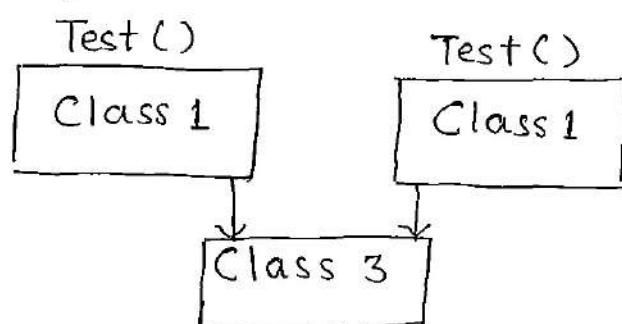
ImplClass c = new ImplClass ();
Inter i1 = c;
Inter i2 = c;
i1.Add (123, 567); i1.Sub (98, 23);
i2.Mul (21, 34); i2.Div (78, 3);
Console.ReadLine();

```

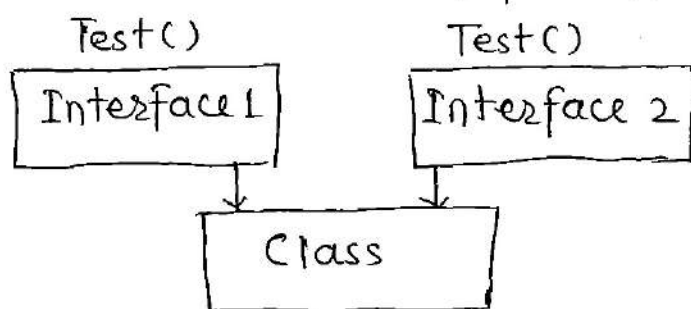

Multiple Inheritance with Interfaces & Ambiguity —

109

- When we discussed about classes & inheritance, we have been discussing that multiple inheritance is not supported through classes because of ambiguity problem, but multiple inheritance is supported through interface, because here there is no chance of coming across the ambiguity problem.
- If a class is inheriting from more than one class at a time, if at all the multiple parent class contain any member with the same name and signature, ambiguity arises because from the multiple parent classes will be pushed into the child & two members with same name & signature will cause a problem.



- Ambiguity will not arise at all if the two parents of interfaces, because in case of an interface, the child class is not consuming members of parent. It only has to implement the members of parent.



If we come across this situation any time, implementation for the method can be given under child class in two different ways:—

- ① Implement the method only for one time under the child class, where both the interfaces assumes the implemented method is of it only, so we can directly call the method using child class object of implementation.

② We can also implement the method under the class separately for each interface by prefixing the method with interface name by implementation, but in this case we cannot call the method by using class object must be called only by using the reference of interface created by using the child class object.

Add a interface in a project naming as interface 1.cs & write following code:-

```
interface interface 1
{
    void Test();
    void Show();
}
```

Add another interface interface 2.cs & write the following :-

```
interface interface 2
{
    void Test();
    void Show();
}
```

Add a class ~~interface~~ interclass.cs & write following :-

```
Class interclass : interface 1, interface 2
{
    public void Test()
    {
        Console.WriteLine("Declared under 2 interfaces");
    }
    void interface 1.show()
    {
        Console.WriteLine("Declared under interface 1");
    }
    void interface 2.show()
    {
        Console.WriteLine("Declared under interface 2");
    }
}
```

```

}
static void Main()
{
    Interclass c = new Interclass();
    c.Test();
    Interface1 i1 = c;
    Interface2 i2 = c;
    i1.Show();
    i2.Show();
    Console.ReadLine();
}
}

```

Structure —

- It is also user defined type similar to a class or interface.
- In C# language, we can define under a class like variables, methods, constructor etc, whereas in traditional language C under structure we can only declare variable.

Difference between a class and a structure —

<u>Class</u>	<u>Structure</u>
<ul style="list-style-type: none"> - It is a reference type. - Memory is allocated on managed heap. So the garbage collector can perform automatic memory management. - Used for representing entity with huge volume of data. - Requires new keyword for creating object. 	<ul style="list-style-type: none"> - It is a value type. - Memory is allocated on the stack, so there is no support of garbage collector for memory management but faster in access. - Used for representing entity with smaller volumes of data. - Using new keyword for object creation is only optional.

- Variable can be declare and those variables can be initialized at the time of declarazation.

- Variable can also initialize under constructor, we can directly assign to the variable refering it through the object.
ie,

`<Object>.<variable> = Value`

- Requires a constructor for ~~collin~~ creating the object, which can be either parameterless or parameterized.

- Programmer can define either parameterless or parameterised constructor also.

- If '0' constructor of define after compilation, there will be '1' constructor & if 'n' constructor will be define after compilation n const only.

- Supports both implementation and ~~inheritance~~ ~~also~~ interface inheritance also.

- All predefined reference type like string object are internally implemented as classes in base class library.

- Variable can be declared but can not be initialized and 112 time of declarazation.

- Variable can only be initialized under constructor or we can directly assign to variable refering it through the object ie,

`<Object>.<variable> = <Value>`

- Requires parameterless constructor at any cost for creating the object which gets used when the object is create d without new keyword.

- Programmer can ~~also~~ define also parameterized constructor because a parameterless constructor is mendatory for creating the object. Here is already an implicit parameterless constructor.

- If '0' constructor of define after compilation there will be '1' constructor & if 'n' constructor will be define after compilation n+1 const only.

- Supports only interface inheritance but interface inherisance doesnot provide Reusability.

- All predefined value type int, float, bool, char etc are internally implemented as struct under BCL.

Note -

C# language suffers from criticism that it is not fully object oriented because of the supports of structure in the language where this structure doesnot provide reusability through inheritance.

Syntax of Structure -

```
[<modifier>] struct <Name>
{
    - members
}
```

- To define a structure under a project, we have not provided with a structure template under the add new item. just like we have given for class & structure. So to define a structure in project we need to use the code file item template which is blank C# file that can be used for defining any thing in it like a class, interface or structure.
- Now open the add new item window and choose a code file item template in it & name it as MyStruct.cs and write the following code:-

```
using system;
Namespace OOPS project
{
    struct MyStruct ()
    {
        int x;
        Public MyStruct (int x)
        {
            this.x = x;
        }
        public void Display ()
        {
            Console.WriteLine ("Method in struct " + x);
        }
    }
}
```

```

static void Main()
{
    MyStruct m1; // Call Parametess Constructor
    m1.x = 10;
    m1.Display();
    MyStruct m2 = new MyStruct(20);
    m2.Display();
    Console.ReadLine();
}
}
}

```

Consuming A Structure —

Even if inheritance is not supported we can still consume a structure from another structure or a class by creating its object. to test this add new class teststruct.cs and write the following :-

```

Class TestStruct
{
    static void Main()
    {
        MyStruct obj = new MyStruct(30);
        obj.Display();
        Console.ReadLine();
    }
}
}

```

Project or Application Management —

While developing an application sometimes code will be written under more than one project also where collection of project is known as solution whenever we open a new project by default vs will create a solution and

- the projects gets added where a solution is a collection of projects & Project is collection of items.

115

Solution

- Collection of projects

Project

- Collection of items/files

- A solution also requires a name which can be specify by us while opening the project or else will take the name of first project that is created under the solution, if not specified.
- In our case solution name is OOPS project because our project name is OOPS project.
- A solution can have projects of different .net languages as well as different project template like windows, Console etc, but a project can't contain items of different .net languages, they must be specific to one language only.
- To add view Project under our OOPS project solution, right click on solution node in solution explorer and select add "New Project" which opens the new project window, under it select language as Visual C#, template as Console Application, name the project as "Second Project" & Click 'OK' which add the new project under OOPS Project Solution.
- By default new project also comes with a class program but under second project namespace, now write the following code in its main method & execute :-

```
Console.WriteLine("Second Project");  
Console.ReadLine();
```
- To run the above class, first we need to set a proper startup property because there are multiple projects

under the solution and VS by default run first project in the solution only i.e. OOPS Project. 116

- To set the startup project property open solution explorer, right click on second project. Select set as Startup project & then run the project.

Note -

If the project is added with new classes we need to again set startup Object property under second projects property window.

Saving Solution and Projects -

Application what we have created right now is saved physically on harddisk in the same hierarchy what we see under solution explorer i.e, first a folder is created representing the solution and under that separate folder is created representing each project & under that items corresponding to that project get saved the following:-

C:\Csharp8\OOPS Project\OOPS Project

C:\Csharp8\OOPS Project\Second Project

Note:-

A solution file gets saved with .sln ~~ext~~ extension and C# project file gets saved with .csproj extension which can contain items specific to C# language only.

Compilation of Projects:-

- Whenever a project is compiled it generated an output file known as assembly that contains IL code of all ~~of~~ the items present in the project.
- The name of assembly file will be same as the project name but will have an extension of either .exe or dll.

117
- Assembly files are what we carry on the client systems when the application has to be installed or deployed there, so they are also referred as unit of deployment.

- The assembly of a project will be present under bin/ debug folder of that project's folder.

C:\Csharp8\OOPs Project\OOPs Project\bin\Debug

C:\Csharp8\OOPs Project\SecondProject\bin\Debug

Note -

All the BCL were installed on our machine in the form of assemblies only and we can find them under following folder

<drive>:\Windows\assembly

Q. Can we consume classes of a project from other classes of same Project?

⇒ Ans- Yes. we can consume them directly because all those classes were under same project and will be considered as a family.

Q. Can we consume the classes of a project from other project?

Ans- Yes. We can consume but not directly as they are under different project. To consume them first we need add reference of the assembly in which the class is present to the project who wants to consume it.

Q How to add reference of an assembly to a project?

Ans- To add reference of an assembly to a project open solution explorer, right click on the project to whom reference has to be added, select add reference option, click browse tab on the add reference window, Select the assembly we want to consume from its physical location & Click ok.

Now we can consume type of that assembly by referring of their namespace or importing the namespaces. 118

- To test this go to our OOPS project solution, right Click on the second project we have newly added, select add reference, select browse tab & Choose the assembly of OOPS project from its physical location.

C:\Csharp7\OOPS Project\OOPS project\bin\debug

Now add a new class under second project as Class1.cs and write the following code in it:-

```
Using OOPS Project ;
Class Class 1
{
    static void Main()
    {
        Rectangle r = new Rectangle (12.34, 56.78);
        Console.WriteLine (r.getArea());
        Console.WriteLine (r.getPerimeter());

        Circle c = new Circle (34.56);
        Console.WriteLine (c.GetArea());
        Console.WriteLine (c.GetPerimeter());
        Console.ReadLine();
    }
}
```

Project (Source Code) → Assembly (Compiled Code)

Note - A project can contain multiple namespaces on it.

ASSEMBLIES AND NAMESPACES-

An assembly is an output file which gets generated after compilation of a project and it is physical. The name of an assembly will be same as project name and cannot be changed at all.

Project : Source Code

Assembly : Compiled Code (IL Code)

- A namespace is a logical container of types which are used for grouping types, By default every project has a namespace and its name is same as project name but we can change namespace names as per our requirement and moreover a project can contain multiple namespace in it.

Test Project (Project) → Compiled → Assembly namespace NSP1

```
namespace NSP1
```

```
{
```

```
Class 1
```

```
Class 2
```

```
}
```

```
namespace NSP2
```

```
{
```

```
Class 3
```

```
Class 4
```

```
}
```

- Whenever we want to consume a type which is defined in a project from other projects, first we need to add reference of assembly corresponding to that project and then only we can consume the types of that assembly in new project.
- While consuming type in the assembly for which reference is added we need to prefix type names with their namespaces ex. import the namespaces and consume the types.

ACCESS SPECIFIERS —

These are also modifiers used to define scope of type as well as their members i.e., who can access them & ~~at~~ who cannot. C# supports 5 access specifiers. Those are:—

- ① Private
- ② Internal
- ③ Protected
- ④ Protected Internal
- ⑤ Public

120

Note -

Members that are defined in a type with any scope or specifier are always accessible within the type, Restrictions comes into picture only when we try to access them outside of the type.

Private -

Members declared as private under a class or structure cannot be accessed outside of the type in which they are defined & moreover their default scope is private only. Types can't be declare as private. So private can be used only on members.

Note -

Interface can't contain any private members and default scope for interface member is public.

Protected -

Members declared as protected under a class can be accessed only within the class or in a child class, non child classes can't consume them. Types can't be declared as protected also, so this can also be used only on members.

Internal -

Members and types that are declared as internal can be consumed only within the project, both from a child or a non child. The default scope for any type in C# is internal only.

Protected Internal -

Member declared as protected internal will have dual scope i.e, within the project they behave as internal providing access to anywhere in the project, outside the project they will change to protected and still provides access to their child classes. Types cannot be declared as protected internal also. So this can also used only on members.

Public -

A type or member of a type if declared as public is global in scope which can be accessed from anywhere.

Case	Private	Internal	Protected	Protected internal	Public
Class 1	✓	✓	✓	✓	✓
Class 2	x	✓	✓	✓	✓
Class 3	x	✓	x	✓	✓
Class 4	x	x	✓	✓	✓
Class 5	x	x	x	x	✓

Case 1: Consuming or accessing members of a class from same class

Case 2: Consuming or accessing members of a class from child of same project.

Case 3: Consuming or accessing members of a class from non child class of same project.

Case 4: Consuming or accessing member of a class from child class of another project.

Case 5: Consuming or accessing members of a class from non child class of another project.

Q How to restrict a class not be accessible for any other class to consume?

Ans- This can be done by declaring all the class constructors as private.

Q How to restrict a class not to be inherited for any other class?

Ans- This can be done by declaring classes as sealed.

Q How to restrict a class not to be accessible for any other class to consume by creating its object?

Ans- This can be done by declaring all the class constructors as protected.

* Open Visual studio → goto file menu → select New Project
 → Select language as C sharp → Choose Console Application
 Project Template, name the project as Access Demo 1
 & rename the solution as My Solution.

- By default the project comes with a class program, write
 following code in it by making the class as public.

// Case 1:

```
public class Program
{
    private void test1()
    {
        Console.WriteLine("Private Method");
    }
    internal void Test 2()
    {
        Console.WriteLine("Internal Method");
    }
    Protected void Test3()
    {
        Console.WriteLine("Protect Method");
    }
    Protected internal void Test 4()
    {
        Console.WriteLine("Protected internal method");
    }
    Public void Test 5()
    {
        Console.WriteLine("Public Method");
    }
    static void Main (string [] args)
    {
        Program p = new Program ();
        p.Test 1(); p.Test 2(); p.Test 3(); p.Test 4();
        p.Test 5();
        Console.ReadLine(); } >
```


Add a new class Two.cs in the project & write following:-
123

// Case 2 :-

```
Class Two : Program
{
    Two obj = new Two();
    obj.Test 2();
    obj.Test 3();
    obj.Test 4();
    obj.Test 5();
    Console.ReadLine();
}
}
```

Add a new class Three.cs in the project & write following

// Case 3

```
Class Three
{
    static void Main()
    {
        Program p = new Program();
        p.Test 2(); p.Test 4(); p.Test 5();
        Console.ReadLine();
    }
}
```

Add a new project under the solution of type Console Application, name it as Access Demo 2, rename default file program.cs as Four.cs. So that Class name also changes as four, now add the reference of Access Demo 1 assembly from its physical location to Access Demo 2 project & then write the following Code:-

// Case 4

```
class Four : Access Demo 1. program
{
    static void Main()
    {
```

```

Four obj = new Four();
obj.Test3(); obj.Test4(); obj.Test5();
Console.ReadLine();
}
}

```

Add a new class under AccessDemo2 project, naming it as Five.cs and write the following :-

```

// Case 5
Using Access Demo 1 ;
class Five
{
    static void Main()
    {
        Program p = new Program();
        p.Test();
        Console.ReadLine();
    }
}

```

LANGUAGE INTEROPERABILITY —

As discussed earlier, the code written in one .net language can be consumed from any other .net language. To test this add a new project under my solution choosing the language as VB, template as class library & name the project as VB Project 1.

Note—

A class library is collection of types that can be consumed but not executed. After compilation the extension of the assembly will be .dll here.

- VB is not a case sensitive language.
- VB does not have any curly braces, the end of a block is represented with a matching End stmt.
- VB does not have any semicolon terminator each statement must be in a new line.

- In VB language methods are divided into 2 types -
 - Functions (Value Returning method)
 - SubFunctions (Non value returning method)
- By default the project comes with a class Class 1 within the file Class1.vb, write the following code in it :-

```

public class Class 1
    Public Function SayHello (name As String) As String
        return "Hello" & name
    End Function

    Public Sub AddNums (x As Integer, y As Integer)
        Console.WriteLine (x+y)
    End Sub

    Public Sub Mathz (a As Integer, b As Integer,
        ByRef c As Integer, ByRef d As Integer)

        c = a + b
        d = a * b
    End Sub
End Class

```

- Now to compile the project, open Solution Explorer; right click on the VB Project 1 & Select Build which compiles and generates an assembly VBProject.dll
- Now add a new class under Access Demo 1 project as TestVB.cs & add the reference of assembly VBProject.dll from its physical location, then write the following code under TestVB class :-

```

using VBProject 1
class TestVB
{
    static void main()
    {
        Class 1 obj = new Class 1()
        obj.AddNums (100, 50);
    }
}

```

```

    int x=0, y=0 ;
    obj.Math(100, 25, ref x, ref y);
    Console.WriteLine(x + " " + y);
    String str = obj.SayHello("VJ");
    Console.WriteLine(str);
    Console.ReadLine();
}
}

```

- Now to test consuming CSharp code in VB.net, add another project under MySolution of language VB.net, type as Console Application and name it as VBProject2.
- By default the project comes with the file Module1.vb, open Solution explorer, delete that file under the project and a new class naming it as TestCS.vb.
- Now add the reference of AccessDemo1 assembly to the current project choosing it from its physical location and write the following code under TestCS.class:-

```

Imports Access Demo 1
Public Class Test CS : Inherits Program
    Shared Sub Main()
        Dim obj As New TestCS()
        obj.Test 3()
        obj.Test 4()
        obj.Test 5()
        Console.ReadLine()
    EndSub
End Class

```

Class Members :-

As we are aware that a class is a collection of member and members of a class be any of the following :-

- Fields (Variables)
- Methods
- Constructors
- Destructors
- Properties
- Indexers
- Events
- Delegates (Types)
- Enumeration (Types)

Destructors -

Just like a constructor, a destructor is also a special method that is present under a class, which get called with the object of class is destroyed. It work opposite to constructor, because constructor is called when object is created.

The name of constructor and destructor exactly the same i.e. name of class in which they defined. To differentiate between both, we use build operator.

Class Test

{

Test()

{

// Constructor (Called when object is created)

}

~ Test()

{

// Destructor (Called when object is destroyed)

}

Note -

A destructor method cannot have any parameter and also we cannot apply any modifiers on them.

Destroying the object of a class is done by Garbage collector, where Garbage Collector destroys or deallocate the object in any of the following cases -

- ① In the end of program execution, each & every object that is associated with the program is automatically destroyed.
- ② Sometimes implicit occurring of Garbage collector takes place when the memory is running low, to identify any unused objects that are associated with the program and destroy them.
- ③ We can also call the garbage collector explicitly into a program to destroy unused objects that are associated with the program & to call garbage collector explicitly in our we are provided with a static method "Collect" under the class GC base class library:
`GC.Collect()`

Note -

Even if there is a chance of calling the Garbage Collector explicitly into the program by using `GC.Collect`. It is not advised to call it, because as we have already discussed in Concerns and Criticism related to .Net. that whenever the Garbage Collector comes into the picture for executing the memory of unused object suspend the execution of the program, so it is advised that never to call the garbage collector explicitly in the program.

Add a class TestDemo.cs and write following code:-

```

Class DestDemo
{
    public Dest Demo()
    {
        Console.WriteLine("Object is created.");
    }
}

```

```

}
~ Dest Demo ( )
{
    Console.WriteLine ("Object is destroyed" );
}
static void Main ( )
{
    Dest Demo D1 = new Dest Demo
    Dest Demo D2 = new Dest Demo
    Dest Demo D3 = new Dest Demo
    // d1 = null; d3 = null; GC.Collect ();
    Console.ReadLine ();
}
}
}

```

Execute the above program by using 'Ctrl + F5' and watch the output, where you will be noticing the constructor getting called for 3 times as we will create 3 objects & then execution stops at the ReadLine() statement. Once we use the enter key, the execution of ReadLine and the end of the program also. So immediately, we will be noticing the destructor getting called for 3 times, which proves in the end of the program's execution, each & every object associated with the program will be destroyed.

Now uncomment the commented code in Main() method & again execute the program by using 'Ctrl + F5', where we will notice that object getting created for 3 times and 2 objects being destroyed before ReadLine(), because we explicitly call the Garbage collector.

And the 3rd object will be destroyed after ReadLine(), because only two objects are marked as Unused.

Inheritance and Destructor -

130

As we are aware that whenever a child class object is created it will first go and call the parent classes constructor. In the same way whenever an object of the class is destroyed, then also its parent classes destructor get called but constructor calling will be top to bottom hierarchy, whereas destructor calling will be bottom to top hierarchy.

To test this add a class Test Demo2.cs & Write the following:-

```
Class DestDemo2 : DestDemo
{
    public DestDemo2()
    {
        Console.WriteLine("Object 2 is created");
    }
    ~DestDemo2()
    {
        Console.WriteLine("Object 2 is destroyed");
    }
    static void Main()
    {
        DestDemo2 obj = new DestDemo2();
        Console.ReadLine();
    }
}
```

Need of Destructor -

- In unmanaged languages like C++, destructor plays an very important ^{role} because they are responsible for destroying or deallocating the memory, whereas in managed languages like Java & .Net, deallocation process is taken care by Garbage Collector. So destructor is not useful in managed languages.

- In some cases we need destructor in managed languages also provided if we dealing with any unmanaged resource like files, databases etc. in our program.

PROPERTIES -

These are members of a class which are used for providing access to the values of a class outside of a class.

- If a class is associated with values in it and if we want those values to be accessible outside of the class, we can provide access to those values in two different ways:-

① By storing the value under the public variable we can view access the value outside of class, but it is a Read, Write access. i.e, any one can get value as well as any one can set the value also.

Ex-

```
public class Test
{
    public int x = 100;
}
```

```
Test obj = new Test();
```

```
int value = obj.x; // Getting the old value
```

```
obj.x = 200; // Setting the new value
```

② By store the value under a private variable also we can provide access to that value outside of that class by defining a property on that variable. The advantage in this mechanism is we can give access to the value in 3 different ways:-

(a) Only get access (Read Only property)

(b) Only Set access (Write Only property)

(c) Both get and set access (Read/Write Property)

Syntax to define a property -

```
[<modifier>] <type> <name>
{
    [get {<stmts>}] // Get Accessor
    [set {<stmts>}] // Set Accessor
}
```

The code we define under the get block or Get Accessor of the property gets executed when we try to capture the value through the property (132)

Ex- String str = Console.Title ;

- The code we write under the set block or Set Accessor of the property gets executed when we try to assign a value to the property.

Ex- Console.Title = "My Console window" ;

If a property is defined with both Set and Get blocks, It is a Read/Write Property, whereas if the property is defined ~~with~~ only with a Get block. It is a Read Only property and if a property is defined only with a Set block it is a write only property.

Add class customer.cs & write the following code -

```
public class Customer
{
    int Custid ;
    String - Cname, - state ;
    bool - Status ;
    double - Balance ;
    Cities - City ;
    public Customer (int Cust-id)
    {
        this._Custid = Custid ;
        this._Cname = "VJ" ;
        this._Status = false ;
        this._Balance = 5000 ;
        this._city = 0 ;
        this._state = "Karnataka" ;
        Country = "India" ;
    }
    // Read Only property
    public int custid
    {
```

```
get { return - Custid; }
}
```

```
// Read/Write Property
```

```
public string Cname
```

```
{
    get { return - Cname; }
    set { - Cname = value; }
}
```

```
// Read/Write Property without any conditions
```

```
public bool status
```

```
{
    get { return - status; }
    set { - status = value; }
}
```

```
// Read/Write Property with conditions
```

```
public double Balance
```

```
{
    get
    {
        if (status == true)
            return - Balance;
            return 0;
    }
}
```

```
set
```

```
{
    if (value >= 500)
        Balance = value;
}
```

```
public Cities City // Enumerated Property
```

```
{
    get { return - city; }
    set { - city = value; }
}
```

```
// Setting the scope of each property
    accessor independently (2,0)
```

```

public string State
{
    get { return -State; }
    protected set { -state = Value; }
}

// Automatic property 3.0
public string Country
{
    get;
    private set;
}
}

```

- Value is a implicit local variable available under a property that provides access to the value that is assigned to a property while consuming the property. The data type of the value variable will be same as the datatype of the property in which we are consuming it.

Enumerated Property —

These are properties that are defined with a list of constant to choose as a value to the property. i.e. we want to assign a value to the property. It will provide with a list of ~~color~~ values to choose from.

Ex-

The background and foreground color properties of the Console class, that provide a list of color to choose from within an enum Console color.

Enum —

It is also a user defined data type like a class or structure but used for defining the constants.

Syntax —

```
[<modifier>] enum <Name> { <List of Constants> }
```

Add a code file in the project naming it as cities.cs and define the following enum in it before defining the Customer class -

135

```
using System;
namespace OOPS project
{
    public enum Cities
    {
        Banglore, Chennai, Delhi,
        Hyderabad, Kolkata, Mumbai
    };
}
```

To define a property as enumerated, we need to follow the below process -

Step 1 :-

Define an enum with list of constants in it.

Ex - Public enum Days { Monday, Tuesday, Wednesday, Thursday, Friday };

Step 2 :-

Declare a variable of new enum datatype days.

Ex - Days _Day = 0
or,

Days _Day = Day.Mondays;

Note :-

Index assessment is possible only for the first value for the enum.

Step 3 :-

Now define a property for providing access to the enum variable.

```
public Days Day
{
    get { return _Day; }
    set { _Day = value; }
```

New Features Related to Properties —

136

- In C# 2.0, we have given with a option of setting the scope of each property accessive independently.

Ex- State Property in our customer class.

- In C# 3.0, we are provided with a feature "Automatic Property" which allows to define a property without any variable for storing the value. In this a variable and stores the values also. So while defining the property as automatic, the get & set block should not contain any body.

Ex- Country Property in our Customer class.

Note -

While defining a property as automatic it is must to define both get & set block under the property.

- To consume the above properties we have define under customer class, Add a new class Test Customer.cs & write the following:-

```
class TestCustomer
{
    static void Main()
    {
        Customer c = new Customer(101);
        Console.WriteLine("Custid:" + c.Custid);
        // Assigning can't be performed b'coz property is read only.
        // c.Custid = 102; // Invalid
        Console.WriteLine("Old Name:" + c.Cname);
        c.Cname = "VJ";
        Console.WriteLine("New Name:" + c.Cname);
        Console.WriteLine("Current status:" + c.status);
    }
}
```


Console.WriteLine("Current Balance:" + c.Balance); 137

c.Balance = 100; //Transaction fails as bal < 500

Console.WriteLine("Balance => transaction failed:" + c.Balance);

c.Balance = 600; //Transaction succeed

Console.WriteLine("Balance => transaction Succeed:" + c.Balance)

Console.WriteLine("Current City:" + c.City);

c.City = Cities.Hyderabad;

Console.WriteLine("Modified City:" + c.City);

Console.WriteLine("Current City:" + c.State);

// Assigned fails as the current class is not a child class
of customer

// c.State = "AP"; //Invalid

Console.WriteLine("Country:" + c.Country);

Console.ReadLine();

}

}

OBJECT INITIALIZERS —

- This is a new feature that has been added in C# 3.0 that allows to initialize the attributes (properties) of a class while creating the object with the help of a parameterless constructor.
- Till now we have been learning that only a parameterized constructor can initialize the attributes of a class with user defined values & parameterless constructor will initialize with the default value; but the introduction of object initializers parameterless constructor also can initialize the attributes of a class with user defined values.
- To test this go to our customer class, we defined above & add a new constructor in the class (parameterless) as following:-

```
public Customer()
```

138

```
{
```

```
}
```

- Now with the help of the parameterless constructor and object initializer we can initialize the attributes of the class in any mismatch combination you want.

To test this add a new class in the project & naming it as object_initializer.cs & write the following code:-

```
class ObjectInitializer
```

```
{
```

```
static void Main()
```

```
{
```

```
Customer C1 = new Customer
```

```
{
```

```
Cname = "VJ", Balance = 10000
```

```
City = Cities.Hydrabad, Status = true
```

```
};
```

```
Customer C2 = new Customer
```

```
{
```

```
Cname = "Smith", Balance = 6000,
```

```
Status = true
```

```
};
```

```
Customer C3 = new Customer
```

```
{
```

```
Cname = "Yo Yo", Balance = 12000
```

```
};
```

```
Customer C4 = new Customer
```

```
{
```

```
Balance = 15000, City = Cities.Kolkata
```

```
};
```

```
}
```

```
}
```

INDEXERS -

- These are similar to properties using which we can provide access to the value of a class outside of the class.
- In the case of indexers, we provide access to the value of the class with the help of class object by specifying the index position of that value whereas in the case of a property, we provide access to the value of the class with the help of the property name which is defined explicitly.
- We define indexers very similar like properties only but property will be having a specific name whereas indexers will not ~~only~~ have any name. In the place of name we use "this" keyword. To test that we are defining the indexes directly on the class itself. so that object of that class starts providing the access to the values that are provided inside of the class.

Syntax:-

```
[<modifiers>] <type> this [int index or String name]
{
    [get {<stmts>}] // Get Accessor
    [set {<stmts>}] // Set Accessor
}
```

Note:-

With the help of indexers we can provide access to the value of the class, either by using the index position or by using the attribute name also.

Add a class employee.cs & Write the following code:-

```
Public class employee
{
    int _empno;
    String -ename, -job;
    double - salary;
    Public Employee (int empno)
    {
        this._empno = empno;
```

```

this._Ename = "VJ";
this._Job = "manager";
this._Salary = "5000";
}
public object this [int index]
{
    get
    {
        if (index == 0)
            return _empno;
        else if (index == 1)
            return _ename;
        else if (index == 2)
            return _job;
        else if (index == 3)
            return _salary;
        return null;
    }
    set
    {
        if (index == 1)
            _Ename = value.ToString();
        else if (index == 2)
            _Job = value.ToString();
        else if (index == 3)
            _Salary = Convert.ToDouble(value);
    }
}

```

