C# .Net

23/12/13

→ What is software app?

Software application is collection of software prgms which are developed by some ~~by some~~ software technologies to fulfill the ~~enduser~~ requirement.

Ex: MS-office, Gmail.com

→ what is Software prgm?

It is collection of logical instructions to the computer

Ex: printf, scanf. (logical instructions.)

→ Example for Software Technologies

.Net ; Java, PHP.

→ Example for Endusers.

all gmail users, all MS. office users

Types of software applications.

→ software app are classified into various types based on application behaviour.

→ Some of they are

1. Windows applications → Ex: MS-office
2. Web application → Ex: Gmail.com, facebook etc..

24/12/13

What is .Net

→ .Net is a software technology which was introduced by Microsoft in the year 2002.

→ MicroSoft is a leading organisation, which was established by Bill Gates in U.S.A.

→ .Net is a frame work technology which is integrated with multiple technologies like below.

1. .Net windows technology (WPF) (Window presentation foundation) (Windows Forms)
2. web technology. (ASP.Net)
3. web service technology (WCF) → windows communication Foundation
4. Mobile technology

⋮

→ .Net is integrated with multiple technologies, due to that reason using single .Net a programmer can develop various type of software applications like below

1. windows applications
2. web applications
3. web services
4. Mobile applications
5. windows Service ---

→ .Net is supporting for multiple programming languages which are called as .Net languages they are

1. C# .Net
2. VB .Net
3. VC++ .Net
4. VJ# .Net
5. VF# .Net
6. C++ .Net ---

→ To develope any type of application using .Net we require one .Net technology and one .net language for Eg if you want to develop a windows application we have to use windows forms and C# .Net.

→ If you want to develop web application we have to use ASP.Net and C#.Net.

→ If we want to develop web Service. we have to use WCF and C#.Net.

26/12/13 **Histoly of .Net (&) Challenges faced by Microsft to Introduce .Net :-**

→ Before .Net technology i.e., from 1990-98 under Microsft family we have 3 popular technologies. They are

1. VB (Visual Basic)
2. VC++ (visual C++)
3. ASP (Active Server Pages)

→ These technologies are having some of the drawback as beln.

1. Platform dependency :-

→ These technologies will work only on windows operating system.

2. Not supporting to develop multi type of app :

→ Using VB and VC++ we can develope only windows applications.

→ Using ASP we can develop only web applications.

→ At the same by time in the yr Sun Microsystem has introduced a technology called JAVA.

→ JAVA is a platform Independent technology and using JAVA we can develop various types of sftware applications.

→ Because of above advantages in JAVA, Microsoft existing technology clintes are attracting towards JAVA.

→ To overcome this microsft has anounced abt .Net first time in the year 1998 in the meeting called PDC (Professional Developer Conference)

→ In PDC Microsoft has announced about .Net features like below.

1. .Net will be platform Independent technology.
2. .Net is supporting to develope various type of applications.
3. .Net ~~to~~ will be supporting for multi languages.

→ From 1998 – 2002 Microsoft R&D team has designed .Net technology and finally they have released first version of .Net i.e., .Net 1.0 in the year 2002.

27/12/13

Versions of .Net technology:

I st release — .Net frame work 1.0 → 2002.
II nd release — " " 1.1 → 2003
III rd release — " " 2.0 → 2005
IV th release — " " " 3.0 → 2006
V th release — " " 3.5 → 2008
VI th release — " " 4.0 → 2010
VII th release — " " 4.5 → 2012.

What is .Net frame work:

→ .Net frame work is an important integral component in software, when we instal .Net software this component will be installed directly.

Definition:

→ .Net frame work is a common platform for developing various types of applications by using .net technologies and .Net languages.

→ .Net frame wrk is integrated with 4 items as below.

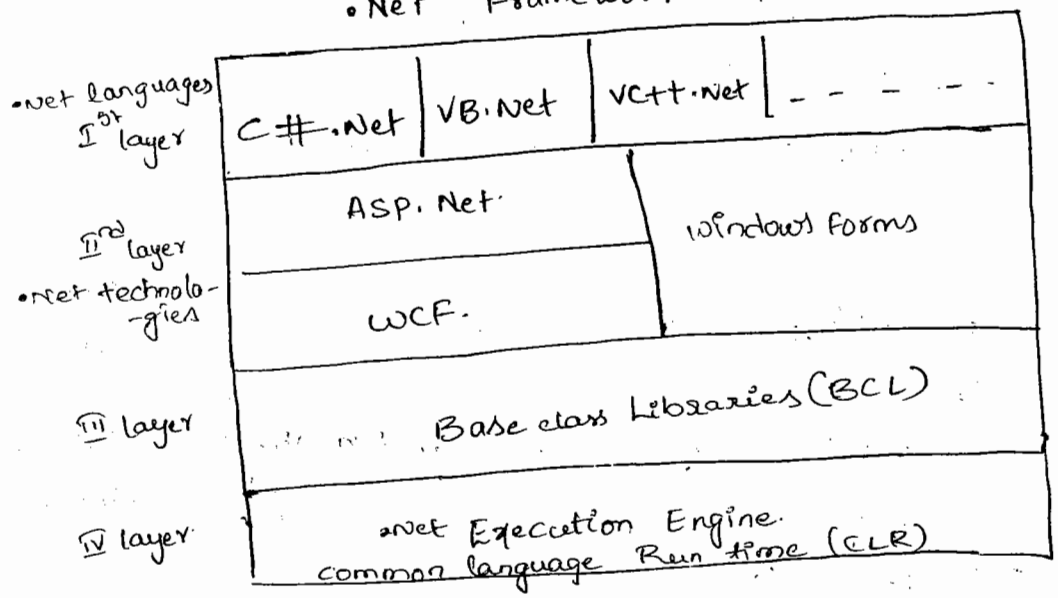
1. .Net languages
2. .Net technologies
3. .Net baseclass libraries
4. .Net exicution Engine.

28/12/13

.Net frame work Architecture Diagram:

.Net frame work will be divided into 4 layers

.Net Framework Architecture.



AWD windows app in Java.

JVM → Exicution Engine of Java.

→ Layer 1 is representing .Net languages

→ To develope any type of application by using .Net, .Net programmer require one .Net language i.e., C#.Net (or) VB.Net.

→ Layer 2 is representing .Net technologies.

1. Windows Forms :-

~~2. ASP.Net~~ → it is a .Net windows technology

→ using windows forms we can develove a windows application

~~→ windows~~ application ~~can be~~

→ using .Net if we want to develope windows forms we have to use .Net windows technology and C#.Net.

2. ASP.Net :-

→ It is a .Net web technology.

→ By using ASP.Net we can develope a web application

→ In .Net if we want to develop a web application, we have to use .Net web technology called ASP.Net and .Net language called C# .Net language.

3. WCF :-

→ WCF stands for "windows Communication Foundation".

→ It is a .Net web service technology.

→ In .Net if we want to develope a web service by using .Net we have to use WCF and C#.Net.

30/12/13

→ Layer 3 :-

→ Layer 3 is representing Base class libraries

* What is a class?

→ class is a collection of variables and methods

→ classes are 2 types.

1. Pre defined class
2. User defined class

Pre defined class :

A class which is defined by Microsoft programmer can be called as predefined class.

User defined class :

A class which is defined by the .Net programmer can be called as user defined class.

What is

→ What is class Libraries?

→ class library is a collection of classes

→ class libraries are 2 types

1. Predefined class libraries
2. User defined class libraries.

Pre-defined class libraries:

→ A class library which is defined by the microsoft programmer can be called as pre-defined class libraries.

→ pre defined class library will contain collection of pre-defined classes.

→ pre defined class libraries are called as Base class Libraries, which are providing by the Microsft and using by the .Net programmer.

How to use Base class Library:

With the help of 'Using' keyword like below.

code window

```
Using System;
↓      ↓
keyword  Base class library
```

User defined class Libraries:

→ A class library which is defined by .Net programmer can be called as user defined class library.

Note: Base class libraries can be called as .Net Frame work class Libraries.

Layer 4:

→ Layer 4 is representing .Net exicution Engine i.e. CLR

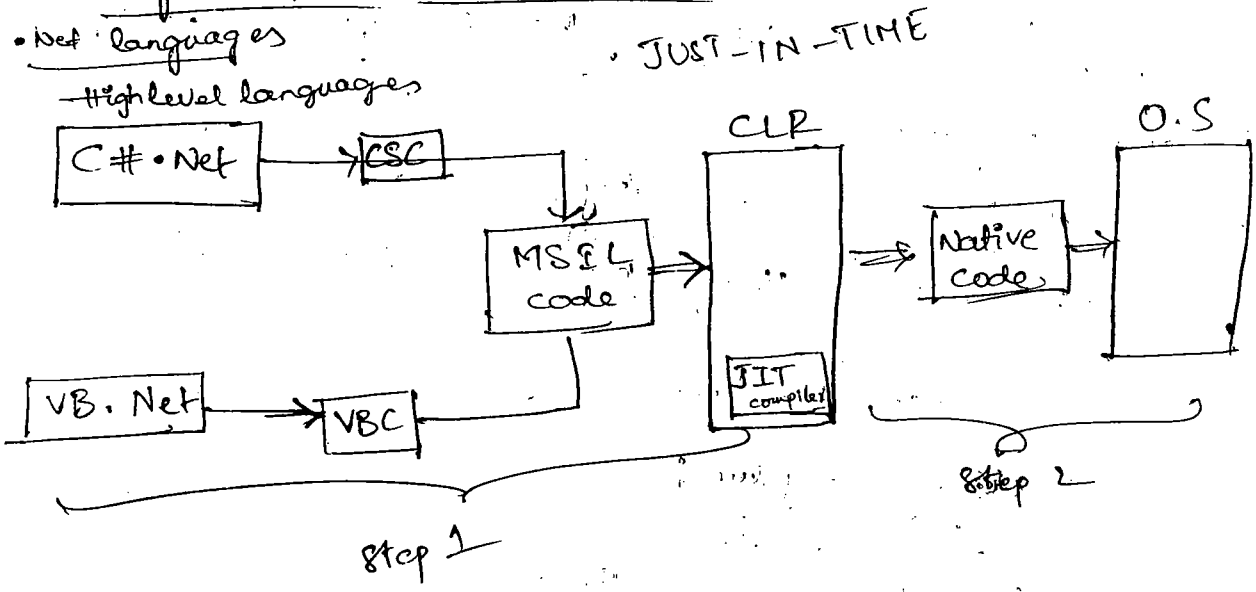
→ CLR is a common execution engine for all .Net languages.

→ Every .Net language developed application has to execute with the help of CLR.

31/12/13

Execution process of .Net application:-

Diagram for .Net application Execution process



.Net application process will be divided into 2 steps

Step 1:

* Language compilers will be converting high level language code (C#.Net ~~compiler~~ code) in to MSIL code (Micro soft Intermediate language).

* Becoz .Net execution engine can understand only MSIL code.

Step 2:

→ JIT compiler [Just In Tym] will convert MSIL code to Native code becoz operating System can understand only native code.

01/01/14

.Net Languages:

→ .Net is supporting for multiple programming languages which are called as .Net languages (or) .Net supporting languages

→ These .Net languages ~~are~~ classified into two types like below

.Net Languages

- Microsoft .Net Languages
- Non-Microsoft .Net Languages

1. Microsoft .Net Languages:

→ Microsoft .Net languages are the languages which are introduced by Microsoft Organisation.

Ex: C# .Net
VB. Net
VC++. Net
C++ . Net

→ Microsoft .Net languages compilers are inbuilt by the microsoft software. So the .Net programmer is not in need to install them again.

2. Non - Microsoft .Net languages:

→ Non - Microsoft .Net languages are the languages introduced by other than Microsoft organisation. These are Non-Micro soft .Net languages (or) third play .Net languages (or) Vendor .Net languages.

Ex: cobal .Net
Pascal .Net
Perl .Net
Delphi .Net

→ These language compilers are not inbuilt by Microsoftware.

→ That means whenever a programmer wants to use non - Microsoft .Net languages, programmer has to purchase the language compiler from the concern organisation website through online.

2/1/14

Types of Software applications:

→ Software applications are classified into various types based on application behaviour.

1. Console application
2. windows application.
3. web application.
4. web Service
5. windows Service
6. Mobile application - - - -

1. Console application:

→ Console application is a single user application

→ To consume this application end user doesnt require internet conectivity

→ No user Interface i.e., input and out put will be within the comand prompt window.

→ will not have real time applications but, to learn C# .Net we are going to develope these applications.

→ To develope console applications by using .Net we have to use console technology and C#.Net

2. Windows applications:

→ Windows applications can be called as desktop applications (or) standard alone applications (or) GUI based application (graphical user Interface) (or) win forms applications & so on - - -

→ Single user applications

→ To consume windows application enduser doesnt require internet conectivity.

→ It is User interface will be there which is called as windows forms.

→ application should be installed at end user machine.

→ when we will develop windows applications?

Ans: whenever an application should be available for single user at a tym we will go for windows applications

Ex: father business accounts applications, Mobile Shoppee day to day transactions, clg library Management, Medical store stock application, MS office.

→ To develope a windows application by using a .Net, we have to use .Net windows technology called windows forms & .Net language called C# .Net.

3/1/14

3. web application:

→ Web applications are web enabled applications.

→ Web applications are multi user applications.

→ web application will have a user interface. which can be called as web page.

→ web application should be installed at remote machine called web server.

⇒ when we will go for web application?

Ans: whenever an application should be available for single or multiple users at ~~that~~ a tym we will go for web application

Ex: www.facebook.com
www.gmail.com

→ To develope a web application by using .Net we have to use ASP.Net and a language called C#.Net

comparision between Console application and windows application:

Console application	windows application
→ Single user	→ single user
→ No user interface	→ Interface is there which is called windows forms
→ No internet conectivity	→ No internet conectivity.
→ No real time application	→ we will develop Real time applications.
→ Light weight applications	→ Heavy weight application.

Comparision b/w windows and web applications :

windows Application	web application.
→ Single user	→ Multi User.
→ user Interface is there called windows forms	→ user interface is there called web page
→ Internet connectivity is not needed	→ Internet conectivity is required
→ To develop we have to use windows forms and C#.Net	→ To develop we have to use ASP.Net and C#.Net.
→ whenever the application should be available for single user we go for windows application. Ex: MS office	→ whenever the application should be available for multi user we go for web application. Ex: Gmail.com facebook.com

6/1/14

CLR Architecture :

→ CLR is part of .Net frame work.

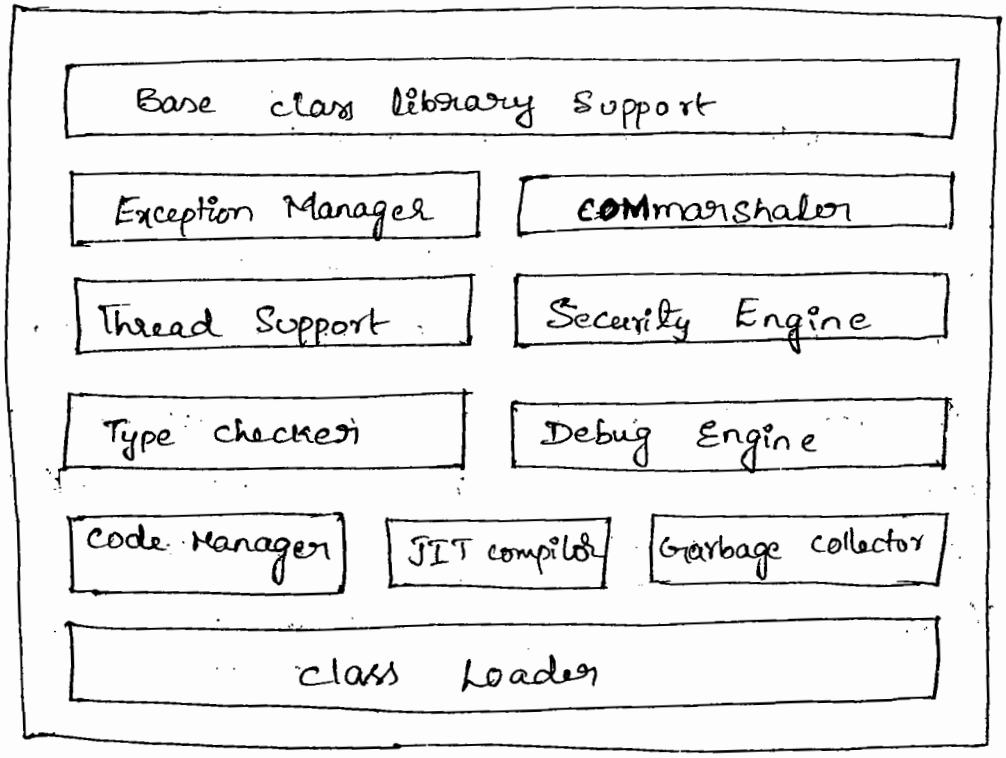
→ .Net frame work is a part of .Net Software

→ The main responsibility of CLR is executing .Net applications, it will provide some other services such as code management, memory management, exception handling support, Multi threading Support, Security, debugging and so on..

Diagram for CLR architecture :-

→ CLR architecture is representing the responsibilities of the CLR in .Net

CLR Architecture.



⇒ **Class Loader :**

→ It loads classes from application to CLR one by one according to execution requirement.

⇒ **Code Manager :**

→ which Manages the code during execution of the application.

⇒ **JIT Compiler :** (Just-In-Time)

→ which converts MSIL code to Native code.

⇒ **Garbage collector :**

→ In .Net, memory management is performing by garbage collector.

→ To perform this memory management garbage collector

is doing 2 duties.

a) allocating memory :-

→ when an object is creating by the application garbage collector will allocate memory for that particular object within the heap datastructure.

b) Deallocating memory :

→ when an object is not using by the application garbage collector will recognize particular object as ~~unus~~ unused object and it will destroy that unused object.

→ To allocate memory and to de-allocate memory programmer need not to write single line of code because which is internal process. Due to that reason a .Net memory management is called as "Automatic memory management"

07/1/14

⇒ Exception Manager :

→ Error are two types

1. Compile time error
2. Run time error

Compile time error :

→ an ~~w~~ error which is occured at the time of compilation is called compile time error.

Ex : Syntax errors.

Run time error :

→ An error which is occured at the time of executing Program is called as run time error.

Ex :

→ Run time error will occur because of invalid input or improper logic.

→ what is an exception?

Ans: An exception can be defined as runtime error.

→ what is exception handling?

Ans: Exception handling is a mechanism to handle runtime errors by using try, catch and finally blocks.

→ Purpose of Exception handling?

Ans: To avoid the abnormal termination when the error is occured

→ Exception manager will provide exception handling facility for .Net applications.

Thread Support:

→ what is a thread?

Ans: Thread is an independent execution path, it able to run simultaneously with other execution paths.

→ what is multi threading?

Ans: Inplementing multiple execution paths simultaneously is called as multi threading.

→ what is purpose of multi threading?

Ans: To improve the performance of application will go for multithreading.

→ Thread Support is providing multi-threading facility for .Net application.

Type checker:

→ which provides strict data type checking.

COM Marshaler :-

→ COM Stands for "Component Object Model."
→ COM is a microsft old traditional technology.
→ Using COM we can develop reusable components (DLL files).
→ COM marshaler allows .Net applications to consume COM components.

08/1/14.

Security Engine:

→ which provides Security to .Net applications.

Debug Engine :

→ Which provides Various types of debugging facilities to .Net application.

Base class Library Support:

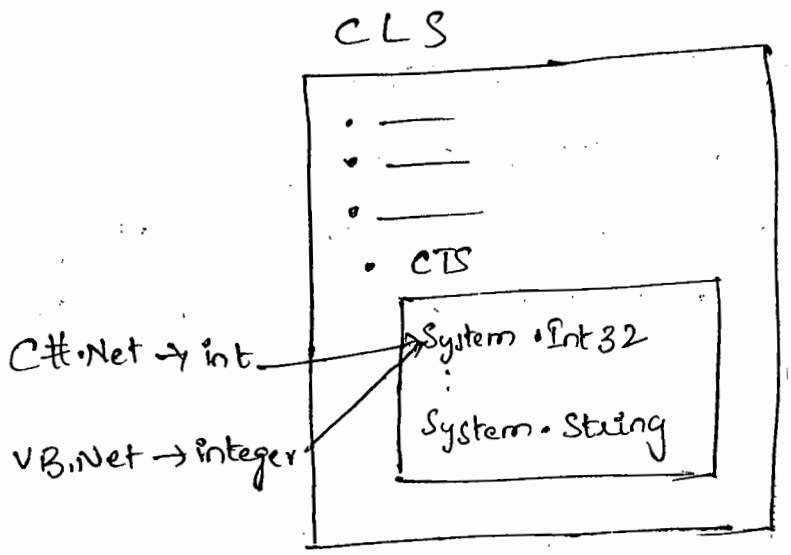
→ which provides predefined class library Support to .Net applications.

CLS (Common Language Specification):-

→ CLS is a set of common language standards defined by micro soft for all .Net languages.
→ whenever a programming language wants to recognize as .Net language it has to follow CLS Standards.

CTS (common Type System) :-

→ CTS is a Subset of CLS.
→ CTS is a Set of common base datatypes defined by microsoft for all .Net languages.
→ Every .Net language data types will have base data types in CTS.
→ Ex: System.int32 is the CTS type which is base type for VB.Net integer and C#.Net int like below.



C#

9/01/14 C# . Net

→ what is C#. Net ?

Ans It is a .Net language.

→ why C#. Net ?

Ans: whenever we want to develop any type of Software application .Net programmer require one .Net language.

→ what type of programming language is C#. Net ?

Ans: C#. Net is a object oriented Programming language

Features of C#. Net :

→ C#. Net is a case sensitive language

Ex: int a ;

A = 10 ; // error

→ It supports block level programming.

Ex: for block :-

```
{
  St 1;
  St 2;
}
```

→ every statement should end with semicolon.

Visual Studio .Net :

→ Visual Studio .Net is a predefind tool providing by Microsoft with .Net Soft ware.

Visual Studio .Net
.Net Frame work

→ Visual Studio .Net is providing development environment for .Net Programmers to develop various types of applications by using .Net technologies and .Net languages

→ .Net frame work is a runtime environment which is required to run the .Net application.

Order to Install .Net Software.

1) Install SQL Server
2) Configure Install IIS.
3) Install .net Software.

How to create a new console application?

⇒ Start → programs → Micro Soft visual studio 2010 → visual Studio 2010

⇒ It will open visual Studio IDE [Integrating developmet environment] enter

⇒ Click on New Project

⇒ It will open new project window

⇒ here Select languages visual C# and type of application as Console application and rename it as, myconsole application, location as D:\. Then click OK.

⇒ Console application development environment will have mainly 2 windows.

1) Solution Explorer window
2) Code window

1) Solution Explorer window:

→ This window will display all the project related files

→ By default it will come with one file called program.cs

Program.cs

→ program.cs is a C#.Net class file.

→ Every C#.Net class file extension will be .cs and VB.Net class file extension will be .vb

What is class file ?

Ans A class file can contain collection of classes, by default every class file will come with a single class.

→ Program.cs file will come with a single class called Program and program class will come with a special method called Main method.

Main ( ) :-

→ Main is a special type of function, console application execution starts by main(), controlling by main() and will stop by main().

→ This main() is also controlling by .Net execution engine called CLR.

2) Code Window :-

→ This window will display the selected class file code.

Structure of program.cs file:

```
Keyword   Base class library
using System;
Keyword
namespace MyConsoleApplication;
                  → project name
{
    Key word
    class Program   → class name
    {
        Key word   Key word   → Method name
        static void Main (String [] arg)
        {
        }
    }
}
```

1) write a console programe to print welcome msg.

```
using System;
using
namespace MyconsoleApplication;
{
    class Program
    {
    Static Void Main (String[ ]args)
        {
            Console. WriteLine ("welcome to C#.Net");
            Console. ReadLine ( );
        }
    }
}
```

o/p

welcome to C#.Net

Compiling the program :-

F6

Executing the program:

F5

→ write line is a predefined method.
→ Read line is a predefined method.
→ console is predefined class
→ class program → is user defined class
→ static void Main → user defined. but it is not be change becoz it
→ System → it is a base class library.

2) write a console prgm to print two welcome msgs?

o/p

```
welcome to C#.Net
welcome to console Appl.
```

Prgm.

```
Static void Main( )
{
console.WriteLine(" welcome to C#.Net ");
Console.WriteLine(" welcome to console Appl");
console.ReadLine( );
}
```

3) write a console prgm to print 3 welcome msgs

o/p

```
welcome to C#.Net
welcome to console Appl.
welcome to satya.
```

Prgm

```
Static void Main( )
{
console.WriteLine(" welcome to C#.Net");
console.WriteLine(" welcome to Console Appl");
Console.WriteLine(" welcome to Satya");
Console.ReadLine( );
}
```

System :

→ It is a base class library.

Console :

→ Console is a predefined class which is a part of System Base class library

Write Line ( ) :-

→ Write Line is a predefined member method of Console class.

→ Write Line method will print the given value, after printing the value, it will move the cursor to the next line.

Read Line( ) :- It holds the comand Prompt window untill we Press an enter key.

4) Write a prgm to print one string value and one numerical value?

Prgm

```
Static Void Main ( )
{
  Console . Write Line (" rama");
  Console . Write Line (" 25");
  Console . Read Line( );
}
```

O/P

```
rama
25
-
```

Variable:

Variable is representing a value which can be changed.

Syntax to declare a variable:

<data type> <variable name> = <value>;

Ex: int a = 10;
String s = "Sathya";

5) Write a Console program to declare 1 int variable and 1 string variable?

Prgm

```
Static void Main()
{
  ~~String s = "rama";~~
  int age = 25;
  String name = "rama";
  Console.WriteLine(name);
  Console.WriteLine(age);
  Console.ReadLine();
}
```

O/P
rama
25
—

6) ~~Wrte~~ Example to display variable values with userdefined String?

O/P
your name is : rama
your age is : 25
—

Prgm

```
Static void Main()
{
  ~~int~~ String a = "your name is : rama";
  string b = "your age is : 25;
```

Prgm

```
Static Void Main()
{
  int age = 25;
  String name = "rama";
  Console.WriteLine("Your Name is:" + name);
  Console.WriteLine("Your Age is:" + age);
  Console.ReadLine();
}
```

(user defined string; concatenation operator; variable)

7) Write a Console program to display employee info. ie., employee no, employee name, Salary?

Prgm

```
Static void Main()
{
  int enumber = 9981263440;
  int esalary = 50000;
  String ename = "Amrutha";
  Console.WriteLine("Employee name is:" + ename);
  Console.WriteLine("Employee number is:" + enumber);
  Console.WriteLine("Employee Salary is:" + esalary);
  Console.ReadLine();
}
```

O/P

Employee name is: Amrutha.
Employee number is: 9981263440.
Employee Salary is: 50000

8) Write a Console program to display student info ie., Student ID, name, location?

Prgm

```
Static Void Main( )
{
  int stuID= 123;
  String stuname = "Pallavi";
  String stulocation= "Hyderabad";
  Console.WriteLine("Student ID is:" +stuID);
  Console.WriteLine("Student name is:"+stuname);
  Console.WriteLine("Student location is:" +stulocation);
  Console.ReadLine();
}
```

o/p

Student ID is: 123
Student name is: Pallavi
Student location is: Hyderabad.

Shortcut to open visual studio.Net

Start → Run → type as devenv → OK.

21/1/14 7) Write a console prgm to store student Marks. They are M1, M2, M3 & calcute total marks and average marks & finally display?

o/p

Total marks are : 190..
Average marks are : 63..

Prgm

```
void Main( )
{
  int m1 = 70;
  int m2 = 65;
  int m3 = 55;
  int totMarks = m1+m2+m3;
  int avg = totMarks/3;
  Console.WriteLin("Total marks are :" + totMarks);
  Console.WriteLine("Average marks are :" + avg);
  Console.ReadLine();
}
```

10) Write a console prgm to represent customer info i.e., customer ID, customer name, customer location, customer ph no, customer mail ID & display.

11) write a console prgm to store product information i.e., product ID, product name, product company, product cost & display?

HW

10) Prgm

```
void Main()
{
  int cusID = 123;
  String cusname = "sandhy";
  String cusloc = "kukatpally";
  double cusnum = 9848032211;
  String cusemail = "sandhya@gmail.com";
  Console.WriteLine("customer ID is :" + cusID);
  Console.WriteLine("customer name is :" + cusname)
```

```
console.WriteLine("Customer location is:"+cusloc);
Console.WriteLine("Customer number is:"+cusnum);
console.WriteLine("Customer emailID is:"+cusemail);
Console.ReadLine();
}
```

O/P.

Customer ID is : 123
customer name is : Sandhya.
customer location is: Kukatpally.
customer number is : 9848032211.
customer emailID is: Sandhya@gmail.com.

HW 11) -

Prgm

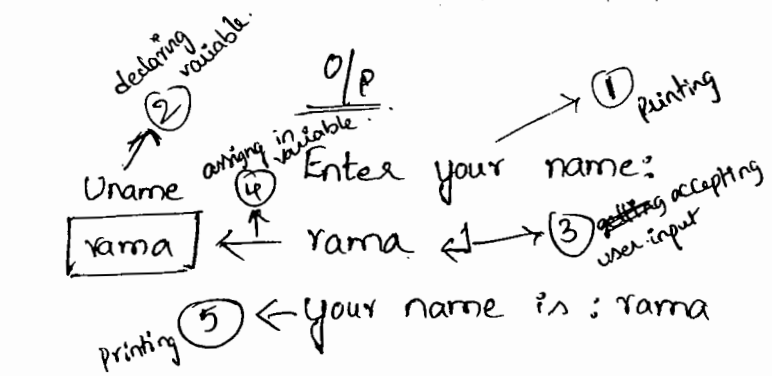
```
Void Main()
{
int proID = 123;
string proname = "TV";
string procom = "Sony";
int procost = 30000;
Console.WriteLine("Product ID is:"+proID);
console.Writeline("Product name is:"+proname);
console.WriteLine("Product Company is:"+procom);
console.Writeline("Product Cost is:"+procost);
console.Readline();
}
```

O/P

Product ID is : 123
Product name is : TV
Product company is : Sony
Product cost is : 30000.

22/1/14

12) Write a console program to accept user name & display



Prgm

```
Void Main( )
{
    Console.WriteLine ("Enter your name : ");
    String uname = console.ReadLine();
    Console.WriteLine ("your name is: "+uname);
    Console.ReadLine();
}
```

13) Write a console program to accept user name and user location & display to user.

O/P

Enter your name:

uname

rama ← rama.