```

public object this [String Name]
{
    get
    {
        if (name == "empno")
            return _empno;
        else if (name == "ename")
            return _ename;
        else if (name == "Job")
            return _job;
        else if (name == "Salary")
            return _salary;
        return null;
    }
    set
    {
        if (name == "ename")
            _Ename = value.ToString();
        else if (name == "Job")
            _Job = value.ToString();
        else if (name == "Salary")
            _Salary = Convert.ToDouble(
                value);
    }
}

```

Add a class Test Employee.cs & Write the following code:-

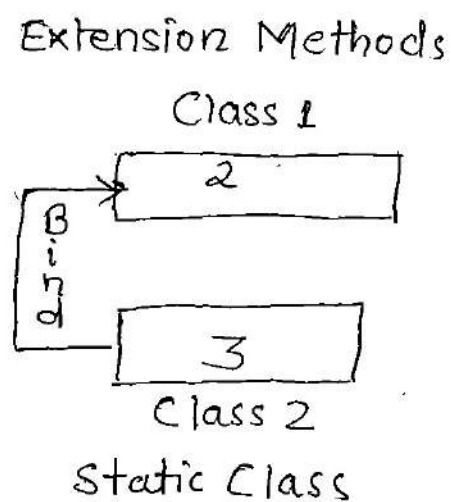
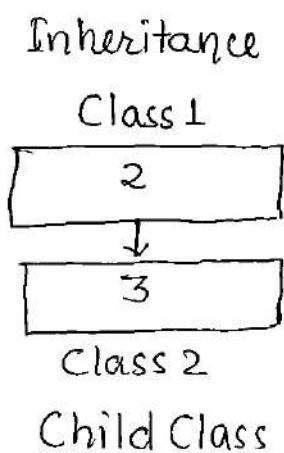
141

```
class Test Employee
{
    static void Main()
    {
        Employee Emp = new Employee(1001);
        Console.WriteLine(Emp[0]);
        Console.WriteLine(Emp[1]);
        Console.WriteLine(Emp[2]);
        Console.WriteLine(Emp[3]);
        Emp[1] = "VJ";
        Emp["Job"] = "Sr. Manager";
        Emp[3] = 6500;
        Console.WriteLine();
        Console.WriteLine(Emp["Empno"]);
        Console.WriteLine(Emp["Ename"]);
        Console.WriteLine(Emp["Job"]);
        Console.WriteLine(Emp["Salary"]);
        Console.ReadLine();
    }
}
```

EXTENSION METHODS-

-It is a new feature that has been added in C# 3.0 which allows us to add new methods into a class without editing the source code of the class i.e., if a class consists a set of members in it & in the future if we wanted to add new methods into the class, we can add those methods without making any changes to the source code of the class.

- Extension method can be used as an approach of extending the functionalities of a class in the future even if ~~into~~ the source code of the class is not available or we don't have any permission in making changes to the class.
- Before Extension methods, inheritance is an approach that is used for the extending the functionalities of a class i.e, if we want to add any new member into an existing class with making modification to the class. We will define a child class to that existing class & then we add the new members in the child class.
- In case of extension method also, we will extend the functionality of a class by defining the methods, we want to add into the class in a new class and then bind them to an existing class.



- Both the above two approaches can be used for extending the functionalities of an existing class whereas in the inheritance, we call the method defined in the old and new classes by using object of new Class i.e, class, whereas in case of extension methods, we can call the old and new method by using object of old class i.e, Class 1.

Defining Extension Methods -

If we want to define extension methods under a class, we need to follow below rules & regulations -

- ① Extension methods must be defined only under static class.
- ② As an extension method is defined under a static class, compulsory the method should be defined as static only whereas once the method is bound with another class. The method changes into non static.
- ③ The first parameter of an extension method is known as binding parameter which should be name of class ~~to which~~ ~~should be name of class~~ to which the method has to be bound and the binding parameter should be prefixed with this keyword.
- ④ An extension method can have only one binding parameter and that should be defined in the first place of the parameter list.
- ⑤ If required, an extension method can be defined with normal parameter also starting from the second place of the parameter list.

Note -

If an extension method is defined with n parameter, while calling method there will be $n-1$ parameter only excluding the binding parameter.

Add a Class OldClass.cs & Write the following :-

```
public class OldClass
{
    public int x=100;
    public void Test1()
    {
        Console.WriteLine("Method One: " + this.x);
    }
    public void Test2()
    {
```



```
Console.WriteLine("Method Two:" + this.x);  
}  
}
```

144

Add a class NewClass.cs & write the following -

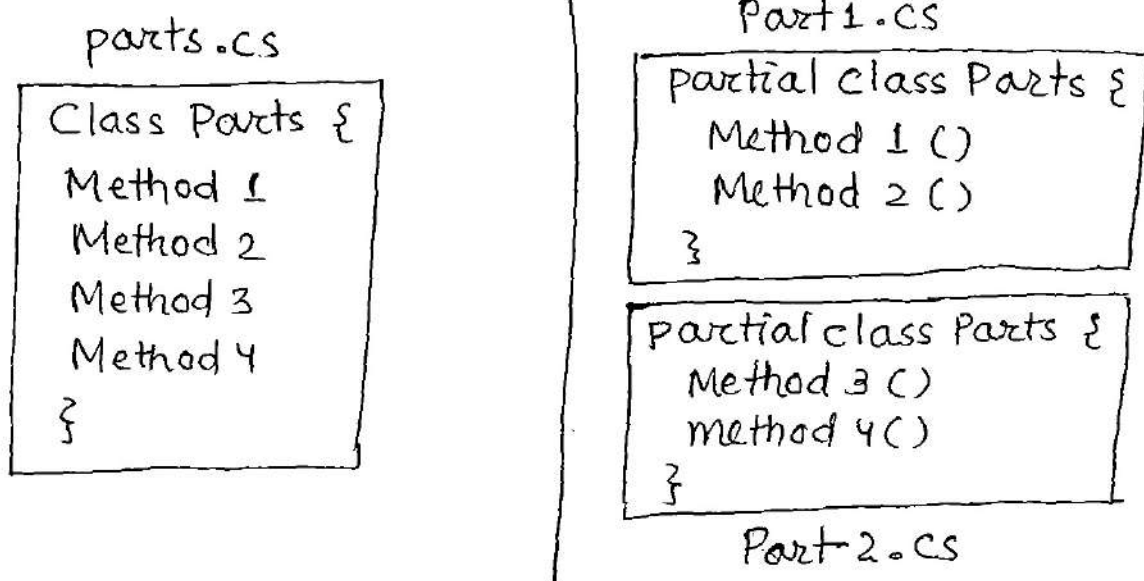
```
public static class NewClass  
{  
    public static void Test3(this OldClass o)  
    {  
        Console.WriteLine("Method Three");  
    }  
    public static void Test4(this OldClass o, int x)  
    {  
        Console.WriteLine("Method Four:" + x);  
    }  
    public static void Test5(this OldClass o)  
    {  
        Console.WriteLine("Method Five:" + o.x);  
    }  
}
```

Add a Class TestOld.cs & Write the following -

```
Class TestOld  
{  
    static void Main()  
    {  
        OldClass obj = new OldClass();  
        obj.Test1(); obj.Test2();  
        // Calling extension methods  
        obj.Test3(); obj.Test4(); obj.Test5();  
        Console.ReadLine();  
    }  
}
```

Partial Classes -

These are newly ~~are~~ introduced in C# 2.0 which allows to define a class on multiple files i.e., we can split a class physically & define it on more than one file.



- In the above case, the class defined in Left hand side is redefined in Right hand side by splitting into two files.
- To define a class as partial in all the files, the class name must be same and also we need to use partial modifier to specify the class is Partial.
- If we want to inherit a partial class from any other class, we need to specify that information in only one file but not required on all the files.
- Partial classes allows multiple programmer to work in the same # class at same time.
- Partial classes are suggested to use for splitting huge volume of codes on separate file, so organising becomes easier.
- Partial classes is only an approach of physically division of code but logically the code in all the files is treated as single class only.

Note -

We use partial classes in Windows application development.

Exception and Exception Handling :-

- While developing an application there are chances of coming across 2 different type of errors:
 1. Compile Time errors
 2. Run Time errors
- An error that occurs in a program at the time of compilation of a program is known as Compile time error. These error occurs due to syntax mistakes under the program.
- An error that occurs in the middle of program's execution is known as Runtime error. These error occurs due to various reasons:-
 1. Wrong implementation of logic.
 2. Missing of required resource.
 3. Wrong input supplied to a program.
- Compile time errors are not dangerous whereas run time errors are dangerous, b'coz whenever they occur in the program, the program terminates abnormally on the same line where the error get occurred without executing next lines of code.
- Who is responsible for abnormal termination of program when runtime errors occurs in a program?

Ans - Exception are responsible for abnormal termination of a program, whenever a runtime errors occurs ~~on~~ under program.

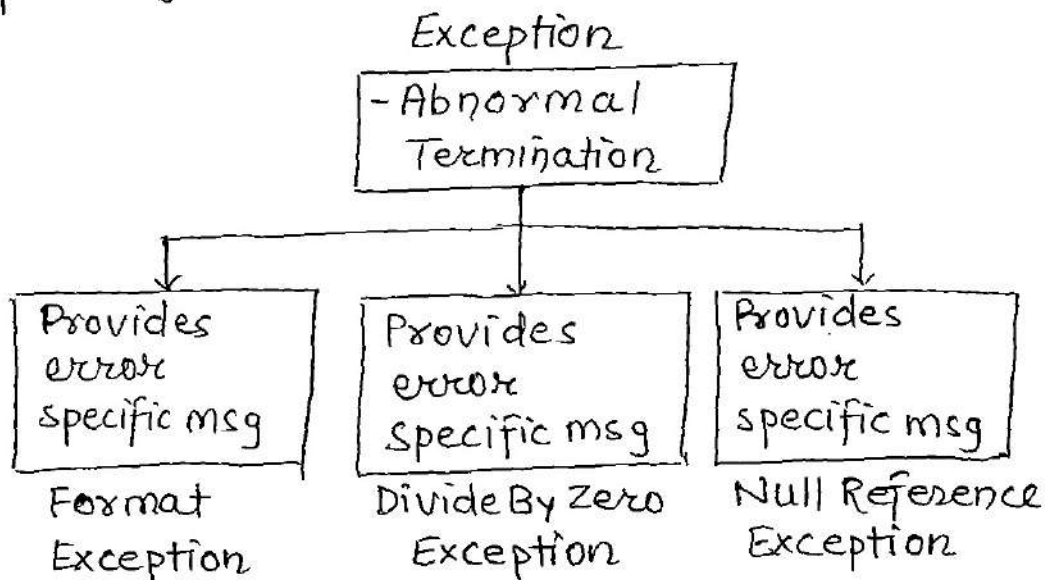
Exception are predefined classes, where we are provided with each & every different type of runtime error that can occur in a program.

Example - Index Out Of Range Exception,

- Format Exception,
- Null Reference Exception
- Divide By Zero Exception

- OverflowException

- Whenever a runtime error occurs in a program, the object of a matching exception class related to the error gets created, which terminates the program abnormally as well as display error message related with error that occurred.
- Under the base class libraries, exception classes are implemented as following:-



- Whenever a runtime error occurs in a program, first the exception manager under CLR identifies the type of error that occurs in the program, creates an object of the exception class related with that error and throws that object which will immediately terminate the program abnormally on the line where error occurred & displays the error message related with that exception class.

Exception Handler -

- It is a process of ~~handling~~ stopping the abnormal termination of a program whenever runtime errors are occurring in a program.
- Once we handle an exception under a program, we will be getting following advantages:-
 1. We can stop the abnormal termination

2. We can perform any corrective action that may resolve problem occurring due to abnormal termination.
3. Display User friendly error message, so that clients can resolve the problem provided if it is under his control.

- ~~●~~ If we want to handle an exception we must enclose the code under two special blocks i.e., Try & Catch blocks, which should be used as following :-

```
try
{
    -stmts which will cause a 'runtime error'
    -stmts which doesn't require execution when the
      runtime error occurs
}
catch ( <exception> <var> )
{
    -stmts which requires execution only when the
      runtime error occurs
}
```

-- <multiple catch blocks if required> --

- Once we enclose the code inside of the try-catch blocks, the execution takes places as following :-
 - If all the statements under the try block are executed successfully from the last statement of try, the control directly jumps to first statement which is present after catch block without executing ~~any~~ any catch block i.e, there is no error in the program.
 - If any statement under try raises an error from that line, control directly jumps to the catch block without executing any other statements inside try block,

checking for a matching catch block to handle the exception.

- If a matching catch block is available to handle the exception, abnormal termination stops then, execute the statement from catch block, from there again control jumps to first statement after all the catch block.
- If a matching catch block is not available, abnormal termination occurs again.

Add a class ExceptionDemo.cs & write following code:-

```
class ExceptionDemo
{
    static void Main ()
    {
        int x, y, z;
        Console.WriteLine("Enter x value :");
        x = int.Parse(Console.ReadLine());
        Console.WriteLine("Enter y value :");
        y = int.Parse(Console.ReadLine());
        z = x / y;
        Console.WriteLine(z);
        Console.WriteLine("End of program");
    }
}
```

- In the above program, if we give the input to the program as a non-numeric value or the divisor value as zero, we will be getting an exception & once we get an exception, immediately abnormal termination takes place.

Add a class tryCatchDemo.cs & rewrite the previous program by handling the exception as following:-

```
class TryCatchDemo
{
    static void Main()
    {
        int x, y, z;
        try
        {
            Console.WriteLine("Enter x value:");
            x = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter y value:");
            y = int.Parse(Console.ReadLine());
            z = x / y;
            Console.WriteLine(z);
        }
        catch (DivideByZeroException ex1)
        {
            Console.WriteLine("Divisor should not be 0");
        }
        catch (FormatException ex2)
        {
            Console.WriteLine("Input must be numeric only");
        }
        catch (Exception ex3)
        {
            Console.WriteLine(ex3.Message); // ex3 predefined property gives error message
        }
        Console.WriteLine("End of Program.");
    }
}
```


- message is a property of exception class which gets the error message associated with the exception that got occurred.
- The property message is defined under the class Exception as virtual giving a chance for all its child classes to override.

```

public class Exception
{
    - Implemented the logic for abnormal termination
    public virtual string Message
    {
        get { return "<some error msg. >"; }
    }
}

```

- As the property is declared as Virtual under the Exception class, all the child classes of exception class have a chance of overriding the property as per their requirements, so the child classes ~~have~~ have overridden the property as following:-

```

public class DivideByZeroException : Exception
{
    public override string Message
    {
        get { return "Attempted to divide by zero"; }
    }
}

```

```

public class FormatException : Exception
{
    public override string message
    {
        get { return "Input string was not in correct format"; }
    }
}

```

- In our above program, when exception are caught by using last catch block, the parent class reference will call the message property of child classes only, because in the overriding, a parent reference will always call child class members only. That is the reason why message property is displaying predefined error message associated with the exception that got occurred.

Note-

If required we can also define our own exception classes same as the previous classes format exception, Divide by zero exception are defined above.

Finally Block -

- This is another block of code which is associated with try and gets executed when an exception raises as well as did not raises also.
- All the statement in the try block will be executed only when there is no exception in the code whereas the statement under catch block will executed only when there is an exception in the code but finally block execute in both two cases :-

```

try
{
    Open the file
    Write on the file
}
catch
{
}
finally
{
    Close the file
}

```

Add a class FinallyDemo.cs & Write a code: ~~Class Finally Demo.cs~~

Class FinallyDemo.cs

```
{
    static void Main ()
    {
        int x,y,z;
        try
        {
            Console.WriteLine("Enter x");
            x = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter y");
            y = int.Parse(Console.ReadLine());
            if(y == 1)
                return;
            z = x/y;
            Console.WriteLine(z);
        }
        catch(exception ex)
        {
            Console.WriteLine(ex.message);
        }
        finally
        {
            Console.WriteLine("finally Block");
        }
        Console.WriteLine("End of program");
    }
}
```

- Execute the above program for two times, by 1st time giving the input which doesnot cause any error and 2nd time give the input which cause the error and watch the output where in both the two cases finally block will be executed.

Note -

In both two cases not only finally block end of the program statement also will be executed.

- Now run the program by third time by giving the value divisor as 1, so that the if condition in the program get satisfied and return statement executes.
- As we are aware that return is a jump statement which will jump out of the method will now jump out of method but only after executing the finally block because once the control enters into try we cannot stop the execution of finally i.e. statement executed at any cost.
(Mandatory execution)

- try
- Catch
- finally

We can use these three blocks in three different combinations:-

① Try & Catch :-

In this case, the abnormal termination stops because exceptions are handled to catch block.

② Try, Catch & finally -

In this case also abnormal termination will not occur because exceptions are handled through catch block as well as set of statements under finally will be executing at any ~~cost~~ cost.

③ Try and finally -

In this case abnormal termination will not stop because there is no catch block to handle the exception but if abnormal termination takes place in the program finally block gets executed.

Types of exceptions -

Exceptions are of two types

① System Exception

② Application Exception

- An exception get raised implicitly by the exception manager is a system exception.

Ex- DivideByZero, FormatException etc.

- An exception get raised explicitly under a program by the program basic his own requirement is known as application exception.

- As a programmer if we want to raise an exception in our program follow the below process:-

① Create the object of an exception class

② throw that object by using the throw statement.

Syntax-

throw <object of exception class>

eg.

```
FormatException ex = new FormatException();
```

```
throw ex;
```

or,

```
throw new FormatException();
```

- When we want to raise exception explicitly, we create the object of an exception class, so in this case which class object has to be created can be decided from any of two options -

① We can create the object of the predefined class 'Application Exception' by passing the error message as a parameter to its constructor so that whenever the exception raises the given error message gets displayed.

ex- Application exception (string errorMsg)

throw new ApplicationException ("<errorMsg>");

② We can also create the object of our own defined exception class and throw it when required, so that the error message, we override under the message property gets displayed when the exception raised on thrown.

- If we want to define an exception class on our own adopt the following process:-

① Define a new class inheriting from predefined class Exception, so that new class also become an exception.

② Override the property message by providing the desired error message.

To test this add new class DivideByOddNoException.cs and write following code:-

```
Public Class DivideByOddNoException : Exception
{
    Public override string message
    {
        get { return "attempted to divide by odd"; }
    }
}
```

Now add new class ThrowDemo.cs and write following code-

```
Class ThrowDemo
{
    static void Main()
    {
        int x, y, z;
        Console.WriteLine("Enter x");
        x = int.Parse(Console.ReadLine());
        Console.WriteLine("Enter y");
```

```

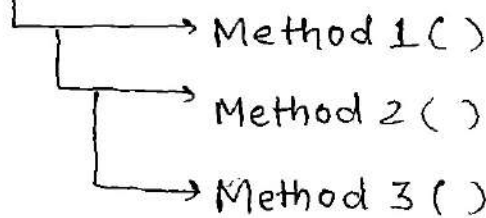
y = int.Parse(Console.ReadLine());
if (y % 2 > 0)
{
    throw new DivideByOddNoException();
}
z = x / y;
Console.WriteLine(z);
Console.WriteLine("End of the program");
}
}

```

Multi-Threading -

- A single program performing multiple actions at a given point of time simultaneously is known as Multithreading.
- Earlier we having a concept known as Multitasking, where in this approach, there will be more than one program executing at a point of time.
- Multitasking is a feature supported by OS, where all OS doesn't support multitasking for example: Windows is a multitasking OS, whereas Dos is a single tasking OS.
- Multithreading is a feature which should be first supported by the language whereas internally it will take the support of OS to execute.
- Thread is a unit of execution where every program will have a thread to execute that program & that thread is Main thread. So every program is by default single thread
- In a single threaded model, the program executes by calling the method or performing the actions one after the other i.e., only after completing the method which is starting the execution, then only it goes to next method for execution. So here if a method is taking more time to execute than expected, the other methods that has to execute next should wait.

Main Method



- To overcome drawback with single threaded model, Multi-threading was introduced, where in this approach we can use a separate thread for each method to execute as following :-

Main Method

T1 \longrightarrow Method 1()

T2 \longrightarrow Method 2()

T3 \longrightarrow Method 3()

- If a program is multithreaded, execution of the program takes place by adopting 2 principles :-

① Time Sharing -

Here OS will allocate some time period for each thread to execute & transfer the thread to other thread for execution when the given time period is completed, so there will be equal preference to be given for each method to execute, bcoz The OS keeps on transferring control between all the thread in execution.

② Maximum Utilization of Resources -

This comes into picture only when the first principle violates i.e. if a method was unable to execute for some reason in its given time period without wasting the time there, the control jumps to the other threads in execution.

T1 5 - 5 5 5 5 \longrightarrow Method 1()

T2 5 5 - - 5 5 \longrightarrow Method 2()

T3 5 5 5 - 5 5 \longrightarrow Method 3()

Note -

The main aim or goal of multithreading is maximum utilization of CPU resources i.e., Utilizing resources to maximum extent without leaving them idle.

How to create a thread ?

If we want to create a thread, we need to create the object of class thread that is present under System.Threading namespace and while creating the object, the method we want to call from the thread should be passed as a parameter to the constructor.

System.Threading.Thread (<method name>)

Thread t1 = new Thread (Method1);

Thread t2 = new Thread (Method2);

Thread t3 = new Thread (Method3);

Members of Thread Class :-

1. Start() :-

This method should be called on the thread object to start the execution of thread.

Note -

After starting a thread completes its work, it will automatically exits from the program.

2. Abort() :-

If this method is called on the thread object will abnormally terminate the execution of thread.

3. Suspend() :-

If this method is called on the thread object, it will temporarily suspend or pause the execution of thread until the method resume is called.

4. Resume():-

This method should be called on a suspended thread object to resume it.

5. Sleep(int milliseconds) -

It is a static method which will suspend the execution of current executing thread until the given time period elapses.

6. Join() -

If this method is called on a thread object, Main thread has to wait without exiting from the program until the thread which called join is exiting from the program.

7. Priority -

It is a property used for setting the priorities between the threads if sharing the CPU resources.

Add a class singleThread.cs and write following code:-

```
using System.Threading;
class SingleThread
{
    public void Main Method 1()
    {
        for (int i = 1; i <= 100; i++)
        {
            Console.WriteLine("Method 1: " + i);
        }
    }
    public void Method 2()
    {
        for (int i = 1; i <= 100; i++)
        {
```

```

        Console.WriteLine("Method 2:" + i);
        // if(i==50)
        // Thread.Sleep(10000);
    }
}

public void Method 3()
{
    for(int i=1; i<=100; i++)
    {
        Console.WriteLine("Method 3:" + i);
    }
}

static void Main()
{
    Single Thread st = new SingleThread();
    st.method1(); st.method2(); st.method3();
}
}

```

Run the above program & watch the output of the program, where in above case we are calling 3 methods in a class in a single threaded approach i.e., main thread only call the 3 methods, so execution takes place by executing one method after the other. ~~The~~ The drawback in this approach is, if a method is taking more time to execute than expected then next method has to wait for the execute. To test this uncomment the commented code & reexecute the program again. In this context method 2 will stop its execution after printing 50 for 10 sec. All the 10 secs. along with Method 2, method 3 also wait b'coz it has to execute after method 2.

- The same program rewritten in multithreaded approach, execution will be different.

To test this MultiThreaded.cs & write the following code:-

```
using System.Threading;
class MultiThread
{
    Thread t1, t2, t3;
    public MultiThread()
    {
        t1 = new Thread(Method 1);
        t2 = new Thread(Method 2);
        t3 = new Thread(Method 3);
        t1.Start(); t2.Start(); t3.Start();
    }
    public void Method 1()
    {
        for( int i=1 ; i<=100; i++)
        {
            Console.WriteLine("Method 1 : " + i);
        }
        Console.WriteLine("Thread 1 is exiting.");
    }
    public void Method 2 ()
    {
        for(int i=1 ; i<=100; i++)
        {
            Console.WriteLine("Method 2: " + i);
            // if (i==50)
            Console // Thread.Sleep(10000);
        }
        Console.WriteLine("Thread 2 is exiting.");
    }
}
```

```

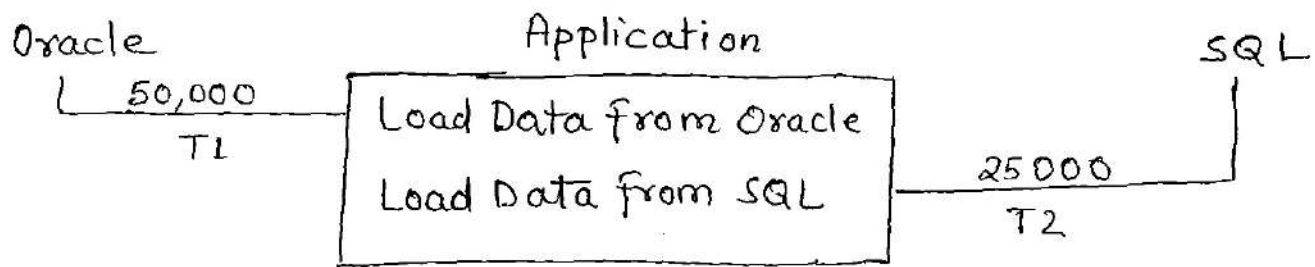
    }
    public void Method 3()
    {
        for (int i = 1; i <= 100; i++)
        {
            Console.WriteLine("Method 3: " + i);
        }
        Console.WriteLine("Thread 3 is exiting");
    }
    static void Main()
    {
        Multithread mt = new Multithread();
        mt.t1.Join(); mt.t2.Join(); mt.t3.Join();
        Console.WriteLine("Thread Main is exiting");
    }
}

```

- ⇒
- In the above case, we are rewriting our previous program by using Multithreading, so in this case, all the 3 methods will be executing at same time by sharing CPU resources (time sharing), so every method is given the same level of preference to execute.
 - In this case if a method could not execute for some reason for point of time, at that time other methods still execute without any problem. To test this uncomment the commented code in Method 2 & reexecute again. Here Method 2 stops execution for 10 sec after printing 50 & ~~at~~ but still at that time Method 1 and Method 3 will be running in the program.

Thread Priorities -

There are multiple threads in execution of a program, by default all the threads will be sharing CPU resources equally i.e., OS gives equal importance to all threads in consuming the CPU resources, But in any case the thread has to do more work than other thread in execution, in such case we request OS to give additional priority for thread doing more work in utilizing the resources with the help of property priority.



- In the above case the thread which has to load from oracle has to do more work that compared with thread loading the data from SQL server. So we give more priority ~~for~~ for thread that Load data from Oracle.

Priority is enumerated property under the thread class which can be send by with 5 different Priority level defined under ~~thread~~ ~~enum~~ thread priority:-

Thread Priority

- Lowest
- Below Normal
- Normal (default)
- Above Normal
- Highest.

By default every thread is created with priority as Normal.

Add a class TPriority.cs & write the following code:-

```
using System.Threading;
class TPriority
{
```



```

long Count 1, Count 2 ;
Thread t1, t2 ;
public TPriority()
{
t1 = new Thread (IncrCount 1);
t2 = new Thread (IncrCount 2);
t1.Start();
t2.Start();
}
public void IncrCount 1 ()
{
while(true)
Count 1 += 1;
}
public void IncrCount 2 ()
{
while(true)
Count 2 += 1 ;
}
static void main()
{
TPriority tp = new TPriority();
Thread.Sleep(1000);
tp.t1.Abort();
tp.t2.Abort();
Console.WriteLine ("Count 1 : " + tp.Count 1);
Console.WriteLine ("Count 2 : " + tp.Count 2);
}
}

```

In the above program, we are declaring two long variable Count 1 & Count 2 and incrementing them under two

separate method in a indefinite loop.

Both the two threads are given a time period of 1 sec to increment the value then we are terminating the threads by calling `Abort()` method on them. So in this case, if we run the program & check the value of `Count 1` & `Count 2`, it is unpredictable to identify whether the value of `Count 1` or `Count 2` will be higher in the context. For test this execute the program ~~execute~~ multiple times and watch the output.

In the execution of the program we will be noticing that sometimes `Count 1` value is higher or sometimes `Count 2` value is higher, becoz both are executing with same level of priority. If we want to make a difference, we can do it by setting the priority between the threads by using the `Priority` property.

To change the priority between the threads in the above program under constructor before calling `start()` method, add the following code:-

```
t2.Priority = ThreadPriority.Highest;
```

Now reexecute the program multiple time, Here most of the time we will be noticing that `Count 2` value is higher than `Count 1`, because it is set with highest priority.

Thread Synchronization or Locking -

In our first two programs of threading, each thread we created in program is calling a separate method for execution. So we will never face any problem in the execution but sometimes multiple threads in a program may be calling same methods for execution in such cases, we will be getting unexpected results.