Enter your location:

uloc

ameerpet ← Ameerpet

your name is : rama

your location is : Ameerpet

Prgm

```
Void Main ()
{
    Console.WriteLine("Enter your name:");
    String uname = Console.ReadLine();
    Console.WriteLine("Enter your location:");
    String uloc = Console.ReadLine();
    Console.WriteLine("your name is:"+uname);
    Console.WriteLine("your location is:"+uloc);
    Console.ReadLine();
}
```

14) Write a console prgm to accept user age and display to the user.

o/p

Ent

uage
25 ← 25

Enter your age:
your age is: 25

Prgm

```
void Main()
{
    Console.WriteLine("Enter your age:");
    int uage = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("your age is:"+uage);
    Console.Readline();
}
```

Console.ReadLine():—

→ ReadLine is a predefined member method of console class.

→ This method will accept input from the user, untill user will press an enter key, once user will press the enter key that accepted value this method will ~~b~~ written as String value becoz ~~Re~~ ReadLine() ~~written typ~~ return type is String.

Convert.ToInt32():-

→ Convert is a predefined class which is a part of System base class library.

→ ToInt32 is a predefined member method of Convert class.

→ This method will convert given value in to int datatype

22/1/14

15) Write a console program to accept employee no & display

O/P

Enter Employee number:

eno 123

123

your empno is: 123

Prgm

```
Void Main()
{
 Console.WriteLine("Enter Employee number:");
 int eno = Convert.ToInt32(Console.ReadLine());
 Console.WriteLine("your empno is:"+eno);
 Console.ReadLine();
}
```

16) Write a Console Prgm to accept user name, age and display.

O/P

String Enter your name:

uname

Sandhya ← Sandhya.

Enter your age:

int

uage ← 25

25

your name is: Sandhya.

your age is: 25

Prgm

```
void Main()
{
    Console.WriteLine("Enter your name:");
    Steing uname = Console.ReadLine();
    Console.WriteLine("Enter your age:");
    int uage = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("your name is:"+uname);
    Console.WriteLine("your age is:"+uage);
    Console.ReadLine()
}
```

17) Write a Console prgm. to accept employee number, name, Salary & display.

O/P

Enter emp number:

123

Enter emp name:

Amrutha.

Enter emp salary:

25000

Employee number is: 123

Employee name is: Amrutha

Employee Salary is: 25000

Prgm

```
Void Main( )
{
    Console.WriteLine("Enter empnumber: ");
    int eno = ~~ToInt32~~ Convert.ToInt32(Console.ReadLine());
    Console.WriteLine(" Enter empname: ");
    String ename = Console.ReadLine();
    Console.WriteLine ("Enter empSalary :");
    int esal = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Employe number is:"+eno);
    Console.WriteLine("Employe name is:"+ename);
    Console.WriteLine(" Employee Salary is:" + esal);
    Console.ReadLine();
}
```

18) Write a Console Prgm to accept Student name, stu 3 sub marks they are m1, m2, m3. and calculate the Student total marks and ~~average~~ display total marks with Student name. calculate Student avg marks & display avg marks of Student with name?

O/P

```
Enter Student name:
Pavithra.
Enter m1 marks:
70
Enter m2 marks:
80
Enter m3 marks:
75
Pavithra ~~Enter~~ total marks is:
Pavithra av.~~name~~ is:
```

Prgm

```
Void Main()
{
  console.WriteLine("Enter Student name:");
  ~~Console.~~
  String sname = console.ReadLine();
  Console.WriteLine("Enter m1 marks:");
  int m1 = Convert.ToInt32(console.ReadLine());
  Console.WriteLine("Enter m2 marks:");
  int m2 = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Enter m3 marks:");
  int m3 = Convert.ToInt32(console.ReadLine());
  int totmarks = m1+m2+m3;
  Console.Writeline("Sname + "total marks is:" + totmarks);
  ~~console~~
  int avg = totmarks/3;
  console.WriteLine(Sname+"average is:"+avg);
  console.ReadLine();
}
```

(a) Console prg train name & train no. ~~& arrival~~.

```
Void Main()
{
  console.WriteLine("Enter train name:");
  String tname = Console.ReadLine();
  Console.WriteLine("Enter train number:");
  int tno = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Train name is:"+tname);
```

```
Console.WriteLine("Train number is : "+tno);
Console.Readline();
}
```

O/P

Enter train name :
Amaravathi
Enter train number:
12344
Train name is : Amaravathi
Train number is: 12344.

Note :→ To move cursor to next line we have to use "\n". Always it should be in double cotes

→ To print the tab space we can use "\t".

24/1/14

20) Write a console prgm to accept 2 no's & perform the addition & display the addition result like below.

O/P

Enter the first num:
a [10] ← 10
Enter the second num:
b [5] ← 5
addition result is: 15.

c [15]

prgm

```
Void Main()
{
    Console.WriteLine("Enter the first num:");
    int a = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Enter the Second num:");
    int b = Convert.ToInt32(Console.ReadLine());
    int c = a+b;
    Console.WriteLine("Addition result is:"+c);
    Console.ReadLine();
}
```

21) Write a Console prgm to accept a number & calculate the Square of the given number & display.

O/P

Enter the number:
2
2 Square is: 4

prgm

```
Void Main()
{
    Console.WriteLine("Enter the number:");
    int a = Convert.ToInt32(Console.ReadLine());
    int b = a x a;
    Console.WriteLine("Square is:"+b);
    Console.ReadLine();
}
```

22) console prgm to find the cube of given no;

o/p

Enter the number:

2

2 Cube is : 8

prgm

```
Void Main()
{
    Console.WriteLine(" Enter the number: ");
    int a = Convert.ToInt32(Console.ReadLine());
    int b = axaxa;
    Console.WriteLine(+a" cube is: " +b);
    Console.ReadLine();
}
```

Datatypes :-

→ Datatypes specifies type of the data and size of the data.

→ In C#.Net datatypes are classified into 5 types based on type of the data and size of the data.

1. Numerical data types
2. Floating Point data types
3. Charater Related data types
4. Logical data types
5. General data types.

1. Numerical data types:

→ A number which is not having any fractional part will come under Numerical data type.

Ex: 10, 20 . . .

→ Again numerical datatypes are classified into 2 types

1. Signed datatype
2. unsigned datatype

1. Signed numerical data type :-

→ These data types will allow positive and negative values

→ According to sizes signed Numerical data types are classified in to 4 types.

1. sbyte → 1 byte → 3 no's
2. short → 2 bytes → 5 no's
3. int → 4 bytes → 10 no's
4. long → 8 bytes → 19 no's.

1. Sbyte:

→ Here 'S' stands for signed.

→ predefined size of sbyte is 1 byte i.e., 1×8 = 8 bits of signed integers it can hold maximum.

→ sbyte base type in CTS is → System.SByte

2. Short :

→ predefined size 2 bytes i.e., 16 bits (2×8 = 16)

→ Base type of short is → System.Int16.

3. int :

→ predefined size is 4 byte ie., 4×8 = 32 bits

→ Base type of int is → System.Int32.

4. long:

→ predefined size is 8 bytes ie., 8x8 = 64 bits

→ Base type of long is → System.Int64.

2. Unsigned Numerical datatypes:-

→ These data types will allow only positive values.

→ According to the sizes, these data types are classified in to 4 types.

1. byte
2. ushort → 5
3. uint → 10
4. ulong → 20

1. byte:

→ Predefined size is 1 byte ie., 1x8 = 8 bits of unsigned integers

→ Base type is → System.Byte

2. ushort:

→ 'u' stands for unsigned.

→ 2 bytes → 2x8 = 16 bits of unsigned integers

→ base type is → System.UInt16

3. uint:

→ 4 byte → 4x8 = 32 bits of unsigned integers

→ Base type is → System.UInt32

4. ulong:

→ 8 byte → 8x8 = 64 bits of unsigned integers

→ Base type is → ~~System.ULong64~~. System.UInt64

27/1/14

2. Floating Point data types :

→ A number which is having fractional part will come under floating point data.

Ex: 10.5, 20.5.

→ Floating point data types will have only signed which will not support unsigned data types.

→ These are 3 types.

1. Float
2. Double
3. Decimal

1. Float :

→ Pre defined size is 4 bytes i.e., 4 x 8 = 32 bits of signed floating values

→ Base type is System.Single

2. Double :

→ pre defined size is 8 byte i.e., 8 x 8 = 64 bits of signed floating values.

→ Base type is System.Double.

3. Decimal :

→ Predefined size is 16 byte i.e., 16 x 8 = 128 bits of signed floating values.

→ Base type is System.Decimal

23) Example for float variable :

Prgm

```
Void Main()
{
  float a = 4.5;
  Console.WriteLine(a);
  Console.ReadLine();
}
```

O/P

4.5

→ In the above prgm will generate a compile tym error becoz by default C#.Net compiler will treat floating value as double.

→ whenever we want to assign a floating value into float variable we should postfix with "f" like below.

```
float a = 4.5f;
(Or)
float a = 4.5F;
```

24) Example for double variable.

Prgm

```
Void Main()
{
    double a = 4.5;
    Console.WriteLine(a);
    Console.ReadLine();
}
```

25) Example for decimal.

Prgm

```
Void main()
{
    decimal a = 4.5m;
    Console.WriteLine(a);
    Console.ReadLine();
}
```

3. Character Related datatypes:

1) char

→ Pre defined size is 2 byte ie., 2×8 = 16 bits of unicode characters.

→ Base type is System.Char

25) Example for char

```
Void Main()
{
    char a1 = 'b'; // valid
    Console.Writeline(a1);
    char a2 = 'ab'; // invalid
    char a3 = '2'; // valid
    char a4 = '@'; // valid
```

4. Logical Datatypes :

1) bool

→ Bool variable can contain either True (or) False value

→ True will be representing as '1' and False will be representing as '0'

→ Predefined size of bool is 1 bit.

→ Base type is System.Boolean

26) Example for bool variable

program

```
Void Main()
{
    bool b1 = true; // valid
    Console.Writeline(b1);
    bool b2 = "true"; // invalid
    bool b3 = 1; // invalid
    Console.ReadLine();
}
```

5) General data types :

→ In general data types we have 2 types

1) String
2) object

1. String :

→ To represent collection of characters we will go for String variable

Eg: String s = "Satya";

→ There is no predefined size.

→ Base type is System.String

2. Object :

→ Base type is System.Object

Note :→ The above 5 datatypes are called premitive data types

→ In these premitive data types 1, 2, 3 & 4 are Supporting Predefined sizes but 5th category i.e, general data types will not support predefined sizes.

MinValue and MaxValue :

→ These two are constants

→ MinValue is writtening the given data type starting range

→ MaxValue is writtening the ending range of given datatype

27) Example for MinValue & MaxValue of byte & Sbyte datatypes

```
27) Void Main()
    {
      Console.WriteLine("byte min value is: "+byte.MinValue);
      Console.WriteLine("byte Max value is: "+byte.MaxValue);
      Console.WriteLine("Sbyte min value is: "+sbyte.MinValue);
      Console.WriteLine("Sbyte Max Value is: "+sbyte.MaxValue);
      Console.ReadLine();
    }
```

O/P

byte min value is : 0

byte max value is : 255

Sbyte min value is : -128

Sbyte max value is : 127

Note :

→ In premitive datatype 1, 2, 3 & 4 data types are providing by the microsoft as predefined structures.

→ And general datatypes ie,, String and object are providing by the microsoft as predefined classes with in a base class library called System like below.

Skeleton of System Base class Library :

```
namespace System
{
  class console
  {
    writeLine()
    {
    }
    ReadLine()
    {
    }
  }
  class convert
  {
    ToInt32()
    {
    }
  }
```

```
Struct int
{
}
Struct long
{
}
Struct double
{
}
Struct char
{
}
Struct bool
{
}
class String
{
}
class object
{
}
}
```

28/1/24

typeof() and sizeof():

typeof()

→ This will return the base type of given data type

sizeof()

→ This will return the size of the given ~~by~~ data type in bytes.

28) Ex of sizeof() & typeof()

pgm

```
Void Main()
{
 Console.WriteLine (typeof (int));
 Console.WriteLine (typeof (float));
 Console.Writeline (sizeof (Int));
 Console.WriteLine (sizeof(long));
 Console.ReadLine();
}
```

o/p

```
System.Int32
System.Single
4
8
```

What is local Variable?

→ A variable which is declared within a method is called as Local variable.

→ local Variable should be initialised with some value before accessing, otherwise compiler will generate an error.

29) Ex:

```
Void Main()
{
 int a;
 Console.WriteLine (a);  → error
 Console.Readline();
}
```

→ To overcome above error we can write the code like below.

```
Void Main()
{
 int a=10;
 Console.WriteLine(a);
 Console.Readline();
}
```

(or)

```
int a; // declaration
a = 10; // assigning.
console.WriteLine(a); // accesing.
Console.ReadLine();
```

30) Ex to declare multiple variables in a single line.

```
int a = 10, b = 20; // valid
int a = 10, string s = "Satya"; // invalid.
int a = 10; string s = "satya"; // valid.
```

Implicite typed Variables:-

→ Using var keyword we can declare implicite typed Variables.

→ Implicit typed variables concept was introduced ~~by~~ with .Net framework 3.0 version

→ Implicit typed variable datatype will be deciding based on the value which we are assigning

30) Ex of Implicit typed variables.

Prgm

```
Void Main()
{
    Var a = 10
    Var b = "rama";
    Var c = true;
    Console.WriteLine(a.GetType());
    Console.WriteLine(b.GetType());
    Console.WriteLine(c.GetType());
    Console.ReadLine();
}
```

O/P

System.Int32
System.String
System.Boolean

GetType():

→ This method will return the base type of given variable.

Difference b/w typeof() and GetType():

→ Using GetType() we can get the base type of given variable.
→ Using typeof() we can get the base type of given data type.

31) Ex for implicit typed variable.

```
Void Main()
{
  Var a = 6.5;
  Var b = 4.5f;
  Var c = 6.5m;
  int d = 10;
  Console.WriteLine (a.GetType());
  Console.WriteLine (b.GetType());
  Console.WriteLine (c.GetType());
  Console.WriteLine (d.GetType());
  Console.ReadLine ();
```

O/P

System.Double
System.Single
System.Decimal
System.Int32

When will we use Implicit Variables.

→ We will use Implicit typed variable at the time of implimenting linq concept.

→ According to data storage location C#.Net datatypes are classified into two types.

1. Value types
2. Reference types

1. <u>Value types</u> :-

→ In value types data will be storing into stack memory, due to that reason value types are called as Stack based data types.

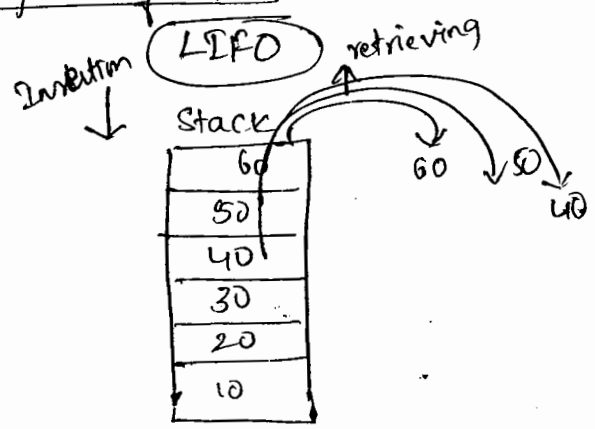
<u>Stack</u>:

→ Stack is a data structure which can contain collection of elements

→ Stack will support only one end i.e., Topend due to that reason, insertion & retrieval we can perform from Topend.

→ Stack is following an approach called LIFO [Last In First Out] that mean last inserted element we can retrieve first

<u>diagram of stack</u>



2. Reference Types:

→ Reference type data will be storing into heap memory. Due to that reason, reference types are called heap based data types.

heap

→ Heap is a data structure which can contain collection of elements.

→ Heap will be representing in binary tree structure format.

→ Heap is a collection of nodes, which are two types.

1. Parent node
2. child node.

→ Heap is not following any approach called LIFO (or) FIFO, which is following its own approach called Random approach that means within the heap we can reach ~~the~~ an element randomley.

Difference b/w value types & Reference types:

Value Types	Reference Types
1. Data is storing in to Stack.	1. Data is staring in to heap.
2. Value type variable will contain the actual data like below. Ex: a [10]	2. Reference type variable will contain the address of the data like below Ex: b → variable [1010] → [20] data, 1010 → address
3. Value types are:- 1. Structure 2. In premitive datatypes following are Value types:- a) Numerical data types b) Floating point data types c) character related data types d) Logical data types.	3. Reference types are:- 1. class. 2. In premitive data types following are Reference type a) General data types Ex: String, object

Ex: Int, float, double, char,
bool.....

3) Enums [Enumerator]

4)

3) Interface

4) delegates

Can we assign Null value into Value type variable:?

→ No

Ex: Int a = Null; // invalid

Can we assign null value into reference type variable?

→ Yes

Ex: String S = null; // valid.

→ Till .Net 1.1 we cannot assign null value in to value type variable.

→ From .Net 2.0 we can assign null value into value type variable with the help of nullable value types.

nullable value types:

→ This was introduced with .Net 2.0 version.

→ Using nullable value types we can assign a null value into value type variable.

→ Example for nullable value types.

32)
```
int Void Main
{
    int? a = null;
    float? b = null;

}
```

When we will go for nullable value types?

→ whenever the input is optional. We will ~~go for~~ declare particular variable as nullable value type.

30/01/14

33) Example to perform division.

```
Void Main()
{
  int a = 4;
  int b = 2;
  div c = a/b;
  Console.WriteLine("Enter first no:");
  ~~Console.ReadLine();~~
  Console.WriteLine("Enter second no:");
  Console.WriteLine("Division is:"+c);
  Console.WriteLine
}
```

```
Void Main()
{
  Console.WriteLine("Enter first no:");
  int a = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Enter second no:");
  int b = Convert.ToInt32(Console.ReadLine());
  div c = a/b;
  Console.WriteLine("division is:"+c);
  Console.ReadLine();
}
```

Note → In the above prgm when user enter second no as zero it is throwing runtime error.

→ We can handle these runtime errors in two ways

1. Using logic
2. Excepting handling mechanism.

→ To solve the run time errors by using logic we should go for control statements.

Control Statements:-

→ In C#.Net Control statements are 3 types.

1. Conditional statements
2. loops
3. Jump statements

1. Conditional statements

1. If Condition
2. Switch condition

→ When we will go for conditional statements.

Ans Whenever we want to exicute a single statement or block of statements based on condition we will go for conditional statements.

If Condition:-

→ We can impliment if condition in 4 ways

1. If (simple if)
2. if else
3. ~~multi~~ if else if (multiple if)
4. nested if

1. Simple if:

Syntax:

```
if(<condition>)
{
    Stm 1;
    Stm 2;
}
```

33) → Example to handle the above division prgm error by using simple if.

o/p

Enter first no:

a [10] ← 10

Enter second no:

b [0] ← 0

Please enter otherthan zero:

b [2] ← 2

div [a/b] → 5

Division result is : 5

Prgm

```
Void Main()
{
    Console.WriteLine("Enter first no:");
    int a = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Enter second no:");
    int b= Convert.ToInt32(Console.ReadLine());
    if(b==0)
    {
      Console.WriteLine("Please Enter otherthan zero:");
      b= Convert.ToInt32(Console.ReadLine());
      ~~Console.ReadLine();~~
    }
    Console.WriteLine("Enter second no:");
    int b= Convert.ToInt32(Console.ReadLine());
    div c= a/b;
    Console.WriteLine("division result is :"+c);
    Console.ReadLine();
}
```

a/b = —

10/2 = 5

↓

2

→ when we will go for simple if ?

Ans → Whenever we have a single option, which we want to execute based on condition we can go for simple if.

2. if else :

Syntax : if (<condition>)

St 1; a

~~St 2~~

else

st2; b

34) Write a console prgm to compare two no's by using if else

o/p

Enter first no:

i 10

10

Enter second no:

j 5

5

i is greater than j.

Prgm

```
Void Main()
{
    Console.WriteLine("Enter first no:");
    int i = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Enter second no:");
    int j = Convert.ToInt32(Console.ReadLine());
```

```
if (i>j)
  {
    Console.WriteLine(" i is greater than j");
  }
  else
    Console.Writeline(" j is greater than i");
    Console.Readline();
  }
```

3. <u>if else if</u>:

Syntax:

```
if (<condition1>)
  { St1;
  else if (<condition2>)
    St2;
  else if (<condition3>)
    St3;
  else
    St4;
```

34) Implement comparing two no's with the help of if else if

```
Void Main()
{
  Console.WriteLine("Enter first no:");
  int i = Convert.ToInt32(Console.ReadLine());
  Conside.WriteLine(" Enter Second no:");
  int j = Convert.ToInt32(Console.ReadLine());
  if (i>j)
    Conside.writeline(i+" is big");
  else if (j>i)
    Conside.Writeline(j+ " is big");
```

```
else
    Console.WriteLine("i+ is equal to " +j);
    Console.ReadLine();
}
```

Q) ~~whenever~~ when we will go for if else?

Ans whenever we have two options, want to execute one option at a time among two options based on condition then we can go for if else.

Q) when we will go for if else if?

Ans whenever we have more than two options, at a time if we want to execute one option among multiple options. based on condition then we can go for if else if.

35) Implement the above example with if else.

Prgm

```
Void main
{
  Console.WriteLine("Enter first no:");
  int a = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Enter Second no:");
  int b = Convert.ToInt32(Console.ReadLine());
  if (a>=b)
  {
    if (a>b)
      Console.WriteLine("a is greater than b");
    else
      Console.WriteLine("a is equal to b");
  }
  else
```

```
Console.WriteLine (" a is lesthan b");
Console.ReadLine();
}
```

→ In the above example we have implemented nested if

Q) when we will go for nested if?

Ans whenever we have multiple options which we want to execute 1 option based on ~~execution~~ condition first choice is if else if, second choice is nested if.

36) implement the above prgm ~~prgm~~ using simple if conditions

prgm

```
Void Main()
{
    Console.WriteLine ("Enter first no:");
    int a = Convert.ToInt32(Console.ReadLine());
    Console.Writeline ("Enter second no:");
    int b= Convert.ToInt32 (Console.ReadLine());
    if (a>b)
        Console.Writeline ("a is greater than b");
    if (a<b)
        Console.Writeline ("b is greater than a");
    if (a==b)
        Console.Writeline ("a is equal to b");

    Console.ReadLine();
}
```

5/1/14 **2. Switch :-**

Q) when will we go for switch?

Ans. Whenever we want to execute single case among multiple cases we can go for switch.

Syntax:

```
Switch(< Expression >)
{
    case 1:   → Label
        St1;
        break;
    case 2:
        St2;
        break;
    case 3:
        St3;
        break;
    default:
        St4
        break;
}
```

37) Example for switch condition to display month name, based on month no -

O/P

Enter your month number:

month: [1] ← 1

Jan

Pgm

```
Void Main()
{
    Console.WriteLine("Enter your month number:");
    ~~Console.~~ int a = Convert.ToInt32(Console.ReadLine());
    Switch (~~month~~ a)
    {
```

```
Case 1:
    Console.WriteLine("January");
    break;
Case 2:
    Console.WriteLine("February");
    break;
Case 3:
    Console.WriteLine("March");
    break;
Case 4:
    Console.WriteLine("April");
    break;
Case 5:
    Console.WriteLine("May");
    break;
Case 6:
    Console.WriteLine("June");
    break;
Case 7:
    Console.WriteLine("July");
    break;
Case 8:
    Console.WriteLine("August");
    break;
Case 9:
    Console.WriteLine("September");
    break;
Case 10:
    Console.WriteLine("October");
    break;
Case 11:
    Consoles.WriteLine("November");
    break;
```

```
Case 12 :
    Console.WriteLine ("December");
    break;
default :
    Console.WriteLine ("Invalid input");
    break;
}

Console.ReadLine();
}
```

38) Write a Console prgm to compare 2 no's by using Switch

Prgm)

```
Void Main()
{
  Console.WriteLine(" Enter first no: ");
  int a= Convert.ToInt32(Console.ReadLine());
  Console.WriteLine (" Enter Second no: ");
  int b= Convert.ToInt32(Console.ReadLine());
  Switch(a>b)
  {
    Case True :
    Console.WriteLine (" a is greater than b");
    break;
    Case False :
    Console.WriteLine (" b is greater than a");
    break;
    default :
    Console.WriteLine (" both are equal");
    break;
  }
Console.ReadLine();
}
```

Note :→ Switch expression can have a condition and it can have bool, char, string, integral, enum (or) corresponding type values but it cannot contain floating type value.

→ Switch case label can be integer value, String value, char value, bool value, enum value and it cannot be floating value.

→ Switch case label cannot have condition.

3) Void Main( )

```
{
  Console.WriteLine("Enter first no:");
  int a= Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Enter second no:");
  int b= Convert.ToInt32(Console.ReadLine());
  Switch (a>b)
  {
    Case true:
      Console.WriteLine("a is greater than b");
      break;
    Case false:
      Switch (b>a)
      {
        case true:
          Console.WriteLine("b is greater than a");
          break;
        case false:
          Console.WriteLine("a is equal to b");
          break;
      } //internal switch close
      break;
    } //external switch close
    Console.ReadLine();
  }
```

Comparison b/w if else if and switch condition.

Ans → Both are executing one option among multiple options.

→ Execution wise switch is faster than if else if because switch is direct branching and if else if is ladder wise branching.

→ Switch is execution wise faster than if else if, but which is having some limitations like below

1. expression of switch cannot be a floating value
2. Switch label cannot be a floating value & condition.

→ whenever we want to execute ~~multiple~~ one options among multiple options first priority is to switch then next priority is if else.

40) Write console prgm to compare two nos with the combination of switch & if else

prgm

```
Void Main()
{
  Console.WriteLine(" Enter first no: ");
  int a=Convert.ToInt32(Console.ReadLine());
  Conside.WriteLine(" Enter second no:");
  int b= Convert.ToInt32(Console.ReadLine());
  Switch (a>b)
  {
    case true:
      Consoleo.WriteLine("a is greater than b");
      break;
    Case false:
      if (b>a)
        Console.WriteLine("b is greater than a");
      else
        Console.WriteLine("both are equal");
```

```
        break;
    }
    Console.ReadLine();
}
```

5/2/14

4.1) Write a Console prgm to accept a letter and display whether it is vowel or not:

Prgm

```
void Main()
{
    Console.WriteLine("Enter your letter:");
    Char ch = Convert.ToChar(Console.ReadLine());
    Switch (ch)
    {
        case 'a':
            Console.WriteLine("it is a vowel");
            break;
        case 'e':
            Console.WriteLine("it is a vowel");
            break;
        Case 'i':
            Console.WriteLine("it is a vowel");
            break;
        Case 'o':
            Console.WriteLine("it is a vowel");
            break;
        Case 'u':
            Console.WriteLine("it is a vowel");
            break;
        default:
            Console.WriteLine("it is not vowel");
            break;
```

O/P

Enter your letter:

ch [i] ← i

it is a vowel.

```
}
Console.Readline();
}
```

Loops :

Q) When we will go for loops ?

Ans Whenever we want to execute a single statement or block of statements multiple times we will go for loops.

42) Ex: write a console prgm to print Satya Technologies 5 times.

Prgm

```
Void Main()
{
    Console.WriteLine("Satya Technologies");
    Console.WriteLine("Satya Technologies");
    Console.WriteLine("Satya Technologies");
    Console.WriteLine("Satya Technologies");
    Console.WriteLine("Satya Technologies");
    Console.ReadLine();
}
```

→ In the above prgm we have written writeline statement 5 times, by using loops we can write it as a single statement.

Benefit of loops :-

→ We can reduce the number of lines of codes with the help of loops

→ In C#.Net we can implement loops in 4 ways.

1) for loop
2) while loop
3) do while loop
4) for each loop

for loop:

Syntax: for(<initialization>; <condition>; <++(or)--->)

```
{
    St 1;
    St 2;
}
```

43) write a Console prgm to print 1 to 10 no's using for loop

prgm

```
Void Main()
{
    Console.WriteLine(" Numbers are:");
    int i = Convert.ToInt32(Console.ReadLine());
    for(i=1; i<11; i++)
    {
        Console.writeline(i + "\n");
        Console.ReadLine();
    }
}
```

(i<=10) (or)

o/p

```
1
2
3
4
5
6
7
8
9
10
```

44) write a prgm to print no's like below by using for loop.

10, 9, 8, 7, 6, 5, 4, 3, 2, 1.

prgm

```
Void Main()
{
    Console.writeline("Numbers are:");
    int i = Convert.ToInt32(Console.ReadLine());
    for(i=10; i>=1; i--)
```

o/p

```
10
9
8
7
6
5
4
3
2
1
```

```
    Console.writeLine (i+ "\n");
    Console.ReadLine();
}
```

Note : For ~~deg~~ debugging F10 or F11

(4) Write prgm to print no's like below

O/p

1
3
5
7
9
11
15
17
19

Prgm:

```
Void Main()
{
    for (int i = 1; i < 20; i=i+2)
        Console.WriteLine (i+ "\n");
        Console.ReadLine();
}
```

while loop:

Syntax :

```
while (<condition>)
{
    St 1;
    St 2;
}
```

(45) write a prgm to print 1 to 10 no's like below

Prgm)

```
Void Main()
{
  int i=1;
  while (int i<= 10)
  {
    Console.Write(i+" ");
    i++
  }
  Console.ReadLine();
}
```

o/p

1 2 3 4 5 6 7 8 9 10

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

Console.Write();-

→ Write is a predefined method of Console class

→ this method will print the given value on output window, after printing the value it will keep the cursor in the same line of next character.

Do while loop:-

Syntax :

```
Do
{
  St 1;
  St 2;
  i++ (or) i--;
}
while (<condition>);
```

(46) write a Console prgm to print 1 to 100 no's like below.

o/p 1, 5, 10, 15, 20 - - - - 100,

Prgm

```
Void Main()
{
  int i=1;
  Do
  {
    Conole.WriteLine(" Numbers are:");
    Conole.Write(i+ ", ");
  }
```

```
while (i<20);
  {if(i==1)
    Console.Write (i);
    i++;
  int a=(i-1)*5;
  Console.write (a);
  }
 Console.ReadLine();
}
```

47) Prgm to print 1 to 10 no's like below using Do while

O/P

1, 2, 3, 4, 5, 6, 7, 8, 9, 20. —

Prgm

```
Void Main()
{
 int i=1;
 Do
  {
   Console.write(i+" , ")
   i++;
  }
 while (i<10)
  {
   Console.write(i+" . ");
   i++;
  }
 Console.ReadLine();
}
```

```
Void Main()
{
  int i=1
  Do
  {
    Console.Write(i+",");
    i++
  }
  while (i<=9)
    Console.Write(i+".");
    Console.ReadLine();
}
```

Q) When we will go for while loop and when we will go for for loop?

Ans → When the no. of interations are fixed & decided we will go for for loop.

→ When the no. of interations are not fixed we will go for while loop.

(48) Purposefull Example for while loop.

division by zero prgm by using while.

Prgm

```
Void Main
{
  Console.WriteLine("first no : ");
  int i = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine(" second no :");
  int j = Convert.ToInt32(Console.ReadLine());
  while (j==0)
  {
    Console.WriteLine(" Enter j other than zero");
    b = Convert.ToInt32(Console.ReadLine());
  }
int div = i/j;
  Console.WriteLine(div);
  Console.ReadLine();
}
```

2(a) Implement the above prgm by using for loop

```
Void Main()
{
// first 4 lines are same as above prgm
    for(   ;b==0;  )
    {
      Console.WriteLine("pls enter other than zero");
      b= Convert.ToInt32(Console.ReadLine());
    }
int div =a/b;
    Console.WriteLine("Result is:"+div);
    Console.ReadLine();
}
```

→ we can implement a for loop without initialization & increment (or) decrement. By default we can work only with condition also. So for the above requirement ~~abov~~ best choice is while loop

Q) Can we implement a for loop without initialization, condition, increment (or) decrement?

Ans Yes possible like below but will be like infinite loop.

```
for(;;)
{
  Console.WriteLine("Hi");
}
Console.ReadLine();
```

Q) When we will go for Do while loop?

Ans whenever the no. of interations are not fixed and even though condition is false it has to execute atleast one interation then we can go for Do while loop.

50) Implement above division prgm using Do while.

51) Write a purposeful console prgm for Do while.

for each loop :-

→ for each is one of the looping concept which we will use to fetch the elements from collections like erase, stack, queue. =

Jump Statements :-

→ Using jump statements we can transfer the prgm control.

→ we have 5 types

1. break.
2. Continue
3. goto
4. return
5. throw

1. break :

→ break statement will transfer the prgm ctrl to out of nearest closing brase and out of switch block.

52) Write a prgm to print 1 to 100 no's but it shud print only 1 to 5 no's like below.

Prgm

```
Void main()
{ int i;
  for(i=1; i<=100, i++)
    if(i==5)
    {Console.writeline(i); if(i==5);
     break;
    }
    Console.ReadLine();
}
```

O/P

1
2
3
4
5

10

2. Continue :-

→ Continue will skip the current interation

53) prgm to print 1 to 10 no's but it has to print the o/p like below

o/p

1
2
3
4
6
7
8
9
10

```
Void Main( )
{
    for(i=1; i<=10; i++)
    {
      ~~Console.writeline(i);~~
      if (i==5)
        {
        continue;
        }
      Console.Writeline(i);
    }

    Console.ReadLine();
}
```

3. goto :-

→ goto will transfer the prgm control to the given label as well as to the switch case.

→ label should be defined by the ~~prg~~ programmer like below

Satya: → label.

54) Example for goto

Prgm

```
Void Main()
{
  Console.WriteLine("I like C#.Net");
  Console.WriteLine("I Like ASP.Net");
  goto skip;
  Console.WriteLine("I Like Java");
  Skip:
   Console.ReadLine();
}
```

O/P

I like C#.Net

I Like ASP.Net

55) Example for division prgm by using if condition with the help of go to.

Prgm

```
Void Main()
{
  Console.WL("first no:");
  int a = Convert.ToInt32(Console.ReadLine());
  Console.WL("second no:");
  int b = Convert.ToInt32(Console.RL());
i:  if (b==0)
    {
      goto Console.WL("Enter other than zero");
        b = Convert.ToInt32(Console.RL());
      goto i;
    }
    Console.RL();
  int div = a/b;
  Console.WL("Result is:"+div);
  Console.RL();
}
```

46) To print 1 to 100 no's as below using do while loop

op

1, 5, 10, 15, 20, 25 - - - - - 99, 100,

Prgm

```
Void main()
{
 int i = 1;
 Console.WriteLine("Numbers are:");
 do
 {
   if (i == 1)
        Console.Write(i + ", ");
   i++;
     int a = (i-1) * 5;
    Console.Write(a + ", ");
 }
 while (i<=20);
 Console.ReadLine();
}
```

50)

```
Void Main()
{
 // first 4 times are same
   do
```

56) Example to transfer prgm control to the targeted Switch case using go to:

```
Void Main()
{
 Console.WriteLine ("Enter your choice 1 or 2");
 int i=Convert.ToInt32(Console.ReadLine());
 Switch (i)
 {
   Case 1:
     Console.WriteLine("This is first case");
     break;
   Case 2:
     Console.WriteLine("This is second case");
     ~~break~~;
     goto case 1;
   default:
     Console.WriteLine("This is default case");
     Break;
 }
 Console.ReadLine();
}
```

4. return:

→ Using return statement we can return the ~~prgm~~ ~~control~~ value.

→ Whenever we are defining a user defined function to return a value. we can use return statement.

→ Once return statement is executed control will transfer the out of the method block.

5. throw :

→ Using throw keyword we can throw an exception explicitly

57) Console prgm to store 5 employee no's. They are 111, 222, 333, 444, 555.

→ To impliment above prgm we have to go for five variables

→ whenever we want to store multiple elements which are same type we will go for a concept called array.

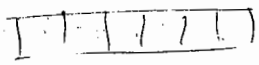
Arrays

→ Array is a collection of elements which are similar datatype

→ every element will be representing with one index value. fist element index value will be '0' and 2nd element index value is 1. and so on..

→ C#.Net will support 3 types of arrays.

1. One dimensional array
2. Multi dimensional array
3. Jagged array.



1. One dimensional array :

→ Arranging collection of elements in a single row is called as one dimensional array.

→ Syntax :

int, string ---

subscription operator

user defined

keyword

no. of elements

<data type>[] <arrayname> = new <datatype>[size];

→ Syntax to declare an array with initialization :-

<datatype>[] <array name> = new <data type>[] { <element 1>, <elem 2>, <elem 3>, ---- };

→ Syntax to access array element:-

arrayname[<index>];

Ex:

a

10	20	30	40	50
0	1	2	3	4

n = 5 - 1
= 4

a[3] → 40
a[0] → 10

57) Console prgm to create one dimensional array like above and display the array elements like below

Prgm

```
Void Main()
{
int [ ] a = new int [ ] {10, 20, 30, 40, 50};
Console.Write("a[0]+" "a[1]+" "a[2]+" "a[3]+" "a[4]);
Console.ReadLine();
}
```

Wrong

O/P

10 20 30 40 50

→ For the above prgm if we have more elements so we go for loops

58)

```
Void Main()
{ int i;
int [ ] a = new int [ ] {10, 20, 30, 40, 50};
for (i=0; i<a.Length; i++)
  Console.Write(a[i]+" ");
  Console.ReadLine();
}
```

5

10 20 30 40 50

Length :

→ length is a predefined member property of array class.

→ length property will return the no of elements with in the no. of elements with in the given one dimensional array.

HW

59) Implement above prgm using while & do while loops.

60) Console prgm to ~~pr~~ create string one dimensional array.

Multi Dimensional array :

→ ~~Multi di~~

→ Representing Collection of elements in matrix format is called as Multi dimensional array.

→ Multi dimensional array is a collection of rows and collection of coloumns.

→ every row should have the same no. of elements.

→ Syntax to declare multi dimensional array :-

<datatype>[,]<array> = new<datatype>[size, size]; → no. of coloumns

no. of rows

61) Example for Multi dimensional array.

a	0	1	2
a:	10	20	30
1:	40	50	60

a[0,0] → 10

a[0,2] → 30

a[1,2] → 60

a[1,1]

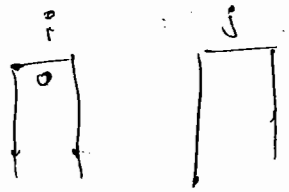
```
Void Main()
{
int i, j;
int[ , ]a = new int[2,3]{{10,20,30},
                         {40,50,60}};
for(i=0; i<2; i++)
{
  for(j=0; j<3; j++)
```

o/p

10 20 30

40 50 60

```
    Console.Write(a[i,j] + " ");
  }Console.WriteLine();

Console.ReadLine();
}
```



→ row

a.GetLength(0);–

→ It will return no. of rows with in multidimensional array.

a.GetLength(1);– → c.o

→ It will return no. of columns with in multidimensional array.

62) Implement above prgm by using while & while loop.

63) above prgm using while & for

64) for – while

65) while – do while

66) dowhile – while

67) for – do while

68) do while – for

HW

59) using while

O/P

10 20 30 40 50 –

Prgm

```
Void Main
{
  int i = 0;
  int[] a = new int[5] {10, 20, 30, 40, 50};
  while (i<5)
    {
      Console.Write(a[i] + " ");
      i++;
    }
    Console.ReadLine();
}
```

Using do while

```
Void Main()
{
  int i = 0;
  int[]a = new int[] {10,20,30,40,50}
  do
   {
     Console.Write(a[i]+" ");
     i++;
   }
  while(i<5);

     Console.ReadLine();
}
```

60)

```
Void Main()
{
 ~~String~~ int i

String[]a = new string[] {"rama", "Sita", "Satya"};
for(i=0; i<a.Length; i++)
   {
     Console.Write(a[i]+"  ");
   }
     Console.ReadLine();
}
```

o/p

rama sita satya —

62)

```
Void Main()
{
 ~~int i=0,j=0;~~
 int[,]a=new int[,]{{10,20,30},{40,50,60}};
 int i=0;
 while(i<2)
  { int j=0;
   while(j<3)
    {
     Console.Write(a[i,j]+" ");
     j++;
    }
```

```
    console.Writeline( );
    i++;
  }
  console.Readline( );
}
```

63)

```
int Void Main( )
{
  int i=0;
  int[,]a=new int[,]{{10,20,30},{40,50,60}};
  while (i<2)
                (or) a.GetLength(0)
  {
    for(j=0 ; j<3 ; j++)
                (or) a.GetLength(1)
    {
      Console.Write (a[i,j]+" ");
    }
    Console.Writeline( );
    i++;
  }
  Console.Readline( );
}
```

64)

```
Void Main ( )
{
  int j=0;
  int[,]a=new int[,]{{10,20,30},{40,50,60}};
  for (i=0 ; i<2 ; i++)  → row
  {
    while ( j<3) → col
    {
      console.Write (a[i,j]+" ");
      j++;
    }
    console.Writeline( );
  }
  console.Readline( );
}
```

	0	1	2
0	10	20	30
1	40	50	60

a[0,1]

65)
```
Void Main()
{
  int i=0;
  int[,] a = new int[,]{{10,20,30},{40,50,60}};
  while (i<2)
   {
    int j=0;
    do
     {
       if (j<3)
       Console.Write(a[i,j]+"    ");
       j++;
     }
    while (j<3);
    Console.WriteLine();
    i++;
    }
  Console.ReadLine();
}
```

67)
```
Void Main()
{
  int[,] a = new int[,]{{10,20,30},{40,50,60}};
  for(int i=0; i<2 ; i++)
  {
     int j=0;
    do
     {
       if (j<3)
       Console.Write(a[i,j]+"    ");
       j++;
     }
    while(j<3);
     Console.WriteLine();
  }
 Console.ReadLine();
}
```

0 1 2
10 20 30
40 50 60

66)

```
int[,] a = new int[,]{{10,20,30},{40,50,60}};
int i=0 -> i
do
{
    int j=0;
    while (j<a.GetLength(1))
    {
      Console.Write(a[i,j]+"    ");
      j++;
    }
    Console.WriteLine();
    i++;
  }
while (i<a.GetLength(0));
      Console.ReadLine();
}
```

68)

```
int[,] a = new int[,]{{10,20,30},{40,50,60}};
int i=0
do
{
  for(int j=0; j<a.GetLength(1); j++)
      Console.Write(a[i,j]+"    ");
  }
  while (i<a.GetLength(0))
    Console.WriteLine();
    i++; }
  while(i<a.GetLength(0));
  Console.ReadLine();
}
```

0
1
2
3

Q) How to get the number of elements with in Multidimensional array?

Ans: Using Length property we can get the no. of elements using multi dimensional array.

69) Write a console prgm to create Multidimensional array like below

a	0	1	2	3
0	10	15	20	25
1	30	35	40	45
2	50	55	60	65

and print this array elements like below.

Prgm

```
Void Main()
{
    int[,]a=new int[,]{{10,15,20,25},{30,35,40,45},{50,55,60,65}};
```

O/P

```
10 15 20 25
30 35 40 45
50 55 60 65
```

O/P (1)

```
10 15 20 25
30
35
40
45
50 55 60 65
```

O/P (2)

```
10 15 20 25
30 35 40 45
50
55
60
65
```

O/P (3)

```
10
15
20
25
30 35 40 45
50 55 60 65
```

O/P (4)

```
10
15
20
25
30
35
40
45
50 55 60 65
```

O/P (5)

```
10
15
20
25
30 35 40 45
50
55
60
65
```

Implement above 5 o/p with diff loop combinations.

3. Jagged Array:

→ Jagged array is a collection of rows which may not contain same no. of elements.

→ That means in jagged array every row may contain different no. of elements.

→ Finally we can say jagged array is a collection of one dimensional arrays.

Advantage:

→ we can save the memory.

→ Syntax for jagged array :-

```
<datatype>[ ][ ]<arrayname> = new <datatype>[size][ ];
```

No. of rows

70) Example for jagged array.

a	0	1	2	3
0	10	25	16	17
1	20	30		
2	40	50	60	

a[0][0] → 10

a[1][1] → 30

a[2][2] → 60

pgm

```
Void Main
{
    int[][] a = new Int[3][ ];
    for(int i=0; i<a.Length(0); i++)
    {
        for(int j=0; j<a.GetLength(1); j++)
            Console.Write(a[i][j] + "  ");
            Console.WriteLine( );
    }
        Console.ReadLine( );
}
```

a[1][ ] 2 3 4

o/p

```
Void Main()
{
  int[][] a = new int[3][];
  // step 2
    // 1st row
    a[0] = new int[] {10, 15, 16, 17};
    //2nd row
    a[1] = new int[] {20, 30};
    // 3rd row
    a[2] = new int[] {40, 50, 60};
  for (int i=0; i<a.Length; i++)
  {
    for(int j=0; j<a[i].Length; j++)
    {
      Console.Write (a[i][j] + "  ");
    }
    Console.WriteLine();
  }
  Console.ReadLine();
}
```

i = 0, 1, 2, 3

i = 0, 1, 2, 3, 4, 0, 1, 2, 0, 1, 2, 3

a[0].L e

a.Length :

→ Here a is jagged array.
→ It will return no. of rows with in the jagged array.

a[i].Length :

→ 1st interation
a[0].Length → 4
↳ 1st one dimensional array
→ 2nd interation
→ 2nd one dimensional array
a[1].Length → 2

→ 3rd interation
a[2].Length → 3
↳ 3rd one dimensional array.

71) Implement the above prgm by using diff loops combinations

72) Write a jagged array to store customer address & implement that array using diff loops combination & display customer address which is available in jagged array with diff o/p formats.

73) Write a console prgm for \t and \n

```
Void Main()
{
    Console.Write("\t Satya\n");
    Console.Write("Hi \t");
    Console.ReadLine();
}
```

o/p

```
        Satya
hi
```

74) Write a Console prgm to accept elements from the user and store into one dimentional array and display to user.

o/p

Enter your Element : 10

Enter your Element : 20

Enter your Element : 30

Enter your Element : 40

Enter your element : 50

Array elements are :

10 20 30 40 50

a

10	20	30	40	50

Prgm.

```
Void Main()
{ int[] a = new int[ ];
  Console.WriteLine("Enter ur element:");
  int a[0] = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Enter your element: \n");
  a[1] = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Enter your element: \n");
  :
  a[4] = Convert.ToInt32(Console.ReadLine());
  Console.WriteLine("Enter your element: \n");
```

```
for (int i=0; i<a.Length; i++)
  {
   Console.Write (a[i] + " ");
   a[i] = Convert.ToInt32(Console.ReadLine());
Console.WriteLine
```

Prgm

```
Void Main ()
{
 int[ ] a = new int[5];
 for(int i=0; i<a.Length; i++)
  {
    Console.Write("Enter your element:");
    a[i] = Convert.ToInt32(Console.ReadLine());
    Console.Write("\n");
  }
  Console.WriteLine("Array elements are:");
  for(int j=0; j<a.length; j++)
    {
     Console.Write(a[i] + " ");
    }
    Console.ReadLine();
}
```

75) Write a console prgm to create two arrays ① names ② ages Accept user name, user age store into concern arrays. Accept the user name and display the concern user age. If user entered invalid name display error msg.

O/P

Enter ur ~~age~~ name : rama
Enter ur age : 20

Enter ur name : sita
Enter ur age : 18

Enter ur name : venkat
Enter ur age : 25

Enter ur name : Gopi
Enter ur age : 26

Enter ur name: ramesh
Enter ur age : 28

find
Gopi

Enter searching user name : Gopi
Gopi age is : 26

Prgm

```
Void Main()
{
 //step 1 declaring arrays
 String[] names = new String[5];
 int [] ages = new int[5];
 // step 2 accepting & storing into arrays.
 for(int i=0; i<5; i++)
  {
   Console.WriteLine (" Enter ur name : ");
   names[i] = Console.ReadLine();
   Console.WriteLine ("Enter ur age:");
   ages [i] = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine ("\n\n");
  }

  Console.Write ("Enter searching user name:");
  String find = Console.ReadLine();
  int ctr = 0;
  for( int i=0; i<5 ; i++)
   {
    if(names[i] == find)
    {
     console.WriteLine (names[i] + " age is:" + ages[i]);
     ctr++
    }
    if(ctr==0) { Console.WriteLine("Invalid user"); }
   } Console.ReadLine();
 }

  if(ctr == 0)
   {Console.WriteLine(
```

Type Casting: (Type Converting)

→ It is a process of converting from one data type to another data type.

Q) When we will go for type casting?

Ans: → whenever we want to assign one data type value into another data type variable we go for type casting

→ In C#.Net will support type casting in 3 ways.

1) Implicit type casting.
2) Explicit type casting
3) Boxing and unboxing.

1. Implicit type casting:-

→ By default C#.Net will support some of the type conversions which are called as implicit type casting.

→ To Implement implicit type casting programmer doesnt require to impliment any special syntax except assignment operation.

Following table describing the possible implicit type convertions in C#.Net

From	To
sbyte	short, Int, Long, Float, Double, Decimal
byte	short, Ushort, Int, UInt, Long, ulong, Float, double, decimal
Short	Int, Long, Float, Double, decimal.
Ushort	Int, UInt, Long, ULong, Float, Double, Decimal.
Int	Long, Float, Double, Decimal.
UInt	Long, ULong, Float, Double, decimal.
Long	Float, double, decimal.
ulong	Float, double, decimal.
Float	Double

char ushort, int, uint, long, float, ulong, double, decimal.

String object

76) Example for implicit type casting from int to long

```
Void Main
{
 int i =10;
Long j = i;
Console.WriteLine("j value is:" +j);
Console.ReadLine();
}
```

77) Example for implicit type casting to convert from char to int & double

```
Void Main()
{
   char c = 'a';
   int i = c;
   Console.WriteLine("int i value is:" + i);
   double d = c;
   Console.WriteLine("double d value is:" + d);
   char e = '1';
   int j = e;
   Console.WriteLine("Int j value is:" + j);
   Console.ReadLine();
}
```

o/p

int i value is: 97

double d value is: 97

int j value is: 46

78) Implement above all possible implicit type conversions.

2. Explicit type casting:

Q) when we go for explicit type casting?

Ans → whenever we are assigning one type value into another type variable which is not possible by implicit type casting then we have to go for explicit type casting.

→ while implementing explicit type casting programmer has to follow some special type conversion syntax.

→ In C#.Net explicit type casting we can implement by using 3 techniques

1. ~~C#~~ C++ style of type casting technique
2. Parsing Technique
3. Converting Technique.

1. C++ Style of type casting ~~syntax~~ technique:

→ Syntax

<To datatype> <To variable> = <To datatype)> <from variable>;

Example for C++ style of type casting from converting from int to byte

79)
```
int i=100;
byte b=(byte) i;
Console.WriteLine("byte b value is:" + b);
Console.ReadLine();
```

2. Parsing:-

→ To implement parsing we will use a predefined method called Parse()

Parse():-

→ It is a predefined member method of every premitive datatype structure.

→ Parse method will accept String value and, it will convert String value into ~~given~~ calling datatype and that converted value it will return.

79) Example for parsing :

```
Void Main()
{
  Console.WriteLine("Enter ur id:");
  Int Id = int.Parse(Console.ReadLine());
  Console.WriteLine(" Enter ur weight:");
  float weight = float.Parse(Console.ReadLine());
  Console.WriteLine("Ur Id is:" + Id);
  Console.WriteLine(" Ur weight is:" + weight);
  Console.ReadLine();
}
```

80) Example :2 for parsing.

```
Void Main()
{
  String S1 = " 123";
  int i1 = int.Parse(S1); //valid
  String S2 = "rama";
  int i2 = int.Parse(S2); //format exception
  String S3 = "4.5";
  float f1 = float.Parse(S3); //valid
  int i3 = int.Parse(S3); //format exception.
  String S4 = " 1234567255432 14";
  int i4 = int.Parse(S4); // overflow
  String S5 = null;
  int i5 = int parse(S5); // Argument null exception.
```

8) Implement multiple programs for different parsings

3. Converting :-

→ To implement converting we have to use a predefined class called Convert.

Convert :

→ Convert is a predefined ~~calls~~ class which is part of system base class library.

→ within convert class we have all the data type conversion methods

Difference between Parsing and converting :-

Parsing	Converting
1) Parsing will convert from String to any other data type	1) Converting will convert from any data type to any other data type

Difference b/w int.Parse & Convert.ToInt32 :-

Int.Parse	Convert.ToInt32
1) It will convert string to int.	1) It will convert any type to ~~str~~ int.
2) when string variable contains null value int.Parse will throw argumentnullexception.	2) when string variable contains null value Convert.ToInt32 will not throw any exception & will convert the null value into zero.

3. Boxing and UnBoxing :

→ Boxing is a process of converting from value type to Reference type

Ex: Converting from int to object.

→ UnBoxing is a process of converting from Reference type to value type

Ex: Converting from object to int.

Note :-

→ To ~~do~~ Boxing and UnBoxing we can use C++ Style of type casting Syntax.

8i) Example for Boxing and unBoxing.

```
Void Main()
{
    int i = 100 ; // value type
    object obj = (object) i ; // boxing
    Console.WriteLine("obj value is :" + obj);
    int j = (int) obj ; // unboxing.
    Console.WriteLine("j value is :" + j);
    Console.ReadLine();
}
```

8ii) Write a Console prgm to accept a range and display the numbers up to that range.

Prgm

```
Void Main ()
{
Console.WriteLine("Enter ur range:");
int n = int.Parse(Console.ReadLine());
for( int i = 1 ; i <= n ; i++)
  {
  Console.WriteLine(i);
  }
Console.ReadLine();
}
```

83) Console prgm to print sum of given range.

Prgm

O/P

ulong ~~int~~ n

Enter ur range :

ulong [5] ← 5

← sum [15] ← Sum is : 15

Prgm

```
Void Main()
{
  Console.WriteLine(" Enter ur Range :");
  ulong ~~int~~ n = ~~Convert.Int~~ ulong.Parse(Console.ReadLine());
  ulong ~~int~~ Sum = 0;
  for(ulong i = 1; i <= n; i++)
  {
    Sum = sum + i;
  }
  Console.WriteLine(" sum is : " + sum);
  Console.ReadLine();
}
```

84) Console prgm to accept a range & display multiplication of range.

```
Void Main()
{
  Console.WriteLine(" Enter ur range :");
  ulong n = ulong.Parse(Console.ReadLine());
  ulong mul = 1;
  for(ulong i = 1; i <= n; i++)
  {
    mul = mul * i;
  }
  Console.WriteLine("multiplication is : " + mul);
  Console.ReadLine();
}
```

85) Console prgm to find factorial of given no;

```
Void Main()
{
  Console.WriteLine("Enter the range:");
  Ulong n = Ulong.Parse(Console.ReadLine());
  Ulong fac = 1;
  for(Ulong i = n; i>=1; i--)
  {
    fac = fac * i;
  }
  Console.WriteLine("factorial is:" + fac);
  Console.ReadLine();
}
```

n = 5

fac = 1

i = 5; i * = 1 i - -

86) Console prgm to display table for given no.

n (int)

[2]

O/P

Enter ur number:

2

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Prgm

```
Void Main()
{
  Console.WriteLine("Enter the number:");
  int n = int.Parse(Console.ReadLine());
  int tab = 1;
  for(int i = 1; i <= 10; i++)
  {
    tab = tab * i;
    Console.WriteLine(n + " * " + i " = " + n x i);
```

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

87) Console prgm to display no's like below.

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Prgm

```
Void Main()
{
  ~~Console~~.. for(int i=1; i<=5; i++)
        {
         for(int j=1; j<=i; j++)
          {
           Console.Write("i+" ");
          }
          Console.WriteLine();
        }

        Console.ReadLine();
    }
```

88) Implement above prgm by using diff loops combinations.

89) Console prgm to print no's like below with in given range.

n

19

Enter ur range:

o/p

```
9
1
2 2
3 3 3
:
9 9 9 9 9 9 9 9 9
```

```
Void Main()
{
    Console.WriteLine("Enter the range:");
    int n = int.Parse(Console.ReadLine());
```

```
for(int i=1; i<=n; i++)
{
  for (int j=1; j<=i; j++)
    {
     Console.Write(i+"  ");
    }
    Console.WriteLine();
  }
  Console.ReadLine();
}
```

90) Console prgm to print the no's like below within given range

o/p

n 5

Enter ur range:
5

```
1
1 2
1 2 3
 :
1 2 3 4 5
```

prgm

```
Void main()
{
 Console.WriteLine("Enter ur range:");
 int n = int.Parse(Console.ReadLine());
 for(int i=1; i<=n; i++)
 {
   for(int j=1; j<=i; j++)
   {
     Console.Write(j+"  ");
   }
   Console.WriteLine();
 }
 Console.ReadLine();
}
```

n=5

i	j
1	1
2	2
3	1
	2
	3
	1

1
1 2

91) Console prgm to print no's like below within given range.

o/p

Enter ur range:

n
5 ← 5

5 5 5 5 5
4 4 4 4
3 3 3
2 2
1

```
Void Main()
{
 Console.writeline("Enter ur range:"));
 int n = int.Parse (Console.ReadLine());
for (int i=n; i>=1; i--)
 {
  for(int j=1; j<=i; j++)
   {
    Console.Write( i);
   }
   Console.Writeline();
  }
  Console.Readline();
}
```

92) Console prgm to display no's like below within given range

o/p

Enter ur range
5
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```
Void Main()
{
 Console.WL("Enter ur range:");
 int n= int.parse(Console.ReadLine());
```

```
for(int i=n; i<=1; i--)
{
  for(int j=0; j>=i; j++)
   {
    Console.Write(j+" ");
   }
  console.Writeline();
}
Console.ReadLine();
```

→ Console prgm to display the no's like below.

```
main()
{
 int k=1;
 for(int i=1; i≤4; i++)
 {
  for(int j=1; j≤i; i++)
   {
    Console.Write(k+" ");
    k++;
   }
 console.Writeline();
 }
 Console.ReadLine();
}
```

o/p

```
1
2 3
4 5 6
7 8 9 10
```

→ console program to display the no's like below within the range

```
main()
{
 console.Writeline("enter your range");
 ulong n = ulong.Parse(console.ReadLine());
 int k=1;
 for(ulong i=1; i<=n; i++)
 {
   for(ulong j=1; j<=i; j++)
    {
      if(k<=n)
      console.Write(k+" ");
      k++;
    }
```

```
    Console.WriteLine();
  }
  Console.ReadLine();
}
```

Hw

1.

```
for(i=1; i<=4; i++)
{ for(j=1; j<=i; j++)
  { Cwl("*");
  }
  C.wL();
}
```

```
*
* *
* * *
* * * *
```

2.

```
* * * *
* * *
* *
*
```

3.

```
    *
  * *
* * *
* * * *
* * * * *
```

4.

```
* * * * *
 * * * *
  * * *
   * *
    *
```

5.

```
      1
    2   2
  3   3   3
4   4   4   4
5   5   5   5   5
```

→ console prgm to accept Student name and accept 3 Sub marks they are m1, m2, m3 and calculate the total and avg, If avg >=60 first class, if avg in between (50 to <60) second class, if avg >=35 and <50 third class If Student is getting in any one of the sub <35 then result is fail.