```

1 Add a class TLock
2 using System.Threading;
3 class TLock
4 {
5     Thread t1, t2, t3;
6     public TLock()
7     {
8         t1 = new Thread(Display);
9         t2 = new Thread(Display);
10        t3 = new Thread(Display);
11        t1.start(); t2.start(); t3.start();
12    }
13    // Asynchronous Method
14    public void Display()
15    {
16        Console.WriteLine(" [CSharp is ]");
17        Thread.Sleep(5000);
18        Console.WriteLine(" Object Oriented ]");
19    }
20    static void Main()
21    {
22        TLock obj = new TLock();
23        obj.t1.join(); obj.t2.join(); obj.t3.join();
24    }
25 }

```

→ In the above program, Display() is a synchronous method i.e., it allows multiple threads to ~~access~~ access same method at same time. So in the program, three methods are calling the same method display, when we execute the program, we will get unexpected result, check it by executing the program.

- To overcome the above problem we need to depend on the concept of thread synchronization or Locking i.e. we need to lock the code that is present under the method by putting it a lock is known as lock Block. So that the method becomes synchronized method, which will not allow multiple threads to access the method at same time. So that only one thread can call the method at a point of time.

- To test this rewrite the code of display method as following:-

// Synchronous Method

```
public void Display()  
{  
    lock(this)  
{  
    Console.WriteLine(" [Csharp is ");  
    Thread.Sleep(5000);  
    Console.WriteLine(" Object Oriented ] ");  
    }  
}
```

In the above case, because the method is synchronised, only one thread will execute the method at a time. So reexecute the program & watch the difference in the output.

Delegates:-

It is also a user defined type.

- Apps should have one entry point. (Program.cs)
- All logic shd. be written under code view. Code view is accessible to programmers only, not for end user.
- Design view can be seen by both programmers & end user.

————— X ————— X —————

Finish.....
Refer Printouts.....

Rich text Box Control -

The Rich text box control provides advanced text entry & advance feature such as character & Paragraph formatting.

Properties -

- ① Text
 - ② SelectedText
 - ③ WordWrap
 - ④ ScrollBar (None, Horizontal, Vertical, both, forcedHorizontal, forcedVertical, forcedBoth)
 - ⑤ Shortcuts enabled (True or False)
 - ⑥ Show Selection Margin (True or False)
 - ⑦ MultiLine (True or False)
 - ⑧ ReadOnly (True or False)
 - ⑨ MaxLength - This specify maximum no. of character that can be entered into the textbox (2147483647)
 - ⑩ AutoWordSelection (True or False)
 - ⑪ BulletIndent - used to define the indent for the bullets in the control.
 - ⑫ EnableAutoDragDrop - Used to get or set a value that indicates whether drag drop of pics, other data is enabled or not (By default set to false).
 - ⑬ ZoomFactor ^(define current scaling factor) - used to get or set a value that indicates the zoom factor of the control.
- These are following properties & methods that can be used to provide character as well as paragraph formatting along with some advanced text entry feature.

Methods - These are the following common methods

- ① AppendText :- used to append text to the current text of a rich textbox.
- ② Copy() :- used to copy the current selection in rich textbox to the Clipboard.
- ③ Cut() :- used to move the current selection in rich text box in Clipboard.
- ④ Paste - It replaces the current selection in the rich text box with the content of the Clipboard.
- ⑤ CanPaste() - indicates whether we can paste info. from the Clipboard for in specified data format or not (Return true or False).
- ⑥ Clear() - Clears all text from RichTextBox.
- ⑦ ClearUndo - Clears info about the most recent operation from UndoBuffer of RichTextBox.
- ⑧ DeselectAll() - used to specify that the value of SelectionLength property is 0, that no characters are selected in the control.
- ⑨ SelectAll() - used to select all the text in the RichTextBox control.
- ⑩ Select - used to activate the control as well as it also allows to select a range of text in the RichTextBox control.
- ⑪ Undo - used to do the undo operation.
- ⑫ Redo.
- ⑬ ResetText() - resets the Text property to its default value.
- ⑭ SaveFile() - used to save the content

- of RichTextBox to in a particular file with specified Type such as PlainText or RichText format (RTF)

Properties

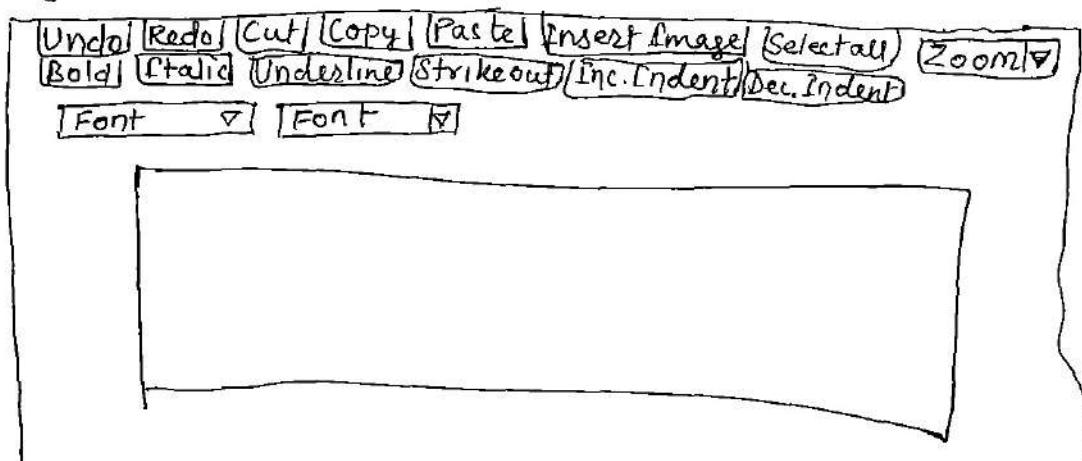
These are the following major properties that are related to selected Text to be applied in the RichTextBox control.

- ① SelectedText :- used to get or set selected text within the RichTextBox control.
- ② SelectedRTF ~~Text~~ :- RichText Format
- ③ SelectionAlignment :- gets or sets the alignment of currently selected Text.
- ④ SelectionBackColor :- get or set background color of the Selected Text in the RichTextBox.
- ⑤ SelectionColor :- get or set Foreground color of selected Text in RichTextBox.
- ⑥ SelectionBullet :- used to apply the bullet style to the currently selected Text.
- ⑦ SelectionFont :- Gets or sets the font of the current selected Text.
- ⑧ SelectionIndent :- used to specify the indent of the currently selected text.
- ⑨ SelectionHangingIndent used to gets or set the distance between the left edge of the first line of the text in the selected paragraph and the left edge of the subsequent lines in the selected paragraph.
- ⑩ SelectionLength - used to get or set the no. of character selected in the control.
- ⑪ Selection Right Indent - used to define the value to specify the right indent of the selected text.

Events -

1. TextChanged - This event occurs when the value of the text property is change on the control.

Design the Windows Form as : —



Code view-

```
private void Form_Load (
{
// Adding all available Fonts:
foreach( FontFamily font in FontFamily.Families )
{
    cboFontName.Items.Add( font.Name );
}
// Adding Font Size :
for( int size = 0; size <= 72; size += 2 )
{
    cboFontSize.Items.Add( size );
}
// Adding Zoom Factor :
for( int zoom = 100; zoom <= 500; zoom += 50 )
{
    cboZoom.Items.Add( zoom + "% " );
}
}
private void btnUndo_Click (
{
    richTextBox.Undo();
}
private void btnRedo_Click (
{
    richTextBox.Redo();
}
private void btnCopy_Click (
{
    richTextBox.Copy();
}
```

```

private void btnPaste_Click
{
    richTextBox1.Paste();
}

private void btnCopySelectAll_Click(
{
    richTextBox1.SelectAll();
    richTextBox1.Focus();
}

private void cboZoom_SelectedIndexChanged(
{
    string strZoomValue = cbo.Zoom.SelectedItem.ToString();
    strZoomValue = strZoomValue.Remove(strZoomValue.Length
                                     - 1, 1);

    richTextBox1.ZoomFactor = float.Parse(strZoomValue) / 100;
}

bool flagBold = true;
private void btnBold_Click(object sender,
{
    if (flagBold == true)
    {
        richTextBox1.SelectionFont = new Font(richTextBox1.
        SelectionFont, richTextBox1.SelectionFont.Style |
        FontStyle.Bold);
        flagBold = false;
    }
    else
    {

```

```

richTextBox1.SelectionFont = new Font ( richTextBox1.
    SelectionFont, richTextBox1.SelectionFont.Style &
    ~ FontStyle.Bold );
    FlagBold = true;
}
}
bool flagItalic = true;
private void btnItalic_Click (
{
    if ( flagItalic == true )
    {
        - same as above only change bold ↔ italic
    }
    else
    {
        - -
    }
}
bool flagUnderline = true;
private void btnUnderline_Click (
{
    if ( flagUnderline == true )
    {
        — only change FontStyle. Underline —
    }
    else
    {
        — —
    }
}
}

```

~~per~~ strikeout same —

```
private void btnIncreaseIndent_Click (
{
```

```
    richTextBox1.SelectionIndent += 5 ;
```

```
}
```

```
private void btnDecreaseIndent_Click (
```

```
{
```

```
    - = 5 ;
```

```
}
```

```
    btnAlignLeft_Click (
```

```
{
```

```
    richTextBox1.SelectionAlignment = HorizontalAlignment.Left;
```

```
}
```

```
    btnAlignRight_Click (
```

```
{
```

```
}
```

. Right,

```
    btnBullet_Click (
```

```
{
```

```
    btn richTextBox1.SelectionBullet = true;
```

```
}
```

```
    bool flag WordWrap = true;
```

```
    private void btnWordWrap_Click (
```

```
{
```

```
        if (flagWordWrap == true)
```

```
{
```

```
            richTextBox1.WordWrap = true;
```

```
            flagWordWrap = false;
```

```

    }
    else
    {
        richTextBox1.WordWrap = False ;
        flag WordWrap = True ;
    }
}

private void btnInsertDateTime_Click (
{
    richTextBox1.SelectedText = DateTime.Now.ToString
("dddd dd MMMM yyyy hh:mm:ss tt");
}

cboFontName_SelectedIndexChanged(
{
    richTextBox1.SelectionFont = new Font(cboFontName
SelectedItem.ToString(), richTextBox1.Selection
Font.Size);
}

cboFontSize_SelectedIndexChanged(
{
    richTextBox1.SelectionFont = new Font( richTextBox1
SelectionFont.FontFamily, float.Parse(cboFontSize.
SelectedItem.ToString()) );
}
}

```

Treeview -

- The ~~Preview~~^{Treeview} Control is used to display hierarchical collection of labeled items with an optional image.

Properties of Treeview -

1. Nodes :- represents a collection of nodes (it is called `TreeNodeCollection`). In the `TreeNodeCollection`, each node represents as `TreeNode` object. This property allows us to add or remove `TreeNode`. Each `TreeNode` has its own properties. These are the following major properties :-

- Name
 - Text
 - ToolTipText
 - Checked (True or False)
 - NodeFont
 - BackColor
 - ForeColor
 - ImageIndex
 - ImageKey
 - SelectedImageIndex
 - SelectedImageKey
- Use one of this
- Use one of this

Component Controls -

① Timer -

The Timer Control is used to raise an event at user defined interval in order to execute any certain code.

Properties -

1. Interval: Used to specify the interval time in milliseconds
(1000 millisec = 1sec)
2. Enabled: Used to enabling or disabling Timer control object
(By default it is set to false).

Events -

1. Tick: This event fires whenever specified interval time elapses.

② Image List Control -

The ImageList control is used to manage a collection of images that are typically used by other control such as ListView, TreeView or ToolStrip.

Properties -

1. Images: represents a collection of images that are stored in Image List control.
2. ImageSize: Define size of images i.e., width & Height in the ImageList Control.

③ Error Provider -

The Error Provider provides a UI to indicate to the User that a control on a Windows Form has an error associated with it.

Properties -

1. Icon: Used to specify the icon to indicate an error.
2. BlinkRate: - Used to define the rate in millisec at which error icon blinks.
3. BlinkStyle: - indicates whether the error icon blinks when an error is set. It contains following properties i.e., BlinkIfDifferentError, AlwaysBlink, NeverBlink.

Ex- Design UI as following :-

Enter Name

Enter Address

ⓧ ErrorProvider

```
private void btnsubmit_Click (Object sender, Event args e)
{
    bool flag = true ;
    if ( txtName .text.Trim() == String.Empty )
    {
        errorProvider1.SetError( txtName, " Please Enter Name" );
        flag = false ;
    }
    else
    {
        errorProvider1.SetError( txtName, "" );
    }
    if ( txtAddress.Text.Trim() == String.Empty )
    {
        errorProvider1.SetError( txtAddress, "Please enter Address" );
        flag = false ;
    }
    else
    {
        error provider 1.SetError( txtAddress, "" );
    }
    if (flag != false)
    {
        MessageBox.Show("Your Name : " + txtName.Text.Trim() +
            "\n Your Address : " + txtAddress.Text.Trim() );
    }
}
```

④ HelpProvider -

This control is used to provides a popup window with Help description for a specified control.

Properties -

1. HelpNamespace -

The HelpNamespace (for eg. HTML ^{files} that will be used). It is only used for controls that have the help keyword set.

- The HelpProvider Control object uses a method i.e. SetHelpString to specify the help string associated with the specified control.

Example -

The diagram shows a rectangular window titled "F1 HelpProvider". Inside the window, there are three elements: a text input field with the label "Enter Name" to its left, another text input field with the label "Enter Address" to its left and a mouse cursor icon pointing at it, and a button labeled "Submit" below the "Enter Address" field.

```
private void Form1_Load (Object sender, EventArgs e)
```

```
{  
    helpProvider1.SetHelpString(txtName, "Please Enter Name!!");  
    helpProvider1.SetHelpString(txtAddress, "Please enter address!!");  
    helpProvider1.SetHelpString(btnSubmit, "Click here to submit  
data!!");  
}
```

- The HelpProvider Control object uses F1 key to be pressed by user to show the HelpString Information for a particular control in which focus is active & Helpstring information is set.

User Controls in Windows Forms Application -

The UserControl in Windows Forms application containing a UI similar to Windows Form to create user defined control, that can be reusable to any number of Windows Form in an Entire Windows Forms application. It is called a Reusable Windows Forms control.

Example -

Create a Windows Form UserControl as Header UserControl that contains HeaderBannerImage with Header Scrolling Text and Reuse on each Windows Forms -

Steps to Create an UserControl in Windows Forms Application -

1. Create a new Windows Forms application or open an existing Windows Forms application.
2. Open Solution explorer, Select Add → New Item, Choose UserControl Template from the Add New Item Templates window & specifies the name of user control as per need or except Default Name.

ex → UCHeader.cs

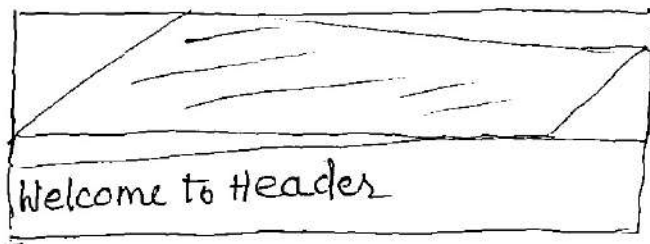
3. Finally Click Add button. Once we click Add Button, it generates two following files similar to Windows Forms

□ UCHeader.cs

□ UCHeader.Designer.cs [Designer generated code]

4. Now Open the UserControl in design View as following and Design UI to create Header & Write following code :-

Design UserControl as following -



Code (UCHeader.cs) -

// Adding all req^d properties to UserControl

```
public Image HeaderImage
{
    get { return this.pictureBox1.Image; }
    set { this.pictureBox1.Image = value; }
}

public string HeaderText
{
    get { return this.label1.Text; }
    set { this.label1.Text = value; }
}

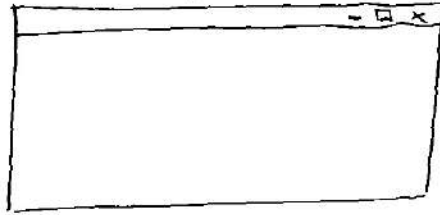
public string HeaderTextBackColor
{
    get { return this.label1.BackColor; }
    set { this.label1.BackColor = value; }
}

public string HeaderTextForeColor
{
    -
    -
}
```

// Write the code on Tick Event of Timer Control Object
to scroll Text from Left to Right

```
private void timer1_Tick(
{
    label1.Left = label1.Left + 50;
    if (label1.Left > this.Width)
        label1.Left = 0;
}
```

- Build the application, as soon as application is builded successfully, All the User defined control will be added into the toolbox as a component of our project on top.
- Now we can use this ~~used~~ user defined control to a Windows Form using Drag & Drop from Toolbox as shown above.
- Once the application builded successfully then particular user defined control can be used to place to a windows forms dynamically also as following:-



Code -

```

ucHeader ucHeader1;
private void Form3_Load(
{
    ucHeader1 = new ucHeader();
    ucHeader1.HeaderImage = Image.FromFile("Images\\Sunset.jpg");
    ucHeader1.HeaderText = "Welcome to project";
    ucHeader1.BackColor = Color.Blue;
    ucHeader1.ForeColor = Color.Yellow;

    ucHeader1.Width = this.Width;
    this.Controls.Add(ucHeader1);

```

Windows Forms Control Library -

- The Windows Forms Control Library is a project template that can be used to create custom controls to be reused in multiple WinForms apps.
- With the help of Windows Forms Control library Project template user can create own customize control (user control)
- The Windows Forms Control Library Project that contains user defined customized controls & it builds in the form of an assembly i.e., Library assembly (.dll), which can be reused in any WinForms application.

Steps to Create Windows Forms Control Library Project

1. Open Visual Studio IDE.
2. Click Main Menu i.e., File → New → Project..
3. Choose Windows Forms Control Library Template from Windows installed template in New project templates window (It is a template to create a project for creating controls to use in WinForms applications). Accept Default Name, Location, Solution Name of the project or change it as per need.
4. Finally "OK".

Once we click OK button a project will be created with following contents

- ☐ Properties
- ☐ Reference
- ☐ UserControl1.cs - It is a default user control

in the project, but in this project we can add n no. of user control as our requirement.

Ex- Create following two User Controls in this project

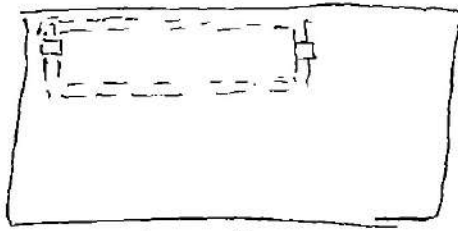
1. UCHeader.cs

2. UCNumericTextBox.cs

- The UCHeader is same as previous example. The design view & code view of UCHeader to create same as the previous example.

- UCNumericTextBox is a user defined control that allows to take input only numeric.

UCNumericTextBox.cs -



public string ^{Numeric}Text

{
get { return this.textBox1.Text; }

set { this.textBox1.Text = value; }

}

private void textBox1_KeyPress (object sender, ^{key}
PressEventArgs e)

{

int key = (int)e.KeyChar;

if ((key >= 48 && key <= 57) || key == 8 || key == 46,

e.Handled = false;

else

e.Handled = true;

int index = textBox1.Text.IndexOf(',');

if (index >= 0)

{

if ((key >= 48 && key <= 57) || key == 8)

e.Handled = false;

else


```
e.Handled = true;
```

```
}
```

```
}
```

After creating above both user controls, we need to build the entire project, once it is build successfully, it generates a library assembly (Ex = Demo WinForms ControlLibrary.dll) & stored in debug folder, which required to use it this Library assembly in WinForms application project.

Steps to use WinForms Control Library in WinForm Apps -

1. Create a new WinForms application or Open an existing WinForms application.
2. Open the Toolbox in the design view environment, Right Click to general tab & Select Add Tab and specify the name of Tab i.e, WinForms ControlLibrary
3. Now Right click on this added tab & then select choose items option to choose the WinForms Control Library Assembly (Demo WinForms ControlLibrary.dll) which contains all uses controls to be added in toolbox in order to use it on WinForm.
4. Click Browse button in opened window to locate WinForms Control Library & Select it.
5. Finally click "OK" button
6. Once we click "OK", all the corresponding uses control belong to Windows Forms control library will be automatically added into the toolbox in a particular selected tab.
7. Now any of the usercontrol from toolbox can be placed on a winform by simply drag & drop.

Collections

Arrays are simple data structures used to store data items of a specific type. Although commonly used, arrays have limited capabilities. For instance, you must specify an array's size, and if at execution time, you wish to modify it, you must do so manually by creating a new array or by using Array class's Resize method, which creates a new array and copies the existing elements into the new array.

Collections are a set of prepackaged data structures that offer greater capabilities than traditional arrays. They are reusable, reliable, powerful and efficient and have been carefully designed and tested to ensure quality and performance. Collections are similar to arrays but provide additional functionalities, such as dynamic resizing – they automatically increase their size at execution time to accommodate additional elements, inserting of new elements, removing of existing elements etc.

Initially .net introduces so many collection classes under the namespace System.Collections like Stack, Queue, LinkedList, SortedList, ArrayList etc, you can work out with these classes in your application where you need the appropriate behaviour.

To use these classes open a new project of type "Console Application" naming it as "CollProject" now under the first class Program.cs write the following code to use the Stack class which works on the principle First In Last Out (FILO) or Last In First Out (LIFO):

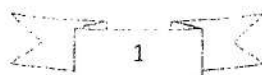
```
using System.Collections;
class Program {
    static void Main(string[] args) {
        Stack s = new Stack(); s.Push(10); s.Push("Hello"); s.Push(3.14f); s.Push(true); s.Push(67.8); s.Push('A');
        foreach (object obj in s) Console.Write(obj + " "); Console.WriteLine();
        Console.WriteLine(s.Pop());
        foreach (object obj in s) Console.Write(obj + " "); Console.WriteLine();
        Console.WriteLine(s.Peek());
        foreach (object obj in s) Console.Write(obj + " "); Console.WriteLine();
        Console.WriteLine(s.Count); s.Clear(); Console.WriteLine(s.Count);
        Console.ReadLine();
    }
}
```

Using Queue class which works on the principle First In First Out (FIFO):

```
using System.Collections;
class Class1 {
    static void Main() {
        Queue q = new Queue();
        q.Enqueue(10); q.Enqueue("Hello"); q.Enqueue(true); q.Enqueue(3.14f); q.Enqueue('A');
        foreach (object obj in q) Console.Write(obj + " "); Console.WriteLine();
        Console.WriteLine(q.Dequeue());
        foreach (object obj in q) Console.Write(obj + " "); Console.ReadLine();
    }
}
```

Auto-Resizing of Collections:

The capacity of a collection increases dynamically i.e. when we keep adding new elements under the collection automatically the size keeps on incrementing. Every collection class has 3 constructors to it and the behavior of collections will be as following when created object using different constructor:



- i. **Default Constructor:** initializes a new instance of the collection class that is empty and has the default initial capacity as zero which becomes 4 after adding the first element and from then when ever needed the current capacity doubles.
- ii. **Collection(int capacity):** Initializes a new instance of the collection class that is empty and has the specified initial capacity, here also when requirement comes current capacity doubles.
- iii. **Collection(Collection):** Initializes a new instance of the collection class that contains elements copied from the specified collection and that has the same initial capacity as the number of elements copied, here also when requirement comes current capacity doubles.

ArrayList: this collection class works same as an array but provides dynamic resizing, adding and deleting of items. using System.Collections;

```
class Class2
{
    static void Main()
    {
        ArrayList al = new ArrayList(); Console.WriteLine("Initial Capacity: " + al.Capacity);
        al.Add(10); Console.WriteLine("Capacity after adding 1st item: " + al.Capacity); al.Add("Hello");
        al.Add(true); al.Add(3.14f); Console.WriteLine("Capacity after adding 4 items: " + al.Capacity);
        al.Add('A'); Console.WriteLine("Capacity after adding 5th item: " + al.Capacity);
        for (int i = 0; i < al.Count; i++) Console.Write(al[i] + " "); Console.WriteLine();
        al.Remove(true); //Removing values from the middle of an ArrayList
        al.Insert(2, false); //Inserting values in the middle of an ArrayList
        foreach (object obj in al) Console.Write(obj + " "); Console.WriteLine();
        ArrayList coll = new ArrayList(al); //Creating new ArrayList passing the old as an parameter
        Console.WriteLine("Initial Capacity of new collection: " + coll.Capacity);
        coll.Add(200); Console.WriteLine("Current Capacity after adding a new item: " + coll.Capacity);
        foreach (object obj in coll) Console.Write(obj + " "); Console.ReadLine();
    }
}
```

Hashtable: it is a collection with stores elements in it as "Key Value Pairs" i.e. Array and ArrayList also has a key to access the values under them which is the index that starts at 0 to number of elements - 1, where as in case of Hashtable these keys can also be defined by us which can be of any data type.

using System.Collections;

```
class Class3
{
    static void Main()
    {
        Hashtable ht = new Hashtable(); ht.Add("Eno", 1001); ht.Add("Ename", "Scott");
        ht.Add("Job", "Manager"); ht.Add("Salary", 5000); ht.Add("Dname", "Sales"); ht.Add("Loc", "Delhi ");
        Console.WriteLine(ht["Salary"]); //Accessing the values of Hashtable using key
        foreach (object obj in ht.Keys) Console.WriteLine(obj + ": " + ht[obj]); Console.ReadLine();
    }
}
```

Generic Collections: these are introduced in C# 2.0 which are extension to collections we have been discussing above, in collections the elements being added is of type object, so we can store any type of values in them, where as in generic collections we can store specific type of values which provides type safety, .Net has re-implemented all the existing collection classes under the namespace System.Collections.Generic but the main difference is while creating object of generic collection classes we need to explicitly specify the type of values we want to store under them. Under this namespace we have been provided with many classes as in System.Collections namespace as following:

Stack<T>, Queue<T>, LinkedList<T>, SortedList<T>, List<T>, Dictionary<Tkey, Tvalue>

Note: <T> refers to the type of values we want to store under them. For example:

```
Stack<int> si = new Stack<int>();    //Stores integer values only
Stack<float> sf = new Stack<float>(); //Stores float values only
Stack<string> ss = new Stack<string>(); //Stores string values only
```

The type of values being stored in a generic collection can be of user-defined type values also like a class type or structure type that is defined to represent an entity as following:

Stack<Customer> sc = new Stack<Customer>(); //Assume Customer is a user-defined class type that represents an entity Customer, so we can store objects of Customer type under the Stack where each object can internally represent different attributes of Customer like Id, Name, Balance, City, State etc.

List: this class is same as ArrayList we have discussed under collections above.

```
class Class4
{
    static void Main()
    {
        List<int> li = new List<int>(); li.Add(10); li.Add(20); li.Add(30); li.Add(30); li.Add(50); li.Add(60);
        foreach (int i in li) Console.WriteLine(i + " "); Console.WriteLine();
        li[3] = 40; //Manipulating List values
        for (int i = 0; i < li.Count; i++) Console.WriteLine(li[i] + " "); Console.ReadLine();
    }
}
```

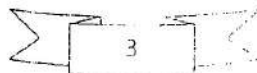
Dictionary: this class is same as Hashtable we have discussed under collections but here while creating the object we need to specify the type for keys as well as for values also, as following:

Dictionary<Tkey, Tvalue>

```
class Class5
{
    static void Main()
    {
        Dictionary<string, object> di = new Dictionary<string, object>();
        di.Add("Eno", 1001); di.Add("Ename", "Scott"); di.Add("Job", "Manager");
        di.Add("Salary", 5000.5); di.Add("Dname", "Sales"); di.Add("Location", "Hyderabad");
        foreach (string key in di.Keys) Console.WriteLine(key + ": " + di[key]); Console.WriteLine();
        di.Remove("Job"); //Removing an element from Dictionary using the Key.
        foreach (string key in di.Keys) Console.WriteLine(key + ": " + di[key]); Console.ReadLine();
    }
}
```

Collection Initializers: this is a new feature added in C# 3.0 which allows to initialize a collection directly at the time of declaration like an array, as following: List<int> li = new List<int>() { 10, 20, 30, 40, 50 };

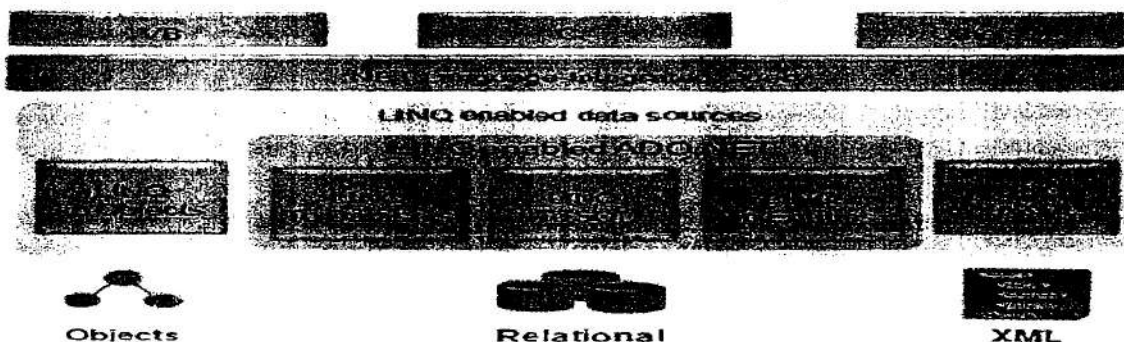
```
class Class6
{
    static void Main()
    {
        List<int> li = new List<int>() { 23, 41, 16, 78, 93, 58, 36, 84, 5, 63, 46, 12, 28, 71, 67 };
        List<int> coll = new List<int>();
        foreach (int i in li) { if (i > 40) coll.Add(i); } //Retrieving values of list > 40
        coll.Sort(); //Sorting the new list values in ascending order
        coll.Reverse(); //Reversing the new list values so that arranges in descending order
        foreach (int i in coll) Console.WriteLine(i + " "); Console.ReadLine();
    }
}
```



In the above program we are filtering the values of a List that are greater than 40 and arranging them in descending order, to do this we have written a substantial amount of code which is the traditional process of performing filters on arrays and collections. In C# 3.0 .net has introduced a new language known as "LINQ" much like SQL which we use universally with relational databases to perform queries. LINQ allows you to write query expressions (similar to SQL queries) that can retrieve information from a wide variety of data sources like objects, databases and xml.

Introduction to LINQ: LINQ stands for Language Integrated Query. LINQ is a data querying methodology which provides querying capabilities to .NET languages with syntax similar to a SQL query. LINQ has a great power of querying on any source of data, where the data source could be collections of objects, database or XML files. The official goal of the LINQ family of technologies is to add "general purpose query facilities to the .NET Framework that apply to all sources of information, not just relational or XML data". Microsoft basically divides LINQ into three areas and that are given below.

LINQ Overview



LINQ to Object: used to perform Queries against the in-memory data like an array or collection.

LINQ to ADO.NET:

- LINQ to SQL is used to perform Queries against the relation database, only Microsoft SQL Server.
- LINQ to DataSet (Supports queries by using ADO.NET data sets and data tables).
- LINQ to Entities.

LINQ to XML (Xling): used to perform Queries against the XML source.

Advantages of LINQ:

- LINQ offers an object-based, language-integrated way to query over data no matter where that data came from. So through LINQ we can query database, XML as well as collections.
- Compile time syntax checking.
- It allows you to query collections, arrays, and classes etc. in the native language of your application, like VB or C # in much the same way as you would query a database using SQL.

LINQ to Objects: using this we can perform queries against the in-memory data like an array or collection and filter or sort the information under them. Syntax of the query we want to use on objects will be as following:

from <alias> in <array | collection> [<clauses>] select <alias>

- Linq queries start with from keyword and ends with select keyword.
- While writing conditions we need to use the alias name we have provided just like we use column names in case of SQL.
- Clauses can be like where, groupby and orderby.
- To use LINQ in your application first we need to import System.Linq namespace.

We can write the above program where we have filtered the data of a List and arranged in sorting order as following using LINQ:

```
class Class7
{
    static void Main()
    {
        List<int> li = new List<int>() { 23, 41, 16, 78, 93, 58, 36, 84, 5, 63, 46, 12, 28, 71, 67 };
        var coll = from i in li where i > 40 select i; //Retrieves all elements greater than 40
        foreach (int i in coll) Console.Write(i + " "); Console.WriteLine();
        coll = from i in li where i > 40 orderby i select i; //Arranging them in ascending order
        foreach (int i in coll) Console.Write(i + " "); Console.WriteLine();
        coll = from i in li where i > 40 orderby i descending select i; //Arranging them in descending order
        foreach (int i in coll) Console.Write(i + " "); Console.ReadLine();
    }
}
```

Note: the values that are returned by a LINQ query can be captured by using implicitly typed local variables, so in the above case "coll" is an implicitly declared collection that stores the values retrieved by the query.

In traditional process of filtering data of an array or collection we have repetition statements that filter arrays focusing on the process of getting the results – iterating through the elements and checking whether they satisfy the desired criteria. LINQ specifies, not the steps necessary to get the results, but rather the conditions that selected elements must satisfy. This is known as **declarative programming** – as opposed to **imperative programming** (which we've been doing so far) in which you specify the actual steps to perform a task. Object oriented programming is a subset of Imperative. The queries we have used above specifies that the result should consist of all the int's in the List that are greater than 40, but it does not specify how those results are obtained – the C# compiler generates all the necessary code automatically, which is one of the great strengths of LINQ.

LINQ Providers: The syntax of LINQ is built into the language, but LINQ queries may be used in many different contexts because of libraries known as providers. A LINQ provider is a set of classes that implement LINQ operations and enable programs to interact with data sources to perform tasks such as projecting, sorting, grouping and filtering elements. When we import the System.Linq namespace it contains the LINQ to Objects provider, without importing it the compiler cannot locate a provider for the LINQ queries and issues errors on LINQ queries.

```
class Class8
{
    static void Main()
    {
        string[] colors = { "Red", "Blue", "Green", "White", "Black", "Yellow", "Pink", "Orange" };
        var coll = from s in colors select s; //Retrieves all colors as it is
        foreach (string str in coll) Console.Write(str + " "); Console.WriteLine();
        coll = from s in colors where s.Length == 5 select s; //Colors with length of 5 characters
        foreach (string str in coll) Console.Write(str + " "); Console.WriteLine();
        coll = from s in colors where s.Substring(0, 1) == "B" select s; //Colors starting with character 'B'
        foreach (string str in coll) Console.Write(str + " "); Console.WriteLine();
        coll = from s in colors where s.EndsWith("e") select s; //Colors ending with character 'e'
        foreach (string str in coll) Console.Write(str + " "); Console.WriteLine();
        coll = from s in colors where s.Contains('e') select s; //Colors that contain character 'e'
        foreach (string str in coll) Console.Write(str + " "); Console.WriteLine();
        coll = from s in colors where s.IndexOf('e') == -1 select s; //Colors that doesnot contain char 'e'
        foreach (string str in coll) Console.Write(str + " "); Console.WriteLine();
    }
}
```

LINQ to SQL

Probably the biggest and most exciting addition to the .Net Framework 3.5 is the addition of the .Net Language Integrated Query Framework (LINQ) into C# 3.0. Basically, what LINQ provides is a lightweight façade over programmatic data integration. This is such a big deal because data is King. Pretty much every application deals with data in some manner, whether that data comes from memory, databases, XML files, text files, or something else. Many developers find it very difficult to move from the strongly typed object-oriented world of C# to the data tier where objects are second-class citizens. The transition from the one world to the next was a kludge at best and was full of error-prone actions.

In C#, programming with objects means a wonderful strongly typed ability to work with code. You can navigate very easily through the namespaces; work with a debugger in the Visual Studio IDE, and more. However, when you have to access data, you will notice that things are dramatically different. You end up in a world that is not strongly typed, where debugging is a pain or even non-existent, and you end up spending most of the time sending strings to the database as commands. As a developer, you also have to be aware of the underlying data and how it is.

Microsoft has provided LINQ as a lightweight façade that provides a strongly typed interface to the underlying data stores. LINQ provides the means for developers to stay within the coding environment they are used to and access the underlying data as objects that work with the IDE, Intellisense, and even debugging. With LINQ, the queries that you create now become first-class citizens within the .NET Framework alongside everything else you are used to. When you work with queries for the data store you are working with, you will quickly realize that they now work and behave as if they are types in the system. This means that you can now use any .NET-compliant language and query the underlying data stores as you never have before.

LINQ to SQL and Visual Studio:

LINQ to SQL in particular is a means to have a strongly typed interface against a SQL Server database. You will find the approach that LINQ to SQL provides is by far the easiest approach to the querying SQL Server available at the moment. It is important to remember that LINQ to SQL is not only about querying data, but you also are able to perform Insert/Update/Delete statements that you need to perform which are known as CRUD operations (Create/Read/Update/Delete). Visual Studio comes into strong play with LINQ to SQL in that you will find an extensive user interface that allows you to design the LINQ to SQL classes you will work with.

To start using LINQ to SQL first open a new Windows Project naming it as "LinqProject", then open the Server Explorer and create a new table under our database naming the table as "Customer" with the following columns and also store some initial data in it:

Custid (Int) [PK]	Cname (Varchar)	City (Varchar)	Balance (Money)
-------------------	-----------------	----------------	-----------------

Adding a LINQ to SQL Class:

To work with LINQ first you need to convert relational types of DB into object oriented types under the language and the process of this conversion is known as ORM (Object Relational Mapping) to perform this we are provided with a tool under visual studio i.e. OR Designer (Object Relational Designer) which does an outstanding job of making it as easy as possible.

To start this task, right-click on LinqProject in Solution Explorer and select 'Add New Item' from the provided menu. In the items of dialog box, you will find LINQ to SQL Classes as an option. Because in this example we are using CSDB database, name the file as "CSDB.dbml" (Database Markup Language). Click on the Add button, and you will see that this operation creates a couple of files for you under the project after adding the "CSDB.dbml" file. Open the solution explorer and watch under the CSDB.dbml file we will find 2 components (CSDB.dbml.layout and CSDB.designer.cs) and also adds the reference of System.Data.Linq assembly.

Introducing the O/R Designer:

Another big addition to the IDE that appeared when you added the LINQ to SQL class to your project was a visual representation of the .dbml file. The O/R Designer will appear as a tab within the document window directly in the IDE. The O/R Designer is made up of two parts. The first part is for data classes, which can be Database, Tables, Views, etc. Dragging such items on this surface will give you a visual representation of the object that can be worked with. The second part (on the right) is for methods, which map to the stored procedures within the database.

Creating the Customer Object:

For this example, we want to work with the Customer table from the CSDB database, which means that you are going to create a Customer class that will use LINQ to SQL map to Customer table. Accomplishing this task is simply a matter of opening up a view of the tables contained within the database from the Server Explorer dialog within Visual Studio and dragging and dropping the Customer table onto the design surface of the O/R Designer in LHS which will prompt with a window asking for storing of 'Connection String' under Config File, so select Yes in it which will add a Config File in the project and adds the connection string into it and also a bunch of code is added to the CSDB.designer.cs file on our behalf with a set of classes in it, and those classes will give you a strongly typed access to the Customer table. When we drag and drop the first table on OR Designer the following actions gets performed internally:

1. Defines a class representing the database from which we are accessing the objects with the name as DataContext and it uses our .dbml file name as a prefix, because our .dbml file name is CSDB the name of DataContext will CSDBDataContext.
2. Defines a class representing the table we have dragged and dropped on the OR Designer where the name of the class will be same as the table name, as we dropped the Customer table on OR Designer Customer class gets defined.
3. Defines properties under the class which is defined representing the table, where each property is defined representing each column of the table.

Note: Now from the second object we place on OR Designer only the 2nd and 3rd steps gets repeated.

Let us have a look into the code added in CSDB.designer.cs file where we will find the classes CSDBDataContext and Customer. CSDBDataContext is an object of type DataContext; we can view this as something that maps to a Connection type object binding with the DB. This object works with the connection string and connects to the database for any required operations when we create object of the class. DataContext class also provides methods like CreateDatabase, DeleteDatabase, GetTable, ExecuteCommand, ExecuteQuery, SubmitChanges etc, using which we can perform action directly on the database.

Customer is the class that represents your table Customer and this class provides required properties mapping with the columns of table and also contains a set of methods DeleteOnSubmit, InsertOnSubmit, GetModifiedMembers, SingleOrDefault etc. for performing CRUD operations on table.

Now place a DataGridView control on the first Form of project, change the name of DataGridView as dgView and write the following code.

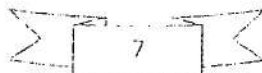
using System.Data.Linq;

Under Form Load:

```
CSDBDataContext dc = new CSDBDataContext();
```

```
Table<Customer> tab = dc.GetTable<Customer>(); dgView.DataSource = tab;
```

Note: In the above case, the DataContext object is used to connect with CSDB database and then the GetTable() method is called to populate the table of type Customer.



Now add a new form in project and design it as following:

Declarations: CSDBDataContext dc; List<Customer> cust; int rno = 0;

Under Form Load:

```
dc = new CSDBDataContext(); cust = dc.GetTable<Customer>().ToList(); ShowData();
```

private void ShowData() {

```
    textBox1.Text = cust[rno].Custid.ToString(); textBox2.Text = cust[rno].Cname;
```

```
    textBox3.Text = cust[rno].City; textBox4.Text = cust[rno].Balance.ToString();
```

}

Under Prev Button: if (rno > 0) {

```
    rno -= 1; ShowData();
```

} else

```
    MessageBox.Show("First record of the table", "Information", MessageBoxButtons.OK,
```

```
    MessageBoxIcon.Information);
```

Under Next Button: if (rno < cust.Count - 1) {

```
    rno += 1; ShowData();
```

} else

```
    MessageBox.Show("Last record of the table", "Information", MessageBoxButtons.OK,
```

```
    MessageBoxIcon.Information);
```

Performing CRUD operations using LINQ:

Create 2 new Forms as following, change the name of DataGridView of first form as dgView and also set its readonly property as True. In second form change the modifier of 4 TextBox's, and Clear Button as Internal.

Code under First Form

Declarations: CSDBDataContext dc;

private void LoadData() {

```
    dc = new CSDBDataContext(); dgView.DataSource = dc.GetTable<Customer>();
```

}

Under Form Load: LoadData();

Under Insert Button: Form4 f = new Form4(); f.ShowDialog(); LoadData();

Under Update Button:

```
if (dgView.SelectedRows.Count > 0) {
    Form4 f = new Form4();
    f.textBox1.Text = dgView.SelectedRows[0].Cells[0].Value.ToString(); f.textBox1.ReadOnly = true;
    f.textBox2.Text = dgView.SelectedRows[0].Cells[1].Value.ToString();
    f.textBox3.Text = dgView.SelectedRows[0].Cells[2].Value.ToString();
    f.textBox4.Text = dgView.SelectedRows[0].Cells[3].Value.ToString();
    f.btnClear.Enabled = false; f.ShowDialog(); LoadData();
}
else
    MessageBox.Show("Select a record for updating", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

Under Delete Button:

```
if (dgView.SelectedRows.Count > 0) {
    if (MessageBox.Show("Do you wish to delete the record?", "Confirmation", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == DialogResult.Yes) {
        int custid = Convert.ToInt32(dgView.SelectedRows[0].Cells[0].Value);
        Customer obj = dc.Customers.SingleOrDefault(C => C.Custid == custid);
        dc.Customers.DeleteOnSubmit(obj); dc.SubmitChanges(); LoadData();
    }
}
else
    MessageBox.Show("Select a record for deleting", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

Under Close Button: this.Close();

Code under Second Form

Under Save Button:

```
CSDBDataContext dc = new CSDBDataContext();
if (textBox1.ReadOnly == false) {
    Customer obj = new Customer();
    obj.Custid = int.Parse(textBox1.Text); obj.Cname = textBox2.Text;
    obj.City = textBox3.Text; obj.Balance = decimal.Parse(textBox4.Text);
    dc.Customers.InsertOnSubmit(obj); dc.SubmitChanges();
    MessageBox.Show("Record added to database table", "Information", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
else {
    Customer obj = dc.Customers.SingleOrDefault(C => C.Custid == int.Parse(textBox1.Text));
    obj.Cname = textBox2.Text; obj.City = textBox3.Text; obj.Balance = decimal.Parse(textBox4.Text);
    dc.SubmitChanges(); ;
    MessageBox.Show("Record modified under database table", "Information", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
```

Under Clear Button: textBox1.Text = textBox2.Text = textBox3.Text = textBox4.Text = ""; textBox1.Focus();

Under Close Button: this.Close();

To Perform CRUD operations on databases using Linq to Sql we need to adopt the following process:

Steps for Inserting:

1. Create an object of Customer entity (Class), which is defined representing the Customer entity (Table) under database into which the record has to be inserted (because each object is a record).
2. Referring to the properties of object assign the values, as we are aware those properties represents columns.
3. Call InsertOnSubmit method on DataContext object referring to the table (Customers) which adds the record to the table in a pending state.
4. Call SubmitChanges method on DataContext object for committing the changes to DB server.

Steps for Updating:

1. Identify the record that has to be updated by calling SingleOrDefault method on DataContext object referring to the table (Customers).
2. Re-assign values to properties so that old values gets changed to new values.
3. Call SubmitChanges method on DataContext object for committing the changes to DB server.

Steps for Deleting:

1. Identify the record that has to be deleted from table by calling SingleOrDefault method on DataContext.
2. Call DeleteOnSubmit method on DataContext object referring to the table (Customers) that deletes the record from table in a pending state.
3. Call SubmitChanges method on DataContext object for committing the changes to DB server.

Calling Stored Procedures thru LINQ:

If we want to call any SP of Sql Server DB using LINQ we need to first drag and drop the SP on RHS panel of OR-designer, so that it gets converted into a method under DataContext class with same name of the SP. If the SP has any parameters those parameters will be defined for the method also, where input parameters of procedure becomes input parameters and output parameters of procedure becomes ref parameters of the method. For example if the below SP was dropped on RHS panel of OR-designer:

Create Procedure Add(@x int, @y int, @z int output)

The method gets created as following:

```
public int Add(int? x, int? y, ref int? z)
```

If the SP contains any non-query operations in it, in such cases the return type of method will be int, where as if the SP has any select statements in it that returns tables as result then the return type of the method will be a ResultSet (Collection of Tables). We can watch the code under Designer.cs file.

Calling Employee Select Procedure: drag and drop Employee_Select SP on the RHS panel of OR-Designer, take a new form place a ComboBox control on top center and add values (All, Active and In-Active) into the ComboBox by using its Items property, place a DataGridView control below the ComboBox, change the name as dgView and write the below code:

Declarations: CSDBDataContext dc;

Under Form Load: dc = new CSDBDataContext(); comboBox1.SelectedIndex = 0;

Under ComboBox SelectedIndexChanged:

```
if (comboBox1.Text == "All")
    dataGridView1.DataSource = dc.Employee_Select(null, null);
else if (comboBox1.Text == "Active")
    dataGridView1.DataSource = dc.Employee_Select(null, true);
else if (comboBox1.Text == "In-Active")
    dataGridView1.DataSource = dc.Employee_Select(null, false);
```

Calling Employee GetSalDetails Procedure: drag and drop Employee_GetSalDetails SP on the OR-Designer, create a new form as following, set the ReadOnly property of 2nd to 5th TextBox's as True and write the following code:

Code under Execute Button:

```
CSDBDataContext dc = new CSDBDataContext(); decimal? sal = null, pf = null, pt = null, nsal = null;
dc.Employee_GetSal(int.Parse(textBox1.Text), ref sal, ref pf, ref pt, ref nsal);
textBox2.Text=sal.ToString();textBox3.Text=pf.ToString();textBox4.Text=pt.ToString();textBox5.Text=nsal.ToString()
```

Code under Close Button: this.Close();

Calling Employee InsertOrUpdate and Employee Delete Procedure's:

Now drag and drop Employee_Insert procedure on the OR-Designer which we have created earlier. Add a new Form, design it as following, place an OpenFileDialog control on the form and write the following code:

using System.IO; using System.Data.Linq;

Declarations: CSDBDataContext dc; string imgPath;

Under Form Load: dc = new CSDBDataContext();

Code under Insert Button:

```
int? Empno = null; byte[] data = File.ReadAllBytes(imgPath); Binary bin = new Binary(data);
dc.Employee_Insert(textBox2.Text, textBox3.Text, decimal.Parse(textBox4.Text), bin, ref Empno);
textBox1.Text = Empno.ToString();
```

Code under Clear Button:

```
textBox1.Text = textBox2.Text = textBox3.Text = textBox4.Text = ""; pictureBox1.Image = null; textBox2.Focus();
```

Code under Close Button: this.Close();

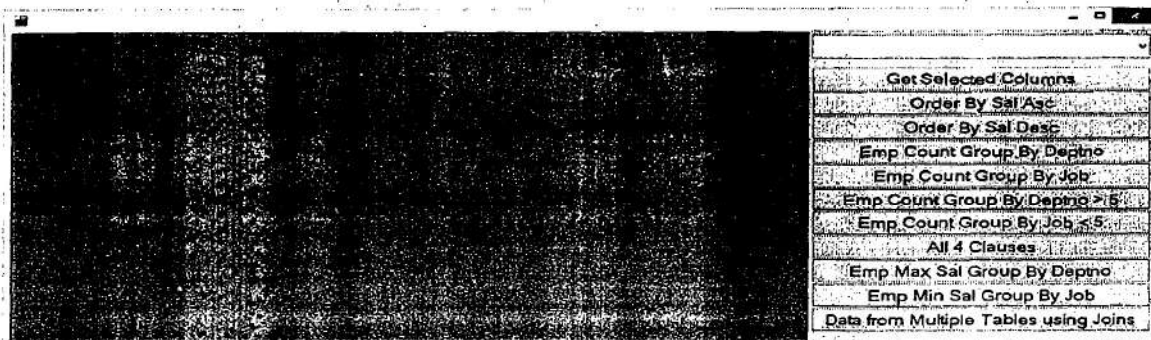
Code under Load Image Button:

```
openFileDialog1.Filter = "Jpeg Images (*.jpg)|*.jpg|Bitmap Images (*.bmp)|*.bmp|All Files (*.*)|*.*";
DialogResult dr = openFileDialog1.ShowDialog();
if (dr == DialogResult.OK) {
    imgPath = openFileDialog1.FileName; pictureBox1.ImageLocation = imgPath;
}
```

Querying data from tables using LINQ to SQL: we can query and retrieve data from table(s) using a query statement which should be used as following:

```
from <alias> in <table> [<clauses>] select <alais> | new { <list of columns> }
```

Now create Emp and Dept tables under CSDB database, drag and drop those tables on LHS panel of OR-Designer, create a new form as following, change the DataGridView name as dgView and write the following code:



Declarations: CSDBDataContext dc; bool flag;

Under Form Load: dc = new CSDBDataContext(); dgView.DataSource = from E in dc.Emps select E;
 comboBox1.DataSource = (from E in dc.Emps select new { E.Job }).Distinct();
 comboBox1.DisplayMember = "Job"; comboBox1.SelectedIndex = -1; flag = true;

Under ComboBox SelectedIndexChanged:

if(flag) { dgView.DataSource = from E in dc.Emps where E.Job == comboBox1.Text select E; }

Under ComboBox KeyPress Event:

if (Convert.ToInt32(e.KeyChar) == 13) {
 if (comboBox1.Text != "All")
 dgView.DataSource = from E in dc.Emps where E.Job == comboBox1.Text select E;
 else
 dgView.DataSource = from E in dc.Emps select E;
 }

Under Button1: dgView.DataSource = from E in dc.Emps select new { E.Empno, E.Ename, E.Job, Salary = E.Sal };

Under Button2: dgView.DataSource = from E in dc.Emps orderby E.Sal select E;

Under Button3: dgView.DataSource = from E in dc.Emps orderby E.Sal descending select E;

Under Button4: dgView.DataSource = from E in dc.Emps group E by E.Deptno into G select new
 { Deptno = G.Key, EmpCount = G.Count() };

Under Button5: dgView.DataSource = from E in dc.Emps group E by E.Job into G select new
 { Job = G.Key, EmpCount = G.Count() };

Under Button6: dgView.DataSource = from E in dc.Emps group E by E.Deptno into G where G.Count() > 5 select new
 { Deptno = G.Key, EmpCount = G.Count() };

Under Button7: dgView.DataSource = from E in dc.Emps group E by E.Job into G where G.Count() < 5 select new
 { Job = G.Key, EmpCount = G.Count() };

Under Button8: dgView.DataSource = from E in dc.Emps where E.Job == "Clerk" group E by E.Deptno into G where
 G.Count() > 1 orderby G.Key descending select new { Job = G.Key, EmpCount = G.Count() };

Under Button9: dgView.DataSource = from E in dc.Emps group E by E.Deptno into G select new
 { Deptno = G.Key, MaxSal = G.Max(E => E.Sal) };

Under Button10: dgView.DataSource = from E in dc.Emps group E by E.Job into G select new
 { Job = G.Key, MinSal = G.Min(E => E.Sal) };

Under Button11: dgView.DataSource = from E in dc.Emps join D in dc.Depts on E.Deptno equals D.Deptno select
 new { E.Empno, E.Ename, E.Job, E.Mgr, E.Sal, E.Comm, D.Deptno, D.DName, D.Loc };

Note: Linq doesn't have having clause, where clause is only provided with the behavior of where as well as having also. If we use where before group by it works like where and if used after group by it works like having clause.

ADO.NET Entity Framework

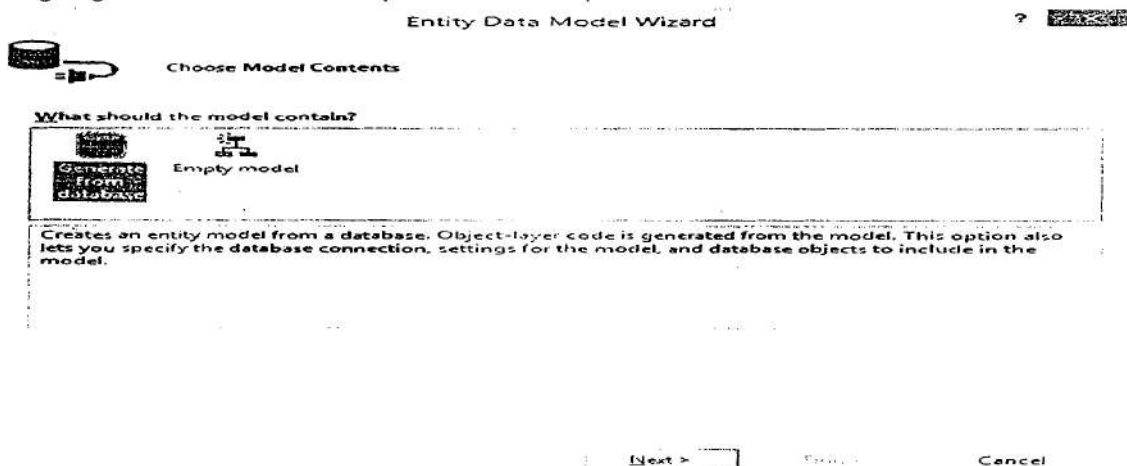
ADO.NET Entity Framework abstracts the relational (logical) schema of the data that is stored in a database and presents its conceptual schema to the application. This abstraction eliminates the object-relational impedance mismatch that is otherwise common in conventional database-oriented programs. For example, in a conventional database-oriented system, entries about a customer and their information can be stored in a Customers table, their orders in an Orders table and their contact information in yet another Contacts table. The application that deals with this database must "know" which information is in which table, i.e., the relational schema of the data is hard-coded into the application.

The disadvantage of this approach is that if this schema is changed the application is not shielded from the change. Also, the application has to perform SQL joins to traverse the relationships of the data elements in order to find related data. For example, to find the orders of a certain customer, the customer needs to be selected from the Customers table, the Customers table needs to be joined with the Orders table, and the joined tables need to be queried for the orders that are linked to the customer. This model of traversing relationships between items is very different from the model used in object-oriented programming languages, where the relationships of an object's features are exposed as Properties of the object and accessing the property traverses the relationship. Also, using SQL queries expressed as strings, only to have it processed by the database, keeps the programming language from making any guarantees about the operation and from providing compile time type information. The mapping of logical schema into the physical schema that defines how the data is structured and stored on the disk is the job of the database system and client side data access mechanisms are shielded from it as the database exposes the data in the way specified by its logical schema.

Entity Data Model: the Entity data model (EDM) specifies the conceptual model (CSDL) of the data via the Entity-Relationship data model, which deals primarily with Entities and the Associations they participate in. The EDM schema is expressed in the Schema Definition Language (SDL), which is an application of XML. In addition, the mapping (MSL) of the elements of the conceptual schema (CSDL) to the storage schema (SSDL) must also be specified. The mapping specification is also expressed in XML. Visual Studio also provides Entity Designer, for visual creation of the EDM and the mapping specification. The output of the tool is the XML file (*.edmx) specifying the schema and the mapping.

Developing an EDM Project configuring with Sql Server:

Open a new project of type windows name it as SqlEDMProject, open the add new item window, select "ADO.NET Entity Data Model", name it as "Sample.edmx" and click Add button, which opens a wizard for configuring with the data source and to perform the ORM operation.



Select "Generate from database" option and click on Next button, which displays a window for configuring with the database as following:

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

lenovo.CSDB.dbc New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☒ Yes, include the sensitive data in the connection string.

Entity connection string:

```
metadata=res://Sample.csdl;res://Sample.ssd;
res://Sample.msl;provider=System.Data.SqlClient;provider connection string="data
source=lenovo;initial catalog=CSDB;persist security info=True;user
id=sa;password=;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save entity connection settings in App.Config as:

CSDBEntities

< Previous Next > Finish Cancel

In this window if our database is already configured under Server Explorer, ComboBox will show that connection details or else click on "New Connection" button beside the ComboBox which opens a Window as following:

Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient) Change...

Server: **lenovo** Refresh

Log on to the server:

☒ Use Windows Authentication

☐ Use SQL Server Authentication

Connect to a database:

Select or enter a database name:

Test Connection Advanced... Cancel

In this window default data source will be Microsoft Sql Server or else click on Change button which opens the following window:

Change Data Source

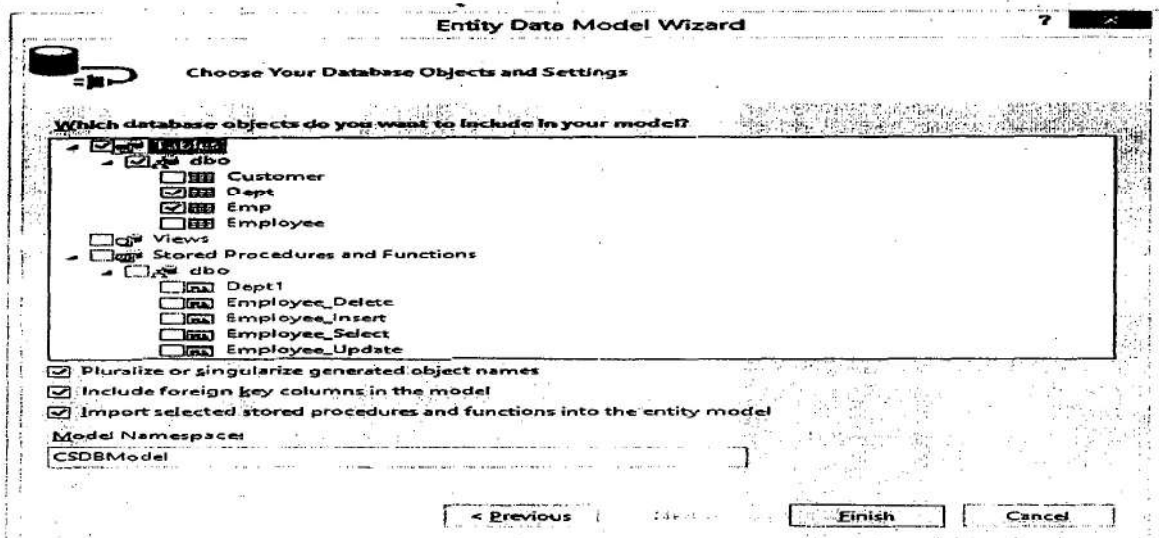
Data source:	Description
Microsoft SQL Server	Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.
Microsoft SQL Server Compact 4.0	
Microsoft SQL Server Database File	
Oracle Server	
<other>	

Data provider:
.NET Framework Data Provider for SQL S v

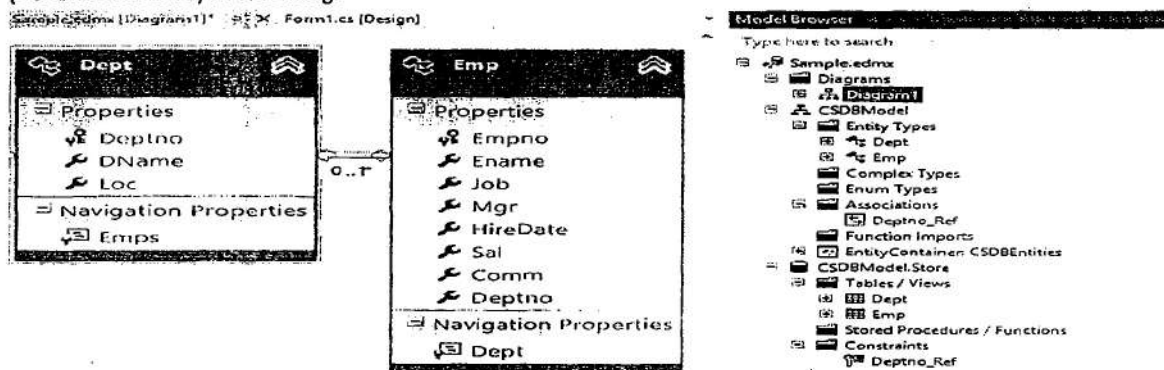
☐ Always use this selection

OK Cancel

Choose Microsoft SQL Server database and click on "OK" button, provide the connection details in the Connection Properties window and click on "Ok" button, now select "Yes, include the sensitive data in the connection string" RadioButton in Entity Data Model Wizard and click on "Next" button which displays a new window as following:



In this window choose the tables Dept and Emp, and click on Finish button which displays the list of tables we have selected, relations between the tables and also on the RHS we will find a window "Model Browser", in it we can find our tables their columns and also the constraint that are used for establishing relations between the tables etc. in object oriented view on the top (CSDBModel) as well as in relational view at the bottom (CSDBModel.Store) as following:



Here under the tables we will be finding some additional properties known as "Navigation Properties" apart from the properties that will be generated for each column of the table. Navigation Properties are used for navigating from one table to the other as these tables have relations between each other, using those properties we can retrieve data from both these table without using any joins.

Note: if working under Visual Studio 2012 under Sample.edmx item we will be finding two additional files Sample.tt and Sample.Context.tt, first delete these 2 files under Solution Explorer and now go to Sample.edmx in document window, right click on it and select Properties and in the property window we will find a property "Code Generation Strategy" which will be having the value as "None" change it as "Default" and build the solution.

Once we build the solution internally all the required classes representing the database with the name as CSDBEntities (same as CSDBDataContext in Linq to Sql), selected tables as well as properties representing all the columns of tables selected as well as methods which are required to perform the database operations gets generated same as we have seen in case of LINQ to SQL. You can check this by expanding the Sample.edmx file in solution explorer and view the code under Sample.Designer.cs file.