```
main()
{
  Console.WriteLine("enter Student name:");
  String Sname = (Console.RL());
  Console.WL("enter m1 marks:");
  int m1 = int.Parse(Console.ReadLine());
  Console.WL("enter m2 marks:");
  int m2 = int.Parse(Console.RL());
```

```
console.WL("enter m3 marks:");
int m3 = int.Parse(Console.ReadLine());
int total = m1+m2+m3;
float avg = total/3;
~~else~~ if (m1 <35 || m2<35 || m3<35)
{
console.WL(Sname+" result is fail");
}
else if (avg>=60)
{
console.WL(Sname +"result is 1st class");
}
else if (50>=avg)
{
Console.WL(Sname+"result is 2nd class");
}
else
{
Console.WL(Sname +" result is 3rd class");
}
console.RL();
}
```

O/P

```
enter student name:
Sneha.
enter m1 marks:
70
enter m2 marks:
70
enter m3 marks;
70
Sneha result is 1st class.
```

Q1) Write a prgm to print no's like below within given range.

console prgm to accept user age & display user status

→ Write a console program to accept user age and display user status.

age >= 58 → senior citizen
age → 25-57 → working
age → 16-24 → clg student
3-15 → school boy
1-3 → playing kid

```
main()
{
  console.writeline("enter u age");
  int age = int.Parse(console.Readline());
  if (age>=58)
  { console.writeline("senior citizen"); }
  else if (age>=25)
  { console.writeline("working citizen"); }
  else if (age>=16)
  { console.writeline("clg student"); }
  else if (age>=3)
  { console.writeline("school student"); }
  else if (age>=1)
  { console.writeline("playing kid"); }
  else
  { console.writeline("invalid age"); }
}
```

→ write a console program to accept customer name and customer consumed units then accept customer type and calculate the electricity bill and display to the user?

Industry

cost per unit is = 7Rs/-

Residential

cost per unit is = 6 Rs/-

agricultural

cost per unit is = free

Total bill = units × 7
= 120 × 7
= 840

O/P

cname
↑
Jhon ← Jhon↵

Enter customer name:

cunits enter customer units:
120 ← 120↵

Enter customer type:
press 1 for Industry:
press 2 for residential
press 3 for agriculture:

ctype
1 ← 1↵

Jhon total bill is 840

ctype

1 ← 1

John total bill is: 840

Industry cost per unit = 7 Rs

Residential cost per unit = 6 Rs

Agriculture cost / unit = free

total bill = cunit * ~~ctype~~ 7

840 = 120 × 7

= 100 (7)

Prgm

```
Void Main()
{
  Console.WriteLine("Enter customer name:");
  String cname = Console.ReadLine();
  Console.WriteLine("Enter customer units:");
  int cunits = Int.Parse(Console.ReadLine());
restart:
  Console.WriteLine("Enter customer type:"+"\n"+"press 1 for Industry"+
        "\n"+"Press 2 for residence"+"\n"+"Press 3 for Agriculture");
  Char ctype = char.Parse(Console.ReadLine());
    int
  Switch(ctype)
  {
    Case '1':
      Console.WriteLine(cname+" total bill is:" (type*7));
      break;
    Case '2':
      Console.WriteLine(cname+" total bill is:" (type* 6));
      break;
    Case '3':
      Console.WriteLine(cname+" total bill is:" (type * 0));
      break;
    default:
      Console.WriteLine("Invalid input");
      ~~break;~~ goto restart;
    }
    Console.ReadLine();
  }
```

Q3) Implement the above prgm with diff requirement like below

for industry

up to 100 cost/unit — 7 Rs

if (> 100) → cost/unit — 8 Rs

for residential

up to 100 cost/unit — 6 Rs

if (> 100) cost/unit — 7 Rs

for agriculture cost/unit — 0/-

Prgm

```
// above lines are same
switch (ctype)
{
  case '1':
      if (cunits > 100)
       {console.write(Cname + " total bill is : " Cunits * 8);
       }
      else
       {console.write(Cname + "total bill is :" cunits * 7);
       }
     break;
  case '2': if (cunits > 100){
      console.write (Cname + " total bill is : " cunits * 7);
      }console.write(
     else
      console.write (Cname + "total bill is : " Cunits * 6);
     break;
  case '3':
      if (cunits > 100)
      console.write (Cname + "total bill is : zero");
      break;
  default:
      Console.write (Cname + "Invalid type");
      goto restart;
}
Console.ReadLine();
}
```

104) Implement above pgm with small changes like below

Industry

if 120 units up to 100 * 7 } tot bill is = 860
20 × 8

Residential.

120 units up to 100 * 6
20 * 7 740/-

Agriculture

bill is 0.

105) Implement above student marks prgm using switch case

106) Implement above user status prgm by using switch case.

107) Implement above electricity bill prgm by using if else if.

108) Implement all above 3 prgms by using nested if.

109) Console prgm to swap 2 no's.

o/p

```
i        Enter i value:
[10] ← 10
j        Enter j value:
[5]      5
         i value is : 5
         j value is : 10
```

Prgm

```
void main()
{
Console.WriteLine("Enter i value:");
int i = Int.Parse(Console.ReadLine());
Console.WriteLine("Enter j value:");
int j = Int.Parse(Console.ReadLine());
int temp = i;
    i = j;
    j = temp;
    Console.WriteLine("i value is :"+i);
    Console.WriteLine("j value is:"+j);
Console.ReadLine();
}
```

i = 5
j = 10

temp = i
[5]
i = j
[10]
j = temp
[5]

110): Implement above prgm without using temp variable.

```
Void Main()
{
Console.WriteLine("Enter i value:");
int i = int.Parse(Console.ReadLine());
Console.WriteLine("Enter j value:");
int j = int.Parse(Console.ReadLine());
i=i+j;
j=i-j;
i=i-j;
Console.WriteLine("i value is:"+i);
Console.WriteLine("j value is:"+j);
Console.ReadLine();
}
```

i = 5
j = 10

i = i+j
= 5+10 ⇒ 15
j = i-j
= 15-10 ⇒ 5
i = i-j
= 15-5 ⇒ 10

111) Console prgm to find biggest among 3 no's.

```
Void Main()
{
Console.Write("Enter first no:");
int a = int.Parse(Console.ReadLine());
Console.Write("Enter second no:");
int b= int.Parse(Console.ReadLine());
Console.Write("Enter third no:");
int c = int.Parse(Console.ReadLine());
if(a>b && a>c)
{Console.Write("a is biggest");
else if(b>c)
Console.Write("b is biggest");
else
Console.Write("c is biggest");
Console.ReadLine();
}
```

Note:- Above prgm is giving wrong o/p when user entered same no's.

112) write the prgm to this above problem. using nested if and also Switch

113) Implement above prgm using simple if

114) console prgm to display smallest no among 3 no's.

```
Void Main()
{
Console.Write("Enter first no:");
int a = int.Parse(Console.ReadLine());
Console.write("Enter second no:");
int b= int.Parse(Console.Read line());
Console.write("Enter third no:");
int c = int.Parse(Console.Read line());
if (a>b && a>c)
  {
    if (a>b)
      Console.WriteLine("a is biggest");
    else if(a==b)
      console.writeLine("a & b is equal");
  }
else if (a>b)
```

a = 5 5 8
b = 5 8 5
c = 8 5 5

115) Implement above smallest no prgm using nested if.

116) Console prgm to print biggest among 4 no's
117) smallest among 4 no's
118) second biggest among 4 no's
119) second smallest among 4 no's

if else if
nested if
nested switch
Switch with if else

120) Biggest no among 5 no's.
121) Smallest no among 5 no's
122) ~~Bigg~~ second biggest & second smallest no.
123) third biggest no.

124) Console prgm is even or odd

Example for even

2, 4, 6 - - -

Example for odd

1, 3, 5 - - -

% → Modular operator

/ → division operator.

divisor (2) 4 (2 → Quotient
4
0 → reminder
↗ divident

→ with the help of modular operator(%) we get reminder

→ with the help of division operator(/) we get Quotent.

Ex:

```
Void Main()
{
  Consol.  int i = 4%2;
           Console.WriteLine(i);
           int j = 4/2;
           Console.WriteLine(j);
           Console.ReadLine();
}
```

o/p

0
2

124) o/p

Enter ur no:

n
4 ← 4

It is even number.

Prgm

```
Void main()
{
Console.Writeline("Enter your no:");
int n = int.Parse(Console.ReadLine());
if (n % 2 == 0)
    Console.WriteLine("It is even number");
```

```
else
    Console.WriteLine(" It is odd number");
    Console.ReadLine();
}
```

125) Console prgm to display the list of even no's with in given range & display the sum of even no's & ~~no~~ number of given no's within that range.

<u>o/p</u>

```
n        Enter the range:
[10] <-- 10
         List of ~~gi~~ Even no's with in 10
         2
         4
         6
         8
         10
Sum
[30] <-- Sum of even numbers are : 30
ctr      no of even numbers are : 5
[5]
```

Sum	ctr	i
~~0~~	~~0~~	~~2~~
~~2~~	~~1~~	~~4~~
~~6~~	~~2~~	~~6~~
~~12~~	~~3~~	~~8~~
~~20~~	~~4~~	~~10~~
~~30~~		

1 — 10

2

```
Void Main()
{
Console.WriteLine("Enter ur range:");
int n = int.Parse(Console.ReadLine());
Console.writeline("List of even no's with in"+n);
int sum=0; ctr=0;
for(int i=2; i<=n; i=i+2)
{
Console.writeline(i);        10        c
}
int sum = sum+i;             0+2       5
ctr++;
Console.writeline("Sum of even numbers are:"+sum);
Console.writeline("No. of even numbers are:"+ctr);
Console.Readline();
}
```

HW 126) Implement above prgm using ~~logical prgm~~ Modular operator.

```
Void Main()
{
Console.WriteLine ("Enter your range:");
int n = int.Parse (Console.ReadLine());
Console.WriteLine (" List of even no's are:");
 int sum=0, ctr=0;
for(int i=2; i<=n; i=i+2)
  {
   Console. ~~WriteLine(i);~~
   if (n % 2 == 0)
   {Console.WriteLine(i);
    Sum = Sum + i;
   } ctr++;
   Console. WriteLine("Sum of even no's are:"+ Sum);
   Console. ReadLine();
  }
```

O/P

```
Enter ur range:
10
List of even no's are:
2
4
6
8
10
Sum of even no's are: 30.
```

127) Console prgm to display the odd no's with in given range and display the sum & list of odd no's. with in range

1) with out using modular operator 2) By using modular operator.

128) Console prgm to check the given no is a prime no or not.

Prime no: Number which is divisible by 1 and it self

```
Void Main()
{
 Console.WriteLine("Enter ur number:");
 int n = int.Parse(Console.ReadLine());

 ~~if(n%n==1 && n!1)~~
int ctr=0;
 ~~Console.Write~~
 for(int i=2; i<=n; i++)
 {  if(n%i=0)
    { ~~Console.WriteLine(~~
      ctr++;
      break;
    }
 } if(ctr==0)
   {  Console.WriteLine("prime no");
   }
   else
     Console.WriteLine("not a prime no");

   Console.ReadLine();
}
```

10|10

n=5 i=2

2)5(2
y
1

2)6(

2

i=2

2%2=0

ctr=1

4
2 4 2
2

#F)

129) Console prgm to display list of prime no's, sum of prime no's, no. of prime no's with in given range.

2,3,5,7,11,13, (15)

1 2 i=2 i<=n i++

2)15(

2)2=0

2)2=2

```
Void Main()
{
Console.WriteLine("Enter ur range:");
int n = int.Parse(Console.ReadLine());
int sum=0, ctr=0;
for(int i=2; i<=n; i++)
 {  if(n%i==0)
    {  Console.WriteLine(i);
       ctr++;
       ~~break~~;
    }
```

130) Console pgm to reverse the given no

O/P

n Enter ur no:

$\boxed{123}$ ← 123

reverse no is : 321

Pgm

```
Void Main()
{
Console.WriteLine("Enter your no:");
int n = int.Parse(Console.RL());
int rem, rev=0;
while (n != 0)
 {
   rem = n % 10;
   rev = rev * 10 + rem;
   n = n/10;
 }
 Console.WL("Reverse number is:"+rev);
 Console.RL();
}
```

1st Interation

123!=0 → T

rem = 123 % 10

$\boxed{3}$

rev = 0 * 10 + rem

↳ 0 = 0 * 10 + 3

rev = 3

n = 123/10

$\boxed{12}$

2nd interation

12 != 0 → T

rem = 12 % 10

$\boxed{2}$

rev = 3 * 10 + rem

↳ 3 = 30 + 2

rev = 32

n = 12/10

$\boxed{1}$

3rd interation

1 != 0 → T

rem = 1

$\boxed{1}$

rev = 32 * 10 + rem

32 = 32 * 10 + 1

rev = 321

n = 1/10

$\boxed{0}$

4th interation

0 != 0

false

131) console prgm to check whether given no is a palindrome or not

```
Void Main()
{
  console.Writeline("Enter your no:");
  int n = int.Parse(console.RL());
  int rem, rev=0, temp=n;
  while(n!=0)
  {
    rem = n%10;
    rev = rev*10+rem;
    n = n/10;
  }
  console.WL("Reverse number is:"+rev);
  if(temp == rev)
      console.WL("It is palindrome");
  else
      console.WL("It is not palindrome");
      console.RL();
}
```

n=121

temp = n
= 121

temp = rev → T
121 → 121

while(n!=0)

rem rev = 0

rem = n%10

rev = rev*10 + rem

n = n/10

132) Console prgm to reverse the given string.

```
Void Main()
{
Console.WL(" Enter a name: ");
String a = Console.ReadLine();
String b = null;
for (int i = a.length-1 ; i>=0 ; i--)
    {
    ~~Console.WriteLine(a[b]);~~
    b = b+a[i];
    }
    Console.WL(b);
    Console.ReadLine();
}
```

o/p

Enter a name:
rama
reverse name is: amar

a = rama
b=4 b<=4

133) Console prgm to check given string is a polyndrome or not.

```
Void Main()
{
Console.WL("Enter a name:");
String a = Console.RL();
String b = null;
for(int i = a.length-1 ; i>=0 ; i--)
  {
    b = b+a[i];
  }
  if (a==b)
    Console.WriteLine("It is polyndrome");
  else
    Console.WL(" not a polyndrome");
    Console.RL();
}
```

mam

a= rama
b=
i=4-1, i>=0, i--
b=ra
amar
mam

134) console pgm to generate fabonoci series

```
void Main()
{
 int a=0, b=1, c=0;
 while (a<=100)
 {Console.Write(a+" ");
   c=a+b;
   a=b;
   b=c;
 }
 Console.ReadLine();
}
```

0 = 0+1
0 = 1

a=0	b=1	c=0
1	2	1
2	3	2
3	5	3
5	8	5
8	13	8
13	21	13
21	34	21
34	55	34
55	89	55
89		89

o/p

0 1 1 2 3 5 8 13 21 34 55 89 –

0+1 = 1, 1+1 = 2, 1+2=3, 2+3 = 5, 3+5=8

Object oriented programming System :-(OOPS)

Q) What is Object Oriented approach?

Ans. Object oriented approach is a methodology to develop computer programs by using classes and objects.

Q) what do you mean by object oriented programming language?

Ans A programming language which will follow object oriented approach is called as object oriented programmig language

Ex: C++, C#.Net, VB.Net.....

Q) what do you mean by Object oriented programming principles?

Ans → OOps principles are 4

1. Encapsulation
2. Abstaction
3. Inheritence
4. Polymorphism.

→ Every object oriented programming language has to follow the above 4 principles.

→ To achive the oops principles every object oriented programming language will use two concepts

1. class
2. Object

1. Class:-

→ class is a collection of data members and member functions.

→ Data member can be called as state (or) Variable (or) field.

Q) what is state?

Ans State will represent some value.

→ Member function can be called as method (or) function. (or) behaviour.

Q) what is behaviour?

Ans Behaviour is nothing but functionality (or) logic.

→ Finally we can say class is a collection of States and behaviours

→ To define a class we have to use class Keyword

→ <Syntax to define class>:-

```
<access modifier> class <class name>
{
    // states   // variables
    // behavious  // methods
}
```

→ Class is a logical representation, when we define class no memory will be allocated for class members.

→ 135) List out the states & behaviours of human being:

States:
→ name
→ age
→ height
→ weight

methods:
→ eating
→ speaking
→ walking

→ 136) List out the states & behaviors of employee & student & customer

employee	Student	customer
States:-	States	States
→ Salary	→ Marks	
→ Position	→ class	
methods	methods	methods
	→ fee payment	bill payment
	→ attendence calc.	

Q) When we will go for State?

Ans Whenever we want to represent some value we will go for State

Ex: empno, emp name, empage - - - -

Types of Variables:-

→ C#.Net will support 3 types of variables

1. Local variable.
2. Instance variable (or) Non static variable
3. Static variable.

1. Local Variable :-

→ A variable which is declared within a method is called as Local Variable.

→ Local Variable should be initialized with some value before accessing. otherwise compiler will generate an error.

2. Instance Variable (or) Non-static variable :-

→ A variable which is declared inside class and outside the method without static Keyword is called instance variable.

3. Static variable :-

→ A variable which is declared inside class and outside the method by using Static Keyword is called as Static variable.

Q) When we will go for method (or) behavior?

Ans Whenever we want to implement some functionality we will go for method or behaviour.

Ex: calculating salary, Calculating age (or) calculating marks - - -

→ Syntax to define a method:

<access modifier> <return type> <method name (type> <arg1>, <type> <arg2>) >

{

Method body { // Logic }

Note:→ Argument and parameter both are Same.

→ According to parameters, methods are classified into 2 types

1. Parameter less method
2. Parameterized method

1. Parameter less method.

→ while defining a method if we dint declare any parameters which is called as Parameter less method.

Q) When we will go for Parameter less method.

Ans → According to method functionality if the method doesnt required any value then we will go for parameter less method.

Ex:
```
class Program
{
    Void Display( )
    {
        Console.WriteLine("Welcome to C#.Net");
        Console.WriteLine("Welcome to ASP.Net");
    }
    Void Main( )
```

2. Parameterized method:

→ while defining a method if we have declared parameter which is called as parameterized method.

Q) When we will go for parameterized Method?

Ans According to requirement whenever a method is required some value for its functionality then we go for parameterized method.

Ex:
```
class Program
{
    Void add(int a, int b)
    {
        int c = a+b;
        Console.WriteLine("Result is : "+c);
    }
    Void Main()
```

→ According to method return types methods are classified in to 2 types

1. Void Method
2. Non-void Method.

1. Void Method:

→ A method which is not returning any value according to its functionality is called as void method

→ Void Method return type should be void.

Q) when we will go for void Method?

Ans According to the requirement if the method doesnt require to return any value we will go for void Method.

2. Non-Void Method:

→ A method which is returning some value is called Non-void method.

Q) when we will go for non-void Method?

Ans → According to method functionality if it requires to return some value we will go for non-void method.

→ The return type of method will be depending on type of the value which is returning by the method.

→ If the method ~~is returning~~ requires to return numerical value then method return type can be any one of the numerical data type like int, long, ---

→ If the method is returning value is floating value then the method return type can be float, double, decimal.

Eg:

```
int fun1 (int a)
{
    return a;
}
```

```
char fun2 (char a)
{
return a;
}
```

→ Again Method are classified into 2 types

1. Instance Method (or) Non-static Method
2. Static Method

1. Instance Method:

→ While defining a method if we dont use static keyword which is called as instance method.

→ If we want to access instance methods we have to access with the help of object

Ex:
```
bool fun3(bool a)
{
return a;
}
```

2. Static Method

→ while defining a method if we are using static keyword which is called as static method.

→ If we want to access static method we have to access with the help of class name.

Memory allocation:-

→ No memory will be allocated for class

→ No memory will be allocated for methods

→ Memory will be allocated for variables

Memory allocation for instance variable:-

→ When the object is created for a class concern class instance variables will be allocating memory.

→ When an object is destroyed concern class instance variables memory will be deallocated ~~memory~~.

→ In .Net object will be destroyed in 2 ways

1) by garbage collector

2) by Programmer.

Memory allocation for local variables :-

→ As part of method execution memory will be allocating for local Variable.

→ when the method execution is completed local variable are destroyed.

Structure of a class with instance variables, instance method, local variables :

```
class Myclass
{
    int a;
    int b;      } → instance variables
    void add (int x, int y)   → instance method
                  └──┴──→ parameters
    {
        a = x;
        b = y;
        int res = a+b;
                  └──→ Local variable
    }
}
```

Note:-

→ Parameter will be acting like a local variable.

→ In the above class we have a method called add which is Consuming 2 instance variables which are a, b

→ If we want to access above my class add method, it requires to allocate memory for instance variables. they are a, b. for this we require Object.

Purpose of object:

→ To allocate memory for instance variables.

What is an Object:-

→ An object is a instance of a class which is physical representation of a class.

→ For example. human being is a class, Ravi is a object of human being class.

→ when we create an object for a class memory will be allocated for concern class instance variables.

What object will contain:

→ An object can contain concern class instance variable and concern class instance methods references.

→ Syntax to create a object:-

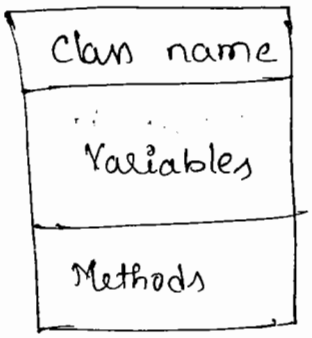
```
<class name> <objectname> = new <classname()>;
```

→ Syntax to invoke instance methods:-

```
<objectname>.<methodname()>;
```

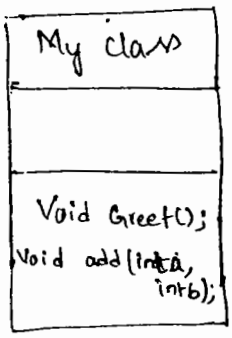
Programming in oops:-

Structure of the UML class diagram.



1) write a console program to define a class with two instance methods.

1) Void Method without parameters
2) Void Method with parameters.



o/p

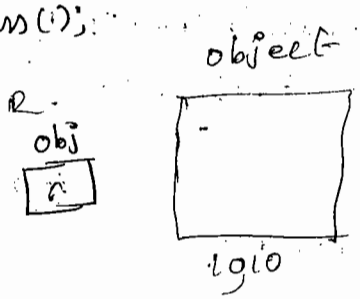
welcome to OOPS.

Addition result is : 15

Prgm)

```
namespace classExample1
{
  class Myclass
  {
internal Void Greet();
        {
          console.writeline("welcome to OOPS");
        }
internal Void add(int a, int b);
        {
          ~~int a = 5,~~
          ~~b = 10~~;
          int res = a+b;
        Console.WriteLine("Result is:"+res);
        }
  }
  class program
  {
    Void Main()
    {
```

```
Myclass obj = new Myclass();
obj.Greet()
obj.Add(10,5);
Console.ReadLine();
} // main
} // class
} // namespace.
```



2) Console program Example to above program by passing a, b values by user.

o/p

~~Addition~~ ~~result is~~ :

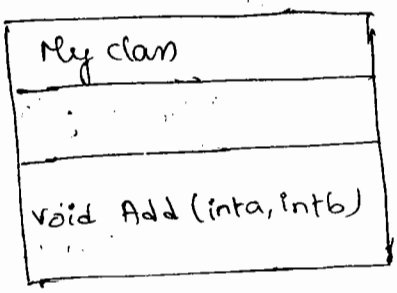
Enter first no :

20 ↵

Enter second no :

10 ↵

Addition result is : 30



Pgm

```
class Myclass
{
  Internal void Add (int a, int b)
  {
    int res = a+b;
    Console.WriteLine("Addition result is: "+res);
  }
}
class program
{
  Void Main()
  {
    Console.WriteLine("Enter first no: ");
    int x = int.Parse(Console.ReadLine());
```

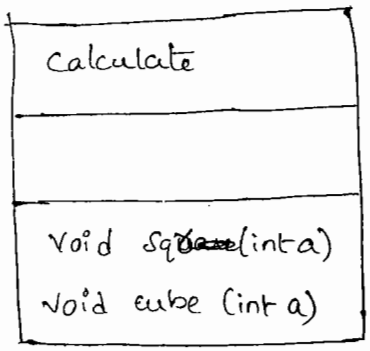
```
        Console.WriteLine("Enter second no:");
        int y = int.Parse(Console.ReadLine());
        Myclass obj = new Myclass();
          obj.Add(x, y);
        Console.ReadLine();
    }
  }
}
```

3) Implement the above Add method as a non-void method.

```
Class Myclass
{
    Internal int add(int a, int b)
    {
        int res = a+b;
        return res;
    }
}
Void Main()
{
    Console.WriteLine("Enter first no:");
    int x = int.Parser(Console.ReadLine());
    Console.WriteLine("Enter second no:");
    int y = int.Parse(Console.ReadLine());
    int i =
    Myclass obj = new Myclass();
      int i = Obj.Add(x,y);
    Console.WriteLine("Addition result is: "+i);
    Console.ReadLine();
}
```

4) Console prgm to calculate the following two.

1) Calculating Square
2) calculating cube.



o/p

Enter your number:
2
Sqr is: 4
Cube is: 8

prgm

```
namespace ConsoleExample4.
{
  class Calculate
  {
    internal Void Sqr(int a)
      {
        int b = a*a;
        Console.writeline("Sqr is:"+b);
      }
    internal void cube(int a)
      {
        int c = a*a*a;
        Console.writeline("Cube is:"+c);
      }
  }
  class program
  {
    Void Main( )
      {
        Console.writeline("Enter ur number:");
        int n = int.Parse(Console.ReadLine());
        calculate obj = new calculate();
          obj.sqr(n);
          obj.cube(n);
        Console.Readline();
      }
  }
}
```

5) Implement above 2 methods as non void Methods

```
namespace Examples
{
  ~~Calcu~~
  ~~Internal~~ class calculate
        {
         internal int Sqr(int a)
          {
            int b = a*a;
            return b;
          }
         internal int cube(int a)
          {
            int c = a*a*a;
            return c;
          }
        }
        class programm
        {
          void Main( )
          {
          Console.WriteLine("Enter ur no:");
          int n = intParse(Console.ReadLine());
          Calculate obj = new calculate();
          int i = obj.Sqr(n);
          Console.WriteLine("Square result is:"+i);
          int j = obj.cube(n);
          Console.WriteLine("cube result is:"+j);
          Console.Readline();
          }
        }
}
```

6) Console prgm to perform following operations

1. addition
2. Subtraction
3. division
4. Multiplication

Calculations
void Add(int a, int b) void Sub(int a, int b) void div(int a, int b) void mul(int a, int b)

o/p

Enter first no:

x [10] ← 10

Enter second no:

y [5] 5

addition result is : 15

Sub. result is : 5

div result is : 2

mul result is : 50

Prgm

```
namespace Example6
{
  ~~Calculatio~~
  class calculations
  {
    internal void Add(int a, int b)
    {
      int res = a+b;
      Console.Writeline("addition result is :"+res);
    }
    internal void Sub(int a, int b)
    {
      int res = a-b;
      Console.Writeline("Sub result is :"+res);
    }
    internal void div(int a, int b)
    {
      int res = a/b;
      Console.Writeline("div result is:"+res);
    }
    internal void mul(int a, int b)
    {
      int res = a*b;
      Console.Writeline("Mul result is :"+res);
    }
```

```
class program
{
    void Main();
    {
     Console.WriteLine("Enter first no:");
     int x= Int.Parse(Console.ReadLine());
     Console.WriteLine("Enter second no:");
     int y= int.Parse(Console.ReadLine());
     Calculations obj = new calculations();
      obj.add(x,y);
      obj.Sub(x,y);
      obj.div(x,y);
      obj.Mul(x,y);
      Console.ReadLine();
    }
   }
  }
```

6) Implement above prgm to define 4 methods as non-void method.

```
namespace Example6
{
  class calculations
   {
     internal int add(int a, int b)
      {
        int res=a+b;
        return res;
      }
     internal int Sub(int a, int b)
     {
       int res = a-b;
       return res;
     }
     internal int div(int a, int b)
     {
       int res = a/b;
       return res;
     }
```

```
internal int mul(int a, int b)
 {
   int res = a * b;
   return res;
 }
}
class program
{
  Void Main()
  {
    Console.WriteLine(" Enter first no:");
    int x = int.Parse(Console.ReadLine());
    Console.WriteLine (" Enter second no:");
    int y = int.Parse (Console.ReadLine());
    calculations obj = new calculations();
    int res = obj.add(x,y);
    Console.WriteLine("Addition is : "+ res);
    int res = obj.Sub(x,y);
    Console.WriteLine ("Sub is : "+ res);
    int res = obj.div(x,y);
    Console.WriteLine ("div is : "+ res);
    int res = obj.mul(x,y);
    Console.WriteLine("mul is : " +res);
    Console.ReadLine();
  }
}
}
```

7) Implement above prgm by defining div method for validating the second number should not be zero.

Prgm

```
namespace example7
{
class calculations
  {
```

```
internal int div(int a, int b)
{
    while ( b==0)
    {
      Console.WriteLine("Enter other than zero");
      b = int.Parse(Console.RL());
    }
    return a/b;
}
}
class program
{
  void main()
  {
    Console.WL(" Enter first no:");
    int x = int.Parse(Console.RL());
    Console.WL(" Enter second no:");
    int y = int.Parse (Console.RL());
    Calculations obj = new calculations();
  int res = Obj.div(x,y);
  Console.WriteLine("div result is : "+res);
  Console.RL();
  }
```

8) Define class called student with two behaviors:

1. Display student info
2. Calculating student result.

O/P

Student
void StudentInfo(int SId, String Sname)
void getresults (int m1, int m2, int m3)

```
namespace Example8
{
  class Student
  {
    internal void StudentInfo(int SId, String sname)
    {
      Console.WL("Student Id is:" + SId);
      Console.WL(" Student name is :" + sname);
    }
    internal void StudentResults (int m1, int m2, int m3)
    {
      Console.Writeline("M1 marks is :"+m1);
      Console.Writeline("M2 Marks is :"+m2);
      Console.writeline (" M3 marks is :"+m3);
      if(m1<35 || m2<35 || m3<35)
        {
          console.WL(" Fail");
        }
      else if (avg>=60)
          Console.WL("First class");
      else if (avg>=50)
          Console.WL(" Second class");
      else
          console.WL("Third class");
    }
  }
class program
  {
    void Main( )
    {
      Student obj = new Student();
      Console.WL("Enter student id : ");
      int sid = int.Parse (Console.RL());
      Console.WL("Enter student name : ");
      String sname = Console.RL();
        obj.StudentInfo(sid, sname)
      // Entering m1, m2, m3 marks by user
        Obj.StudentResults (m1, m2, m3);
        Consoleo.RL();
    }
  }
}
```

int tot = m1+m2+m3;
int avg = tot/3;

9) Define a class called employee with two ~~fur~~ behaviours.

1. accepting empId, empname, emp Sal.
2. Displaying emp Id, empname, empsal.

~~Q) when to~~

employee
int EmpId; String EmpName; double Sal;
void AcceptEmpInfo(); void displayEmpInfo();

Prgm

```
namespace instance Ex1
{
  Class employee
  {
     int EmpId;
     String Empname;
     double Sal;
internal void AcceptEmpInfo( )
     {
      Console.WL(" Enter Employee ID: ");
      ~~int A =~~ EmpId = int.Parse(Console.ReadLine());
      Console.WL(" Enter Empname:");
      Empname = Console.RL();
      Console.WL(" Enter empsal: ");
      Sal = double.Parse(Console.RL());
     }
internal void displayEmpInfo()
     {
      Console.WL(" Employee Id is: "+ EmpId);
      Conside.WL (" Eployee name is: "+ empname);
      Console.WL(" Salary is:"+ Sal);
     }
}
```

```
class program
{ Void Main().
  {
    employee obj = new employee();
    obj.AcceptEmpinfo();
    obj.displayEmpinfo();
    employee obj2 = new employee();
    obj2.Acceptempinfo();
    obj2.displayempinfo();

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

→ In the above example we have declared 3 instance variables
1. Empid 2. Empname 3. Empsal.

→ Instance variable will be related to instance object.

→ That means memory allocation and memory deallocation will be at the tym of creating object and deallocation is at the time of destroying object.

→ Similarly instance variable should be initialized as part of object creation.

→ But in the above example instance variables are initializing after creating the object because we have used a mechanism called method to initialize instance variables.

→ At the time of creating object to initialize instance variables we have a separate mechanism called constructor.

Constructor:

→ Constructor is a member of a class which is a special type of function

→ Constructor will invoke automatically when an object is created

→ Constructor will not return any value so it will not have return type.

→ class name and constructor name should be same.