Now place a DataGridView control on a form and write the following code in its form load event:

```
CSDBEntities db = new CSDBEntities ();
dataGridView1.DataSource = db.Emps.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");
or
dataGridView1.DataSource = db.Emps.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno").Where("it.Deptno=30");
or
dataGridView1.DataSource = db.Emps.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno").OrderBy("it.Deptno");
or
dataGridView1.DataSource = db.Emps.Select("it.Empno, it.Ename, it.Sal, it.Deptno").OrderBy("it.Deptno Desc");
```

Note: "it" is an alias name representing the table and this is a standard alias name that should be used for any table which cannot be changed. Select, Where and OrderBy are methods that predefined to access the data from required tables.

Performing CRUD Operations: Create 2 forms as following:

In the second Form change the modifier as Internal for all the 5 TextBox's, Save and Clear button also so that they can be accessed from first form and in the first form change the DataGridView name as dgView.

Code under First Form

Declarations: CSDBEntities db;

Under Form Load: db = new CSDBEntities();

dgView.DataSource = db.Emps.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");

Under Insert Button: Form3 f = new Form3(); f.btnSave.Text = "Insert"; f.ShowDialog();

dgView.DataSource = db.Emps.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");

Under Update Button: if (dgView.SelectedRows.Count > 0) {

Form3 f = new Form3();

f.textBox1.Text = dgView.SelectedRows[0].Cells[0].Value.ToString();

f.textBox2.Text = dgView.SelectedRows[0].Cells[1].Value.ToString();

f.textBox3.Text = dgView.SelectedRows[0].Cells[2].Value.ToString();

f.textBox4.Text = dgView.SelectedRows[0].Cells[3].Value.ToString();

f.textBox5.Text = dgView.SelectedRows[0].Cells[4].Value.ToString();

f.btnSave.Text = "Update"; f.ShowDialog();

dgView.DataSource = db.Emps.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");

}

else

MessageBox.Show("Select a record from GridView to update", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

```

Under Delete Button: if (dgView.SelectedRows.Count > 0) {
    if (MessageBox.Show("Do you wish to delete the record?", "Confirmation", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == DialogResult.Yes) {
        int empno = Convert.ToInt32(dgView.SelectedRows[0].Cells[0].Value);
        Emp obj = db.Emps.SingleOrDefault(E => E.Empno == empno); db.Emps.DeleteObject(obj);
        db.SaveChanges(); dgView.DataSource=db.Emps.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");
    }
}
else
    MessageBox.Show("Select a record from GridView to delete", "Warning", MessageBoxButtons.OK,
        MessageBoxIcon.Warning);

```

Code under Second Form

Under Save Button:

```

CSDbEntities db = new CSDbEntities();
if (btnSave.Text == "Insert") {
    Emp obj = new Emp(); obj.Empno = int.Parse(textBox1.Text); obj.Ename = textBox2.Text;
    obj.Job = textBox3.Text; obj.Sal = decimal.Parse(textBox4.Text); obj.Deptno = int.Parse(textBox5.Text);
    db.Emps.AddObject(obj); db.SaveChanges();
}
else {
    int empno = int.Parse(textBox1.Text); Emp obj = db.Emps.SingleOrDefault(E => E.Empno == empno);
    obj.Ename = textBox2.Text; obj.Job = textBox3.Text; obj.Sal = decimal.Parse(textBox4.Text);
    obj.Deptno = int.Parse(textBox5.Text); db.SaveChanges();
}

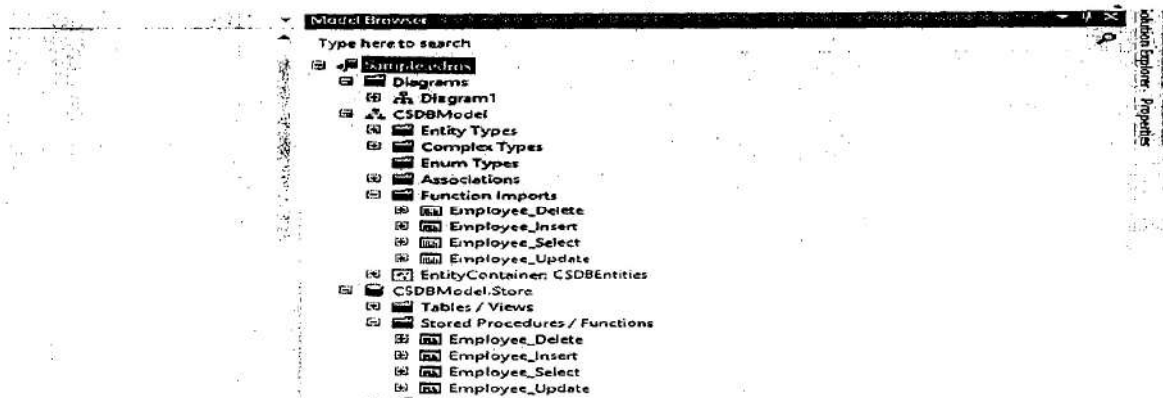
```

Calling Stored Procedures:

To call Stored Procedures, go to Sample.edmx in document window, right click on it and select the option "Update model from the Database" which opens a window as following:



In this window expand the Stored Procedure and Functions node and select the procedures we want to consume in our application and click finish, now in the Model Browser window all the stored procedures gets added in CSDbModel.Store under the node Stored Procedures/functions as well as in CSDbModel under the node Function Imports we can find all the Stored Procedures selected which got converted into methods and we can find them with in the class CSDbEntities, same as we seen in case of Linq to Sql.



To call the Employee_Select procedure add a new Form under the project, place a ComboBox control at "Top Center" of the form and add the values (All, Active and In-Active) by using the items collection property, now place a DataGridView control below the ComboBox setting its dock property as bottom and write the following:

Declarations: CSDbEntities db;

Under Form Load:

db = new CSDbEntities(); comboBox1.SelectedIndex = 0;

Under ComboBox SelectedIndexChanged:

```
if(comboBox1.SelectedIndex == 0) { dataGridView1.DataSource = db.Employee_Select(null, null); }
else if(comboBox1.SelectedIndex == 1) { dataGridView1.DataSource = db.Employee_Select(null, true); }
else if (comboBox1.SelectedIndex == 2) { dataGridView1.DataSource = db.Employee_Select(null, false); }
```

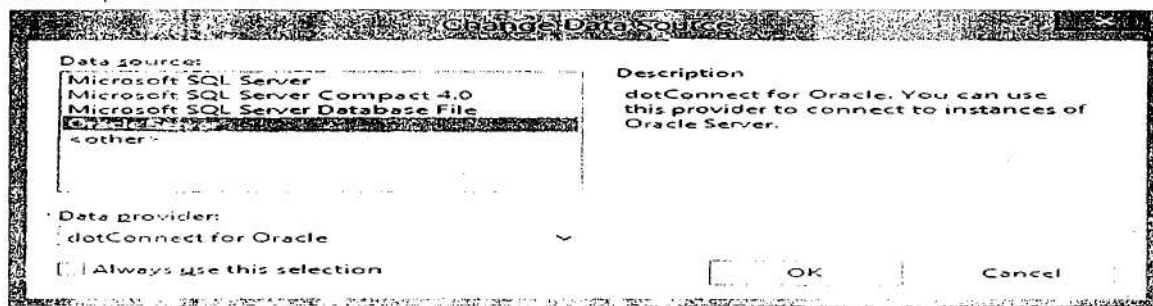
Retrieving data from Tables:

We can retrieve data from table(s) using "Linq to Entities" query language which will be same as we have used in case of "Linq to Sql" for querying the data. We can use all the queries and clauses as it is here also the only difference is retrieving data from multiple tables is simplified here i.e. by using the navigation properties we have seen above we can access the data from multiple tables without using any join statement. To test this add a new form under the project, place a DataGridView Control on it and write the following code under its Form Load:

```
CSDbEntities db = new CSDbEntities();
dataGridView1.DataSource = from E in db.Emps select new { E.Empno, E.Ename, E.Job, E.Mgr, E.HireDate, E.Sal,
E.Comm, E.Dept.Deptno, E.Dept.DName, E.Dept.Loc };
```

Developing an EDM Project configuring with Oracle:

Open a new project of type windows; name it as "OracleEDMProject". Open "Add New Item" window under the project, select "ADO.NET Entity Data Model", name it as "Sample.edmx" and click on Add Button which opens a wizard for configuring with the data source and to perform the ORM operation, select "Generate from database" option in it and click Next Button, in the Next Window click on "New Connection" Button, select Oracle Database in the window opened and click OK:



Provide the connection details for Oracle and then click on Test Connection button to check the connection details and then click on OK button:

Once you click on Ok button it opens the following window, in it select the radio button "Yes, include the sensitive data in the connection string" and click on the Next Button:

Now it displays a window with the list of table and stored procedures present under the database, select Emp and Dept tables in it and click finish button which displays the list of tables we have selected, relations between the tables, also on the RHS we will find a window "Model Browser" and in it we can find our tables their columns and also the constraint that are used for establishing relations between the tables etc.

Here under the tables we will be finding some additional properties known as "Navigation Properties" apart from the properties that will be generated for each column of the table. Navigation Properties are used for navigating from one table to the other as these tables have relations between each other, using those properties we can retrieve data from both these table without using any joins.

Once the tables are selected and clicked on the finish button internally all the required classes representing the database with the name as Entities, selected tables as well as properties representing all the columns of tables selected as well as methods which are required to perform the database operations gets generated same as we have seen in case of LINQ to SQL. You can check this by expanding the Sample.edmx file in solution explorer and view the code under Sample.Designer.cs file.

Note: if working under Visual Studio 2012 under the Sample.edmx we will be finding two additional files Sample.tt and Sample.Context.tt, first delete these 2 files under Solution Explorer and also under the Model Properties we will find a property "Code Generation Strategy" which will be having the value as "None" change it as "Default".

Now Place a DataGridView control on a form change the name of the control ad dgView and write the following code under the form load event:

Entities db = new Entities (); //Creating object of Entities class for establishing connection

dgView.DataSource = db.EMPes.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");

or

dgView.DataSource = db.EMPes.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno").Where("it.Deptno=30");

or

dgView.DataSource =

db.EMPes.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno").Where("it.Deptno=30").OrderBy("it.Sal");

or

dgView.DataSource =

db.EMPes.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno").Where("it.Deptno=30").OrderBy("it.Sal Desc");

Performing CRUD Operations: Create 2 new forms as following:

In the second Form change the modifier as Internal for all the 5 TextBox's, Save and Clear button also so that they can be accessed from first form and in the first form change the DataGridView name as dgView.

Code under First Form

Declarations: Entities db;

Under Form Load: db = new Entities();

dgView.DataSource = db.EMPes.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");

Under Insert Button: Form3 f = new Form3(); f.btnSave.Text = "Insert"; f.ShowDialog();

dgView.DataSource = db.EMPes.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");

Under Update Button: if (dgView.SelectedRows.Count > 0) {

Form3 f = new Form3();

f.textBox1.Text = dgView.SelectedRows[0].Cells[0].Value.ToString(); f.textBox1.ReadOnly = true;

f.textBox2.Text = dgView.SelectedRows[0].Cells[1].Value.ToString();

f.textBox3.Text = dgView.SelectedRows[0].Cells[2].Value.ToString();

f.textBox4.Text = dgView.SelectedRows[0].Cells[3].Value.ToString();

f.textBox5.Text = dgView.SelectedRows[0].Cells[4].Value.ToString();

f.btnSave.Text = "Update"; f.btnClear.Enabled = false; f.ShowDialog();

dgView.DataSource = db.EMPes.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");

}

else

MessageBox.Show("Select a record in GridView for update.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);


```

Under Delete Button: if (dataGridView.SelectedRows.Count > 0) {
    if (MessageBox.Show("Are you sure of deleting the record?", "Confirmation", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == DialogResult.Yes) {
        int Empno = Convert.ToInt32(dgView.SelectedRows[0].Cells[0].Value);
        EMP obj = db.EMPs.SingleOrDefault(E => E.EMPNO == Empno);
        db.EMPs.DeleteObject(obj);
        db.SaveChanges(); dgView.DataSource=db.EMPs.Select("it.Empno, it.Ename, it.Job, it.Sal, it.Deptno");
    }
}
else
    MessageBox.Show("Select a record in GridView for delete.", "Warning", MessageBoxButtons.OK,
        MessageBoxIcon.Warning);

```

Code under Second Form

```

Under Insert Button:
Entities db = new Entities();
if (btnSave.Text == "Insert") {
    EMP obj = new EMP(); obj.EMPNO = int.Parse(textBox1.Text); obj.ENAME = textBox2.Text;
    obj.JOB = textBox3.Text; obj.SAL = double.Parse(textBox4.Text); obj.DEPTNO = int.Parse(textBox5.Text);
    db.EMPs.AddObject(obj); db.SaveChanges();
}
else {
    int Empno = int.Parse(textBox1.Text); EMP obj = db.EMPs.SingleOrDefault(E => E.EMPNO == Empno);
    obj.ENAME = textBox2.Text; obj.JOB = textBox3.Text; obj.SAL = double.Parse(textBox4.Text);
    obj.DEPTNO = int.Parse(textBox5.Text); db.SaveChanges();
}

```

Under Clear Button:

```

textBox1.Text = textBox2.Text = textBox3.Text = textBox4.Text = ""; textBox1.Focus();

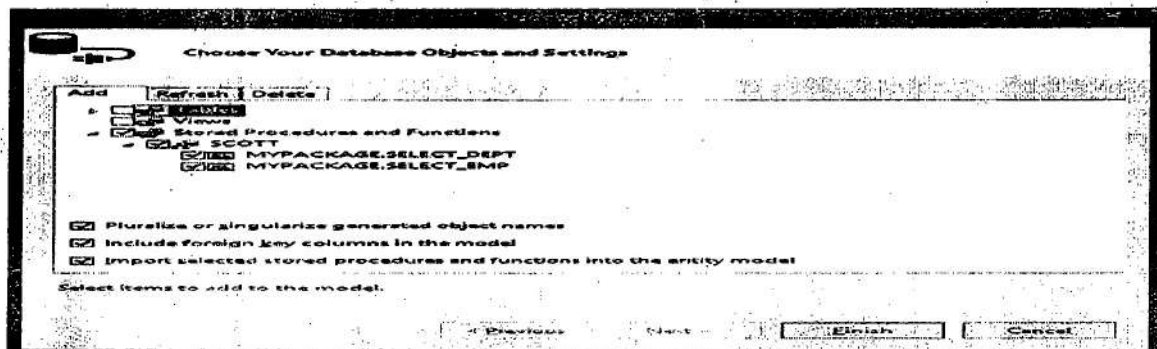
```

Calling Stored Procedures:

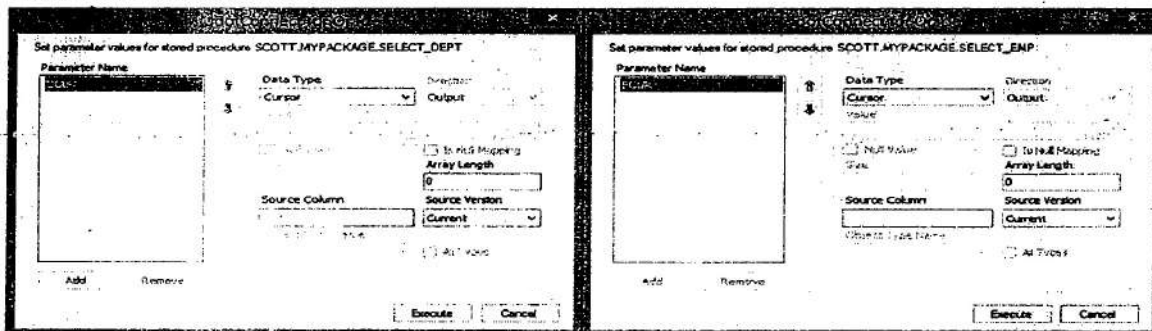
We can call stored procedures also using EDM just like in "Linq to Sql", to call the Select_Emp stored procedure we have defined under the MyPackage earlier while discussing of ADO.Net, first open Sample.edmx file right click on the file and select "Update Model from Database":



Now it opens a "Update Wizard" which we have seen earlier while creating a Model, in that expand the node Stored Procedures and select the SP "MYPACKAGE.SELECT_DEPT" and "MYPACKAGE.SELECT_EMP" and click Finish button:



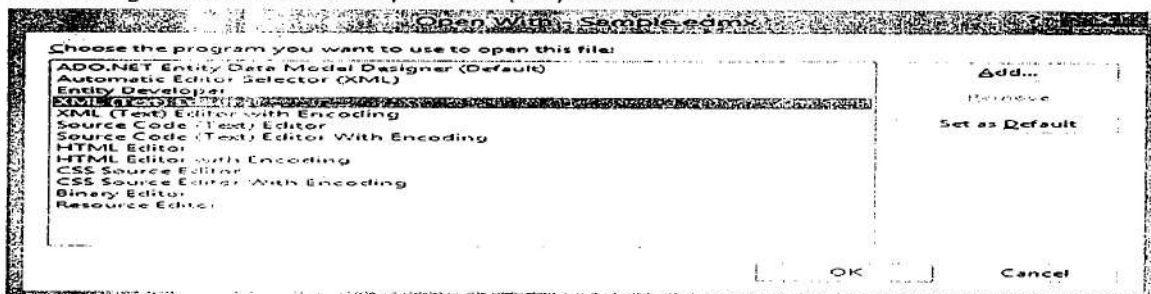
Now a window gets opened asking about the Output Parameter for both the procedures we selected, click on Execute button in both the windows:



Now under the Model Browser below Model.Store we will find the 2 procedures we have selected. As we have learnt in "Linq to Sql" that Stored Procedures gets converted into methods here also the same thing happens, so in Model Browser under the node Sample.edmx we will find the 2 procedures as Function Imports.



Here our Stored Procedures SELECT_DEPT AND SELECT_EMP are returning the data from the database in the form of a cursor as an Output parameter, so first we need to convert them into a return type of our method. To do this open the Solution Explorer right click on the Sample.edmx file and select the option "Open With", which opens the following window init select the option "XML (Text) Editor" and click on the Ok button:



Now it opens the Sample.edmx file in a XML format, and in it we will find a Tag "Schema" as following:

```
<Schema Namespace="Model.Store" Alias="Self" Provider="Devart.Data.Oracle" ProviderManifestToken="Oracle,
10.2.0.3" xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
xmlns="http://schemas.microsoft.com/ado/2009/02/edm/ssdl">
```

Now under the Schema tag add the following content in the last line which should look as following:

```
<Schema Namespace="Model.Store" Alias="Self" Provider="Devart.Data.Oracle" ProviderManifestToken="Oracle,
10.2.0.3" xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
xmlns="http://schemas.microsoft.com/ado/2009/02/edm/ssdl"
xmlns:devart="http://devart.com/schemas/edml/StorageSchemaExtensions/1.0">
```

Actually we are adding a new XML Namespace in the above case. Now go down the file and there we will find a tag Function as following:

```
<Function Name="MYPACKAGE_SELECT_DEPT" Aggregate="false" BuiltIn="false" NiladicFunction="false"
IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion"
StoreFunctionName="MYPACKAGE.SELECT_DEPT" Schema="SCOTT">
  <Parameter Name="DCUR" Type="REF CURSOR" Mode="Out" />
</Function>
```

```
<Function Name="MYPACKAGE_SELECT_EMP" Aggregate="false" BuiltIn="false" NiladicFunction="false"
IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion"
StoreFunctionName="MYPACKAGE.SELECT_EMP" Schema="SCOTT">
  <Parameter Name="ECUR" Type="REF CURSOR" Mode="Out" />
</Function>
```

Under the Function Tag we will find a sub tag Parameter which is our Output parameter DCUR and ECUR what we defined as Cursor which should be changed as a return type of the methods, so do the following to change it and after modification it should look as below:

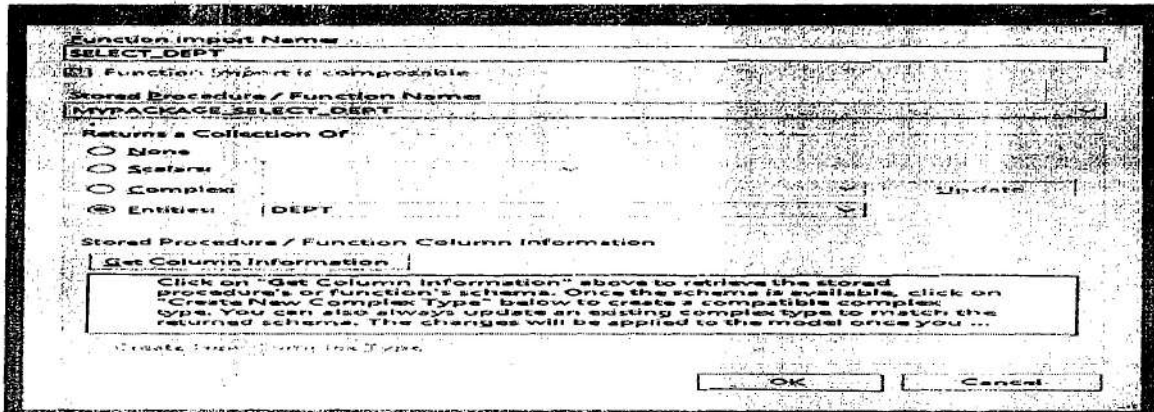
```
<Function Name="MYPACKAGE_SELECT_DEPT" Aggregate="false" BuiltIn="false" NiladicFunction="false"
IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion"
StoreFunctionName="MYPACKAGE.SELECT_DEPT" Schema="SCOTT" devart:ResultSetParameterName="DCUR">
  <!-- <Parameter Name="DCUR" Type="REF CURSOR" Mode="Out" /> -->
</Function>
```

```
<Function Name="MYPACKAGE_SELECT_EMP" Aggregate="false" BuiltIn="false" NiladicFunction="false"
IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion"
StoreFunctionName="MYPACKAGE.SELECT_EMP" Schema="SCOTT" devart:ResultSetParameterName="ECUR">
  <!-- <Parameter Name="ECUR" Type="REF CURSOR" Mode="Out" /> -->
</Function>
```

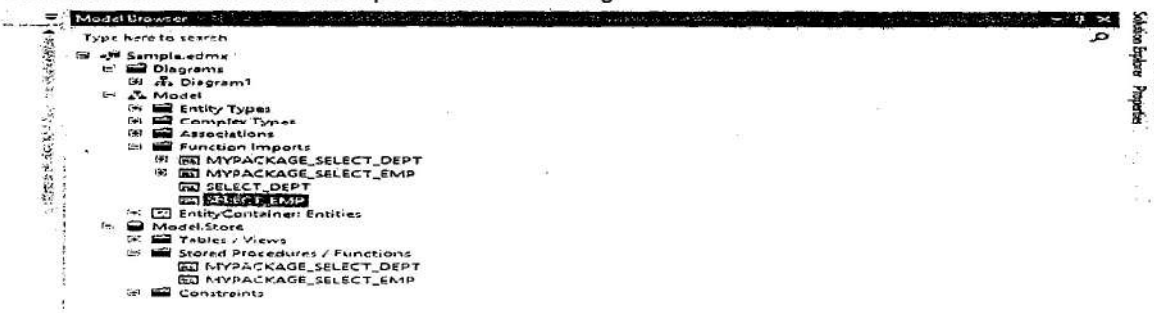
Now save and close the Sample.edmx, re-open it from solution explorer again, go to Model Brower and under Model.Store, expand Stored Procedures / Functions node, right click on the stored procedure "MYPACKAGE.SELECT_DEPT" and select the option "Add Function Import" which opens a window as following:



Now under the window opened change the Function Import Name as Select_DEPT, change the Returns a collection of as "Entities", and choose the Entity "DEPT". After making the changes it should look as following.



Click on Ok button to close the window and do the same for "MYPACKAGE.SELECT_EMP" stored procedure also by specifying the Function Import Name as "Select_EMP" and Entity as "EMP", which will add 2 new functions under the Function Imports node as following:



After adding the 2 functions under Function Imports, delete the first 2 functions "MYPACKAGE_SELECT_DEPT" and "MYPACKAGE_SELECT_EMP" from Function Imports, which should be as below:



Invoking the above Functions:

Add a new Windows Form in the project, place a SplitContainer on it, change the Orientation property as horizontal, place a DataGridView on each panel by setting their Dock property as Fill and write the following code:

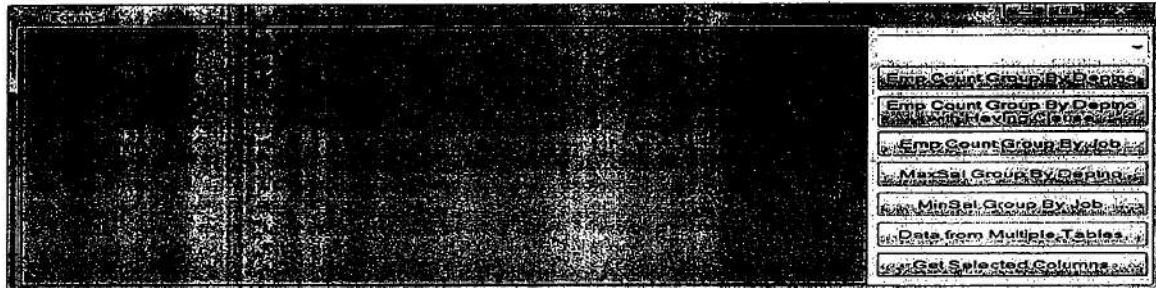
Under Form Load:

```
Entities db = new Entities();
```

```
dataGridView1.DataSource = db.SELECT_DEPT(); dataGridView2.DataSource = db.SELECT_EMP();
```

Retrieving data from tables:

Take a new form and design it as following, change the DataGridView name as dgView and write the code:



Declarations: Entities db;

Under Form Load: db = new Entities(); comboBox1.DataSource = db.EMP_S.Select("it.Job").Distinct();

comboBox1.DisplayMember = "Job";

dgView.DataSource = from E in db.EMP_S select new { E.EMPNO, E.ENAME, E.JOB, E.SAL, E.COMM, E.DEPTNO };

Under ComboBox SelectedIndexChanged:

dgView.DataSource = from E in db.EMP_S where E.JOB == comboBox1.Text select new { E.EMPNO, E.ENAME, E.JOB, E.SAL, E.COMM, E.DEPTNO };

Under Emp Count Group By Deptno Button:

dgView.DataSource = from E in db.EMP_S group E by E.DEPTNO into G orderby G.Key select new { Deptno = G.Key, EmpCount = G.Count() };

Under Emp Count Group By Deptno with Having Clause Button:

dgView.DataSource = from E in db.EMP_S group E by E.DEPTNO into G where G.Count() > 3 orderby G.Key select new { Deptno = G.Key, EmpCount = G.Count() };

Under Emp Count Group By Job Button:

dgView.DataSource = from E in db.EMP_S group E by E.JOB into G orderby G.Key select new { Job = G.Key, JobCount = G.Count() };

Under Max Sal Group By Deptno Button:

dgView.DataSource = from E in db.EMP_S group E by E.DEPTNO into G orderby G.Key select new { Deptno = G.Key, MaxSal = G.Max(i => i.SAL) };

Under Min Sal Group By Job Button:

dgView.DataSource = from E in db.EMP_S group E by E.JOB into G orderby G.Key select new { Job = G.Key, MinSal = G.Min(i => i.SAL) };

Under Data from Multiple Tables Button:

dgView.DataSource = from E in db.EMP_S select new { E.EMPNO, E.ENAME, E.JOB, E.MGR, E.HIREDATE, E.SAL, E.COMM, E.DEPT.DEPTNO, E.DEPT.DNAME, E.DEPT.LOC };

Under Get Selected Columns Button:

dgView.DataSource = from E in db.EMP_S select new { E.EMPNO, E.ENAME, E.JOB, E.SAL, E.DEPTNO };

LINQ Vs EDM:

- LINQ to SQL was developed for rapid application development (RAD), whereas Entity Framework was developed for enterprise application development.
- LINQ to SQL works with the objects in a database whereas Entity Framework works with the conceptual model of a database. As explained earlier, these are two different things which further mean that the Entity Framework allows you to perform queries against relationships between entities, mapping single entities to multiple tables, and so on.

ADO.Net

Pretty much every application deals with data in some manner, whether that data comes from memory, databases, XML files, text files, or something else. The location where we store the data can be called as a Data Source or Data Store where a Data Source can be a file, database, or indexing server etc.

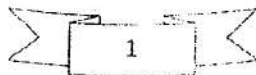
Programming Languages cannot communicate with Data Sources directly because each Data Source adopts a different Protocol (set of rules) for communication, so to overcome this problem long back Microsoft has introduced intermediate technologies like JET, Odbc and Oledb which works like bridge between the Applications and Data Sources to communicate with each other.

The Microsoft Jet Database Engine is a database engine on which several Microsoft products have been built. A database engine is the underlying component of a database, a collection of information stored on a computer in a systematic way. The first version of Jet was developed in 1992, consisting of three modules which could be used to manipulate a database. JET stands for Joint Engine Technology, sometimes being referred to as Microsoft JET Engine or simply Jet. Microsoft Access and Excel uses Jet as their underlying database engine. Over the years, Jet has become almost synonymous with Microsoft Access, to the extent where many people refer to a Jet database as an "Access database". MS developed Jet database system, a C-based interface allowing applications to access that data, and a selection of driver DLLs that allowed the same C interface to redirect input and output to databases, like Paradox and xBase. However, Jet did not use SQL; the interface was in C and consisted of data structures and function calls.

ODBC (Open Database Connectivity) is a standard C programming language middleware API for accessing database management systems (DBMS). ODBC accomplishes DBMS independence by using an ODBC driver as a translation layer between the application and the DBMS. The application uses ODBC functions through an ODBC driver manager with which it is linked, and the driver passes the query to the DBMS. An ODBC driver will be providing a standard set of functions for the application to use, and implementing DBMS-specific functionality. An application that can use ODBC is referred to as "ODBC-compliant". Any ODBC-compliant application can access any DBMS for which a driver is installed. Drivers exist for all major DBMSs as well as for many other data sources like address book systems and Microsoft Excel, and even for text or CSV files. ODBC was originally developed by Microsoft during the early 1990s.

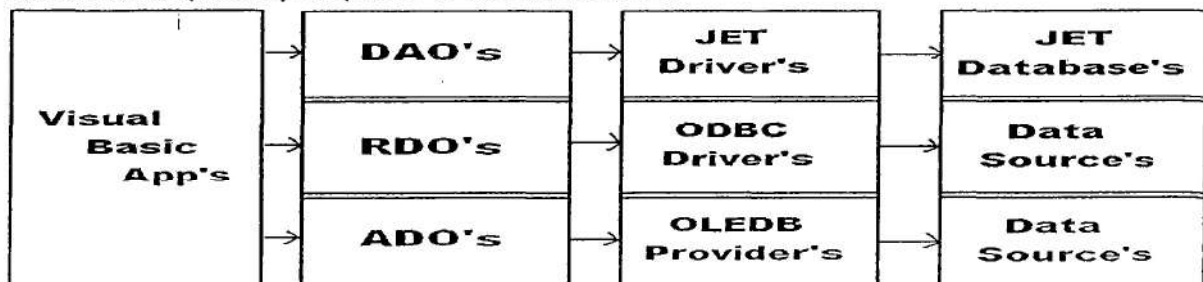
OLE DB (Object Linking and Embedding, Database, sometimes written as OLEDB or OLE-DB), an API designed by Microsoft, allows accessing data from a variety of sources in a uniform manner. The API provides a set of interfaces implemented using the Component Object Model (COM). Microsoft originally intended OLE DB as a higher-level replacement for, and successor to, ODBC, extending its feature set to support a wider variety of non-relational databases, such as object databases and spreadsheets that do not necessarily implement SQL.

OLE DB is conceptually divided into consumers and providers. The consumers are the applications that need access to the data, and the providers are the software components that implement the interface and thereby provide the data to the consumer. An OLE DB provider is a software component enabling an OLE DB consumer to interact with a data source. OLE DB providers are alike to ODBC drivers or JDBC drivers for Java. OLE DB providers can be created to access such simple data stores as a text file and spreadsheet, through to such complex databases as Oracle, Microsoft SQL Server, Sybase ASE, and many others. It can also provide access to hierarchical data stores such as email systems.



DAO's, RDO's and ADO's in Visual Basic Language:

Visual Basic Language used DAO's, RDO's and ADO's for data source communication without having to deal with the comparatively complex JET or ODBC or OLEDB API.



Data Access Objects is a deprecated general programming interface for database access on Microsoft Windows systems using Joint Engine Technology.

Remote Data Objects (abbreviated RDO) is the name of an obsolete data access application programming interface primarily used in Microsoft Visual Basic applications on Windows 95 and later operating systems. This includes database connection, queries, stored procedures, result manipulation, and change commits. It allowed developers to create interfaces that can directly interact with Open Database Connectivity (ODBC) data sources on remote machines.

Microsoft's ActiveX Data Objects (ADO) is a set of Component Object Model (COM) objects for accessing data sources. It provides a middleware layer between programming languages and OLE DB (a means of accessing data stores, whether they be databases or otherwise, in a uniform manner). ADO allows a developer to write programs that access data without knowing how the database is implemented; developers must be aware of the database for connection only. ADO is positioned as a successor to Microsoft's earlier object layers for accessing data sources, including RDO (Remote Data Objects) and DAO (Data Access Objects). ADO was introduced by Microsoft in October 1996.

ADO.NET Providers:

ADO.NET providers can be created to access such simple data stores as a text file and spreadsheet, through to such complex databases as Oracle, Microsoft SQL Server, MySQL, PostgreSQL, SQLite, DB2, Sybase ASE, and many others. They can also provide access to hierarchical data stores such as email systems. However, because different data store technologies can have different capabilities, every ADO.NET provider cannot implement every possible interface available in the ADO.NET standard. Microsoft describes the availability of an interface as "provider-specific," as it may not be applicable depending on the data store technology involved.

ADO.Net:

It is a set of classes that expose data access services to the .NET programmer. ADO.NET provides functionality to developers writing managed code similar to the functionality provided to native COM developers by ADO. ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed through OLE DB and XML.

Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data. ADO.NET provides a rich set of components for creating distributed, data-sharing applications. It is an integral part of the .NET Framework, providing access to relational data, XML, and application data. ADO.NET supports a variety of development needs, including the creation of front-end database clients and middle-tier business objects used by applications, tools, languages, or Internet browsers.

ADO.Net provides libraries for Data Source communication under the following namespaces:

- System.Data
- System.Data.OleDb
- System.Data.SqlClient
- System.Data.OracleClient
- System.Data.Odbc

Note: System.Data, System.Data.OleDb, System.Data.SqlClient and System.Data.Odbc namespaces are defined under the same assembly System.Data.dll where as the namespace System.Data.OracleClient is defined under the assembly System.Data.OracleClient.dll.

System.Data: types of this namespace are used for holding and managing of data on client machines. This namespace contains following set of classes in it: DataSet, DataTable, DataColumn, DataRow, DataView, DataRelation etc.

System.Data.OleDb: types of this namespace can communicate with any Data Source like files, databases, and indexing servers etc. using OleDb Providers (Un-Managed COM Provider).

System.Data.SqlClient: types of this namespace can purely communicate with Sql Server database only using SqlConnection Provider (Managed ADO.Net Provider).

System.Data.OracleClient: types of this namespace can purely communicate with Oracle database only using OracleClient Provider (Managed ADO.Net Provider).

System.Data.Odbc: types of this namespace also can communicate with any Data Source using Un-Managed Odbc Drivers.

All the above 4 namespaces contains same set of types as following: Connection, Command, DataReader, DataAdapter, CommandBuilder etc, but here each class is referred by prefixing with their namespace before the class name to discriminate between each other as following:

OleDbConnection	OleDbCommand	OleDbDataReader	OleDbDataAdapter	OleDbCommandBuilder
SqlConnection	SqlCommand	SqlDataReader	SqlDataAdapter	SqlCommandBuilder
OracleConnection	OracleCommand	OracleDataReader	OracleDataAdapter	OracleCommandBuilder
OdbcConnection	OdbcCommand	OdbcDataReader	OdbcDataAdapter	OdbcCommandBuilder

Performing Operations on a DataSource: Each and every operation we perform on a Data Source involves in 3 steps, like:

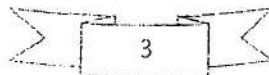
- Establishing a connection with data source.
- Sending request to data source as a sql statement.
- Capturing the results given by data source.

Establishing a Connection with Data Source:

It's a process of opening a channel for communication between Application and Data Source that is present on a local or remote machine to perform any operations. To open the channel for communication we can use the Connection class.

Constructors of the Class:

Connection()
Connection(string connectionString)



Note: `ConnectionString` is a collection of attributes that are used for connecting with a `DataSource`, those are:

- Provider
- Data Source
- User Id and Password
- Database or Initial Catalog
- Trusted_Connection or Integrated Security
- DSN

Provider: as discussed earlier provider is required for connecting with any data source's, we have a different provider available for each data source.

Oracle	Msdasora	Sql Server	SqlOledb
MS-Access or MS-Excel	Microsoft.Jet.Oledb.4.0	MS-Indexing Server	Msidxs

Data Source: it is the name of target machine to which we want to connect with but it is optional when the data source is on a local machine.

User Id and Password: as db's are secured places for storing data, to connect with them we require a valid user id and password.

Oracle: Scott/tiger

Sql Server: Sa/<pwd>

Database or Initial Catalog: these attributes are used while connecting with Sql Server. Database to specify the name of database we want to connect with.

Trusted Connection or Integrated Security: these attributes are also used while connecting with Sql Server or Database only to specify that we want to connect with the Server using Windows Authentication. In this case we should not again use User Id and Password attributes.

DSN: this attribute is used to connect with data sources by using Odbc Drivers.

Connection String for Oracle: "Provider=Msdasora;User Id=Scott;Password=tiger[;Data Source=<server>]"

Connection String for Sql Server: "Provider=SqlOledb;User Id=Sa;Password=<pwd>;Database=<db name>;Data Source=<server>]"

Note: in case of Windows Authentication in place of User Id and Password attributes we need to use Integrated Security = SSPI (Security Support Provider Interface) or Trusted_Connection = True.

Members of Connection class:

1. **Open():** a method which opens a connection with data source.
2. **Close():** a method which closes the connection that is open.
3. **State:** an enumerated property which is used to get the status of connection.
4. **ConnectionString:** a property which is used to get or set a connection string which is associated with the connection object.

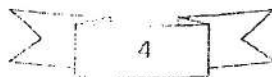
The Object of class Connection can be created in any of the following ways:

`Connection con = new Connection(); con.ConnectionString = "<connection string>;"`

or

`Connection con = new Connection("<connection string>");`

Testing the process of establishing a connection: open a new project of type Windows Forms Application and name it is an DBOperations. Place 2 buttons on the form and set their caption as "Connect with OLEDB Provider" and "Connect with Sql Server using OLEDB Driver". Go to code view and write the following code:



using System.Data.OleDb;

Declarations: OleDbConnection ocon, scon;

Under Connect with Oracle using OLEDB Provider:

```
ocon = new OleDbConnection("Provider=Msdaora;User Id=Scott;Password=tiger;Data Source=<server>");  
ocon.Open(); MessageBox.Show(ocon.State.ToString()); ocon.Close(); MessageBox.Show(ocon.State.ToString());
```

Under Connect with Sql Server using OleDb Provider:

```
scon = new OleDbConnection(); scon.ConnectionString =  
    "Provider=SqlOledb;User Id=Sa;Password=<pwd>;Database=Master;Data Source=<server>";  
scon.Open(); MessageBox.Show(scon.State.ToString()); scon.Close(); MessageBox.Show(scon.State.ToString());
```

Sending request to Data Source as a Sql Statement: In this process we send a request to Data Source by specifying the type of action we want to perform using a Sql Statement like Select, Insert, Update, and Delete or by calling a Stored Procedure. To send and execute those statements on data source we use the class Command.

Constructors of the class: Command() Command(string CommandText, Connection con)

Note: CommandText means it can be any Sql Stmt like Select or Insert or Update or Delete Stmts or Stored Procedure Name.

Properties of Command Class:

1. Connection: sets or gets the connection object associated with command object.
2. CommandText: sets or gets the sql statement or SP name associated with command object.

The object of class Command can be created in any of the following ways:

Command cmd = new Command(); cmd.Connection = <con>; cmd.CommandText = "<sql stmt or SP Name>";

or

Command cmd = new Command("<sql stmt or SP Name>", con);

Methods of Command class:

- ExecuteReader() -> DataReader
- ExecuteScalar() -> object
- ExecuteNonQuery() -> int

Note: after creating object of Command class we need to call any of these 3 methods to execute that statement.

Use ExecuteReader() method when we want to execute a Select Statement that returns data as rows and columns. The method returns an object of class DataReader which holds data that is retrieved from data source in the form of rows and columns.

Use ExecuteScalar() method when we want to execute a Select Statement that returns a single result. The method returns result of the query in the form of an object.

Use ExecuteNonQuery() method when we want to execute any SQL statement other than select, like Insert or Update or Delete etc. The method returns an integer that tells the no. of rows affected by the statement.

Note: The above process of calling a suitable method to capture the results is our third step i.e. capturing the results given by data source.



Accessing data from a DataReader: DataReader is a class which can hold data in the form of rows and columns, to access data from DataReader it provides the following methods:

1. GetName(int columnIndex) -> string

Returns name of the column for given index position.

2. Read() -> bool

Moves record pointer from the current location to next row and returns a bool value which tells whether the row to which we have moved contains data in it or not, that will be true if data is present or false if data is not present.

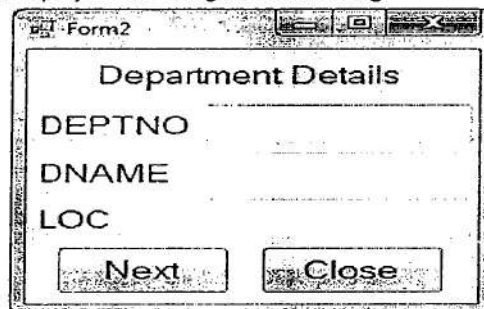
3. GetValue(int columnIndex) -> object

Used for retrieving column values from the row to which pointer was pointing by specifying the column index position. We can also access the row pointed by pointer in the form of a single dimensional array also, either by specifying column index position or name, as following:

<DR>[column index] -> object

<DR>[column name] -> object

Add a new Windows Form under the project and design it as following:



using System.Data.OleDb;

Declarations: OleDbConnection con; OleDbCommand cmd; OleDbDataReader dr;

Under Form Load: con=new OleDbConnection("Provider=Msdaora;User Id=Scott;Password=tiger");
cmd = new OleDbCommand("Select Deptno, Dname, Loc From Dept", con); con.Open(); dr = cmd.ExecuteReader();
label1.Text = dr.GetName(0); label2.Text = dr.GetName(1); label3.Text = dr.GetName(2); ShowData();

```
private void ShowData() {  
    if(dr.Read()) {  
        textBox1.Text = dr.GetValue(0).ToString(); textBox2.Text = dr[1].ToString(); textBox3.Text = dr["Loc"].ToString();  
    }  
    else    MessageBox.Show("Last Record");  
}
```

Under Next Button: ShowData();

Under Close Button: if(con.State != ConnectionState.Closed) { con.Close(); } this.Close();

Working with Sql Server

Sql Server is a collection of databases, where a database is again collection of various objects like tables, views, procedures etc.; users of Sql Server can be owner of 1 or more databases at a time, so while connecting with sql server from a .net application within the connection string we need to specify name of the database we want to connect either by using Database or Initial Catalog attributes.

Sql Server provides 2 different modes of authentication for connecting with the DB Server those are:

1. Windows Authentication
2. Sql Server Authentication

When a user connects through windows authentication, Sql Server validates the account name and password using the windows principal token in the operating system; this means that the user identity is confirmed by windows, Sql Server does not ask for the password and does not perform the identity validation. When using Sql Server authentication, logins are created in sql server that is not based on windows user accounts, both the user name and password are created by using sql server and stored in sql server database. Users connecting with sql server authentication must provide their credentials every time they connect with DB Server.

Note: if we want to connect using windows authentication mode, within the connection string in the place of User Id and Password attributes use "Trusted_Connection=True" or "Integrated Security=SSPI" attributes.

Connecting String for Sql Server Authentication:

"Provider=SqlOledb;User Id=Sa;Password=<pwd>;Database=<dbname>;Data Source=<server name>]"

Connecting String for Windows Authentication:

"Provider=SqlOledb;Trusted_Connection=True;Database=<dbname>;Data Source=<server name>]"

Or

"Provider=SqlOledb;Integrated Security=SSPI;Database=<dbname>;Data Source=<server name>]"

Creating a database on Sql Server:

Go to Start Menu -> Programs -> MS Sql Server -> Sql Server Management Studio, open it and provide the authentication details to login. Once the studio is opened in the LHS we find a window "Object Explorer", in that right click on the node Databases, select "New Database" that opens a window asking for the name, enter the name as: <DB Name>, click ok which adds the database under databases node. Now expand the database node, right click on Tables node and select "New Table" which opens a window asking for column names and data types enter the following:

Eno (Int), Ename (Varchar), Job (Varchar), Salary (Money), Photo (Image), Status (Bit)

Now select Eno Column, right click on it and select the option "Set Primary Key" and make it as an identity or key column of the table. Select Status column, go to its properties in the bottom and set "Default value or Binding" property as 1, which takes the default value for status column as true. Click on the save button at top of the studio which will prompt for table name enter name as "Employee" and click Ok which adds the table under tables node. Now right click on the table created and select "Edit" which opens a window, in that enter the data we want ignoring Photo and Status columns. Close the studio.

Note: We can connect with Sql Server from .net applications either by using OleDb or SqlConnection classes also. If use SqlConnection or OracleConnection classes to connect with databases then connection string doesn't require Provider attribute to be specified as these classes are designed specific for those databases.

Add a new form in the project and design it as following:

The diagram shows a form titled "Employee Details" with the following fields and controls:

- Empno: (text box)
- Ename: (text box)
- Job: (text box)
- Salary: (text box)
- Next (button)
- New (button)
- Insert (button)
- Update (button)
- Delete (button)
- Close (button)

Annotations:

- An arrow points from the Empno field to a box labeled "Set Read-only Property as True".
- An arrow points from the Insert button to a box labeled "Set Enabled Property as False".

using System.Data.SqlClient;

Declarations: SqlConnection con; SqlCommand cmd; SqlDataReader dr; string SqlStr;

Under Form Load: con = new SqlConnection("User Id=Sa;Password=<Pwd>;Database=<DB Name>;Data Source=<Server Name>"); cmd = new SqlCommand(); cmd.Connection = con; con.Open(); LoadData();

private void LoadData() {
 cmd.CommandText = "Select Eno, Ename, Job, Salary From Employee Order By Eno";
 dr = cmd.ExecuteReader(); ShowData();
}

private void ShowData() {
 if (dr.Read()) {
 textBox1.Text = dr[0].ToString(); textBox2.Text = dr[1].ToString();
 textBox3.Text = dr[2].ToString(); textBox4.Text = dr[3].ToString();
 }
 else MessageBox.Show("No data exists.");
}

Under Next Button: ShowData();

Under New Button:

textBox1.Text = textBox2.Text = textBox3.Text = textBox4.Text = "";
dr.Close(); cmd.CommandText = "Select IsNull(Max(Eno), 1000) + 1 From Employee";
textBox1.Text = cmd.ExecuteScalar().ToString(); btnInsert.Enabled = true; textBox2.Focus();

private void ExecuteDML() {
 DialogResult d = MessageBox.Show("Are you sure of executing the below Sql Statement?\n\n" + SqlStr,
 "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
 if (d == DialogResult.Yes) {
 cmd.CommandText = SqlStr; int count = cmd.ExecuteNonQuery();
 if (count > 0) MessageBox.Show("Statement executed successfully");
 else MessageBox.Show("Statement failed execution");
 LoadData();
 }
}

Under Insert Button:

SqlStr = "Insert Into Employee (Eno, Ename, Job, Salary) Values(" + textBox1.Text + ", " + textBox2.Text + ", " + textBox3.Text + ", " + textBox4.Text + ")"; ExecuteDML(); btnInsert.Enabled = false;

or

SqlStr = String.Format("Insert Into Employee (Eno, Ename, Job, Salary) Values({0}, '{1}', '{2}', {3})", textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text); ExecuteDML(); btnInsert.Enabled = false;

Under Update Button:

SqlStr = "Update Employee Set Ename=" + textBox2.Text + ", Job=" + textBox3.Text + ", Salary=" + textBox4.Text + " Where Eno=" + textBox1.Text; dr.Close(); ExecuteDML();

or

SqlStr = String.Format("Update Employee Set Ename='{0}', Job='{1}', Salary={2} Where Eno={3}", textBox2.Text, textBox3.Text, textBox4.Text, textBox1.Text); dr.Close(); ExecuteDML();

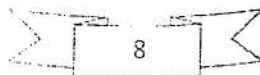
Under Delete Button:

SqlStr = "Delete From Employee Where Eno=" + textBox1.Text; dr.Close(); ExecuteDML();

or

SqlStr = String.Format("Delete From Employee Where Eno={0}", textBox1.Text); dr.Close(); ExecuteDML();

Under Close Button: if (con.State != ConnectionState.Closed) { con.Close(); } this.Close();



DataReader: it's a class designed for holding the data on client machines in the form of Rows and Columns.

Features of DataReader:

1. Faster access to data from the data source as it is connection oriented.
2. Can hold multiple tables in it at a time. To load multiple tables into a DataReader pass multiple select statements as arguments to command separated by a semi-colon.

E.g.: `Command cmd = new Command("Select * From Student;Select * From Teacher", con);`

`DataReader dr = cmd.ExecuteReader();`

Note: use `NextResult()` method on data reader object to navigate from current table to next table.

E.g.: `dr.NextResult();`

Drawbacks of DataReader:

1. As it is connection oriented requires a continuous connection with data source while we are accessing the data, so there are chances of performance degradation if there are more no. of clients accessing data at the same time.
2. It gives forward only access to the data i.e. allows going either to next record or table but not to previous record or table.
3. It is a read only object which will not allow any changes to data that is present in it.

Dis-Connected Architecture: ADO.Net provides 2 different models for accessing data from Data Sources:

1. Connection Oriented Architecture

2. Disconnected Architecture

In the first case we require a continuous connection with the data source for accessing data from it, in this case we use DataReader class for holding the data on client machines, where as in the 2nd case we don't require a continuous connection with data source for accessing of the data from it, we require the connection only for loading the data from data source and here DataSet class is used for holding the data on client machines.

Working with DataSet

DataSet: It's a class present under System.Data namespace designed for holding and managing of data on client machines apart from DataReader. DataSet class provides the following features:

1. It is designed in disconnected architecture which doesn't require any permanent connection with the data source for holding of data.
2. It provides scrollable navigation to data which allows us to move in any direction i.e. either top to bottom or bottom to top.
3. It is updatable i.e. changes can be made to data present in it and those changes can be sent back to DB.
4. DataSet is also capable of holding multiple tables in it.
5. It provides options for searching and sorting of data that is present under it.
6. It provides options for establishing relations between the tables that are present under it.

Using DataSet's: The class which is responsible for loading data into DataReader from a DataSource i.e. Command in the same way DataAdapter class is used for communication between DataSource and DataSet.

DataReader <- Command -> DataSource

DataSet <-> DataAdapter <-> DataSource

Constructors of DataAdapter class:

`DataAdapter(string selectcommand, Connection con)`

`DataAdapter(Command cmd)`

Note: selectcommand means it can be a select statement or a Stored Procedure which contains a select statement



Methods of DataAdapter:

Fill(DataSet ds, string tableName)

Update(DataSet ds, string tableName)

DataAdapter's can internally contain 4 Commands under them associated with a single table, those are:

-Select Command

-Insert Command

-Update Command

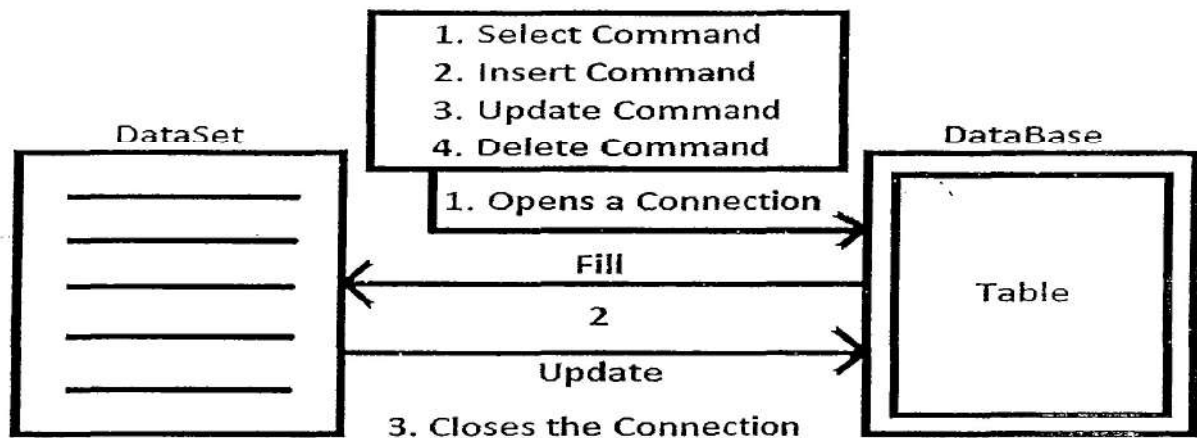
-Delete Command

When we call Fill method on DataAdapter following actions takes place internally:

- Opens a connection with the Data Source.
- Executes the SelectCommand present under it on the DataSource and loads data from table to DataSet.
- Closes the connection.

As we are discussing DataSet is updatable, we can make changes to the data that is loaded into it like adding, modifying and deleting of records, after making changes to data in DataSet if we want to send those changes back to DataSource we need to call Update method on DataAdapter, which performs the following:

- Re-opens a connection with the Data Source.
- Changes that are made to data in DataSet will be sent back to corresponding table, where in this process it will make use of Insert, Update and Delete commands of DataAdapter.
- Closes the connection.



Accessing data from DataSet: Data Reader's provides pointer based access to the data, so we can get data only in a sequential order whereas DataSet provides index based access to the data, so we can get data from any location randomly. DataSet is a collection of tables where each table is represented as a class DataTable and identified by its index position or name. Every DataTable is again collection of Rows and collection of Columns where each row is represented as a class DataRow and identified by its index position and each column is represented as a class DataColumn and identified by its index position or name.

- Accessing a DataTable from DataSet: <dataset>.Tables[index] or <dataset>.Tables[name]
E.g.: ds.Tables[0] or ds.Tables["Employee"]
- Accessing a DataRow from DataTable: <datatable>.Rows[index]
E.g.: ds.Tables[0].Rows[0]
- Accessing a DataColumn from DataTable: <datatable>.Columns[index] or <datatable>.Columns[name]
E.g.: ds.Tables[0].Columns[0] or ds.Tables[0].Columns["Eno"]
- Accessing a Cell from DataTable: <datatable>.Rows[row][col]
E.g.: ds.Tables[0].Rows[0][0] or ds.Tables[0].Rows[0]["Eno"]

Add a new form in the project design it as below, then add reference of Microsoft.VisualBasic assembly from .Net tab of add reference window and write the code:

using System.Data.SqlClient; using Microsoft.VisualBasic;

Declarations: SqlConnection con; SqlDataAdapter da; SqlCommandBuilder cb; DataSet ds; int rno = 0;

Under Form Load:

```
con = new SqlConnection("User Id=sa;Password=<pwd>;Database=<DB Name>;Data Source=<Server Name>");
da = new SqlDataAdapter("Select Eno, Ename, Job, Salary From Employee Order By Eno", con);
ds = new DataSet(); da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
da.Fill(ds, "Employee"); ShowData();
```

private void ShowData() {

```
textBox1.Text=ds.Tables[0].Rows[rno][0].ToString(); textBox2.Text=ds.Tables[0].Rows[rno][1].ToString();
textBox3.Text=ds.Tables[0].Rows[rno][2].ToString(); textBox4.Text=ds.Tables[0].Rows[rno][3].ToString();
}
```

Under First Button: rno = 0; ShowData();

Under Prev Button:

```
if (rno > 0) {
    rno -= 1; if (ds.Tables[0].Rows[rno].RowState == DataRowState.Deleted) {
        MessageBox.Show("Deleted row data cannot be accessed."); return;
    }
    ShowData();
} else MessageBox.Show("First record of the table.");
```

Under Next Button:

```
if (rno < ds.Tables[0].Rows.Count - 1) {
    rno += 1; if (ds.Tables[0].Rows[rno].RowState == DataRowState.Deleted) {
        MessageBox.Show("Deleted row data cannot be accessed."); return;
    }
    ShowData();
}
else
    MessageBox.Show("Last record of the table.");
```

Under Last Button: rno = ds.Tables[0].Rows.Count - 1; ShowData();

Under New Button:

```
textBox1.Text = textBox2.Text = textBox3.Text = textBox4.Text = "";
int index = ds.Tables[0].Rows.Count - 1;
int eno = Convert.ToInt32(ds.Tables[0].Rows[index][0]) + 1;
textBox1.Text = eno.ToString(); textBox2.Focus();
```

Adding a DataRow to DataTable of DataSet:

To add a DataRow to the DataTable of DataSet adopt the following process:

1. Create a new row by calling the NewRow() method on DataTable.
2. Assign values to the new row by treating it as a single dimensional array.
3. Call the Rows.Add method on DataTable and add the row to DataRowCollection.

Under Insert Button:

```
DataRow dr = ds.Tables[0].NewRow();  
dr[0] = textBox1.Text; dr[1] = textBox2.Text; dr[2] = textBox3.Text; dr[3] = textBox4.Text;  
ds.Tables[0].Rows.Add(dr); rno = ds.Tables[0].Rows.Count - 1;  
MessageBox.Show("DataRow added to DataTable of DataSet.");
```

Updating a DataRow in DataTable of DataSet:

To update an existing DataRow in DataTable of DataSet we need to re-assign the modified values back to the DataRow in data table, so that the old values get modified with new values.

Under Update Button:

```
ds.Tables[0].Rows[rno][1] = textBox2.Text; ds.Tables[0].Rows[rno][2] = textBox3.Text;  
ds.Tables[0].Rows[rno][3] = textBox4.Text; MessageBox.Show("DataRow updated in DataTable.");
```

Deleting a DataRow in DataTable of DataSet:

To delete an existing DataRow in DataTable of DataSet call Delete() method pointing to the row that has to be deleted on DataRowCollection.

Under Delete Button:

```
ds.Tables[0].Rows[rno].Delete(); MessageBox.Show("DataRow deleted in DataTable of DataSet.");
```

Saving changes made in DataTable of DataSet back to DataBase:

If we want to save changes made in DataTable of DataSet back to Data Base we need to call Update method on DataAdapter by passing the DataSet which contains modified values as a parameter. If Update method of DataAdapter has to work it should contain the 3 commands under it i.e. Insert, Update and Delete, these 3 commands have to be written by the programmers explicitly or can be generated implicitly with the help of CommandBuilder class. CommandBuilder class constructor if given with DataAdapter that contains a SelectCommand in it will generate the required 3 commands.

Note: CommandBuilder can generate update and delete commands for a given select command only if the table contains Primary Key Constraints on it.

Under Save To DB Button:

```
cb = new SqlCommandBuilder(da); da.Update(ds, "Employee"); MessageBox.Show("Data saved to DB Server");
```

Under Close Button: this.Close();

Searching for a DataRow in DataTable of DataSet:

To search for a DataRow in DataTable of DataSet call Find method on DataRowCollection this searches for the DataRow on Primary Key Column(s) of table and returns a Row.

Find(Object key) -> DataRow

Find(Object[] keys) -> DataRow

Use the first method if the primary key constraint is present on a single column or else use the second method if it is a composite primary key.

Note: if the Find method has to work we need to first load the Primary Key information of table into DataSet by setting the property value as "AddWithKey" for MissingSchemaAction of DataAdapter.

Under Search Button:

```
string value = Interaction.InputBox("Enter Employee No.", "Employee Search", "", 150, 150);
if (value.Trim().Length > 0) {
    int eno = int.Parse(value); DataRow dr = ds.Tables[0].Rows.Find(eno);
    if (dr != null) {
        textBox1.Text = dr[0].ToString(); textBox2.Text = dr[1].ToString();
        textBox3.Text = dr[2].ToString(); textBox4.Text = dr[3].ToString();
    }
    else
        MessageBox.Show("Employee does not exists for given Employee No.");
}
```

Configuration Files

While developing applications if there are any values in application which requires changes in future, should not be hard coded i.e. should not be maintained as static values within the application, because if any changes are required for those values in future client will not be able to make changes because they will not have the source code of application for modification. To overcome this problem we need to identify those values and put them under a special file known as Configuration File, it's an XML file which stores values in it in the form of Key/Value pairs. The values that are present under configuration file can be read from applications in runtime. We store values like Company Name, Address, Phone No, Fax No, connection strings etc., in these files. When an application is installed on the client machines along with it the configuration file also will be installed there and because the configuration file is a text file clients can edit those files and make modification to the values under them at any time and those values will be taken into the application for execution.