→ Constructor can contain parameters

→ Syntax to define constructor :-

```
<accessmodifier> <classname()>
  {
    // intialization code.
  }
```

10) Example to understand constructor Execution

```
namespace constructorEx
{
  class Myclass
  {
    // instance variables
    int a;
    int b;
    Internal Myclass()
    {
      Console.Writeline("Constructor is calling");
      a=10;
      b=20;
    }
    //Instance Method
    Internal void display()
    {
      Console.Writeline("a value is :"+a);
      Console.Writeline("b value is :"+b);
    }
  }
  Class program
  {
    Void Main()
    {
      Console.Writeline("Before object is created");
```

```
Myclass obj = new Myclass();
```

step 4 → = new; step 2 → Myclass(); step 3 → Myclass obj; step 1

```
        Console.WriteLine("After object creation");
        Obj.Display();
        Console.ReadLine();
    }
}
}
```

op

Before object is created
Constructor is calling
After object creation
a value is : 10
b value is : 20

```
Myclass obj = new Myclass();
```

step 4 → = new; Step 1 → Myclass obj; Step 2 → new; Step 3 → Myclass()

The above statement will execute in 4 steps.

Step 1: Creating reference variable

Step 2: Creating object [as part of this memory is allocating for instance variables]

Step 3: Invoking Constructor & will execute constructor [as part of constructor execution it will initialize default values to instance variables & after that it will initialize the given values to instance variables].

Step 4:

Step 4: Assigning object address into reference variable.

Types of Constructors :-

→ Constructors are mainly 2 types

1. Instance constructor (or) non-static constructor
2. Static Constructor.

1. Instance Constructor:

→ While defining a constructor if we dont use static keyword it is called as instance constructor.

→ Purpose of instance constructor is to initialize instance variables

→ Within instance constructor, we can initialize instance & static variables.

→ But if we will initialize static variables within instance constructor that static variables will loose the static nature.

→ Instance constructor can contain parameters.

→ Instance constructors are classified into 4 types.

1. Default constructor (or) parameterless constructor.
2. Parameterized constructor
3. Copy constructor
4. Private constructor.

1. Default Constructor:

→ If a constructor is not having any parameters, which is called as parameter less constructor

→ A class can contain only one default constructor

→ Default constructors are two types

1. User defined default constructor
2. System defined default constructor.

1. User defined default Constructor:

→ If a programmer defines a constructor which is not having any parameters is called as user defined default constructor.

11) Example for Userdefined default Constructor

```
namespace UserDefinedDefaultConstructor
{
  class Employee
  {
    int eno;
    string ename;
    double esal;
    internal Employee()
    {
      eno = 111;
      ename = rama;
      esal = 1000;
    }
    internal void Display()
    {
      Console.WriteLine("Employee no is: "+eno);
      Console.WriteLine("Ename is: "+ename);
      Console.WriteLine("Esal is:"+esal);
    }
  }
  class program
  {
    void Main()
    {
      Employee obj = new Employee();
      obj.Display();
      Console.ReadLine();
    }
  }
}
```

drawback :

→ Default constructor will initialize the same values to all the objects.

→ To overcome this we will go for parameterised constructor.

Purpose of user defined default constructor

→ In Real time we will not use default constructor to initialize instance variables.

→ we will use default constructor to declare the conection string and soon.

2. System defined default constructor :

→ when we compile the program within a class, if programmer dont define any instance constructor then compiler will generate one default constructor which is called as system defined default constructor.

Ex: namespace systemdefineddefaultconstructor

```
{
  class Employee
  {
    int eno;
    string ename;
    double esal;
    internal void Display()
    {
      Console.WL("Emp NO is:"+eno);
      Console.WL("Ename is:"+ename);
      Console.WL("Esal is:"+esal);
    }
  }
  class Program
  {
    Void Main()
    {
```

system defined default constructor

```
internal Employee()
{
  eno = 0;
  esal = 0;
  ename = Null;
}
```

```
        Employee emp1 = new Employee();
        emp.Display();
        Console.ReadLine();
    }
  }
}
```

O/P

Emp no is : 0
Ename is : ~~Null~~
Esal is : 0

Purpose of system defined default constructor :

→ To Initialize default values to the instance variables

2. Parameterized Constructor:

→ While defining a constructor. if we have declared parameters which is called as parameterized constructor.

→ A class can contain multiple Parameterized constructors but we should differentiate with no. of parameters (or) ~~order of~~ type of Parameters (or) order of parameters is nothing but diff signature like below.

→ For Example

```
Class employee
{
    int eno;
    String ename;
    double esal;
    // 1. Parameterized constructor
    Internal Employee (int Eno)
    {  eno = Eno;
    }
    internal Employee (String Ename)
    {  ename = Ename;
    }
    internal Employee (double Esal)
    {  esal = Esal;
    }
```

→ Signature

Q) what signature will represent?

Ans- Signature will represent no. of parameters, type of parameters, order of parameters.

What is constructor overloading

→ Defining multiple constructors with different constructors with different signature in a single class is called as Constructor overloading.

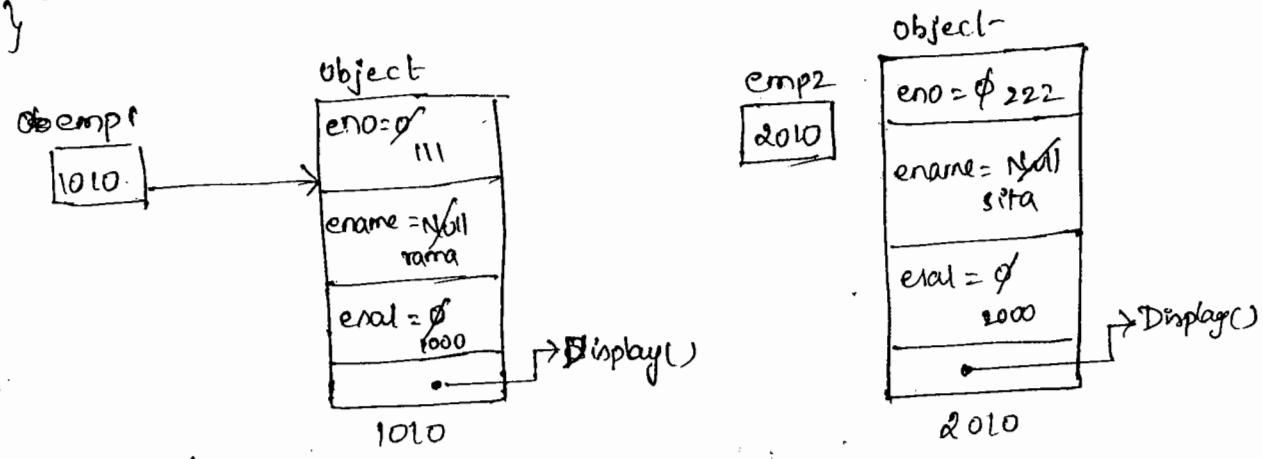
Purpose of Parameterized Constructor (or) when we will go for Parameterized Constructor :-

→ To initialize different values, Object to object we will go for Parameterized constructor.

→ Example for parameterized Constructor

```
namespace parameterizedConstructor
{
  class parameterconst
  {
    int eno;
    string ename;
    double esal;

    internal parameterconst( int eno, string ename, double esal)
    { eno = eno; ename = ename; esal = esal; }
    internal void Display( )
    { Console.WL(" Employee no."+eno);
      Console.WL(" Employee name:"+ename);
      Console.WL(" Employee salary:"+esal);
    }
  }
  Class program
  {
    void Main()
    {
```

```
Employee emp1 = new Employee(111, "rama", 1000);
emp1.Display();
Console.WriteLine("object address is:"+emp1.GetHashCode());
Employee emp2 = new Employee(222, "sita", 2000);
emp2.Display();
Console.WriteLine("object address is:"+emp2.GetHashcode());
Console.RL();
}
}
}
```



O/P

Employee no : 111

Employee name : rama.

Employee salary : 1000

Q) How to get the address of object.

Ans → By using GetHashCode().

→ GetHashCode is a predefined member method of object class

→ This method will return the given object address

Q) What is the super class for all .Net classes.

Ans Object

```
class object
{
  int GetHashCode()
  {
  }
}
class Employee: object
{
}
class Program : object
{
}
```

3. Copy Constructor.

→ Using copy constructor we can copy values from one object to another object.

→ Example for CopyConstructor.

```
namespace CopyConstructor
{
  class Employee
  {
    int eno;
    string ename;
    double esal;
    internal Employee(int Eno, String Ename, double Esal)
    {   eno = Eno;
        Ename = Ename;
        esal = Esal;
    }
```

```
//Copy Constructor
Internal Employee(Employee obj)
{
    eno = obj.eno;
    ename = obj.ename;
    esal = obj.esal;
}
Internal void Display()
{
    // Printing values
}
}
Class program
{
    void Main()
    {
        Employee emp1 = new Employee(111, "Rama", 1000);
        emp1.Display();
        Employee emp2 = new Employee(emp1);
        emp2.Display();
        Console.RL();
    }
}
}
```

Q) when we will go for copy constructor?

Ans whenever we want to copy the values from one object to another object we will go for copy constructor.

4. Private Constructor :-

→ while defining a constructor if we have used private access modifier that constructor is called Private constructor.

→ If a class is having Private constructor we cannot Create an object for concern class.

→ Example for Private Constructor.

```
namespace PrivateConstructor
{
  class Employee
  {
    Private Employee()
    {
    }
  }
  class Program
  {
    void Main()
    {
      Employee empl = new Employee();
      Console.ReadLine();
    }
  }
}
```

Note : The above code cannot create an object because we have an private constructor. So it will throw a runtime error.

Static Constructor :-

→ while defining a constructor if we have used static keyword which is called as static constructor.

→ Static constructor purpose is to initialize static variables when the memory will be allocated for static variables :-

→ when the class is loading memory will be allocated for static variables.

→ Memory will be deallocated for static variables when the class is unloaded.

→ When the class is loading first memory will be allocated for static variable then static constructor will execute.

→ If a class is having one static constructor and one instance constructor, first control will execute static constructor then will execute instance constructor becoz class will load first then object will create.

→ Static constructor cannot contain parameters which is by default parameter-less constructor.

→ A class can contain maximum one static constructor.

→ While defining static constructor we should not use access modifier.

→ Within the static constructor we can initialize only static variables we cannot initialize instance variables becoz static constructor will execute before creating the object

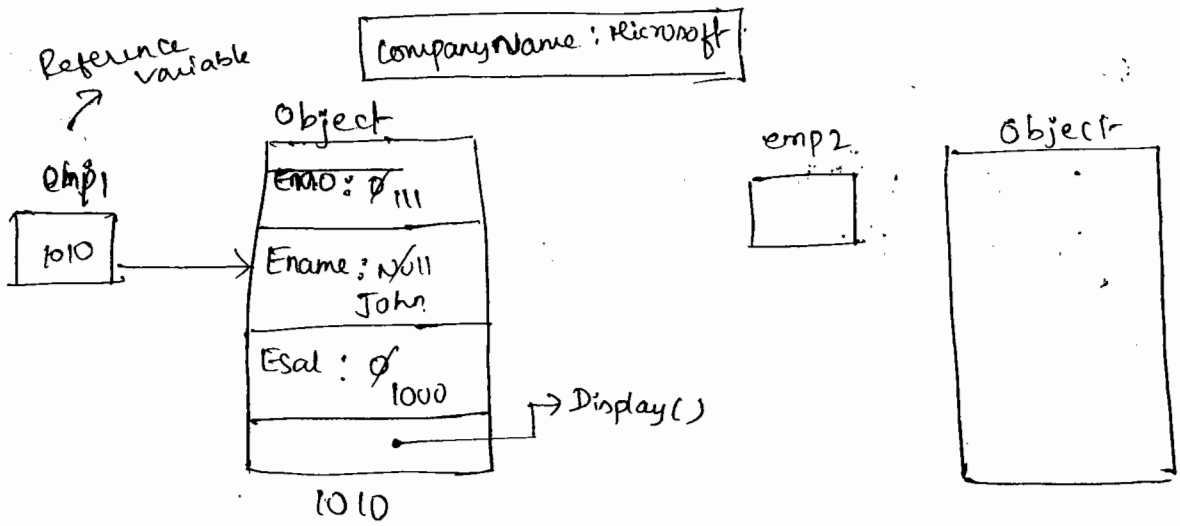
→ Example for Static Constructor.

```
namespace StaticConstructor
{
  class Employee
  {
    int Eno;
    String Ename;
    double Esal;
    Static String CompanyName;
  Static Employee( )
  {
    CompanyName = "MicroSoft";
  }
  internal Employee(int EmpNo, String EmpName, double EmpSal)
  {
      Eno = EmpNo;
      Ename = EmpName;
      Esal = EmpSal;
  }
  internal void Display( )
  {
      Console.WL("Employee No is:" + Eno);
      Console.WL("Employee Name is:" + Ename);
      Console.WL("Employee Salary is:" + Esal);
      Console.WL("Company name is:" + CompanyName);
  }
  }
  class Program
  {
      Void Main( )
      {
          Employee empl = new Employee(111, "John", 1000);
          empl.Display( );
```

```
        Employee emp2 = new Employee(222, "David", 2000);
        emp2.Display();
        Console.ReadLine();
    }
  }
}
```

O/P

```
Employee number is: 111
Employee name is: John
Employee Salary is: 1000
Company name is: Microsoft
Employee number is: 222
Employee name is: David
Employee Salary is: 2000
Company name is: Microsoft
```



⇒ When we will go for static variable?

→ whenever a value is common for all the objects then we can declare particular variable as static variable.

⇒ what is the advantage of static variable:

We can save the memory.

→ Example to understand Static Constructor & instance Constructor Execution

Prgm

```
namespace Constructor Ex1
{
    Class Myclass
    {
        int a;
        Static int b;
        Static Myclass()
        {
            Console.WriteLine(" Static constructor is calling");
            b = 10;
        }
        Internal Myclass(int x);
        {
            Console.WriteLine("Instance constructor is calling");
            a = x;
        }
        internal void Display()
        {
            Console.WL("a Value is:"+a);
            Console.WL("b value is:"+b);
        }
    }
    class Program
    {
        Void Main()
        {// Console.WriteLine(" Before object is created");
            Myclass mc = new Myclass(10);
            mc.Display();
            Console.ReadLine();
        }
    }
}
```

O/P

Static constructor is calling.

Instance constructor is calling

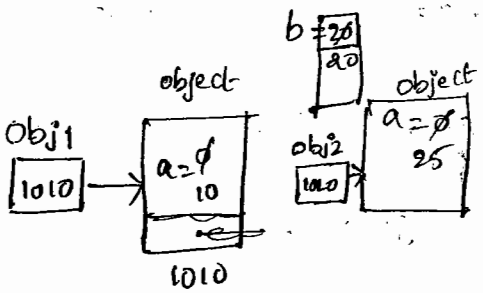
a value is: 10

b value is: 20

→ Example to initialize static variable & instance variable within the instance constructor.

Prgm

```
namespace Constructor Ex2
{
  Class Myclass
  {
     int a;
     Static int b;
     Internal Myclass(int x)
     {
        a = x;
        b = 20;
     }
  // internal void Display()
     {
      Console.WL("a value is:"+a);
      Console.WL("b value is:"+b);
     }
  }
  class program
  {
     Void Main()
     {
       Myclass obj1 = new Myclass(10);
       Myclass obj2 = new Myclass(25);
       Console.ReadLine();
     }
  }
}
```



→ In the above example we have initialized a static variable called b within instance constructor, because of that for two objects two times the value is initializing to static variable.

→ But the static variable nature is It has to create once & it have to initialize once.

→ Because of initializing static variable within the instance constructor we are losing the static nature.

→ Example for static constructor with parameter.

```
class Myclass
{
    Static int a;

    static Myclass (int x)
    {
      a = x
    }
}
class program
{
    void Main ()
    {
      Myclass obj = new Myclass ();
      Console.ReadLine ();
    }
}
```

Note: The above code will generate an error because static constructor should not have parameters

Importance of this Keyword :-

→ this is a keyword which is representing current class instance or object.

Q) when we will use this Keyword?

Ans: whenever we want to access current class instance members within the class by using object we can use this Keyword

→ Syntax :-

```
this.<instance member>;
```

→ Example for this Keyword.

Program

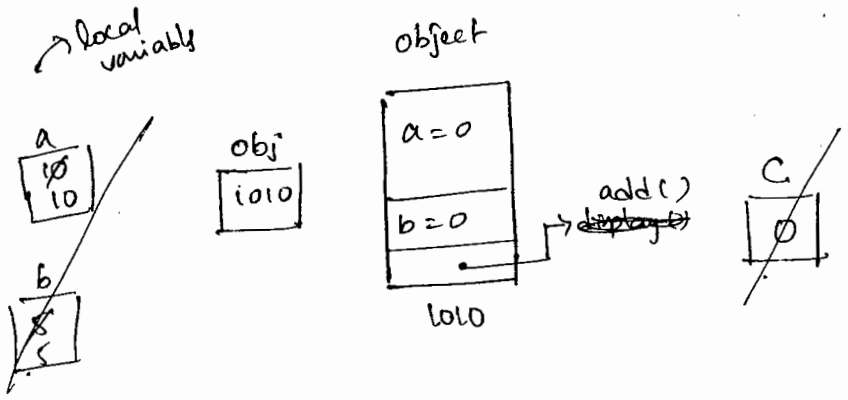
```
namespace thiskeyword
{
    class myclas
    {
        int a;
        int b;      } → instance variables
        internal myclass (int a, int b)
                          ↓
                       local variables
        {
            a = a;
            b = b;
        }
        internal void add()
        {
            int c = a+b;
                ↑ local variable
                  ↓ instance variables
            Console.writeline("Addition result is :" + c);
        }
    }
    class Program
    {
        void Main()
        {
```

instance variables ← a = a; b = b; → local variables

```
Myclass obj = new myclass(10,5);
 Obj.add();
 Console.ReadLine();
 }
 }
}
```

o/p

Addition result is : 0



Note : To overcome above problem we have to access a and b instance variables by using this keyword like below.

```
internal myclass (int a, int b)
{
 this.a = a;
 this.b = b;
 }
```

Note :-

→ while initializing the instance variable with the help of parameterized constructor, according to coding standards instance variable name and parameter name should be same.

→ To differentiate instance variable with parameter we should use this keyword like above.

Q) when object class default constructor will invoke?

Ans → Object is a predefined class and which is Super class for all .Net classes.

→ within object class microsoft defined a default constructor like below.

```
class object
{
  Public object()
  {
  }
}
```

→ In .Net when we will invoke any instance constructor, automatically it will invoke object class default constructor.

→ For example

```
namespace objectclassConstructorEx
{
  class Myclass
  {
    int a;
    internal myclass()
    {
      Console.WriteLine("Default Constructor is calling");
    }
    internal Myclass(int a)
    {
      this.a = a;
      Console.WriteLine("Parameterized Constructor is calling");
    }
  }
  class program
  {
```

```
Void Main()
{
    Myclass obj1 = new Myclass();
    Myclass obj2 = new Myclass(10);
    Console.ReadLine();
}
}
}
```

→ when we compile the above prgm compiler will add base of keyword within every instance constructor declaration like below

```
internal myclass():base()
{
}
internal myclass(int a):base()
{
}
```

<u>base</u> :- → base represents super class instance.

→ ~~base()~~

<u>base()</u> :-

→ It represents super class default constructor.

→ for all .Net classes super class is object.

→ when we ~~run~~ CLR is executing below statement

```
Myclass obj1 = new myclass();
```

→ Here control will go to Mydass default constructor like below

```
internal Mydass():base()
{
}
```

→ In declaration because of base() keyword it will goto object class and will execute object class default constructor

→ After executing object class default constructor it will come back to Myclass and will execute myclass default constructor.

→ While executing below statement

```
Myclass obj2 = new Myclass(10);
```

→ It will go to myclass parameterized constructor like below

```
internal Myclass (int a) : base()
{
}
```

→ Here within the constructor declaration because of base() keyword it will ~~goto~~ execute first object class default constructor then it will execute Myclass parameterized constructor.

→ With the above constructors calling and execution we can say that constructor calling will be bottom to top and constructors execution will be top to bottom.

→ Example to create object in two lines.

```
namespace objectex1
{
  class Myclass
  {
  internal int a=10;
  }
}
class Program
{
```

```
Void Main( )
{
  ~~My class~~
   Myclass obj;
   obj = new myclass();
   Console.WriteLine(obj.a);
   Console.ReadLine();
  }
 }
}
```

O/P

10

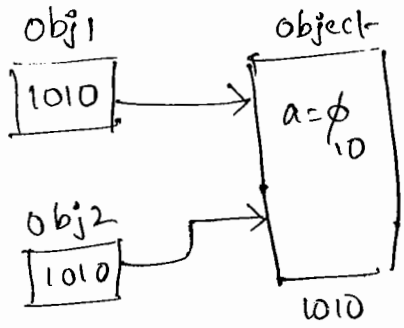
→ Example to hold one object ~~address~~ by two reference variables.

```
namespace ObjectEx2
{
 Class Myclass
 {
   ~~int a~~;
   internal int a =10;
 }
}
class program
{
 Void Main()
 {
  Myclass obj1 = new Myclass();
  ~~obj~~ Myclass obj2 = obj1;
  Console.WL("obj1 value is :"+obj1.a);
  Console.WL(" obj2 value is:" +obj2.a);
  Console.RL();
 }
}
}
```

O/P

obj1 value is : 10

obj2 value is : 10



→ Example for object

Prgm

```
namespace objectEx3
{
  class Myclass
  {
    internal int a=10;
  }
}
class program
{
  void Main()
  {
    Myclass obj1 = new Myclass();
    Myclass obj2 = obj1;
    obj1.a = 20;
    Console.WL("obj1 value is :"+obj1.a);
    Console.WL(" obj2 value is :" + obj2.a);
    Console.RL();
  }
}
}
```

o/p

obj1 value is : 20
obj2 value is : 20

→ Example for object

```
namespace objectEx4
{
  class Myclass
  {
    internal int a=10;
  }
  class program
  {
    void Main()
    {
      Myclass obj1 = new Myclass();
      Myclass obj2 = new Myclass();
      obj1 = null;
    // Console.WL("obj1 value is:"+ obj1.a);//it will throw null reference Exception
      Console.WL("obj2 Value is:"+obj2.a);
      Console.RL();
    }
  }
}
```

Q) when Null reference exception will throw.

Ans Whenever we are accessing an object with the help of a reference variable which contains Null value.

→ Example for object

```
namespace objectEx5
{
  class Myclass
  {
    internal int a=10;
  }
```

```
class program
  {
    void Main()
    {
      Myclass obj1;
      Console.WL("obj1 value is :"+obj1.a);
      Console.RL();
    }
```

→ o/p is compile time error becoz we are not initialized object for reference variable which is local.

Static Method :-

Q) When we will go for static method?

Ans → According to the requirement whenever we want to access all static variables we will go for static method.

→ According to the requirement whenever we want to display ~~we will go for~~ static text we will go for static method.

→ for Ex help(), we will go for static method.

→ Example for staticmethod.

```
namespace StaticmethodEx.
  {
    class employee
    {
      int Eno;
      String Ename;
      double Esal;
      Static String CompanyName;
      Static string Comploc;
```

```
Static  Employee( )
 {
   CompanyName = "Microsoft";
   Comploc = "Manikonda";
 }
internal Employee(int Eno, string Ename, double Esal)
 {  this.Eno = Eno;
    this.Ename = Ename;
    this.Esal = Esal;
 }
internal void Employeeinfo( )
 {
   Console.WL("Employee No is:"+ Eno);
   Console.WL("EmpName is: "+ Ename);
   Console.WL("Salary is :"+Esal);
 }
internal Static void Compinfo( )
 {
   Console.WL("Company name is : "+CompanyName);
   Console.WL("Company location is:"+Comploc);
 }
}
Class program
 {
   void Main( )
   {
     Employee empl = new Employee(111, "david", 1000);
     Empl.Employeeinfo();
     Employee.Compinfo();
     Console.RL();
   }
```

→ Example for predefined static methods

WriteLine ( )
Write ( )
ReadLine ( )
ToInt32 ( )
Parse ( )

→ Example to access instance variable with in static method

```
namespace staticmethodEx2
{
  class Myclass
  {
    ~~static~~ int a=10;  →  internal void show( )
                              { console.WL("instance method is calling:");
                              }
    internal static void display( )
    {  Myclass obj = new Myclass( )
       obj.show( );
       console.WL("a value is :"+a);
    }
  }
  class program
  {
    void Main( )
    {
      Myclass.display( );
      console.ReadLine( );
    }
```

O/P

Instance method is calling

a value is : 10

Note : ~~to~~.

→ With the above Example we can say that while accessing instance members of the same class within the static method we ~~have~~ required object.

Advantage of Static method :-

→ We can save the memory. because for static members we dont require to create an object.

Q) Why main method is static method?

→ To run console application CLR has to invoke main method

→ If main method is instance method every time object has to create it is wastage of memory and main method is an entry point for programmer execution which is not using any instance variables due to that reason main is declared as static method.

Accessing Instance members :-

1. Accessing instance members within the class

```
class Myclass
{
  int a = 10;
  Internal void Print()
  {
    Console.Writeline (a);
  }
 internal void show()
  {
     Print ();
  }
}
```

2. Accessing instance members from outside the class

```
class Myclass
{
   int a=10;
   Internal void print().
   {
     console.WL(a);
   }
```

```
class class2
{
  internal void. show()
  {
    Myclass obj = new Myclass();
    obj. print();
  }
}
```

Accessing static Members:

1. Accessing static members with in the class

```
Class Myclass
{
  Static int a=10;
  Static Internal void print()
  {
    Console. WL (a);
  }
  Static internal void show()
  {
    Print();
  }
}
```

2. Accessing static members from outside the class

```
class Myclass
{
  Static int a=10;
  Static internal void print()
  {
    Console. WL(a);
  }
}
```

```
class class2
{
static internal void show()
  {
    Myclass.Print();
  }
}
```

Access Modifiers (or) Access Specifiers

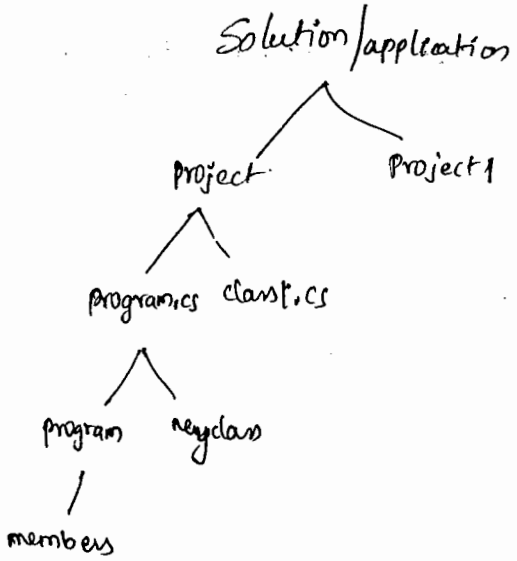
→ Access Modifiers (or) Access specifiers are Keywords which are specifing the accessability (or) access level of class (or) class members.

→ while defining a class (or) while declaring a class member we can define the accessability with the help of access modifiers.

→ In C#.Net we have 5 access modifiers : they are

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

1. Public



→ .Net solution/application is a collection of Projects

→ Project is a collection of class files

→ class files is collection of classes

→ class is a collection of members.

1. Public

→ If we declare a class (or) class members as public which can be accessed by all the classes of current project & all the projects of application.

→ To access a public member there is no restriction with in the application

→ Ex of access modifier Public.

```
namespace Publicmodifierex
{
  class class1
  {
    Public int a = 10;
    Public void method1()
    {
      Console.WL("method 1 value is :"+a);
    }
  }
  class class2
  {
    ~~internal~~
    Public void method2()
    { class1 obj = new class1();
      Console.WL("method 2 value is :" obj.a);
    }
  }
  class Program
  {
```

```
void Main )
{
  class1 objc1 = new class1();
  Console.Writeline ("~~method2 value~~
                        main method value is :"+ objc1.a);
  Objc1.method1();
  class2 objc2 = new class2();
  objc2.method2();
  Console.ReadLine();
}
```

o/p

main method value is : 10
method1 value is : 10
method 2 value is : 10

2. Private

→ If we declare class (or) class member access modifier as Private, which can be accessed only within that class.

→ Example for private access modifier

```
namespace privateEx
{
  class class1
  {
    Private int a=10;
    Private Public void method1()
      {
        Console.WL("method1. value is :"+a);
      }
  }
  class Program
  {
```

```
        Void Main()
        {
          Class1 obj = new class1();
          ~~int (obj.a)~~ //
          obj.method1();

          Console.Readline();
        }
      }
    }
```

3. Protected :-

→ If we declare class & class member access modifier as protected which can be accessed by current class members as well as derived class members.

→ Protected is depending on inheritance concept.

→ Example for protected.

```
namespace protectedEx
{
  Class class1
  {
    Protected int a = 10;
    Public void Method1()
    {
      Console.WL(" Method1 value is:"+a);
    }
  }
  Class class2 : class1
  {
    Public void Method2()
    {
      Console.WL("Method2 value is:"+a);
    }
  }
```

```
class program
{
    void Main()
    {
     class1 obj1 = new class1();
     obj1.method1();
     class2 obj2 = new class2();
     obj2.method2();
     obj2.method1();
     Console.WL("Main
     Console.RL();
    }
```

4. Internal :-

→ If we declare class (or) class member access modifier as Internal which can be accessed by all the classes of current project.

→ Ex for Internal

```
namespace InternalEx
{
    class class1
    {
      internal int a=10;
      Public void Method1()
      { Console.WL("Method 1 value is :"+a);
      }
    }
    class class2
    {
      Public void Method2()
      { class1 obj = new class1();
        Console.WL("Method2 value is :"+ obj.a);
      }
    }
```

```
class program
{
    void Main()
    {
        class1 obj2 = new class1();
        Console.WL("Main method value is:"+obj2.a);
        ~~class~~ obj2.Method1();
        Class2 obj3 = new class2();
        obj3.Method2();
        Console.ReadLine();
    }
```

Difference b/w public and internal access modifiers:-

→ Public members can be accessed by all the projects of the application.

→ Internal members can be accesed only with in that Project.

Default accessmodifiers in C#.Net:-

→ Default accessmodifier of a class is internal.

→ Default access modifier of a method is private.

→ Default access modifier of a variable is private

→ Default accessmodifier of a constructor will be private.

→ Finally we can say default access modifier of a class member will be private.

Why static constructor cannot contain access modifier:

→ We dont require to access static constructor from outside the class.

Default values :-

1. Numerical data type default value is Zero.
2. Default value of string is Null.
3. Default value of char is Null.
4. Default value of floating datatype is Zero
5. Default value of bool is False.
6. Default value of Object is Null.

Passing Parameter mechanism :-

→ Passing a value to a function is called as passing Parameter mechanism.

what is a parameter :-

→ A variable which is declared with in method signature is called as parameter.

Why Parameter :

→ whenever a method is required some value for its functionality from outside we will go for parameter.

Types of parameters :-

→ Two types

1. Formal parameter
2. Actual Parameter.

1. Formal Parameter

Called function parameter is formal parameter.

2. Actual Parameter :

Calling function parameter is actual parameter.