Note: Name of configuration file will be "app.config", and it will be available implicitly under the project. If working with below versions of VS 2012 it should be explicitly added under the project, but before adding a configuration file under a project verify the availability of it in the project and if not present then only add it.

Adding configuration file in a project: to add a configuration file in the project open the "Add New Item" window and select the option "Application Configuration File" which adds a file with the name as app.config. By default the file comes with a tag <configuration></configuration> store values under it as following:

```
<appSettings>
    <add key="Cname" value="Naresh I Technologies"/>
    <add key="Address" value="Ameerpet, Hyderabad - 38"/>
    <add key="Phone" value="23746666"/>
    <add key="Email" value="m.bangarraju@gmail.com"/>
</appSettings>
```

Note: if the config file is already added implicitly we can directly open it and maintain our values same as above.

Reading configuration file values from applications: to read configuration file values from applications we are provided with a class ConfigurationManager within the namespace System.Configuration present under the assembly System.Configuration.dll. To consume the class we need to first add reference of the assembly using the "Add Reference" window and we find the assemblies under .Net Tab, after adding the reference of assembly import the namespace and read the values as following:

ConfigurationManager.AppSettings.Get("<key>") -> **string** (returns the value for given key as string)

To test the above process add a configuration file in the project and store values in it as shown above within the <configuration></configuration> tags. Now add a new form in the project place a button on it setting the Text as "Read Configuration Values" and write the following code in code view:

```
using System.Configuration;
```

Under Read Configuration Values Button:

```
string cname = ConfigurationManager.AppSettings.Get("Cname");  
string addr = ConfigurationManager.AppSettings.Get("Address");  
string phone = ConfigurationManager.AppSettings.Get("Phone");  
string email = ConfigurationManager.AppSettings.Get("Email");  
MessageBox.Show(cname + "\n" + addr + "\n" + phone + "\n" + email);
```

Storing Connection Strings under configuration files:

We can store the connection string values also in configuration files so that we don't require to specify the connection strings in all forms and whenever we want to change it we can make the change directly under the configuration file. To store connection strings in the configuration file we are provided with a tag <connectionStrings> same as <appSettings> tag, so we can maintain the connection string values in the configuration file under <configuration> tag as following:

```
<connectionStrings>
```

```
  <add name="OConStr" connectionString="User Id=Scott;Password=tiger;Data Source=<Server Name>"  
        providerName="Msdaora"/>
```

```
  <add name="SConStr" connectionString="User Id=Sa;Password=<your password>;  
        Database=<Your DB Name>;Data Source=<Server Name>" providerName="SqlOleDb"/>
```

```
</connectionStrings>
```

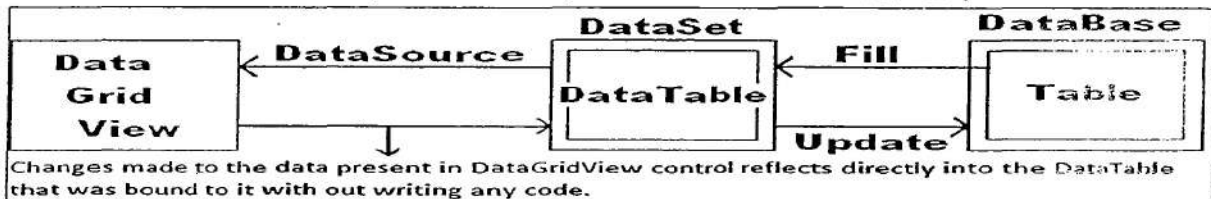
Note: we can read the connection string values from our application as following and use them:

```
string oconstr = ConfigurationManager.ConnectionStrings["OConStr"].ConnectionString;  
string oprovider = ConfigurationManager.ConnectionStrings["OConStr"].ProviderName;  
string sconstr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;  
string sprovider = ConfigurationManager.ConnectionStrings["SConStr"].ProviderName;
```

DataGridView: this control is used for displaying the data in the form of a table i.e. rows and columns. To display data in the control first we need to bind the DataTable of DataSet to the GridView control by using its DataSource property as following:

```
dataGridView1.DataSource = <datatable>
```

DataGridView control has a specialty i.e. changes performed to data in it gets reflected directly to the data of DataTable to which it was bound, so that we can update dataset back to database directly.



To test this process add a new Form in the project and place a DataGridView control on it setting its dock property as top. Now place 2 buttons on the form setting the text as Save and Close and write the code:

```
using System.Data.SqlClient; using System.Configuration;
```

Declarations: SqlConnection con; SqlDataAdapter da; SqlCommandBuilder cb; DataSet ds;

Under Form Load:

```
string constr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;  
con = new SqlConnection(constr); ds = new DataSet();  
da = new SqlDataAdapter("Select Eno, Ename, Job, Salary From Employee Order By Eno", con);  
da.Fill(ds, "Employee"); dataGridView1.DataSource = ds.Tables[0];
```

Under Save Button:

```
cb = new SqlCommandBuilder(da); da.Update(ds, "Employee"); MessageBox.Show("Data saved");
```


Connecting with Oracle Database using Classes of System.Data.OracleClient Namespace:

If we want to connect with Oracle Database using classes of OracleClient namespace we need to first add the reference of System.Data.OracleClient.dll.

Note: In VS 2010 we will not find that assembly by default in the add reference window so to get it into add reference window first you need to change the "Target Framework" of our project to .Net Framework 4, default will be .Net Framework 4 Client Profile and in this we cannot consume all assemblies of .net, after changing the target framework we can access all .net assemblies. To change target framework of project open project properties window and under that we find a ComboBox Target Framework showing different framework's in which we can develop an application, select .Net Framework 4 and select "Yes" in the confirmation window.

Loading multiple tables into DataSet:

A DataSet can hold any no. of tables in it; if we want to load multiple tables into a DataSet we have 2 different approaches.

1. Using a single DataAdapter we can load any no. of tables into the DataSet by changing the SelectCommand of adapter each time after calling Fill and loading a table into DataSet. In this approach if we want to make any changes to the data of dataset and send it back to database we can do it on last table of the dataset only because an adapter can hold all the 4 commands only for a single table i.e. for the last SelectCommand we have given only the required insert, update and delete commands will be generated by CommandBuilder class.
2. Using a separate DataAdapter for each table being loaded into DataSet we can load multiple tables. In this approach it will be possible to update all the tables data, back to the database because each table is using an individual DataAdapter that will hold the required insert, update and delete commands for a table and more over here each table can also be loaded from a different data source i.e. 1 table from Oracle 1 table from Sql Server etc.

DataView: Just like we have Views in SQL, we have DataView object in ADO.Net. A DataView object represents a customized view of DataTable object. Operations like Sorting; Searching can be performed on a DataView object. In scenarios like retrieval of a subset of data from a DataTable, we can make use of DataView to get this data. Note that the DefaultView property of a DataTable returns the default data view for the DataTable.

Using a DataView for filtering or sorting the data under DataTable:

Step1: Create an object of class DataView by calling DefaultView property on the DataTable which will return a DataView object with same structure of the table on which the property is called.

E.g: `DataView dv = ds.Tables["Emp"].DefaultView;`

Step2: Specify a condition for filter by making use of the RowFilter property or specify a column for sorting the data using Sort property of DataView class.

E.g: `dv.RowFilter = "Job = 'Manager'"; dv.RowFilter = "Sal > 2500";
dv.RowFilter = "Job = 'Manager' And Sal > 2500"; dv.RowFilter = "Job = 'Manager' Or Sal > 2500";
dv.Sort = "Sal"; or dv.Sort = "Sal Desc"; dv.Sort = "Sal, Comm"; or dv.Sort = "Sal, Comm Desc";`

Loading multiple tables into a DataSet using single DataAdapter and filtering the data using DataView: Add a new form in the project, place a ComboBox control at top center, place a DataGridView control below and write the following code after adding the reference of System.Data.OracleClient.dll assembly.
using System.Data.OracleClient; using System.Configuration;

Declarations: `OracleConnection con; OracleDataAdapter da; DataSet ds; bool flag = false;`

Under Form Load: `string constr = ConfigurationManager.ConnectionStrings["OConStr"].ConnectionString;
con = new OracleConnection(OConStr); ds = new DataSet();
da = new OracleDataAdapter("Select * from Dept", con); da.Fill(ds, "Dept");
da.SelectCommand.CommandText = "Select * From Emp"; da.Fill(ds, "Emp");`

```
comboBox1.DataSource = ds.Tables["Dept"]; comboBox1.DisplayMember = "Dname";
comboBox1.ValueMember = "Deptno"; comboBox1.SelectedIndex = -1; comboBox1.Text = "Select a Department";
dataGridView1.DataSource = ds.Tables["Emp"]; flag = true;
```

Under ComboBox SelectedIndexChanged:

```
if(flag) {
    DataView dv = ds.Tables["Emp"].DefaultView; dv.RowFilter = "Deptno=" + comboBox1.SelectedValue;
    dv.Sort = "Sal" or dv.Sort = "Sal Desc"; or dv.Sort = "Sal, Comm"; or dv.Sort = "Sal, Comm Desc";
}
```

Note: Just like we can bind a DataTable to DataGridView control in the same way it can also be bound to ComboBox and ListBox controls using DataSource property but these controls even if bound with the table they can display only a single column. So using DisplayMember property of the controls we need to specify which column has to be displayed. We can also bind another column of the table using ValueMember property of the controls but that column values will not be visible to end user where as we can access them in code using SelectedValue property of control for the selected DisplayMember.

Loading multiple tables into a DataSet from different DataSources using multiple DataAdapter's:

Add a new form in the project, place a SplitContainer on it which comes with 2 panels in it. Now place a button on each panel and set the dock property of button as top. Set the caption of the first button as "Save Data to Sql Server" and caption of second button as "Save Data to Oracle". Then add a DataGridView control on each panel and set their dock property as Fill. Then write the following code:

using System.Data.SqlClient; using System.Data.OracleClient; using System.Configuration;

Declarations:

```
SqlConnection scon; SqlDataAdapter sda; SqlCommandBuilder scb;
OracleConnection ocon; OracleDataAdapter oda; OracleCommandBuilder ocb; DataSet ds;
```

Under Form Load:

```
string sconstr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;
string oconstr = ConfigurationManager.ConnectionStrings["OConStr"].ConnectionString;
scon=new SqlConnection(sconstr); sda=new SqlDataAdapter("Select Eno,ENAME,Job,Salary From Employee", scon);
ocon = new OracleConnection(oconstr); oda = new OracleDataAdapter("Select * From Salgrade", ocon);
ds = new DataSet(); sda.Fill(ds, "Employee"); oda.Fill(ds, "Salgrade");
dataGridView1.DataSource = ds.Tables["Employee"]; dataGridView2.DataSource = ds.Tables["Salgrade"];
```

Under Save Data to Sql Server Button:

```
scb=new SqlCommandBuilder(sda);sda.Update(ds, "Employee");MessageBox.Show("Data saved to Sql Server DB.");
```

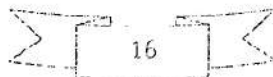
Under Save Data to Oracle Button:

```
ocb=new OracleCommandBuilder(oda);oda.Update(ds, "Salgrade");MessageBox.Show("Data saved to Oracle DB.");
```

DataRelation:

DataRelation is used to relate two DataTable objects to each other through DataColumn objects. For example, in a Dept/Emp relationship, the Dept table is the parent and the Emp table is the child of the relationship. This is similar to a primary key/foreign key relationship. Relationships are created between matching columns in the parent and child tables. That is, the DataType value for both columns must be identical.

To use foreign key constraint we need a parent table that contains master data and a child table that contains detailed data corresponding to master data. Parent table should contain a Reference Key Column with a Primary Key or Unique Key Constraints imposed on it, and child table should contain a Foreign Key Column with a Foreign Key Constraint imposed on it which refers into the values of Reference Key Column. If relationships are established between tables following rules comes into picture:



1. Cannot store a value in the foreign key column of child table, provided the given value is not present in reference key column of parent table.
2. Cannot delete a row from parent table provided the given reference key value of the record being deleted has child records in the child table without addressing what to do with the corresponding child records.
3. Cannot update reference key value of the parent table provided the given reference key value being updated has child records in the child table without addressing what to do with the corresponding child records.

If we want to establish same type of relations between tables of a DataSet also, we can do it with the help of DataRelation class. **DataRelation (string name, DataColumn RKcol, DataColumn FKcol)**

After creating object of DataRelation it has to be added explicitly to the DataSet under which tables were present using Relations.Add method of DataSet. **<dataset>.Relations.Add(DataRelation dr)**

For deleting or updating reference key values in parent table if the reference key value has an child records in the child table some rules comes into picture for delete and update known as DeleteRules and UpdateRules, those were:

1. **None:** Cannot delete or update reference key value of parent table when corresponding child records exists in child table, this rule is applied by default under DB's.
2. **Cascade:** In this case we can delete or update reference key values of parent table, but the corresponding child records in child table will also be deleted or updated, this rule is applied by default in case of DataSet's.
3. **SetNull:** In this case also we can delete or update reference key values of parent table but the corresponding child records foreign key value changes to Null.
4. **SetDefault:** This is same as SetNull, but in this case the corresponding child records foreign key value changes to default value of the column.

Note: we need to apply required rule for delete or update on DataRelation using following statements:

```
<datarelation>.ChildKeyConstraint.DeleteRule = Rule.<rule>;
<datarelation>.ChildKeyConstraint.UpdateRule = Rule.<rule>;
```

To use SetDefault rule first we need to set a default value for foreign column using the following statement:

```
<datatable>.Columns[name].DefaultValue=<value>
```

Loading multiple tables into a DataSet and establishing relation between the tables:

Add a new form in the project, place a SplitContainer on it and change the Orientation property of the control as Horizontal so that the panels under the SplitContainer will be horizontally aligned (default is vertical). Place a DataGridView control on each panel and set their Dock property as Fill. Now write the following code: using System.Data.OracleClient; using System.Configuration;

Declarations: OracleConnection con; OracleDataAdapter da1, da2; DataSet ds; DataRelation dr;

Under Form Load: string constr = ConfigurationManager.ConnectionStrings[OSConStr"].ConnectionString;

con = new OracleConnection(constr); ds = new DataSet();

da1 = new OracleDataAdapter("Select * From Dept", con); da1.Fill(ds, "Dept");

da2 = new OracleDataAdapter("Select * From Emp", con); da2.Fill(ds, "Emp");

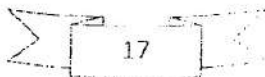
dr = new DataRelation("EmpDept", ds.Tables["Dept"].Columns["Deptno"], ds.Tables["Emp"].Columns["Deptno"]); ds.Relations.Add(dr);

dr.ChildKeyConstraint.DeleteRule = Rule.None; dr.ChildKeyConstraint.UpdateRule = Rule.None;

dataGridView1.DataSource = ds.Tables["Dept"]; dataGridView2.DataSource = ds.Tables["Emp"];

Note: Use the following statement if required for setting a default value to foreign key column:

ds.Tables["Emp"].Columns["Deptno"].DefaultValue = 40;



Establishing relations between tables of DataSet and displaying data in Parent/Child view:

It is possible to establish relations between tables of a DataSet and display the data in Parent/Child view with the help of a class known as BindingSource to test this add a new form in the project, place a SplitContainer on it and change the Orientation property as Horizontal. Place a DataGridView control on each panel and set their Dock property as Fill. Now write the following code:

using System.Data.OracleClient; using System.Configuration;

Declarations: OracleConnection con; OracleDataAdapter da1, da2; DataSet ds;

Under Form Load: string constr = ConfigurationManager.ConnectionStrings["OConStr"].ConnectionString;

con = new OracleConnection(constr);

da1=new OracleDataAdapter("Select * From Dept", con); da2=new OracleDataAdapter("Select * From Emp", con);

ds = new DataSet(); da1.Fill(ds, "Dept"); da2.Fill(ds, "Emp");

ds.Relations.Add("MasterDetail",ds.Tables[0].Columns["Deptno"],ds.Tables[1].Columns["Deptno"]);

BindingSource bsDept = new BindingSource(); bsDept.DataSource = ds; bsDept.DataMember = "Dept";

BindingSource bsEmp = new BindingSource(); bsEmp.DataSource = bsDept; bsEmp.DataMember = "MasterDetail";

dataGridView1.DataSource = bsDept; dataGridView2.DataSource = bsEmp;

Major Differences between DataSet and DataReader:

- DataSet is disconnected architecture while DataReader has live connection while reading data.
 - If we want to cache data and pass to a different tier DataSet's will be the best choice.
 - If we want to move back while reading records, DataReader does not support this functionality.
 - One of the biggest drawbacks of DataSet is speed as DataSet carry considerable overhead because of relations; multiple tables' etc speed is slower than DataReader, so always try to use DataReader where ever possible as it's meant especially for speed performance.
-

Communication with Data Sources using ODBC Drivers

In the case of OleDb, SqlClient and OracleClient classes we use providers for Data Source communication where as in case of Odbc classes we use drivers for Data Source communication. Providers will be on server side so within the connection string if we specify the provider name along with other details like server name, user id, password we can communication with data sources, whereas drivers will be sitting on client machines so to use them first we need to configure the appropriate driver with data source and then from our application using Odbc classes we should communicate with drivers which will in turn communicate with the data sources.

Odbc offers different drivers for connecting with different data sources which comes along with OS, to use them first we need to configure an appropriate driver to its specific data source by supplying all the required connection details and they are stored internally with a name known as DSN (Data Source Name), this should be specified by us only while configuring with the driver. After configuring the DSN, we can use that DSN in our .net app so that Odbc Classes can talk with the Data Source making use of the driver that has been configured as following:

OdbcConnection con = new OdbcConnection("Dsn=<name of the dsn we have created>");

Configuring a DSN: to configure DSN go to Control Panel -> Administrative Tools -> Data Sources (ODBC), click on it to open 'ODBC Data Source Administrator' window. In the window opened click on Add button -> choose a driver for Oracle or Sql Server or MS Excel etc., and click Finish, which opens a window in that first enter a name for DSN and then supply connection details like User Id, Password, Database, Server etc., and Click finish, which adds DSN in Data Source Administrator window.

Configuring a DSN with Sql Server and Oracle Databases:

Open ODBC Data Source Administrator window, click on Add button, select a driver for Sql Server and click Finish button, which opens a window, in it enter the following details, Name: SqlDsn, Description:

with Sql Server Database, Server: <Server Name>, click on Next button, select the RadioButton "Using Sql Server Authentication", enter the Login ID: <User Name>, Password: <Pwd>, click on Next button, select the CheckBox "Change the default database to", and select the Database to which we want to configure with below, click on Next button and Click on Finish button which displays a window showing the connection details, click on Ok button which adds the DSN under ODBC Data Source Administrator window.

Again click on Add button, select a driver for Oracle and click Finish button, which opens a window, in it enter the following details, Data Source Name: OraDsn, Description: Connects with Oracle Database, TNS Service Name: <Server Name>, User ID: Scott/tiger, click on Ok button which adds the DSN under ODBC Data Source Administrator window.

Now add a new form in the Project place 2 buttons on it and set their caption as "Connect with Oracle using ODBC Driver" and "Connect with Sql Server using ODBC Driver". Go to code view and write the following code:

using System.Data.Odbc;

Declarations: OdbcConnection ocon, scon;

Under Connect with Oracle using ODBC Driver: ocon = new OdbcConnection("Dsn=OraDsn");

ocon.Open(); MessageBox.Show(ocon.State.ToString()); ocon.Close(); MessageBox.Show(ocon.State.ToString());

Under Connect with Sql Server using ODBC Driver: scon = new OdbcConnection("Dsn=SqlDsn");

scon.Open(); MessageBox.Show(scon.State.ToString()); scon.Close(); MessageBox.Show(scon.State.ToString());

Accessing MS Excel data from .Net Application

MS Excel is a file system which stores data in the form of rows and columns same as a database table. An Excel document is referred as Work Book that contains Work Sheets in it, work books are considered as databases and work sheets are considered as tables. First row of work sheet can store column names.

Creating an Excel document:

Go to Start Menu -> Programs -> Microsoft Office -> Microsoft Office Excel, click on it to open, by default the document contains 3 work sheets in it. Now in the first row of the sheet1 enter column names for Student table as Sno, Sname, Class, Fees and from the second row enter few records in it. Now in bottom of the document change the sheet name Sheet1 as Student and select 'Save As' choose 'Excel 97 - 2003 Workbook', name the document as School.xls in your desired location.

Connecting with Excel document from .Net Application:

We can connect with Excel documents from .Net application by using Drivers or Providers also. To connect with drivers first we need to configure ODBC driver for Excel. To configure driver go to Start Menu -> Control Panel -> Administrative Tools -> Data Sources (ODBC), click on it to open ODBC Data Source Administrator window, Click Add button, select Microsoft Excel (*.xls) driver, Click finish and Enter the following details, Data Source Name: ExcelDsn, Description: Connects with Excel document, and click on Select Workbook button to choose the School.xls document from its physical location and click on the Ok button which adds the DSN under ODBC Data Source Administrator window. Now add a new windows form in the project and design it as following:

using System.Data.ODBC;

Declarations: ODBCConnection con; ODBCCommand cmd; ODBCDataReader dr; string SqlStr;

Under Form Load:

```
con = new ODBCConnection("Dsn=ExcelDsn;ReadOnly=0");
cmd = new ODBCCCommand(); cmd.Connection = con; con.Open(); LoadData();
label1.Text=dr.GetName(0); label2.Text=dr.GetName(1); label3.Text=dr.GetName(2); label4.Text=dr.GetName(3);
private void LoadData() {
    cmd.CommandText = "Select * From [Student$]"; dr = cmd.ExecuteReader(); ShowData();
}
private void ShowData() {
    if (dr.Read()) {
        textBox1.Text = dr[0].ToString(); textBox2.Text = dr[1].ToString();
        textBox3.Text = dr[2].ToString(); textBox4.Text = dr[3].ToString();
    }
    else
        MessageBox.Show("No data exists.");
}
```

Under Next Button: ShowData();

Under Clear Button: textBox1.Text=textBox2.Text=textBox3.Text=textBox4.Text = ""; textBox1.Focus();

```
private void ExecuteDML() {
    dr.Close(); cmd.CommandText = SqlStr;
    if (cmd.ExecuteNonQuery() > 0)
        MessageBox.Show("Insert Or Update operation was successful.");
    else
        MessageBox.Show("Insert or Update operation failed.");
    LoadData();
}
```

Under Insert Button: sqlstr = String.Format("Insert Into [Student\$] Values ({0}, '{1}' {2}, {3})", textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text); dr.Close(); ExecuteDML();

Under Update Button: sqlstr = String.Format("Update [Student\$] Set Sname='{0}', Class={1}, Fee={2} where Sno={3}", textBox2.Text, textBox3.Text, textBox4.Text, textBox1.Text); ExecuteDML();

Connecting with Excel using OLEDB Provider:

To connect with Excel documents using OLEDB Provider, Connection String is should be as following:

"Provider=Microsoft.Jet.Oledb.4.0; Data Source=<path of excel file>; Extended Properties=Excel 8.0"

Note: OdbcConnection class opens connection with Excel Document in read only mode so if we want to perform any manipulations to data in the document we need to open it in read/write mode by setting the attribute "readonly=0" under the connection string, whereas OleDbConnection will open the document in read/write mode only so no need of using readonly attribute there.

Accessing XML Data from .Net Application

XML stands for Extensible Markup Language. It's a language much like HTML which was designed to carry data, not to display data and also self-descriptive. XML does not do anything. XML was created to structure, store, and transport information. XML tags are not predefined; you must define your own tags. XML is not a replacement for HTML, XML and HTML were designed with different goals:

- XML was designed to transport and store data, with focus on what data is.
- HTML was designed to display data, with focus on how data looks.
- HTML is about displaying information, while XML is about carrying information.

To create an Xml doc we need to satisfy a set of rules that are prescribed by W3C, as following:

1. An Xml doc has to be saved with .xml extension.
2. Data under Xml doc should be present only under tags, where every tag should have a start and end element. e.g.: <Tag>1</Tag> Or <Tag Id="1"/>
3. Tags can also be defined with attribute which are also user-defined where value to attributes should be enclosed under double quotes.
4. While defining tags start and end tag should match in case.
e.g.: <Abc>Hello</Abc> //Valid
<Abc>Hello</abc> //Invalid
5. An Xml doc can have only one root element.

We can also define our own rules on XML data with the help of schemas. A XML schema describes the structure of an XML document. XML Schema language is also referred to as XML Schema Definition (XSD). An XML Schema defines elements that can appear in a document, defines attributes that can appear in a document, defines which elements are child elements, defines the order of child elements, defines the number of child elements, defines whether an element is empty or can include text, defines data types for elements and attributes, defines default and fixed values for elements and attributes.

An XML document with correct syntax is referred as "Well Formed" XML Document and an XML document validated against a schema is referred as "Valid" XML Document

Accessing Xml files from .Net Applications:

Within a .Net application DataSets hold the data in Xml format only, so they are referred as In memory and volatile DB's. We can view content of a dataset using following methods of DataSet class which copies data present in it into a file: WriteXml (string path) WriteXmlSchema (string path)

e.g.: ds.WriteXml("C:\\<folder>\\Employee.xml");
ds.WriteXmlSchema("C:\\<folder>\\Employee.xsd");

Open any form where we used a dataset with Employee table and add the above last 2 stat's under WriteXml method. In the same way we can also read Xml files into a dataset by using following methods:

ReadXml(string path) ReadXmlSchema(string path)

Create an XML document as Emp.xml storing data of employee as following:

```
<Emps>
  <Emp>
    <Eno>1</Eno>
    <Ename>Ajay</Ename>
    <Job>Manager</Job>
    <Salary>5000</Salary>
    <Photo>Any jpg image name with .jpg extension</Photo>
  </Emp>
  -----<Add Multiple Records>-----
</Emps>
```

To generate a schema for above XML document open the document under visual studio, go to XML menu, select "Generate Schema" option which generates a schema for the document, store the XML file (Emp.xml) as well as XML Schema file (Emp.xsd) in one folder and also copy the image file of each employee into the folder where we saved the XML and XSD files. Create a new form as following, add an OpenFileDialog control and write the code:

Form15

Employee Details

Eno:

Ename:

Job:

Salary:

Declarations: DataSet ds; int rno = 0; string dirPath, xmlPath, xsdPath;

Under Load Button: openFileDialog1.Filter = "Xml Files (*.xml)|*.xml|All Files (*.*)|*.*";

DialogResult dr = openFileDialog1.ShowDialog(); if(dr != DialogResult.Cancel) {

xmlPath = openFileDialog1.FileName; xsdPath = xmlPath.Replace(".xml", ".xsd");

dirPath = xmlPath.Substring(0, xmlPath.LastIndexOf("\\") + 1);

ds = new DataSet(); ds.ReadXml(xmlPath); ds.ReadXmlSchema(xsdPath); ShowData();

}

private void ShowData() {

textBox1.Text=ds.Tables[0].Rows[rno][0].ToString();textBox2.Text=ds.Tables[0].Rows[rno][1].ToString();

textBox3.Text=ds.Tables[0].Rows[rno][2].ToString();textBox4.Text=ds.Tables[0].Rows[rno][3].ToString();

pictureBox1.ImageLocation = dirPath + ds.Tables[0].Rows[rno][4].ToString();

}

Under Prev Button: if(rno > 0) {

rno -= 1; ShowData();

} else

MessageBox.Show("First Record of the table.");

Under Next Button: if(rno < ds.Tables[0].Rows.Count - 1) {

rno += 1; ShowData();

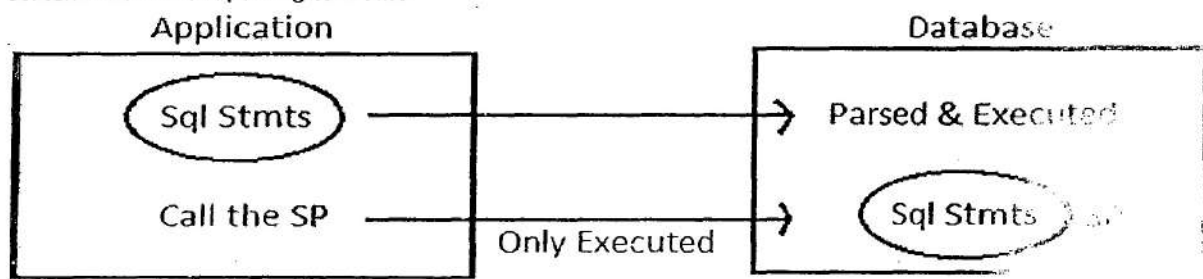
} else

MessageBox.Show("Last Record of the table.");

Under Save to DB Button: write code to insert record retrieved from XML document into Employee table.

Stored Procedures

Whenever we want to interact with a database from an application we use sql stmts. When we use sql statements within the application we have a problem i.e. when the application runs sql Statements will be sent to db for execution where the stmts will be parsed (compile) and then executed. The process of parsing takes place each time we run the application, because of this performance of our application decreases. To overcome this above drawback write sql statements directly under db only, with in an object known as Stored Procedures and call them for execution. As a SP is a pre-compiled block of code that is ready for execution will directly execute the statements without parsing each time.



Syntax to define a Stored Procedure:

Create Procedure <Name> [(<parameter definitions>)

As

Begin

<Stmts>

End

- SP's is similar to a method in our language.
- If required we can also define parameters but only optional. If we want to pass parameters to a Sql Server SP prefix the special character "@" before parameter name.

public void Test(int x) //CSharp

Create Procedure Test(x number) //Oracle

Create Procedure Test(@x int) //Sql Server

- A SP can also return values, to return a value we use out clause in Oracle and Output clause in Sql Server.

public void Test(int x, ref int y) //CSharp

Create Procedure Test(x number, y out number) //Oracle

Create Procedure Test(@x int, @y int output) //Sql Server

Creating a Stored Procedure:

We can create a SP in Sql Server either by using Sql Server Management Studio or Visual Studio.Net also. To create a SP from Visual Studio first we need to configure our Database under Server Explorer, to do this go to view menu, select Server Explorer which gets launched on LHS of the studio. To configure it right click on the node "Data Connections", select "Add Connection" which opens a window asking to choose a Data Source select MS Sql Server, click ok, which opens "Add Connection" window and under it provide the following details:

1. Server Name: <Name of the Server>
2. Authentication: Windows or Sql Server (provide User Name and Password)
3. Database: <DB Name>

Click on the OK button which adds the DB under Server Explorer, expand it, and right click on the node Stored Procedures, select "Add New Stored Procedure" which opens a window and write the following:

Create Procedure Employee_Select

As

Select Eno, Ename, Job, Salaray, From Employee

Now click on the save button at top of the studio which will create the procedure on Database Server.

Calling a SP from .Net application: if we want to call a SP from .net application we use DataAdapter class. If it is a Select SP to load data into a DataSet or else we can use Command class if the SP is of any operation but here if it is Select SP we can load data either into a DataReader or DataSet also. To call the SP we adopt the below process:

Step 1: Create an object of class Command or DataAdapter by passing SP name as argument to their constructor.

DataAdapter da = new DataAdapter("Employee_Select", con);

or

Command cmd = new Command("Employee_Select", con);

Step 2: Change the CommandType property of SelectCommand of DataAdapter object or CommandType property of Command object as StoredProcedure because by default CommandType property is configured to execute Sql Statements only after changing the property we can call Stored Procedures.

da.SelectCommand.CommandType = CommandType.StoredProcedure;

or

cmd.CommandType = CommandType.StoredProcedure;

Step 3: If the SP has any parameters we need to pass values to those parameters by adding the parameters with their corresponding values under SelectCommand of DataAdapter or Command.

```
da.SelectCommand.Parameters.AddWithValue(string <pname>, object <pvalue>)
```

or

```
cmd.Parameters.AddWithValue(string <pname>, object <pvalue>)
```

Step 4: To call the Select SP using DataAdapter we can directly call Fill method on DataAdapter object and load data into DataSet. If we want to call Select SP using Command call the ExecuteReader() method on Command object so that data gets loaded into a DataReader or else if we want to load the data into a DataSet create object of DataAdapter by passing this command object as a parameter and then call Fill method on DataAdapter. If the SP contains any non-query operations like Insert or Update or Delete then call ExecuteNonQuery method on command to execute the SP.

Calling above Stored Procedure using DataAdapter: In a new form place a DataGridView and write below code:
using System.Configuration; using System.Data.SqlClient;

Declarations: SqlConnection con; SqlDataAdapter da; DataSet ds;

Under Form Load: string constr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;

con = new SqlConnection(constr); da = new SqlDataAdapter("Employee_Select", con);

da.SelectCommand.CommandType = CommandType.StoredProcedure;

ds = new DataSet(); da.Fill(ds, "Employee"); dataGridView1.DataSource = ds.Tables[0];

Calling the above Stored Procedure using Command: take a new form and design it as following.

using System.Configuration; using System.Data.SqlClient;

Declarations: SqlConnection con; SqlCommand cmd; SqlDataReader dr;

Under Form Load: string constr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;

con = new SqlConnection(constr); cmd = new SqlCommand("Employee_Select", con);

cmd.CommandType = CommandType.StoredProcedure; con.Open(); dr = cmd.ExecuteReader(); ShowData();

label1.Text = dr.GetName(0); label2.Text = dr.GetName(1);

label3.Text = dr.GetName(2); label4.Text = dr.GetName(3);

private void ShowData() {

if (dr.Read()) {

textBox1.Text = dr[0].ToString(); textBox2.Text = dr[1].ToString();

textBox3.Text = dr[2].ToString(); textBox4.Text = dr[3].ToString();

}

else

MessageBox.Show("Last record of the table");

}

Under Next Button: ShowData();

Parameters to Stored Procedures:

SP can be defined with parameters either to send values for execution or receiving values after execution. While calling a SP with parameters from .net application for each parameter of the SP we need to add a matching parameter either under Command or DataAdapter i.e. for input parameter matching input parameter.

added and for output parameter a matching output parameter has to be added. Every parameter that is added to Command or DataAdapter has 5 attributes to it like Name, Value, DbType, Size and Direction which can be Input(d) or Output or InputOutput (in case of oracle only).

- Name refers to name of the parameter that is defined in SP.
- Value refers to value being assigned in case of input or value we are expecting in case of output.
- DbType refers to data type of the parameter in terms of the DB where the SP exists.
- Size refers to size of data.
- Direction specifies whether parameter is Input or Output or InputOutput.

If a SP has Input or Output parameters we need to specify the following attributes while adding parameters to Command or DataAdapter:

	Input	Output	InputOutput
Name	Yes	Yes	Yes
Value	Yes	No	Yes
DbType	No	Yes	Yes
Size	No	Yes	Yes [Only in case of variable length types]
Direction	No	Yes	Yes

Adding Input Parameter under .Net Application:

```
da.SelectCommand.Parameters.AddWithValue(string <pname>, object <pvalue>);
```

or

```
cmd.Parameters.AddWithValue(string <pname>, object <pvalue>);
```

Adding Output Parameter under .Net Application:

```
da.SelectCommand.Parameters.Add(string <pname>, <dbtype>).Direction = ParameterDirection.Output;
```

or

```
cmd.Parameters.Add(string <pname>, <dbtype>).Direction = ParameterDirection.Output;
```

Setting parameter size if it is a variable length type:

```
da.SelectCommand.Parameters[<pname>].Size = <size>;
```

or

```
cmd.Parameters[<pname>].Size = <size>;
```

After executing the SP we can capture Output parameter values as following:

```
Object obj = da.SelectCommand.Parameters[<pname>].Value;
```

or

```
Object obj = cmd.Parameters[<pname>].Value;
```

Performing Select and DML Operations using Stored Procedures: to perform select, insert, update and delete operations using SP's first define the following procedures in Database.

```
Alter PROCEDURE Employee_Select (@Eno Int=NULL, @Status bit=NULL)
```

```
As
```

```
Begin
```

```
    If @Eno Is Null And @Status Is Null
```

```
        Select Eno, Ename, Job, Salary From Employee
```

```
    Else If @Eno Is Null
```

```
        Select Eno, Ename, Job, Salary From Employee Where Status=@Status
```

```
    Else
```

```
        Select Eno, Ename, Job, Salary, Photo From Employee Where Eno=@Eno And Status=@Status
```

```
End
```

```
Create Procedure Employee_Insert(@Ename Varchar(50), @Job Varchar(50), @Salary Money, @Photo Image,
@Eno Int Output)
```

```
As
```

```
Begin
```

```
Select @Eno = IsNull(Max(Eno), 1000) + 1 From Employee
```

```
Insert Into Employee (Eno, Ename, Job, Salary, Photo) Values (@Eno, @Ename, @Job, @Salary, @Photo)
```

```
End
```

```
Create Procedure Employee_Update(@Eno Int, @Ename Varchar(50), @Job Varchar(50), @Salary Money, @Photo
Image)
```

```
As
```

```
Update Employee Set Ename=@Ename, Job=@Job, Salary=@Salary, Photo=@Photo Where Eno=@Eno
```

```
CREATE PROCEDURE Employee_Delete (@Eno Int)
```

```
As
```

```
Update Employee Set Status=0 Where Eno=@Eno
```

Take a new form, design it as following by adding an OpenFileDialog control and then write below code:

using System.Configuration; using System.IO; using System.Data.SqlClient; using Microsoft.VisualBasic;

Declarations: SqlConnection con; SqlCommand cmd; SqlDataAdapter da; DataSet ds; string imgPath; byte[] data;

Under Form Load: string constr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;

con = new SqlConnection("ConStr"); cmd = new SqlCommand();

cmd.Connection = con; cmd.CommandType = CommandType.StoredProcedure;

Under Select Button: int Eno = int.Parse(Interaction.InputBox("Enter an Employee No."));

cmd.Parameters.Clear(); cmd.CommandText = "Employee_Select"; ds = new DataSet();

cmd.Parameters.AddWithValue("@Eno", Eno); cmd.Parameters.AddWithValue("@Status", true);

da = new SqlDataAdapter(cmd); da.Fill(ds, "Employee");

if (ds.Tables[0].Rows.Count > 0) {

textBox1.Text = ds.Tables[0].Rows[0][0].ToString(); textBox2.Text = ds.Tables[0].Rows[0][1].ToString();

textBox3.Text = ds.Tables[0].Rows[0][2].ToString(); textBox4.Text = ds.Tables[0].Rows[0][3].ToString();

if (ds.Tables[0].Rows[0][4] != System.DBNull.Value) {

data = (byte[])ds.Tables[0].Rows[0][4];

MemoryStream ms = new MemoryStream(data); pictureBox1.Image = Image.FromStream(ms);

}

else { pictureBox1.Image = null; }

}

else

MessageBox.Show("Employee doesn't exist, please check the given employee number.", "Warning",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

Under Clear Button: textBox1.Text = textBox2.Text = textBox3.Text = textBox4.Text = "";
pictureBox1.Image = null; data = null; imgPath = ""; textBox2.Focus();

Under Insert Button:

```
try {
    cmd.Parameters.Clear(); cmd.CommandText = "Employee_Insert";
    cmd.Parameters.AddWithValue("@Ename", textBox2.Text);
    cmd.Parameters.AddWithValue("@Job", textBox3.Text);
    cmd.Parameters.AddWithValue("@Salary", textBox4.Text);
    if (imgPath.Trim().Length > 0) {
        data = File.ReadAllBytes(imgPath); cmd.Parameters.AddWithValue("@Photo", data);
    }
    else {
        cmd.Parameters.AddWithValue("@Photo", DBNull.Value);
        cmd.Parameters["@Photo"].SqlDbType = SqlDbType.Image;
    }
    cmd.Parameters.Add("@Eno", SqlDbType.Int).Direction = ParameterDirection.Output;
    con.Open(); cmd.ExecuteNonQuery(); textBox1.Text = cmd.Parameters["@Eno"].Value.ToString();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally { con.Close(); imgPath = ""; data = null; }
```

Under Update Button:

```
try {
    cmd.Parameters.Clear(); cmd.CommandText = "Employee_Update";
    cmd.Parameters.AddWithValue("@Eno", textBox1.Text);
    cmd.Parameters.AddWithValue("@Ename", textBox2.Text);
    cmd.Parameters.AddWithValue("@Job", textBox3.Text);
    cmd.Parameters.AddWithValue("@Salary", textBox4.Text);
    if (imgPath.Trim().Length == 0 && data == null) {
        cmd.Parameters.AddWithValue("@Photo", DBNull.Value);
        cmd.Parameters["@Photo"].SqlDbType = SqlDbType.Image;
    }
    else if (imgPath.Trim().Length > 0) {
        data = File.ReadAllBytes(imgPath);
        cmd.Parameters.AddWithValue("@Photo", data);
    }
    else if (data != null) {
        cmd.Parameters.AddWithValue("@Photo", data);
    }
    con.Open(); cmd.ExecuteNonQuery();
    MessageBox.Show("Record updated under the database table.", "Information Message", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally { con.Close(); imgPath = ""; data = null; }
```

Under Delete Button:

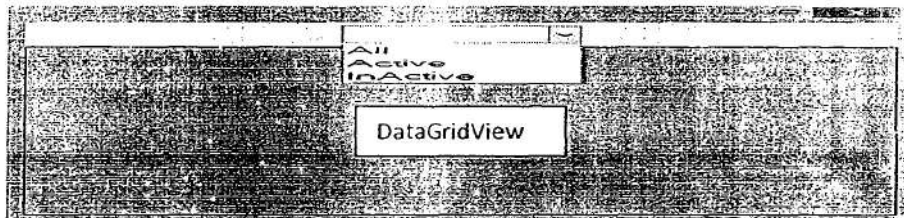
```
try {
    cmd.Parameters.Clear(); cmd.CommandText = "Employee_Delete";
    cmd.Parameters.AddWithValue("@Eno", textBox1.Text);
    con.Open(); cmd.ExecuteNonQuery(); btnClear.PerformClick();
    MessageBox.Show("Record deleted under the database table.", "Sql Message", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally { con.Close(); }
```

Note: when we call PerformClick() method on a button internally the Click of that button occurs.

Under Load Image Button:

```
openFileDialog1.Filter = "Jpeg Images (*.jpg)|*.jpg|Bitmap Images (*.bmp)|*.bmp|All Files (*.*)|*.*";
DialogResult dr = openFileDialog1.ShowDialog();
if (dr == DialogResult.OK) {
    imagePath = openFileDialog1.FileName; pictureBox1.ImageLocation = imagePath;
}
```

Create a new form as below for accessing all, active and in-active records from the table using select SP.



using System.Configuration; using System.Data.SqlClient;

Declarations: SqlConnection con; SqlDataAdapter da; DataSet ds;

Under Select Button: string constr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;
con = new SqlConnection(constr); da = new SqlDataAdapter("Employee_Select", con);
da.SelectCommand.CommandType = CommandType.StoredProcedure; comboBox1.SelectedIndex = 0;

Under ComboBox SelectedIndexChanged: da.SelectCommand.Parameters.Clear();
if (comboBox1.SelectedIndex == 1) { da.SelectCommand.Parameters.AddWithValue("@Status", true); }
else if (comboBox1.SelectedIndex == 2) { da.SelectCommand.Parameters.AddWithValue("@Status", false); }
ds = new DataSet(); da.Fill(ds, "Employee"); dataGridView1.DataSource = ds.Tables[0];

Creating a SP with both Input and Output parameters:

Create a SP under Sql Server DB which takes the Employee No. and returns the Salary, Provident Fund, Professional Tax and Net Salary of the Employee by performing the calculations. In this case writing the SP provides advantages of making the changes easier in the future if required.

CREATE PROCEDURE Employee_GetSalDetails (@Eno Int, @Salary Money Output, @PF Money Output, @PT Money Output, @NetSal Money Output)

As

Begin

Select @Salary = Salary From Employee Where Eno = @Eno And Status = 1;

Set @PF = @Salary * 0.12; Set @PT = @Salary * 0.05; Set @NetSal = @Salary - (@PF + @PT);

End

Calling the above Stored Procedure: Create a new form as following & set read-only of 2nd to 5th textbox's as true.

using System.Data.SqlClient;

Declarations: SqlConnection con; SqlCommand cmd;

Under Form Load: string constr = ConfigurationManager.ConnectionStrings["SConStr"].ConnectionString;

con = new SqlConnection(constr); cmd = new SqlCommand("Employee_GetSalDetails", con);

cmd.CommandType = CommandType.StoredProcedure;

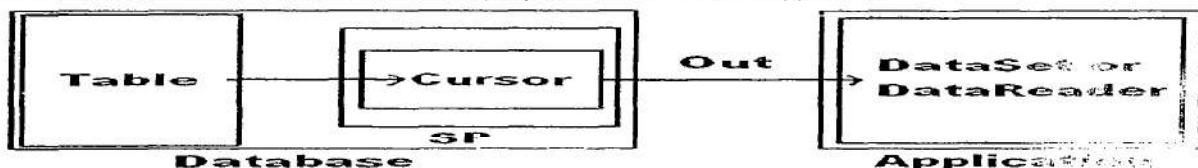
Under Execute Button:

```
try {
    cmd.Parameters.Clear(); cmd.Parameters.AddWithValue("@Eno", textBox1.Text);
    cmd.Parameters.Add("@Salary", SqlDbType.Money).Direction = ParameterDirection.Output;
    cmd.Parameters.Add("@PF", SqlDbType.Money).Direction = ParameterDirection.Output;
    cmd.Parameters.Add("@PT", SqlDbType.Money).Direction = ParameterDirection.Output;
    cmd.Parameters.Add("@NetSal", SqlDbType.Money).Direction = ParameterDirection.Output;
    con.Open(); cmd.ExecuteNonQuery();
    textBox2.Text = cmd.Parameters["@Salary"].Value.ToString();
    textBox3.Text = cmd.Parameters["@PF"].Value.ToString();
    textBox4.Text = cmd.Parameters["@PT"].Value.ToString();
    textBox5.Text = cmd.Parameters["@NetSal"].Value.ToString();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally { con.Close(); }
```

Under Close Button: this.Close();

Calling Select Stored Procedures from Oracle:

We can't use select statements directly in Oracle SP's as we used in Sql Server, so to define a select SP in Oracle for retrieving any data from the table(s) first we need to load the data from table into a cursor under SP and then that cursor should be returned as an output parameter back to the application.



To send cursor as an output parameter from a SP we need to use Ref Cursors, because Ref Cursor is a type (user defined), which can be used as a procedure parameter, so first we need to define a Ref Cursor type and that should be used as an Output parameter to the SP. We define Ref Cursor types in Oracle Database as following:

Type <type name> is Ref Cursor;

eg: Type DynaCur is Ref Cursor;

To define the SP with Ref Cursor Output Parameter first we need to define the Ref Cursor under PL/SQL known as Package which is a group of Procedures, Functions, Variables and Sql Statements created as a single unit used for storing of related objects. A package has two parts:

1. Package Specification or Spec or Package Header

2. Package Body

Package Specification acts as an interface to the package which contains only declaration of types, variables, constants, exceptions, cursors and sub-programs. Package Body is used to provide implementation for the sub-programs, queries for the cursors declared in the package specification.

Now let us define a package header with a Ref Cursor type definition and also 2 SP's for selecting data from DEPT and EMP tables using the Ref Cursor type as an Output Parameter and then implement the SP's under the Package Body.

Step 1: Defining Package Specification:

Create or Replace Package MyPackage

As

Type DynaCur is Ref Cursor;

Procedure Select_Emp(ecur out DynaCur);

Procedure Select_Dept(dcur out DynaCur);

End MyPackage;

/

Step 2: Defining the Package Body:

Create or Replace Package Body MyPackage

As

Procedure Select_Emp(ecur out DynaCur)

Is

Begin

Open ecur For Select * From Emp;

End Select_Emp;

Procedure Select_Dept(dcur out DynaCur)

Is

Begin

Open dcur for Select * From Dept;

End Select_Dept;

End MyPackage;

/

Step 3: Displaying the data by loading it into the application:

Now take a new windows form place a SplitContainer on it, set its Orientation property as Horizontal, place a DataGridView control on each panel by setting their Dock property as fill and write the following code: using System.Data.OracleClient;

Declarations: OracleConnection con; OracleDataAdapter da1, da2; DataSet ds;

Under Form Load: string constr = ConfigurationManager.ConnectionStrings["OConStr"].ConnectionString;
con = new OracleConnection(constr); da1 = new OracleDataAdapter("MyPackage.Select_Emp", con);
da1.SelectCommand.CommandType = CommandType.StoredProcedure;
da1.SelectCommand.Parameters.Add("ecur", OracleType.Cursor).Direction=ParameterDirection.Output;
da2 = new OracleDataAdapter("MyPackage.Select_Dept", con);
da2.SelectCommand.CommandType = CommandType.StoredProcedure;
da2.SelectCommand.Parameters.Add("dcur", OracleType.Cursor).Direction=ParameterDirection.Output;
ds = new DataSet(); da1.Fill(ds, "Emp"); da2.Fill(ds, "Dept");
dataGridView1.DataSource = ds.Tables["Dept"]; dataGridView2.DataSource = ds.Tables["Emp"];

Crystal Reports

Crystal Reports is a business intelligence application used to design and generate reports from a wide range of data sources. Microsoft Visual Studio bundles an OEM version of Crystal Reports as a general purpose reporting tool which became the de facto standard report writer when Microsoft released it with Visual Studio.

The product was originally created by Crystal Services Inc. as Quik Reports when they could not find a suitable commercial report writer for their accounting software. After producing versions 1.0 through 3.0, the company was acquired in 1994 by Seagate Technology. Crystal Services was combined with Holistic Systems to form the Information Management Group of Seagate Software, which later rebranded as Crystal Decisions, and produced versions 4.0 through 9.0. Crystal Decisions was acquired in December 2003 by Business Objects, which has so far produces versions 10, 11 and current version 12. Business Objects was acquired by SAP on Oct 8, 2007.

To add report under a project open "Add New Item" window, select Reporting in LHS panel, then select Crystal Report from RHS panel, which adds the report file, with an extension.rpt. A report file by default has 5 sections in it when opened. Those are:

- | | | | | |
|------------------|----------------|------------|------------------|----------------|
| 1. Report Header | 2. Page Header | 3. Details | 4. Report Footer | 5. Page Footer |
|------------------|----------------|------------|------------------|----------------|

Report Header: content placed under this section gets displayed on top of the report i.e. on top of the first page in report. We use it for placing contents like Company Name, Logo, and Address etc.

Page Header: content placed under this section gets displayed on top of every page into which the report extends. We use it for placing contents like Column Names, Date and Time when the report is generated etc.

Details: content into this section generally comes from Databases; this was the actual information that has to be presented or displayed to the clients.

Report Footer: this was same as Report Header but gets displayed on bottom of the report. We use it for placing contents like Date, Place, and Signatures etc.

Page Footer: this was same as Page Header but gets displayed on bottom of every page into which the report extends. We use it for placing contents like Page No's, Phone No's, and Fax No's etc.

Creating a Report:

Open a new project of type Windows, name it as "ReportsProject", open add new item window, select Reporting on LHS, choose Crystal Report on RHS, name the report as Employee.rpt and click Add button which opens "Crystal Report Gallery" window asking to choose how we want to design the report select "As a Blank Report" Radio Button and click Ok. Once the report is added to the Project all the required assembly references are also implicitly added, check them in solution explorer under references node.

As the report has to get its information from DB first we need to configure the report with appropriate Data Source using the "Field Explorer" window that gets added along with your Report.

Configuring the Report with DB: open the Field Explorer, right click on DB Fields Node, select 'Database Expert' which opens a window, expand Create New Connection node, double click on 'OLE DB (ADO)' node that displays a list of providers, choose a provider for Sql Server, click next, specify the connection details like server name, user id, password and DB, click Finish which adds a node under OLE DB (ADO) with your Server Name expand it and under it expand your database, "dbo", "Tables", double click on the table we want to consume, e.g.: Employee which adds the table under selected tables list on RHS, click ok which adds Table under DB Fields node of Field Explorer.

Designing the Report:

Go to Report Header section right click in it and select Insert Text Object that adds an object like a Label, you can enter content in it and do the required alignments using the format toolbar on top.

Go to Page Header section right click on it, select Insert, Special Field, "DataDate" place it on LHS of the section, right click on it select format object and choose the format in which we want to display the date, in the same way select "DataTime" and place it on RHS.

Now open Field Explorer and expand the table Employee we have selected, that displays the list of columns below, drag and drop each column on to details section as per your design which will create 1 Field on the Page Header (Column Name) and 1 Field on the Details (Data), do all the required alignments to header and details.

Now go to Report Footer and provide the option for users to enter Date, Place and Signature using the Text Objects and Insert Line Options.

Now go to Page Footer section right click Insert, Special Field, select Page Number and place it on the Center of the section.

Launching the Report:

To display report in a windows application use "CrystalReportViewer" control. You find the control in reports tab of ToolBox. To show a report under the control we need to assign path of the report to controls "ReportSource" Property. Now place a CrystalReportViewer control on form which by default uses dock property value as fill and write the following code under Form Load Event:

```
crystalReportViewer1.ReportSource = <path of the report>;
```

Note: to get the path of your report go to Solution Explorer, right click on report file and select properties, copy the value under "FullPath" property and use it.

Run the project to watch the output but first it will prompt for the password confirmation so enter password. The reason why it asks for password is because right now when we launch the report Crystal Report Viewer needs to connect with the Database for getting the information, so it needs all connection details. If we want to launch the Report without prompting the end user for connection details we need to supply all that information to Crystal Report Viewer control explicitly from code with the help of 3 classes ConnectionInfo, TableLogOnInfo and TableLogOnInfos which are present under the namespace CrystalDecisions.Shared. Now rewrite the code again in Form1 under Load Event as following:

using CrystalDecisions.Shared;

```
ConnectionInfo ci = new ConnectionInfo();
ci.ServerName = "<Server Name>" ci.UserID = "Sa"; ci.Password = <pwd>; ci.DatabaseName = "<DB Name>";
TableLogOnInfo ti = new TableLogOnInfo(); ti.ConnectionInfo = ci; ti.TableName = "Employee";
TableLogOnInfos tis = new TableLogOnInfos(); tis.Add(ti);
crystalReportViewer1.ReportSource = "<path of the report>"; crystalReportViewer1.LogOnInfo = tis;
```

Note: Maintain the Server Name, User Id, Password and Database Name attributes under the Configuration File and then read them into the application to make it more dynamic.

Selection Based Reports: these are reports which get generated based on the selections made by users which may get the data from single or multiple tables. Creating a selection based report using EMP and Dept tables of Oracle.

Step 1: Add a new form in the project "Form2.cs" and add reference of System.Data.OracleClient.dll to the project. Design the form as following and write the below code.

using System.Data.OracleClient;

Declarations: OracleConnection con; OracleDataAdapter da; DataSet ds;

Under Form Load: `con = new OracleConnection("User Id=Scott;Password=tiger");`
`da = new OracleDataAdapter("Select Empno From Emp", con); ds = new DataSet(); da.Fill(ds, "Emp");`
`comboBox1.DataSource = ds.Tables[0]; comboBox1.DisplayMember = "Empno";`

Under Show Report Button: write this code after creation of second form i.e. Form3.

`Form3 f = new Form3(); f.Empno = int.Parse(comboBox1.Text); f.ShowDialog();`

Step 2: Designing the Report, add a new Blank Crystal Report under the project and configure the report with DB:

Open the field explorer, right click on database fields, database expert, create new connection, OLE DB (ADO), Microsoft Oledb Provider for Oracle, click next, provide the connection information, click Finish. Adds a node under OLE DB (ADO) as Connection or Server Name, double click on "Add Command" node under connection or Server Name, opens a window where we can write a Select Statement on LHS TextArea.

Parameter Fields: these are used for sending values to a report for execution to make the report more dynamic where those values can be used for displaying title, address, phone no's and still values to queries to execute etc.

Our select stmt should execute basing on the Empno selected under ComboBox of above Form which should be sent as a parameter to report, so first we need to create a parameter and then use it under the select stmt. To create a parameter click on "Create" button which opens a window under which we need to specify following details:

1. Parameter Name: Empno
2. Prompting Text: Enter Employee No.
3. Value Type: Number
4. Default Value: 7566 (optional)

Click ok which adds the parameter on RHS ListBox, now in LHS TextBox write following Sql Stmt:

`Select E.Empno, E.Ename, E.Job, E.Mgr, E.HireDate, E.Sal, E.Comm, D.Deptno, D.Dname, D.Loc From Emp E`
`Inner Join Dept D On E.Deptno=D.Deptno Where E.Empno = {?Empno}`

Click Ok, Ok, Ok which will add the Command under Database Fields Node, if we expand it will display the columns we mentioned under Select Stmt.

Designing the Report:

We can also Add Parameters using Field Explorer window, to create a Parameter right click on the field Parameters and Select Add which will prompt for Name of Parameter, enter a name and click ok. Following the above process create 2 parameters CompanyName and Address, now drag and drop them on ReportHeader and do the necessary alignments.

Place DataDate and DateTime Fields on PageHeader section and do all the necessary alignments by settings the formats.

Drag and drop Columns under Command into the details section and pull back column names from Page Header to Details section.

Right click on the ReportFooter section and select suppress, so that our report doesn't have report footer. Any section can be suppressed like this if not required.

Place a PageNumber Special Field on PageFooter section and do the necessary alignments, if required we add any special fields or parameters fields here also.

Step 3: to launch the above report add a New Form under the project (Form3), place a CrystalReportViewer control on the form. Go to "Application Configuration File" i.e. app.config and add the following code inside configuration tag.

```
<appSettings>
  <add key="Cname" value="NIT Technologies"/>
  <add key="Address" value="Ameerpet, Hyderabad"/>
</appSettings>
```

In this example we need to launch the report as well as send few parameter values to the report for execution, to send values to parameters of report we need to use the class ReportDocument which is present under CrystalDecisions.CrystalReports.Engine namespace. Now write the following code under Form:

using System.Configuration; using CrystalDecisions.CrystalReports.Engine; using CrystalDecisions.Shared;

Declarations: internal int Empno;

Under Form2 Load:

```
string Cname = ConfigurationManager.AppSettings.Get("Cname");
string Addr = ConfigurationManager.AppSettings.Get("Address");
ConnectionInfo ci = new ConnectionInfo();
ci.ServerName = "<Server Name>"; ci.UserID = "Scott"; ci.Password = "tiger";
TableLogOnInfo ti = new TableLogOnInfo(); ti.ConnectionInfo = ci; ti.TableName="Command";
TableLogOnInfos tis = new TableLogOnInfos(); tis.Add(ti);
ReportDocument obj = new ReportDocument(); obj.Load("<path of the report>");
crystalReportViewer1.ReportSource = obj; crystalReportViewer1.LogOnInfo = tis;
//Sending values to parameters of report:
obj.SetParameterValue("CompanyName", Cname);
obj.SetParameterValue("Address", Addr); obj.SetParameterValue("Empno", Empno);
```

Note: using the SetParameterValue method of ReportDocument class we can pass values to Parameter Fields of report: SetParameterValue(string pname, object value) or SetParameterValue(int index, object value)

Cross Tab Reports:

These are reports that are specially provided for summarizing the data or analysis of data. To create this type of report add a new report under the project and when the Crystal Report Gallery window opens select the Radio Button "Using the Report Wizard", now in the bottom we find and option "Choose an Expert" Select "Cross Tab" in it and click ok, which opens "Cross-Tab Report Creation Wizard".

In the wizard expand the node "Create New Connection", double click on the node OLE DB (ADO), which opens a new window with a list of providers, select the provider for Oracle, Click Next, which opens a new window for Connection Information, enter the connection details in it and click finish which adds a new node under OLE DB (ADO) with the name as Server or Connection, expand it we will find a list of users, expand the user "Scott", expand the node tables, double click on the "Emp" table node which adds the table under "Selected Tables" List Box on RHS, Click Next which displays all the columns corresponding to the Emp table asking for the columns to be added into the report, select Job and add to Columns List Box, select Deptno and add to Rows List Box, select Sal and add to Summary Fields List Box, now in the below Combo Box select Sum and click Next which will ask for including a Chart into the report for summarizing the report, select Pie Chart Radio Button, provide a title to the chart or leave the same, under "On change of" select the column Deptno, under "Subdivided by" select Job, under "Show Summary" select Sum of Emp.Sal and click next which will ask for "Record Selection" using which we can set filters to reports data or else it is optional, so leave it and click Next which displays for Grid Style, select the style we want and click Finish which display the report with sample data, make any changes if required or add any required special fields into the report. Now to display the report in our application add a new windows form in the project "Form4.cs", place a Crystal Report Viewer Control on it and write the following code under load event:

using CrystalDecisions.Shared;

```
ConnectionInfo ci = new ConnectionInfo();
ci.ServerName = "<Server Name>"; ci.UserID = "Scott"; ci.Password = "tiger";
TableLogOnInfo ti = new TableLogOnInfo(); ti.ConnectionInfo = ci; ti.TableName="Emp";
TableLogOnInfos tis = new TableLogOnInfos(); tis.Add(ti);
crystalReportViewer1.ReportSource = "<path of the crystal report>"; crystalReportViewer1.LogOnInfo = tis;
```