→ For Example

```
class Myclass
{
  internal void add(int a, int b)
    {int c = a+b;
      Console.WriteLine("Result is:"+c);
    }
}
class Program
{
  void Main()
  {
    Myclass obj = new myclass();
      int x = 10;
      int y = 20;
      obj.add(x, y);
  }
}
}
```

called function

formal parameters

Actual Parameters

calling function

Note: Argument & Parameter both are same.

→ C#.Net will support three types of passing parameter mechanism.

1. Call by value
   (or)
   Pass by value.
2. Call by reference (or) Pass by reference
3. Call by out (or) pass by out.

1. Call by value :-

→ In call by value when formal parameters are modified the modifications will not be reflected back to actual Parameters.

→ In call by value while passing the parameters we will Pass the actual values.

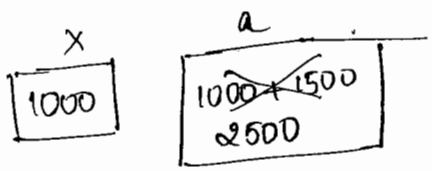
→ Example for call by value.

```
namespace CallbyValueEx
{
  class Myclass
  {
    Internal void PayFee(int a) //formal Parameter
    {
      a = a+1500;
      Console.WriteLine("a value is :"+a);
    }
  }
  class program
  {
    Void Main()
    {
      Myclass obj=new Myclass();
      int x=1000;
      Obj.PayFee(x);
      Console.WriteLine("x value is :"+x);
      Console.ReadLine();
    }
```

O/P

a value is : 2500

x value is : 1000



When can I go for call by value?

→ whenever we want to pass some value to the function and that called function modification should not reflect back to calling function. then we will go for call by value. Means we call particular parameter is called call by value.

2. Call by reference :

→ In call by reference: when formal parameters are modified those modifications will be reflecting back to actual parameters.

→ In call by reference while passing the parameters we will pass the address of the value.

→ Example for call by reference.

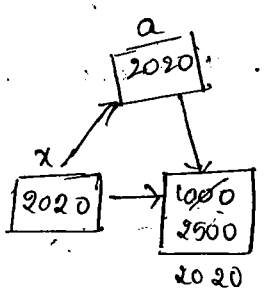
Note : Call by value parameter and call by reference parameter should be initialize with some value before passing otherwise compiler will generate an a error. Because which are local variables.

```
namespace    callbyreferenceEx
{
  class   Myclass
  {
                                  Keyword
    internal   void   Add (ref int a)
    {
      a = a + 1500;
      Console.Writeline("a value is : "+a);
    }
  }
class   Program
  {
      void Main()
      {
        Myclass obj = new Myclass();
        ~~obj.Add~~
        int x = 1000;
        obj.Add(ref x);
        Console.Writeline("x value is : "+x);
        Console.ReadLine();
      }
```

O/P

a value is : 2500

x value is : 2500



Q) When we will go for call by reference?

Ans whenever we want to pass some value to the function and if we want to get back the modified value then we will pass it as a call by reference.

3. Call by out :-

→ Call by out is same as call by reference, i.e., in call by out also when formal parameter is modified those modifications will be reflected back to actual parameter.

→ While passing the parameter & while catching the parameter we should use out keyword.

→ In call by out also ~~we will pass th.~~ with the help of Parameter we will pass the address.

→ Out parameter is not required to initialize, if we initialize also CLR will ignore that value.

Q) When we will go for call by out?

Ans. Whenever we dont want to pass some value to function, but we want to get back the modified value we will go for call by out.

→ Example for call by out.

```
namespace callbyoutEx.
{
    class Myclass
    {
        internal void Pafee(out int totfees)
        {
```

```
        int admFee = 1500;
        int SemFee = 1200;
        totFee = admFee + semFee;
        Console.WriteLine("Total fee is:" + totFee);
    }
}
class program
{
    void Main()
    {
        Myclass obj = new Myclass();
        int x;
        obj.PayFee(out x);
        Console.WriteLine("x value is:" + x);
        Console.ReadLine();
    }
```

O/P

Total fee is : 2700

x value is : 2700

→ Comparision b/w call by value, call by reference, call by out

Call by value	Call by reference	Call by out
1. formal Parameter modification will not be reflected to actual Parameters	1. will be reflected	1. will be reflected
2. Passing the value	2. Passing address	2. Passing address
3. No keyword	3. ref keyword	3. Out keyword.
4. Should be initialized	4. ref parameter should be initialized	4. out Parameter is not required to initialize.

Note :→ Default passing parameter in c# is call by value.

→ VB.Net will support only call by value & call by reference and will not support call by out.

Oops Case Study :-

Requirement :

we have a bank for that bank we have require following functionalities

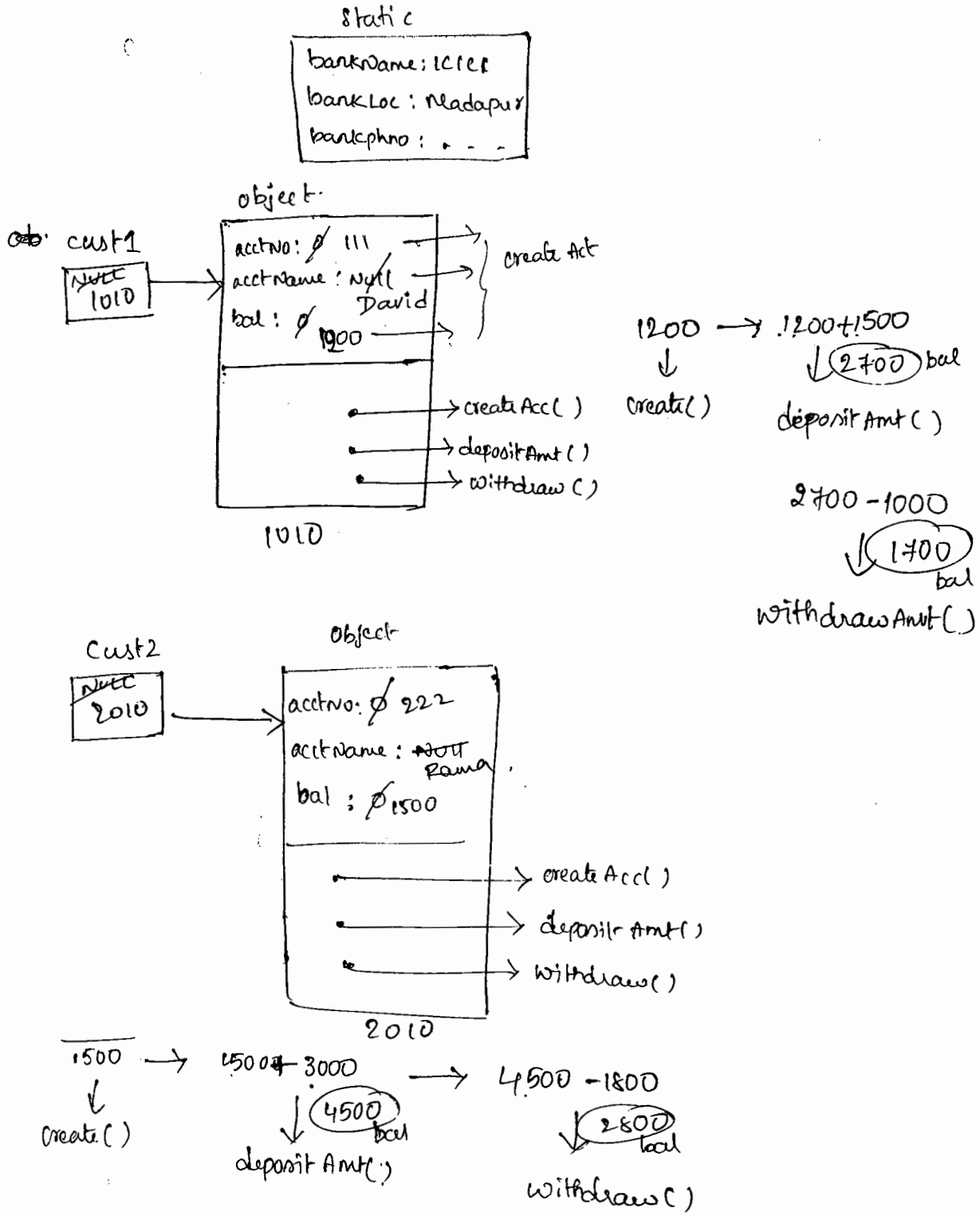
1. Creating account.
2. Depositing amount.
3. Withdraw amount.
4. Displaying the bank information

CustomerAccount
int acNo; String acName; double bal; Static String bankName; Static string bankLoc; Static long bankPhno;
Void createAccount(double intlAmt). Void DepositAmount(double Amt) void withdrawAmount(double Amt) Static void DisplayBankInfo()

Prgm 10

```
namespace customer&BankEx
{
  Class customer
  {
    int actNo;
    String actName;
    double bal;
    Static String bankName;
    Static String bankLoc;
    Static long bankphno;

  internal static customer()
    {
      bankName = "ICICI";
      bankLoc = "Madapur";
      bankphno = 0402361452;
    }

  internal Customer(int actNO, String actName)
  {
    this.actNO = actNO;
    this.actname = actname;
  }

  internal void createAcct (double IntlAmt)
  {
    bal = bal + IntlAmt;
    Console.WL(" Your acct is created Successfully");
    Console.WriteLine(" Account no is :" + actNO);
    Console.WL(" Account name is:" + actName);
    Console.WL(" Account bal is:" + bal);
  }
```

```
    else
    {
      Console.WL(" Account is not created");
    }
  }
  internal void depositAmt(double Amt)
  {
     bal = Amt+bal;
    Consde.WL(" Your total balance is:"+bal);
  }
  internal void WithdrawAmt(double Amt)
  {
     if(bal >=Amt)
      {
        Console.WL(" Your Withdrawn Amount is:"+Amt);
      }
     else
      {
        Console.WL(" Insufficient funds");
  }
 Static void DisplayBankInfo()
 {
   Console.WL("Bank Name is:"+ bankName);
   Console.WL(" Bank location is:"+ bankLoc);
   Console.WL("Bank Ph NO is:"+ bankphno);
  }
}
  class Program
  {
     Void Main()
     {
       Customer obj1 = new customer(111,"Rama");
       obj1.CreateAcct(1000);
       obj1.depositAmt(1000);
       Obj1.WithdrawAmt(500);
       Customer.DisplayBankInfo();
       Console.ReadLine();
     }
```


static
bankName: ICICI
bankLoc : Nadapur
bankphno : . . .
object
cust1
NULL
1010
acctNo: ~~0~~ 111
acctName : ~~Null~~ David
bal : ~~0~~ 1200
create Act
1200 → 1200+1500
2700 bal
create()
deposit Amt ( )
2700 - 1000
1700
bal
withdraw Amt ( )
create Acc ( )
deposit Amt ( )
withdraw ( )
1010
Cust2
NULL
2010
Object
acctNo: ~~0~~ 222
acctName : ~~Null~~ Rama
bal : ~~0~~ 1500
create Acc ( )
deposit Amt ( )
withdraw ( )
2010
1500 → 1500 + 3000 → 4500 - 1800
4500
bal
2800
bal
create ( )
deposit Amt ( )
withdraw ( )

2. We have a requirement for a company. In that company we have to perform following operations.

1. Employee joining and display employee details
2. Increment the employee salary.
3. Displaying the employee company info like company name, Company location, Company phno.

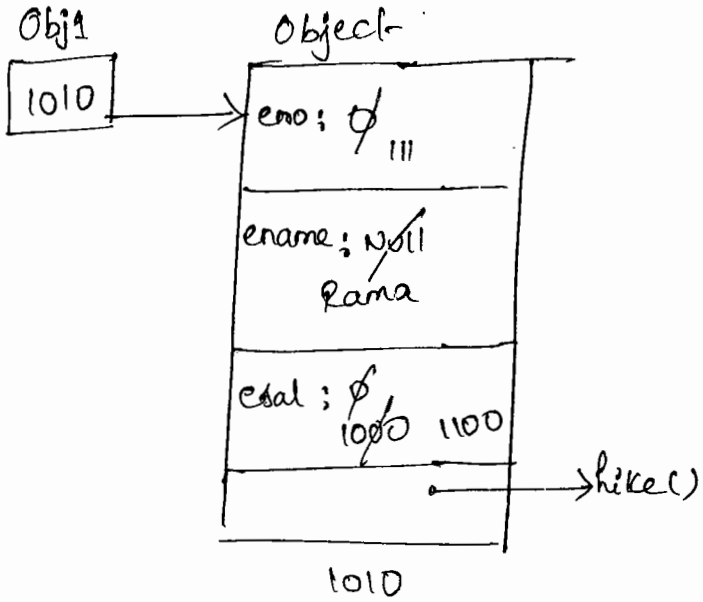
employee
int eno; String ename; double esal; double hikesal; Static companyName; Static Comploc; Static compno;
Void empinfo(int eno, String ~~ename~~, double esal) void emphike(int Static compinfo(~~String empname~~, String emploc, long ~~compNO~~)

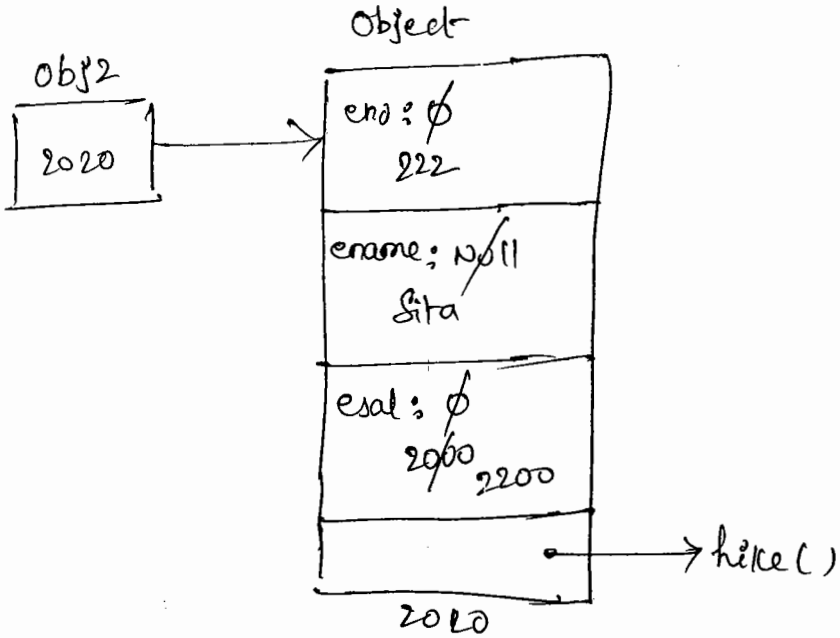
```
namespace employeeInfoEx
{
  class employee
  {
    int eno;
    string ename;
    double esal;
    Static string cmpName;
    Static string cmploc;
    Static long. cmpNo;

    Static employee( )
    {
      CmpName = " Microsoft";
      CmpLoc = "Hitechcity";
      CmpNo = 123456;
    }
    internal employee( int eno , string. ename , double esal)
    {
    this.eno = eno;
    this. ename = ename;
    this. enal = esal;
    Console. WriteLine (" Employee NO is: "+eno);
    Console. WriteLine (" Employee Name is: "+ ename);
    Console. WriteLine (" Employee Salary is: "+ esal);
    }
```

```
internal void hike()
{
    float i = 0.1f;

double tot = esal + (esal * i);
    Console.WriteLine("Employee incremented salary is :" + tot);
}
Static void compInfo()
{
 Console.WriteLine(" Employee Company Name is : " + cmpName);
 Console.WriteLine("Company Location is :" + Cmploc);
 Console.WriteLine("Company Phno is :" + cmpNo);
}

}
Class program
{
  Void Main()
  {
     Employee obj1 = new Employee(111, "Rama", 1000);
     obj1.hike();
     ~~obj1.compInfo();~~
     Employee.CompInfo();

     Employee obj2 = new Employee(222, "Sita", 2000);
     obj.hike();
     Employee.CompInfo();
     Console.~~WriteLi~~ ReadLine();
  }
}
}
```

Company Name : Microsoft
Company Loc : Hitech City
Company No : 12345

Static Variables





Constant :-

→ To declare a constant we have to use const keyword.

→ By default constant is static

→ Constant should be initialized at the time of declaration that is in design time. and the value will be never changed.

Q) when we will go for constant?

Ans

Example for constant:

```
namespace Company
{
  Class Employee
  {
    ~~internal~~ Const int MinWorkHrs = 9;
    internal static void display( )
    {
      Console.WriteLine (" Minimum working hours is :" + MinWorkHrs);
    }
  }
  Class Program
  {
    Void Main( )
    {
      Employee.Display( );
      Console.ReadLine( );
    }
```

→ In the above example we have declared Minworkhrs as constant, but, there is a chance of changing this value for this requirement we cannot declare as constant.

→ Finally we can say in business applications we will declare constants very rarely.

→ Especially in scientific applications, to represent universal fixed values like below we will declare constants.

Ex:- const double Pi = 3.14;
const long SpeedofLight = 300000; // km per sec

→ Example for predefined constants are

1. MinValue
2. Max value

· Differences between constant and static variable.

Constant	Static Variable
1. For using constant we use const keyword.	1. For using static variable use use static keyword.
2. Const can be used, when a field value is not to be changed forever.	2. Static ~~as~~ value can ~~be~~ also be fixed value, but that value can be changed by static method.
3. Const can be used mostly, to represent universal fixed values.	

3. Constant should be initialized while declaration.	3. Static can be initialized using constructor (or) during declaration
4. By default it is static	4. Static keyword should be declared explicitly.
5. whenever we want to have same value to all the objects but not required to change forever.	5. whenever we want to represent common value for all the objects but required to change in future.
6. Declared during design time.	6. Declared during run time.

readonly:-

→ To declare readonly we have to used readonly keyword

→ By default readonly is a instance member.

→ readonly can be initialized at the tym of declaration (design tym) as well as readonly can be initialized in runtime but only within the constructor but not within the method.

→ readonly value cannot be changed.

When we will go for readonly?

Ans Whenever we want to have a field for multiple objects with different values but values should not be changed. then we can go for readonly field.

→ Example for readonly.

```
namespace readonlyEx
{
  class employee
  {
     readonly int eno;
     readonly string ename;
     double esal;

    internal employee(int eno, string ename, double esal)
    {
      this.eno = eno;
      this.ename = ename;
      this.esal = esal;
    }
    internal void Hike()
    {
      double incsal = esal * 10/100;
      Console.WriteLine("Your increment salary is:" + incsal);
      ~~double~~ esal = esal + incsal;
      Console.WriteLine("Updated salary is:" + esal);
    }
  }
 Class Program
  {
    void Main()
    {
      employee obj1 = new employee(111, "John", 1000);
      obj1.Hike();
```

```
employee obj2 = new Employee(222, "Rama", 2000);
  obj2.Hike();
Console.ReadLine();
}
```

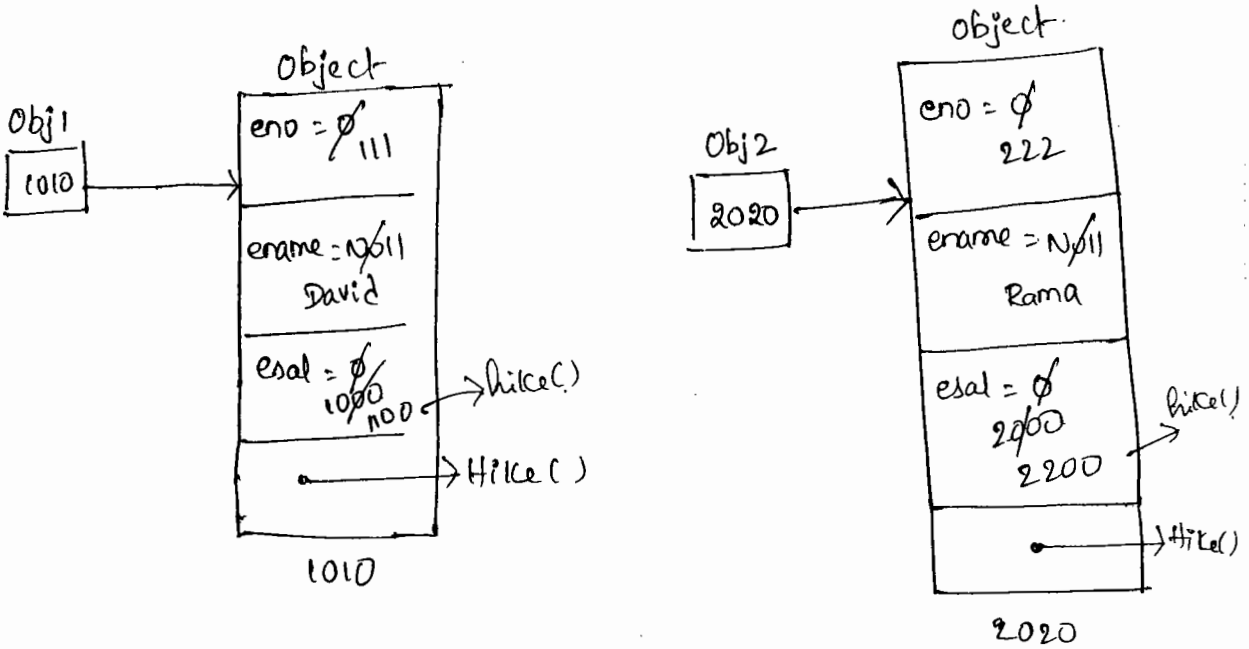
O/P

Your increment salary is : 100

Updated salary is : 1100

Your increment salary is : 200

Update salary is : 2200



readonly Vs instance variable

readonly

1. readonly keyword.
2. Initialization can be done during declaration and also in run time but only in Constructor.
3. Value cannot be changed.
4. when?

   Whenever we require a field for all the objects with different values but, value doesnt required to change then we will go for readonly.

   Ex: Empno, empname, SId, Sname...

instance variable

1. No keyword.
2. Initialization can be done during declaration and can also in run time i.e, in Constructor (or) in method also.
3. value can be changed.
4. when?

   whenever we require a field for all the objects with different values but, value is required to change in future then we will go for instance variable.

   Ex: esal, sloc, sm1, sm2 - - -

readonly

1. readonly Keyword.
2. By default instance.
3. Readonly can be initialized at the time of declaration as well as runtime i.e., in constructor.
4. Readonly value ~~can~~ will be differ from one object to another object.
5. when?

Constant.

1. const Keyword.
2. By default static
3. Constant can be initialized only at the time of declaration i.e., during design time. Cannot be initialized in runtime.
4. Constant value will be same for all the objects.
5. when?

Differences b/w instance variable & Static variable.

Instance Variable	Static variable
1. No Keyword.	1. Static Keyword.
2. initialization is done in instance constructor.	2. for initialization is done in Static Constructor.
3. memory is allocating at the time of object is creating.	3. memory is allocating when the class is loading.
4. Value will be different from one object to another object.	4. Value is same for all the objects.
5. memory will be deallocated when the object is destroyed.	5. memory will be deallocated when the class is unloading.
6. when?	6. when?

Instance Method	Static Method
1. No Keyword	1. Static Keyword.
2. To access instance variables we go for instance method.	2. To access static variables
3. Purpose of instance method is to change the instance variable values.	3. purpose of Static method is to change the static variable values.
4. instance method we have to invoke by using object.	4. Static method we have to invoke with the help of class name.
5. when?	5. when?

Instance Constructor	Static Constructor
1. No Keyword.	1. Static Keyword.
2. It can contain access modifiers	2. It cannot contain access modifies
3. it can contain parameters	3. It cannot contain parameters.
4. Within instance constructor. we can Initialize instance and also static variables.	4. In static constructor we can initialize only static variables.
5. when?	5. when?
6. Instance constructor will invoke when the object is creating.	6. Static constructor will invoke when the class is loading.
7. within a single class we can have multiple instance constructors	7. within a single class we can have only one static constructor.

Constructor	Method
1. Constructor name same as class name.	1. Method name will be user defined.
2. It will contain initialization logic.	2. It will contain modification logic.
3. No return types.	3. Method can contain return types.
4. Constructor will invoke automatically if instance at the time of creating the object and if static at the time of loading the class.	4. method can be invoked by programmer according to his requirement.
5. when?	5. when?

Properties :-

→ Property is a member of a class which we will use to assign the value to a variable, as well as which we will use to retrive the value from variable.

→ Property name and variable name should be same.

→ But variable name should start with small letter and property name should start with capital letter.

→ for Ex

eno → variable name

Eno → property name

Syntax to define a property:

```
<accessmodifier> <return type> <property name>
{
    get  → retriving the value
    {
    }
    Set  → assigning given value.
    {
    }
}
```

→ Properties are 3 types.

1. Read only Property

It will contain only get access

2. Write only:

→ Contain only ~~get~~ set accessor.

3. Read and write properties:

→ Contain get accessor and set accessor.

→ Again properties are 2 types.

1. Instance property
2. Static Property.

1. Instance property:

→ While defining a property if we dint use Static Keyword is called as instance property.

→ The purpose of instance property is to initialize and to retrive values to instance variables.

2. Static Property:-

→ While defining a property if we have used static Keyword is called as static property.

→ purpose of static property is to initialize as well as to retrive values from static variables.

→ Example for static and instance property.

```
namespace propertyEx.
{
    class Employee
    {
        int eno;
        string ename;
```

```
int age;
Static string CompName;
internal int Eno
{
    get
    {
      return eno;
    }
    Set
    {
      eno = value;
    }
}
internal string Ename
{
    get
    {
      return ename;
    }
    Set
    {
      ename = value;
    }
}
internal int Age
{
    get
    {
      return age;
    }
    Set
    {
      age = value;
    }
}
```

```
while(value < 1 || value > 150)
  { Console.WriteLine("Please enter valid age.");
    value = int.Parse(Console.ReadLine()); }
```

```
internal static string CompanName
{
    get
    {
      return compname;
    }
    Set
    {
      compname = value;
    }
  }
}
Class Program
{
    void Main ()
    {
      Employee emp1 = new Employee();
      Console.WriteLine("Enter ur ~~age~~ Empno:");
      ~~emp1.Age~~ = ~~i~~
      emp1.Eno = int.Parse (Console.ReadLine());
      Console.WriteLine("EmpNO is:"+emp1.Eno);
      Console.WriteLine("Enter ur EmpName:");
      emp1.Ename = Console.ReadLine();
      Console.WriteLine("EmpName is:"+emp1.Ename);
      Console.WriteLine("Enter your age:"),
      emp1.Age ~~i~~ = int.Parse(Console.ReadLine());
      Console.WriteLine("Emp age is:"+emp1.Age);
      Console.WriteLine("Enter your Company Name:");
```

```
Employee.CompName = Console.ReadLine();
Console.WriteLine("Company name is:" + Employee.CompName);
Console.ReadLine();
}
}
}
```

Purpose of Property :-

→ Whenever we want to assign the value to a class level variable or whenever we want to retrive the value from class level variable from outside the class we can go for property.

→ As well as whenever we want to perform validations while assigning the value to class level variable from outside the class we can use property.

→ Console prgm within the Notepad.

Step 1: Open note pad.

Step 2: Write below code with in notepad.

```
Using System;
class Program
{
    Static void Main(string[] args)
    {
      for(int i=0; i<=10; i++)
        {
          Console.WL("Welcome to Satya");
        }
      Console.RL();
    }
}
```

Step3: Save Program file with .cs.

Step 4:

Start → Program → Visual Studio 2010 → Visual Studio 2010 → Visual Studio command prompt.

→ change to d drive (d:) ↵

D:\>csc Program.cs ↵

with this process compilation is completed it will generate program.exe file

Step 5: Executing the Program

D:\> Program.exe ↵

Welcome to Satya

⋮

welcome to Satya.

HW.

Implement all above programs using Notepad.

CLR inside (or) CLR Architecture (or) Memory allocation in CLR :-

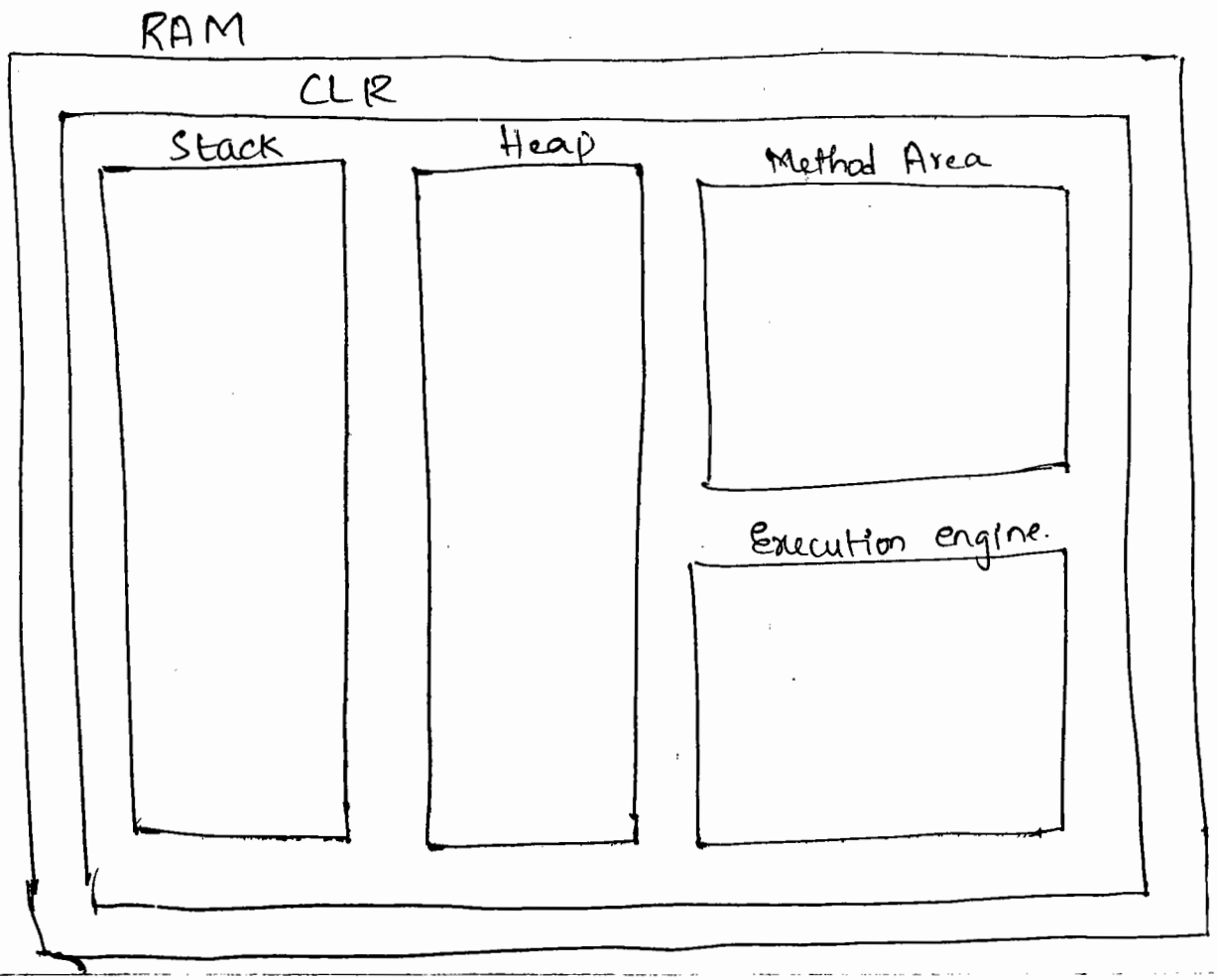
→ When we run the .Net application, .Net execution engine called CLR will be loading in to RAM memory

→ When we run Java application, Java execution engine JVM will load into RAM memory.

→ CLR will maintain various blocks for memory allocation or memory management while executing the application.

→ Among the various blocks of the CLR, 4 are important

1. Stack
2. Heap
3. Method Area
4. Execution Engine. like below.



Stack :

→ Stack CLR will use to store local variables and reference variables.

Heap:

→ CLR will use heap to store objects.

→ Object will contain instance variables & reference of instance methods

Method Area :

→ CLR will use method area to ~~store~~ maintain static variables and methods and constructors. like below.

Method Area

Methods	constructors	Static variables

Execution Engine:

→ CLR will use execution engine to maintain the block which is executing currently, it can be a method (or) it can be a constructor.

→ Example to understand Execution engine.

```
class Myclass
{
static Method1( )
    {
    }
Static Method2( )
    {
    }
}
class Program
{
    void Main( )
    {
        Myclass.Method1( );
        Myclass.Method2( );
    }
}
```

→ When we run the above program, first CLR required Program class, due to that reason CLR will load program class into method area means, main method will place into method area.

→ Now CLR has to start execution with main method, due to that reason main method will be moving from method area to execution engine.

→ Now CLR will has to execute main method first statement that is Myclass.Method1. Now it requires my class, due to

that reason CLR will load Myclass methods i.e., Method1 and Method2 into Method area.

→ Now CLR will execute Method1 for this it will move method1 from method area to execution engine.

→ Now CLR will execute Method1 which is in execution engine

→ Once method1 execution is completed CLR will remove method1 block from execution engine. and now CLR will execute Method2 like below.

```
Myclass.Method2();
```

→ Due to that reason CLR will move the Method2 from method area to execution engine and so on.

→ Once main method execution is completed main block will be removed from the execution engine.

→ Once application execution is completed (or) closed, all the classes will unload nothing but method area will be cleared.

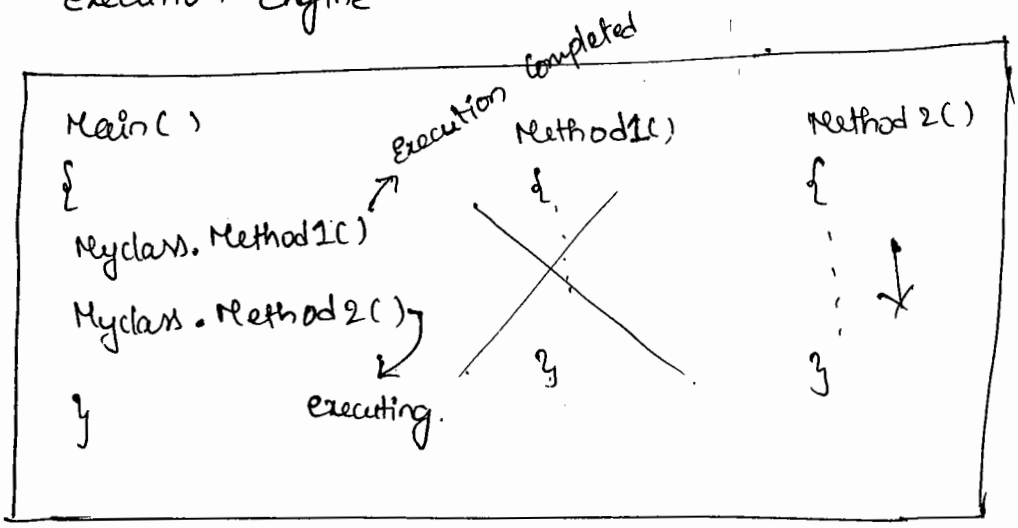
Method Area

```
Main()
{
 Myclass.Method1();
 Myclass.Method2();
}
```

```
Method1()
{
 .
 .
 .
}
```

```
Method2()
{
 .
 .
 .
}
```

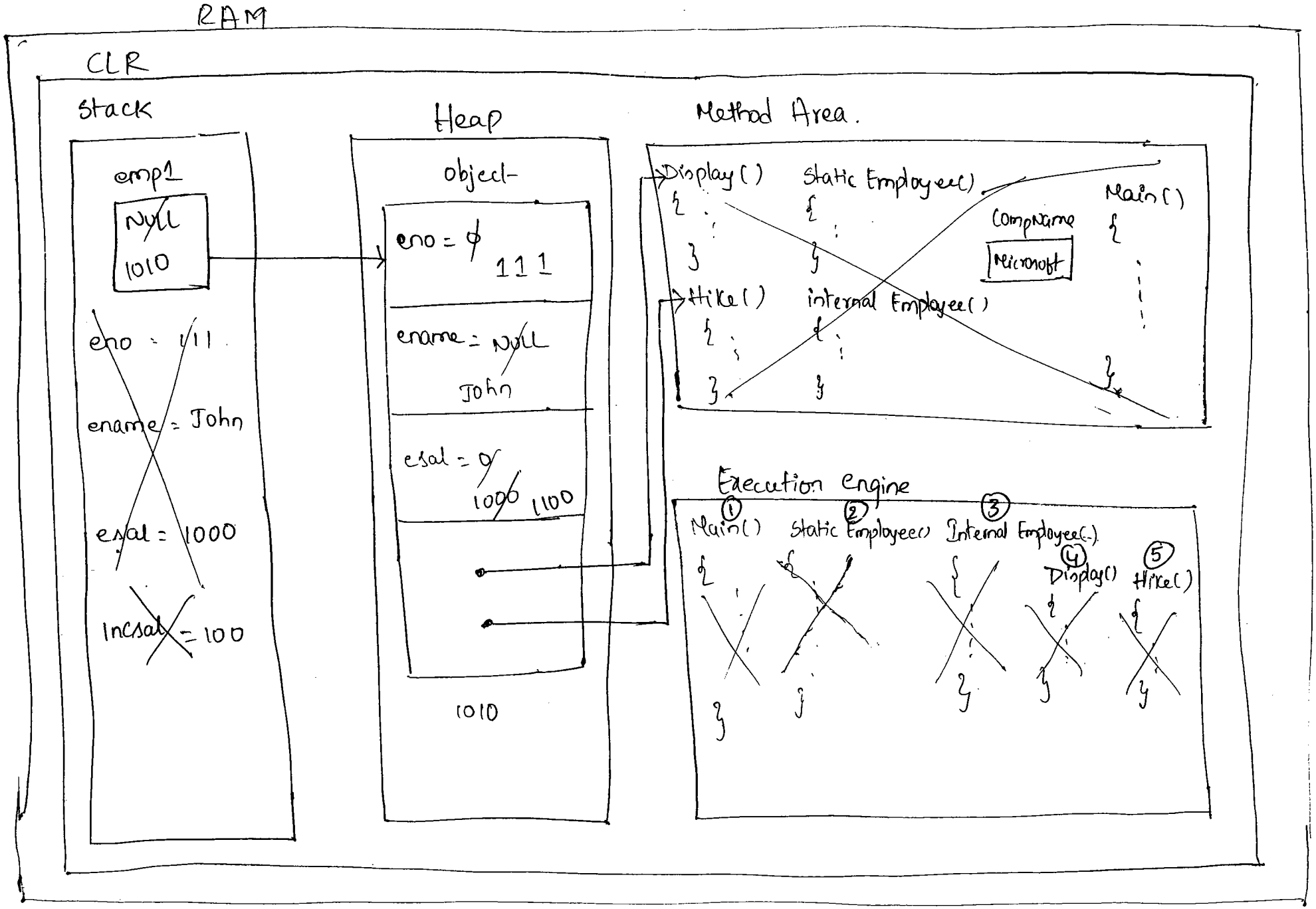
Execution Engine



→ Example for understanding the memory allocations within CLR.

```
namespace memoryallocationEx
{
  class Employee
  {
     int eno;
     String ename;
     double esal;
     Static String CompName;
     Static Employee()
     {
       CompName = "Microsoft";
     }
     internal Employee(int eno, String ename, double esal)
     {
        this.eno = eno;
        this.ename = ename;
        this.esal = esal;
     }
```

```
internal void Display( )
{
  Console.WriteLine("EmpNo is :" + eno);
  Console.WriteLine("Ename is :" + ename);
  Console.WriteLine("EmpSal is :" + esal);
  Console.WriteLine("CompName is : " + CompName);
}
~~static~~

internal void Hike( )
{
   double incsal = esal * (10/100);
   Console.WriteLine("Incremented salary is :" + incsal);
   esal = incsal + esal;
   Console.WriteLine("Updated salary is :" + esal);
}
}
class Program
{
   void Main( )
   {
      Employee emp1 = new Employee(111, "John", 1000);
      emp1.Display();
      emp1.Hike();
      Console.ReadLine();
   }
}
}
```


RAM
CLR
Stack
emp1
Null
1010
eno = 111
ename = John
esal = 1000
Incsal = 100
Heap
object
eno = 111
ename = Null
John
esal = 1000
1100
1010
Method Area.
Display()
Static Employee()
Main()
CompName
Microsoft
Hike()
internal Employee()
Execution engine
Main()
Static Employee()
Internal Employee()
Display()
Hike()

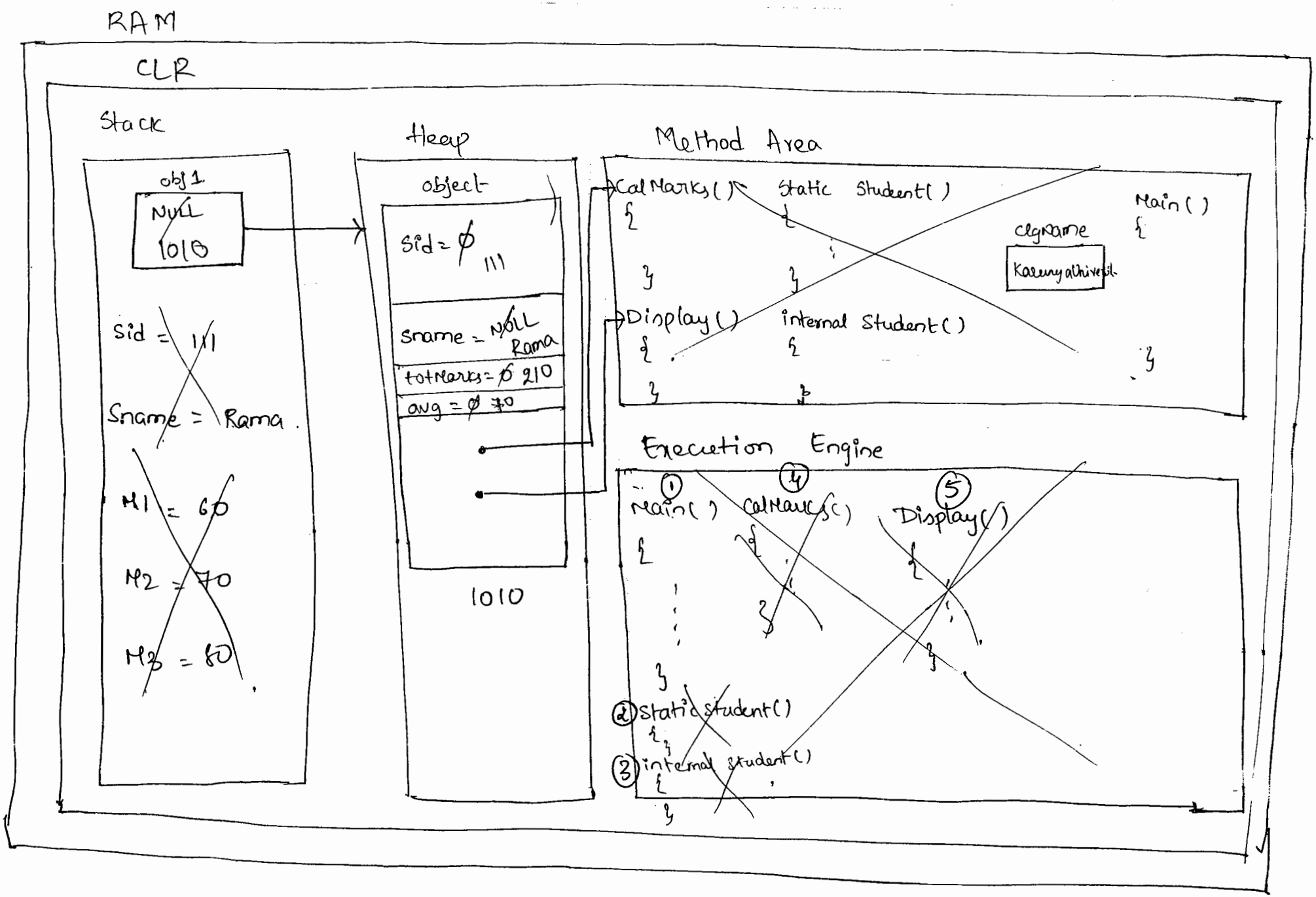
→ Example 2 for memory allocation in CLR.

Student
int sid; String sname; double totMarks; double avg; Static String clgName;
Void CalMarks(double M1, double M2, double M3) void Display();

~~name class~~

```
namespace memoryallocationEx2
{
  Class Student
  {
    int Sid;
    String Sname;
    double totMarks;
    double avg;
  Static String clgName;

    Static Student()
    {
      clgName = " Karunya University";
    }
```

```
internal Student(int sid, string sname)
{
    this.sid = sid;
    this.sname = sname;
}
internal void calMarks(double M1, double M2, double M3)
{
    totMarks = M1 + M2 + M3;
    avg = totMarks/3;
}
internal void Display()
{
    Console.WriteLine("Student Id is :"+ sid);
    Console.WriteLine("Student name is: "+sname);
    Console.WriteLine("Student total Marks is: "+totMarks);
    Console.WriteLine("Student average is :"+ avg);
    Console.WriteLine("Student College is :"+ clgName);
}
}
class Program
{
    void Main()
    {
        Student obj1 = new Student(111,"Rama");
        obj1.calMarks(60,70,80);
        obj1.Display();
        Console.ReadLine();
    }
```


RAM
CLR
Stack
obj 1
NULL
1010
Sid = 111
Sname = Rama
M1 = 60
M2 = 70
M3 = 80
Heap
object
Sid = 111
Sname = NULL Rama
totMarks = 210
avg = 70
1010
Method Area
CalMarks()
Static Student()
Main()
clgName
Kareny athiversit.
Display()
internal Student()
Execution Engine
1 Main()
4 calMarks()
5 Display()
2 Static student()
3 internal student()

→ Draw the CLR diagram for Bank case study.

→ Draw the CLR diagrams for all the above oops examples.

Oops Principles

1. Encapsulation :-

→ Wrapping states and behaviours into a single unit is called Encapsulation.

→ Binding variables and methods is called as Encapsulation

How to encapsulate?

By using class.

2. Abstraction :-

→ Hiding the unwanted data is called as Abstraction.

→ Abstractions are two types.

1. Data Abstraction
2. Method Abstraction.

1. Data abstraction :-

Hiding the unwanted data is nothing but data abstraction

2. Method Abstraction:-

Hiding unwanted method is called as method abstraction.

→ Example data abstraction.

```
namespace dataAbstractionEx.
{
  class Employee
  {
     int eno;
     String ename;
     String designation;
     double esal;

     internal Employee(int eno, String ename, String designation, double esal);
     {
       this.eno = eno;
       this.ename = ename;
       this.designation = designation;
       this.esal = esal;
     }
    internal void display()
    {
     Console.WriteLine("EmpNo is:"+eno);
     Console.WL("EmpName is:"+ename);
     Console.WL("Empdesignation is:"+designation);
     Console.WL("Emp salary is:"+esal);
    }
    internal void Hike()
    {
       double Incsal = esal *(10/100);
       esal = Incsal + esal;
      Console.WL(~~Emp~~ + "EmpNo :"~~+~~eno+ "EmpName :"+ename+
                                "updated salary is:"+esal);
    }
}
```

```
Class Program
{
    Void Main()
    {
     Employee emp1 = new Employee( 111, "John", "Software Engineer",
                                   1000);
     emp1.Display();
     emp1.Hike();
     Console.ReadLine();
    }
}
}
```

Note : In the above prgm, Display method does not have any data abstraction, but Hike method has data abstraction i.e., designation of employee is not needed in Hike method. So we are hiding designation in Hike().
It is nothing but data abstraction.

3. Inheritance :

→ Inheritence is nothing but deriving (or) inheriting the members from one class to another class.

→ A class which is giving the members is called as Super class (or) parent class (or) base class.

→ A class which is receiving the members is called as Sub class (or) derived class (or) Child class.

→ Because of inheritence Sub Class can access Super class members as well as using Sub class objects we can access Super class members.

→ Because of inheritance super class cannot access subclass members as well as using super class object we cannot access sub class members

→ Types of inheritence.

1. Single inheritence
2. Multilevel inheritence
3. Multiple inheritence
4. Hybrid inheritence
5. Hierarchial inheritence.

1. <u>Single inheritence</u> :-

→ Inheriting from one class to another class is called single inheritence.

→ <u>Syntax</u>.

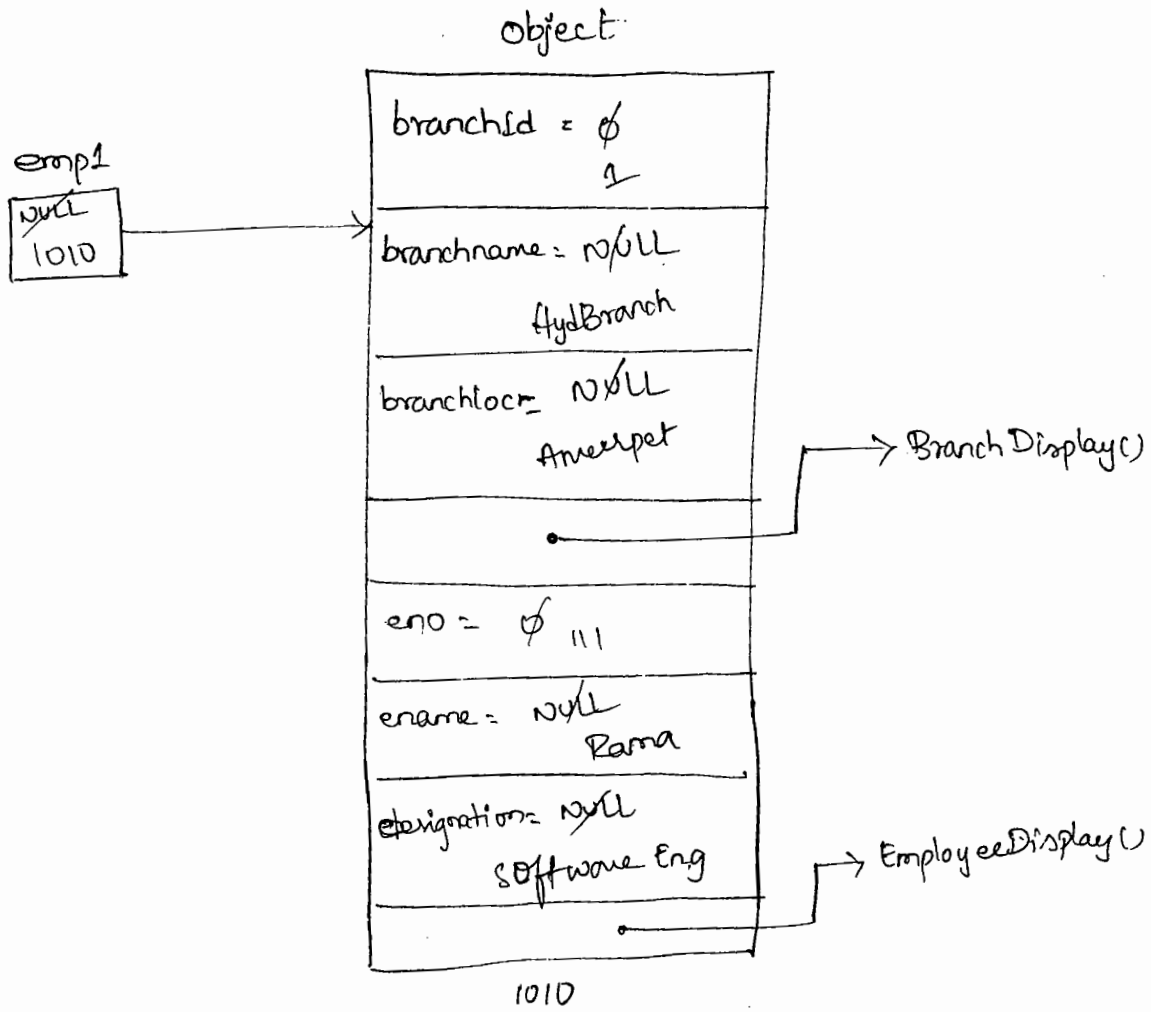
```
Class C1
{
}
Class C2 : C1
{

}
```

→ Example for single inheritence.

```
namespace SingleinheritenceEx
{
    class Branch
    {
        int branchId;
        String branchname;
        String branchloc;
```

```
internal Branch(int branchid, String branchname, String branchloc)
 {
    this.branchid = branchid;
    this.branchname = branchname;
    this.branchloc = branchloc;
 }
 internal void BranchDisplay()
 {
    Console.WL("Branch Id is : "+branchid);
    Console.WL(" Branch name is:"+branchname);
    Console.WL(" Banch location is:"+branchloc);
 }
}
Class Employee : Branch
 {
   int eno;
   String ename;
   String Designation;
   internal Employee(int eno, String ename, String Designation) : base(
   {                                  111, "HydBranch", "Ameerpet");
      this.eno = eno;
      this.ename = ename;
      this.designation = designation;
    }
    internal void EmployeeDisplay()
    {
        Console.WL("Emp No is: "+eno);
        Console.WL("Emp Name is:"+ename);
        Console.WL(" Emp designation is:"+designation);
        base.BranchDisplay();
    }
 }
```

```
class Program
{
    Void Main()
    {
        Employee emp1 = new Employee(111, "Rama", "Software Engg");
        emp1.EmployeeDisplay();
        Console.ReadLine();
    }
}
}
```



2. Multi Level Inheritence :-

→ Inheriting from one class to another class, from that class to someother class is called as MultiLevel inheritence

→ Syntax :

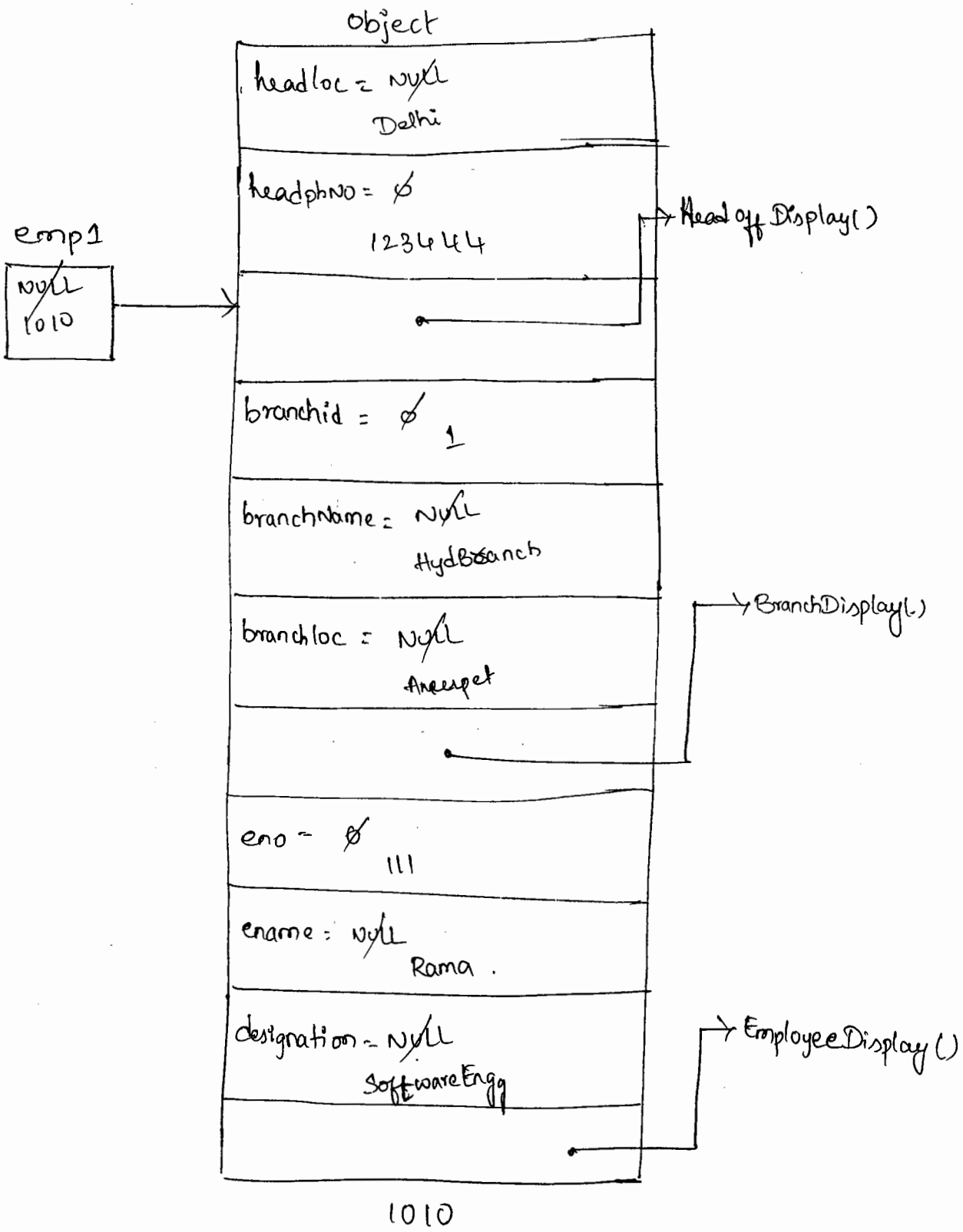
```
class c1
{

}
class c2 : c1
{

}
Class c3 : c2
{

}
```

→ Example for multilevel inheritence.

```
namespace MultiLevelInheritenceEx
{
    Class HeadOffice
    {
      String headloc;
      ulong headPhNO;
      internal HeadOffice (String headloc, ulong headPhNO)
      {
        this. headloc = headloc;
        this . headPhNO = headphNO;
      }
```

```
internal void HeadoffDisplay( )
 {
   Console.WL("Headoffice location is:"+headloc);
   Console.WL("Headoffice Phone No is:"+headPhNO);
  }
}
class Branch:HeadOffice
 {
   int branchid;
   String branchName;
   String branchloc;
   internal Branch(int branchid, String branchName, String branchloc):
                                base("Delhi", 123444)
    {
      this.branchid = branchid;
      this.branchName = branchName;
      this.branchloc = branchloc;
    }
    internal void BranchDisplay( )
    {
      Console.WL(" Branch id is :"+branchid);
      Console.WL(" Branch Name is:" +branchName);
      Console.WL(" Branch location is:"+branchloc);
      base.HeadoffDisplay();
    }
  }
```

```
Class  Employee : Branch
{
  int  eno;
  String ename;
  String designation;

internal  Employee(int eno, String ename , String designation): base
                                  (1, "HydBranch", "Ameerpet");
{
   this. eno =  eno;
   this. ename = ename;
   this. designation = designation;
}
internal void  EmployeeDisplay( )
{
  Console. WL(" Emp NO is :"+ eno);
  Console. WL(" Emp Name is :"+ ename);
  Console. WL(" Emp designation is :"+designation);
  base. BranchDisplay( );
}
}
Class  program
  {
      Void  Main ( )
      {
         Employee  emp1  =  new  Employee(111, "Rama", "SftwareEng");
         emp1. EmployeeDisplay();
      } Console. ReadLine();
```


Object
headloc = NULL
Delhi
headphNO = ∅
123444
Head off Display()
emp1
NULL
1010
branchid = ∅
1
branchName = NULL
HydBranch
branchloc = NULL
Ameerpet
BranchDisplay()
eno = ∅
111
ename = NULL
Rama.
designation = NULL
SoftwareEngg
EmployeeDisplay()
1010

3. Multiple Inheritence :-

→ Inheriting multiple classes into single class is called as multiple inheritence.

→ Syntax :

```
Class C1
{

}
Class C2
{

}
Class C3 : C1, C2
{

}
```

Q) Can we implement multiple inheritence in C#.Net-?

Ans Yes. But not possible by using classes, which is possible with the help of interfaces.

Q) Why multiple interface inheritance is not possible by using C#.Net ?

Ans

HW 1) Implement one more real time case study by using all the oops concepts.

2) Implement one real time case study by using single inheritence

3) Implement one real time case study by using MultiLevel inheritence

4) Hybrid Inheritence :-

→ A combination of two inhenritence is nothing but Hybrid inheritence

5) Hierarchial Inheritence :-

→ Inheriting ~~real~~ single class into multiple classes is called as Hierarchial inheritence.

→ Syntax :

```
class C1
{

}
class C2 : C1
{

}
class C3 : C1
{

}
```

Example 1 :-

→ Object is super class for all .Net classes.

Example 2 :-

→ In ASP.Net Page is the super class for all web page classes like WebForm1, WebForm2 - - - , WebForm n.

Example 3 :-

→ In Windows forms, Form is the super class for all the forms like Form1, Form2 - - - - Form n.

→ In all the above 3 examples, MicroSoft has implemented Hierarchial inheritence.

Advantages of inheritence.

Re
→ Code Reusability.

Reusability:

→ Because of inheritence a method which is defined in the Superclass can be accessed by derived classes. That means here we are achieving code reusability with inheritence.

4. Polymorphism :-

→ Polymorphism means one name many forms.

→ Implementing multiple functionalities with the same name can be called as polymorphism.

~~→ Implemen~~

Def :

→ Implementing multiple methods with the same name with different behaviour is called as polymorphism.

→ Polymorphism can be classified into two ways

1. Static Polymorphism (or) Compile time Polymorphism
2. Dynamic Polymorphism (or) run time polymorphism.

1. Static Polymorphism :-

→ In static polymorphism, a method which will binded at compile time will execute.

→ Compile time polymorphism we are achieving in function overloading.

2. Dynamic Polymorphism :-

→ In dynamic polymorphism a method which is binding at runtime will execute.

→ Run time polymorphism we are achieving in function overriding

When we will go for Polymorphism?

Ans: Whenever we want to implement multiple methods with the same name in a single class (or) a combination of base & derived class we will go for polymorphism.

Q) when we will define multiple methods with same name.

Function Overloading :-

Def:

→ Implementing multiple methods with the same name but with the different signature in a single class (or) a combination of base and derived class is called as function overloading.

Q) what do u mean by different Signature?

Ans 1. Differentiating with no. of parameters.

2. Differentiating with type of parameters.

3. Differentiating with order of parameters.

Example 1 for function Overloading.

```
namespace functionOverloadingEx1.
{
  Class Calculate
  {
    internal void Add (int a, int b)
    {
      int res = a+b;
      Console.WL("Addition result of two integers is:"+res);
    }
    internal void Add( int a, int b, int c)
    {
      int res = a+b+c;
```

```
    Console.WL("Addition result of 3 integers is:"+res);
  }
internal void Add(double a, double b) //3rd Method
  {
    double res = a+b;
    Console.WL("Addition result of 2 double values is:"+res);
  }
internal void Add(double a, double b, double c)
  {
    double res = a+b+c;
    Console.WL("Addition result of 3 double values is:"+res);
  }
}
Class Program
  {
    void Main()
    {
      Calculate obj1 = new calculate();
      obj1.Add(10,5);
      obj1.Add(10,5,2);
      obj1.Add(0.002,0.003);
      obj1.Add(0.001, 0.002, 0.0004);
      Console.ReadLine();
    }
```

→ In the above program we have implemented function overloading

→ when we compile the above program, ~~CLR~~ compiler will bind a method based on reference variable type like below.

obj.Add (10, 5); → 1st Add

obj.Add (10, 5, 2); → 2nd Add

obj.Add (0.002, 0.003); → 3rd Add

obj.Add (0.001, 0.002, 0.004); → 4th Add

→ when we run the above prgm, because of new calculate();
it is creating object of calculate.

→ Due to that reason in run time also which are invoking the methods which are binded at compile time because here reference variable type and object type will be same.

→ Due to that reason function overloading is called as compile time binding.

→ Function overloading will depend on following 3 things

1. No-of Parameters
2. Type of Parameters
3. Order of Parameters

→ Example for no. of Parameters

```
namespace noofParameters
{
  class Myclass
  {
    internal void Print()
    {
      Console.WriteLine("zero parameter function is calling");
    }
    internal void Print(int a)
    {
      Console.WL("one parameter function value is:"+a);
    }
  }
  class Program
  {
    void Main()
    {
      Myclass obj = new Myclass();
      Obj.Print();
      Obj.Print(10);
      Console.ReadLine();
    }
```

→ Example for type of parameter to a function

```
namespace typeofParameters
{
  class Myclass
  {
    internal void Print(String s)
    {
      Console.WL("S value is:"+s);
    }
    internal void Print(int a)
    {
      Console.WL("a value is:"+a);
    }
  }
```

```
Class Program
{
    void Main()
    {
        Myclass obj = new Myclass();
        obj.Print("satya");
        obj.Print(10);
        Console.RL();
    }
```

-> Example of for Order of Parameters

```
namespace Order of Parameters
{
  Class Myclass
  {
    internal void Print(int a, String s)
    {
      Console.WL("a value is: {0}, s value is: {1}", a, s);
    }
    internal void Print(string s, int a)
    {
      Console.WL("s value is: {0}, a value is: {1}", s, a);
    }
  }
Class Program
{
    void Main()
    {
        Myclass obj = new Myclass();
        obj.Print(10, "satya");
        obj.Print("satya", 10);
        Console.RL();
    }
```

→ Function overloading will not depend on following 2 things

1. Return type of a function.
2. Parameter name.

→ Example for return type of a function.

```
namespace Returntypeof function
{
    Class Myclass
    {
        internal void Print()
        {
            Console.WL("void function is calling");
        }
        internal int Print()
        {
            return 10;
        }
    }
    class Program
}
```

→ The above code will generate a compile time error because, we cannot overload two functions because of only diff return types with this we can say function overloading will not depend on return types.

→ Example for parametername

```
namespace parameternamesEx
{
    Class Myclass
    {  int x;
        internal void Print(int a)
        {
```

```
    Console.WriteLine("a value is :"+a);
  }
  internal void Print(int b)
  {
    Console.WriteLine(" b value is:"+b);
  }
}
```

→ The above code will generate a compile time error because function overloading will not depend on parameter names.

→ Realtime ex for function Overloading.

```
namespace functionOverloading
{
  Class Employee
  {
    double basic, hra, ta, da, gross;
    internal void calGrossSal(double basic, double hra, double da)
    {
        gross = basic + hra + da;
        Console.WL(" Software Engineer gross salary is: "+gross);
    }
    internal void calGrossSal(double basic, double hra, double da,
                                double ta)
    {
        gross = basic + hra + da + ta;
        Console.WL("Manager gross salary is:"+gross);
    }
  }
```

```
Class Program
{
    void Main()
    {
        Employee se = new Employee();
        se.CalGrossSal(10000, 400, 300);

        Employee mgr = new Employee();
        mgr.CalGrossSal(20000, 800, 600, 400);

        Console.RL();
    }
```

List of the Predefined overloaded methods in C#.Net.

1. WriteLine() ⟶ 19
2. Write() ⟶ 18
3. ToInt32() ⟶ 19

⋮

Questions in Oops

1. Why class?
2. Procedural oriented approach vs Object oriented approach
3. why object?
4. Why instance variable!
5. Why static variable
6. when we will go for local variables
7. Why method return type.
8. Why parameter
9. What is the role of access modifier
10. what is the drawback of default access modifier

11. when will go for parameterized constructor

12. why static constructor does not have access modifier and parameter.

13. why we cannot change the main method name.

14. what is access modifier of object class and object class Object class default constructor and GetHashcode()

15. When we will declare internal and when we will go for public, Protected.

16. what is access modifier of console class and WriteLine()

17. Why main method is static?

18. when we will go for instance and Static methods.

19. Can we initialize values with the help of a method if yes why constructor

20. when readonly.

21. Can we declare static readonly? when?

22. when call by value, when call by reference, when call by out?

23. what is the importance of properties in C#.Net.

24. when we will go for constant.

25. what is the importance of this and base and this() and base()?

→ Example for method Abstraction.

→ Hiding unwanted method is called method Abstraction

```
namespace    methodAbstraction
{
    class    Myclass
    {
      internal void Print()
      {
        Console.WriteLine("Zero parameter method is calling");
      }
      internal void Print(int a)
      {
        Console.WriteLine("One parameter method is calling");
      }
      internal void Print(int a ,int b)
      {
        Console.WriteLine("Two parameter method is calling");
      }
    }
    class Program
    {
        void Main()
        {
            Myclass mc = new Myclass();
            mc.Print();
            Console.ReadLine();
        }
```

→ In the above program when control is executing below statement

mc.Print();

→ It will invoke first print method, but will hide second and third methods which is called as method Abstraction.

→ With this we can say method hiding can be achieved with function overloading.

2. Function Overriding :-

→ Having multiple methods with same name, same signature in a combination of base and derived class, while ~~overloading~~ overriding for base class method we should use virtual keyword and for derived class method we should use override keyword.

Q) Can we override in single class?

Ans No.

Q) what do you mean by same signature?

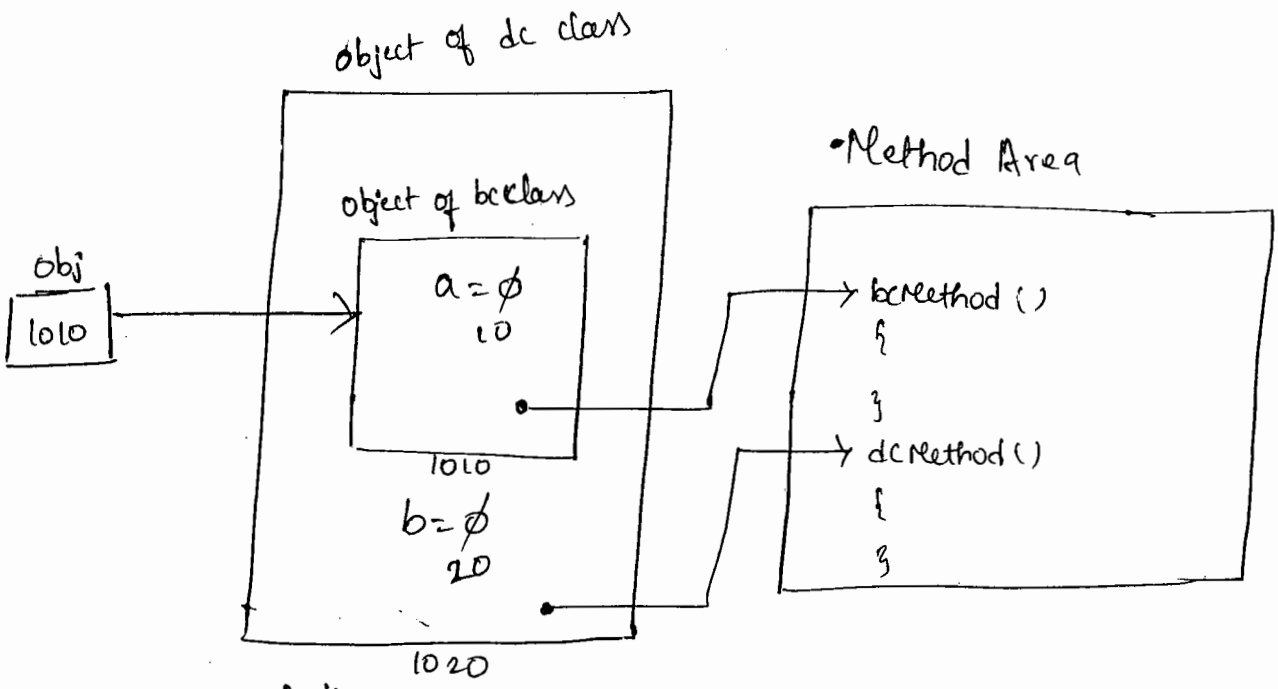
Ans Same no. of Parameters ~~(or)~~ and same type of parameters and same order of parameters.

→ In function overriding both methods return type should be same.

→ Function overriding is depending on inheritence concept.

→ Example to create super class reference variable and subclass object.

```
namespace inheritenceEx
{
  class bc
  {
    int a = 10;
    internal void bcmethod()
    {
      Console.WL("bcmethod value is:"+a);
    }
  }
  class dc:bc
  { int b = 20;
    internal void dcmethod()
    {
      Console.WL("dcmethod value is:"+b);
    }
  }
  class Program
  {
    void Main()
    {
      bc obj = new dc();
      Obj.bcmethod();
      obj.dcmethod(); //error
      Console.ReadLine();
    }
```



→ Example for function overriding

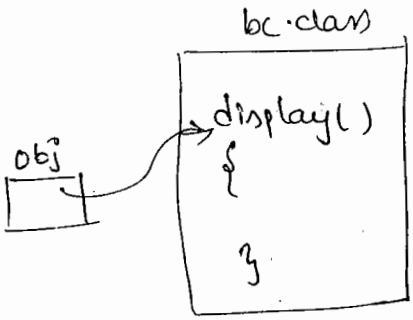
```
namespace functionoverriding1.
{
  class bc
  {
     internal virtual void Display()
     {
       Console.WL(" bc display is calling");
     }
  }
  class dc : bc
  {
    internal ~~void~~ override void Display() :
    {
      Console.WL(" dc display is calling");
    }
  }
  class Program
  {
      Void Main()
```

```
{
    bc obj = new dc();
    obj.Display();
    Console.ReadLine();
}
```

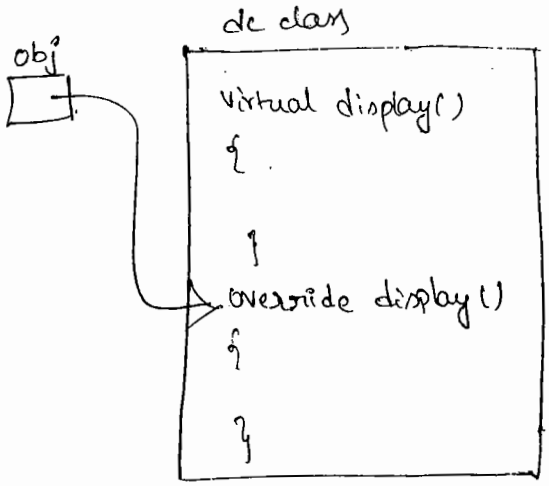
→ when we compile the above program, compiler will bind the method based on reference variable type. According to that here obj is a reference variable of bc class. Due to that reason below statement

```
obj.Display();
```

will bind to display method of bc class like below. in compile time



→ when we run the above program because of new dc(); it will binde to dc class display()



→ Here we have to identify that override method is hiding the virtual method.

→ In runtime below statement

obj.Display();

will invoke the, a method which is binding at runtime that is nothing but dc display.

→ Finally we can say in function overriding a method which is binding at ~~bin~~ compile time is not executing. Instead of that a method which is binding at run time is executing. Due to that reason function overriding is called as Dynamic Polymorphism

o/p

dc display is calling.

Q) Can we say below is dynamic Polymorphism.

```
bc obj = new bc();
obj.Display();
```

→ No. It is no dynamic

Q) Can we say below statement is dynamic Polymorphism.

```
dc obj = new dc();
obj.Display();
```

→ No.

→ Realtime example for function overriding.

```
namespace RealtimeEx
{
  class Employee
  {
    double grossSal;
    internal virtual void calGrossSal(double basic)
    {
      double hra = 0.4 * basic;
      grossSal = basic + hra;
      Console.WriteLine("Employee Gross Salary is :"+grossSal);
    }
  }
  Class Manager : Employee
  {
    internal override void calGrossSal(double basic)
    {
      double hra = 0.4 * basic;
      double da = 0.3 * basic;
      base.grossSal = basic + hra + da;
      Console.WL("Manager gross Salary is :"+grossSal);
    }
  }
  Class Program
  {
    Void Program
    {
```

```
Employee emp1 = new Manager();
emp1.calGrossSal(1000);
Console.ReadLine();
}
```

o/p

Manager Gross Salary is: 1400

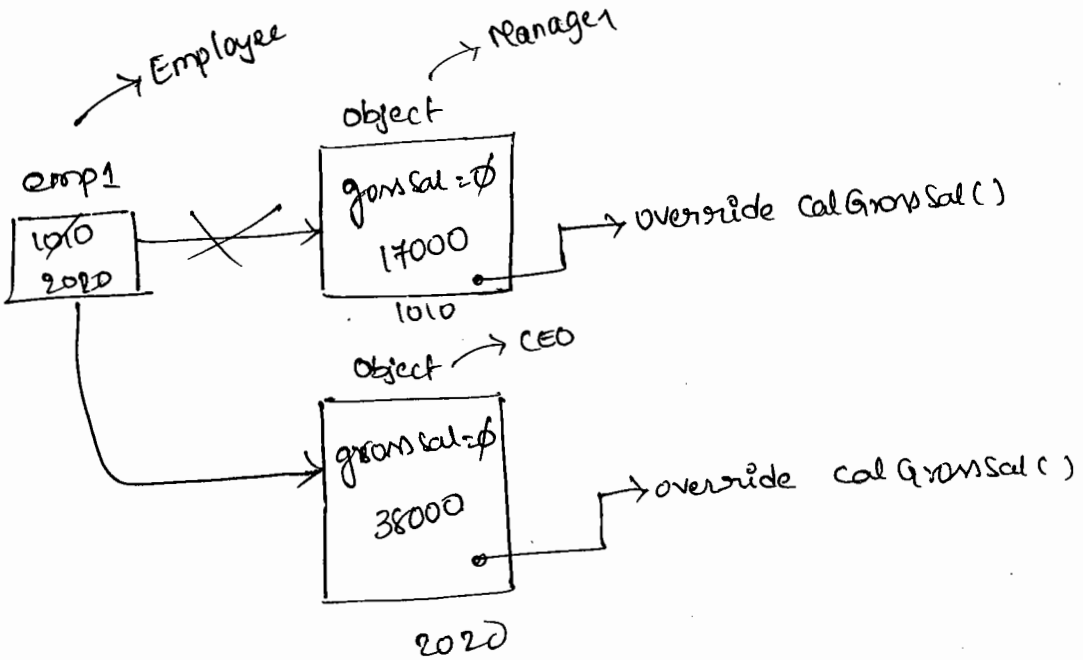
Q) In function overriding why super class reference variable and sub class object.

→ In function overriding we can override a method in multiple derived classes like below.

```
namespace RealTimeEx2
{
    class Employee
    {
        int eno;
        string
Protected double grossSal;

Internal virtual void calGrossSal(double basic)
    {
        double hra = 0.4* basic;
        grossSal = hra + basic;
        Console.WL("Employee Gross Salary is:"+grossSal);
    }
```

```
class Manager : Employee
{
    internal override void CalGrossSal(double basic)
    {
        double hra = 0.4 * basic;
        double da = 0.3 * basic;
    base.grossSal = hra + basic + da;
Console.WL("Manager Gross Salary is : " + grossSal);
    }
}

class CEO : Employee
{
    Internal override void CalGrossSal(double basic)
    {
        double hra = 0.4 * basic;
        double da = 0.3 * basic;
        double ta = 0.2 * basic;
        base.grossSal = hra + da + ta + basic;
        Console.WL("CEO Gross Salary is : " + grossSal);
    }
}
class Program
{
    void Main()
    {
        Employee emp1 = new Manager();
        emp1.CalGrossSal(10000);
```

```
emp1 = new CEO();
emp1.CalGrossSal(20000);
Console.ReadLine();
}
```



→ In function overriding we will create the super class reference variable and sub class object like above. Because of this we can unreference the previous objects always only current object will be in live and with this we can save the memory.

Method Hiding :-

→ Method hiding and function overriding will be similar.

→ In function overriding for base class method we will use virtual keyword for derived class method we use override keyword.

→ But in method hiding for base class method we will not use any keyword, but for derived class method we have to use new keyword.

→ Hiding superclass method in subclass by using new keyword is called as method hiding.

→ Example for method hiding.

```
namespace MethodHidingEx1.
{
  class bc
  {
    internal void display()
    {
      Console.WL("bc display is calling");
    }
  }
  class dc:bc
  {
    internal new void display()
    {
      Console.WL("dc display is calling");
    }
  }
  class Program
  {
    void Main
    {
      dc obj = new dc(); obj.display();
      Console.RL();
    }
```

op

dc display is calling.

→ Example 2.

```
namespace MethodHidingEx2
{
  // defining bc , dc classes as above

  class program
  {
    void Main()
    {
      bc obj= new dc();
      obj.display();
      console.RL();
    }
```

o/p

bc display is calling

→ whenever we want to implement superclass method in one subclass we can use Method hiding.

→ whenever we want to implement superclass method in multiple derived classes better to go for function overriding.

```
namespace MethodHiding Ex3
{
  // defining bc , dc as above

  class tc : bc
  {
    new internal void display()
    {
      Console.WL("tc display is calling");
    }
  }
  class Program
  {
    void Main()
    {
      bc obj = new dc(); obj.display();
      obj = new tc(); obj.display(); C.RL(); }
```

o/p

bc display calling

bc display calling

ctd...

Exception Handling

Q) what is an Exception?

Ans An exception is nothing but runtime error.

Q) what is exception Handling?

Ans Exception handling is a Predefined mechanism to handle runtime errors by using try, catch and finally blocks..

Q) Purpose of Exception handling?

Ans → To handle runtime errors we will go for exception handling

→ Whenever an error is occured while executing the Program, Program execution will not terminate if we have implemented exception handling mechanism and it will display user friendly error messages.

→ To implement exception handling mechanism Microsoft is providing a collection of predefined classes for .Net programmers like below.

1. DivideByZeroException class
2. OverflowException class
3. FormatException class
4. ArithmeticException class
5. ... and so on

→ All the above predefined classes are part of System base class library.

→ Syntax for try, catch, finally blocks

```
Try
{

}
Catch(<ExceptionClass>, <object name>)
{

}
finally
{

}
```

Predefined → ExceptionClass

User defined → object name

→ Within the Try block we have to write the statements which may through an error.

→ Within the Catch block we have to write the code to handle the error by displaying the user friendly message.

→ Within the finally block we have to write the error free code (or) the code which we want to execute irrespective of error occurence.

<u>Execution Flow of Try, catch and finally:</u>

→ When there is an error

→ While executing try block when there is an error control will comeout from the try block and it will execute appropriate catch block then it will execute finally block.

→ While executing try block if there is no error then it will execute full try block and it will skip catch block and it will execute finally block.

→ Catch block will execute only when there is an error, but finally block will execute always irrespective of error occurance.

→ A try block can follow with one catch block (or) multiple catch blocks and with finally (or) without finally

→ Write a Console program by using try, catch handle the divide by zero error.

```
Void Main()
{
    Console.WriteLine (" Enter first no:");
    int a = Convert.ToInt32(Console.ReadLine());
    Console.Writeline (" Enter second no:");
    int b= Convert.ToInt32 (Console.ReadLine());
    try
    {
      int div = a/b;
      Console.WL("div result is:"+div);
    }
    catch (DivideByZeroException de)
     {
       Console.WL(" A value which cant divide by zero");
     }
    Console.ReadLine();
}
```

→ Example to handle divide by zero error by using try, catch and finally blocks.

```
Void Main()
{
    Console.writeline ("Enter first no:");
    int a = int.parse (Console. ReadLine());
    Console.Writeline (" Enter second no:");
    int b = int.Parse (Console. ReadLine();
    try
     {
        int div = a/b;
        Console.writeline (" Division result is :"+ div);
     }
    catch (DivideByZeroException de)
     {
        Console. WL(" A value which cannot be divided by zero");
     }
    finally
     {
        Console. WL ("finally block is calling");
     }
    Console. ReadLine();
}
```

→ With the above program we can say that finally block will execute always irrespective of error occurence.

→ Example for only try block

```
Void Main()
{
    Console.WL(" Enter first no:");
    int a = int.parse(Console.RL());
    Console.WL(" Enter second no:");
    int b= int.Parse(Console.RL());
    try
    {
        int div = a/b;
        Console.WL(" Division result is :"+div);
    }
    Console.RL();
}
```

→ O/P is compile time error. try block should follow with either catch block (or) finally block.

→ Example for try and finally.

```
Void Main()
{
    // first 4 statements same as above
    try
    {
        int div = a/b;
        Console.WL(" Division result is :"+div);
    }
    finally
    {
        Console.WL("finally block is calling");
    }
```

```
    Console.RL();
  }
```

→ With the help of only try and finally we cannot handle the exceptions.

Multiple Catch blocks :-

When we will go for multiple catch blocks.

→ Whenever we are expecting more than one error from the try block, we require to implement multiple catch blocks.

→ Example for multiple catch blocks.

```
Void Main()
{ int a, b, div;
  try
  {
    Console.WL("Enter first no:");
    a = int.Parse(Console.RL());
    Console.WL("Enter second no:");
    b = int.Parse(Console.RL());
    div = a/b;
    Console.WL("division result is :"+div);
  }
  Catch (OverFlowException oe)
  {
    Console.WL(oe.Message);
  }
  Catch (FormatException fe)
  {
    Console.WL(fe.Message);
  }
```

```
Catch(DivideByZeroException de)
  {
    Console.WL(de.Message);
  }
  Console.RL();
} Catch (Exception ee)
  {
    Console.WL(ee.Message);
  }
  Console.RL();
}
```

Example to handle all the errors by using Single catch block.

→ To handle all the errors by using single catch block we will write generalized catch block, in generalised catch block we will write a class called Exception.

Exception class :-

→ It is a predefined class, which is capable to handle all the errors because Exception class is the Super class for all .Net exception classes like OverflowException class, FormatException class, DivideByZeroException class ----

Message:

→ It is a predefined member property of exception class which will display the concern error predefined message.

```
Void Main()
{
  int a, b, div;
  try
  {
    Console.WL("Enter first no:");
    a = int.Parse (Console.RL());
    Console.WL("Enter second no:");
    b = int.Parse (Console.RL());
    div = a/b;
    Console.WL("Division result is:"+div);
  }
  catch (Exception ee)
  {
    Console.WL(ee.Message);
  }
  Console.RL();
}
```

Example for empty catch block.

```
Void Main()
{
  int a, b, div;
  try
  {
    Console.WL("Enter first no:");
    a = int.Parse (Console.RL());
```

```
Console.WL("Enter second no:");
b= int.Parse(Console.RL());
div = a/b;
Console.WL("division result is:"+div);
}
catch
{
  Console.WL("error is occured");
}
Console.RL();
}
```

<u>Note</u>: empty catch block can also handle all the type of errors because it will act like a generalized catch block.

What ~~is the~~ should be the order of catch blocks?

→ When we are implementing multiple catch blocks first we have to write appropriate catch blocks and last catch block should be generalized catch block.

Execution flow of multiple catch blocks:

→ whenever there is an error is occured in try block, control will comeout of the try block then it will search for the appropriate catch block, and it will execute appropriate catch block. then finally...

→ If the appropriate catch block is not available then control will execute generalized catch block and finally block.....

Example to throw an exception explicitly.

→ Whenever a programmer wants to throw an exception explicitly he can throw by using throw keyword.

```
Void Main ()
{
  try
  {
     throw new DivideByZeroException();
  }
  Catch(DivideByZeroException de)
  {
     Console.WL(de.Message);
  }
  Console.RL();
}
```

List out the available predefined Exception classes in .Net.

Implement multiple examples with exception handling mechanism to handle various types of exceptions.

ctd..

```
namespace MethodHidingEx4
{
  class bc
  {
    Internal virtual void display()
      {
        Console.WL(" bc display is calling");
      }
  }
 class dc : bc
  {
    Internal override void display()
     {
       Console.WL (" dc display is calling");
     }
  }
 class tc : bc
   {
     internal override void display()
      {
        Console.WL(" tc display is calling");
      }
   }
 class program
  {
    void Main()
    {
     // as above
     }
  }
```

```
class bc
{
        // Same as above with virtual keyword.
}
class dc : bc
{
internal new void display()
    {
      CWL(
    }
}
class tc : dc
{
    internal override void display()
      {
        CWL(
      }
}
class program
{
    void Main()
      {
          bc obj = new tc();
          obj.display();
          Console.RL();
      }
```

O/P

Compile time error.

Differences between function loading and function overriding.

Over Loading	over riding
→ ~~Stat~~	
→ Multiple methods with same name but with different signature.	→ Can have multiple methods with same name and with same signature.
→ To implement overloading we doent require any key words	→ To implement function overriding we use virtual and override keywords.
→ It is static polymorphism.	→ It is dynamic polymorphism
→ This can be implemented in single class and also in combination on base and derived class.	→ It is not possible in a single class. we have to go for base and derived class
→ Static methods can be overloaded.	→ Static methods cannot be overloaded.

~~Advanced~~

Abstract class

→ while defining a class if we have used abstract ~~class~~ keyword which is called as Abstract class.

→ Abstract class is a collection of Abstract members and non-Abstract members.

What do you mean by Abstract member?

→ While declaring a class member if we have used Abstract Keyword which is called as Abstract member

→ Abstract members are empty members. which will have only declaration within the abstract class, we have to define (or) we have to implement these abstract members within the derived class by using override keyword.

What do you mean by declaration and what do mean by definition

Ex:

```
internal void show()        → declaration
{
    Console.WL("Show method is calling");
}
```

definition

→ A field cannot be abstract.

→ We can have abstract Methods.

→ Example to declare Abstract Method.

```
internal abstract void show();
```

→ We can have Abstract properties.

Ex:

```
internal abstract int x;
{
    get;
    Set;
}
```

→ We cannot have abstract constructors.

→ Abstract class can contain a constructor which is non abstract

→ Abstract ~~class~~ members access modifiers should not be private because we should implement these abstract members in derived class by using override keyword.

→ Static member cannot be abstract.

→ Bydefault abstract member is a virtual member.

→ A class which contains normal members (or) non-abstract members (or) concrete members is called as normal class (or) non-abstract class (or) concrete class and it is fully implemented class.

→ A class which contains abstract members and non-abstract members is called as abstract class which is partially implemented class.

→ Abstract class cannot be instanciated but we can create reference variable.

→ Example for abstract class

```
namespace AbstractClassEx1
{
    Abstract class Vehicle
    {
        internal void Start()
        {
            Console.WL(" Vehicle has started ");
        }
```

```
internal void Drive()
{
  Console.WL("Vehicle has been driving");
}
 // abstract members

 internal abstract void Park();
 internal abstract string Colour
  {
     get;
     set;
  }
 Internal abstract void Stop();
}
class dc : Vehicle
{
    // implementing abstract members in derived class using
       override keyword.
   internal override void Park()
   {
     Console.WL(" vehicle has parked");
   }
   Internal string colour;
   internal override string Colour
    {
      get
        {
          return colour;
        }
      Set
        {
          colour = value;
        }
    }
```

```
    internal override void Stop()
    {
        Console.WL("Vehicle has stopped");
    }
}
class program
{
    void Main()
    {
        Vehicle obj = new dc();
        Obj.Start();
        obj.Drive();
        obj.Park();
        obj.Colour = "Red";
        Console.WL("Vehicle colour is :"+Colour);
        obj.Stop();
        Console.RL();
    }
}
}
```

When we will declare Abstract class?

→ According to the requirement whenever we want to implement some methods in the current class and some methods required to implement in derived classes in future then we will declare particular class as a Abstract class

Example.

```
namespace   Electricity
 {
    abstract class Customer
     {
        Internal void AboutInfo()
         {
          Console.WL(" Electricity department information");
         }
       internal abstract void CalBill(int totunits);
     }
   Class IndustryCustomer : Customer
    {
       internal override void CalBill (int totunit ~~totBill~~)
        {
            int bill = totunits * 7;
            Console.WL(" Your total bill is : " + bill);
        }
    }
  Class Residential Customer : Customer
    {
       internal override void CalBill (int totunits)
        {
            int bill = totunits * 5;
            Console.WL(" Your total bill is : " + bill);
        }
    }
```

```
class Program
{
    void Main()
    {
        Customer cust1 = new ResidentialCustomer();
        Cust1.AboutInfo();
        Cust1.CallBill(100);

        Customer cust1=new IndustryCustomer();

        Cust1.AboutInfo();
        Cust1.CallBill(200);
        Console.ReadLine();
    }
```

O/p

Electricity department Information
Your total bill is : 500

Electricity department Information
Your total bill is : 1400

Interface

→ To define a Interface we have to use interface keyword.

→ Interface looks like a class but it has no implementation.

→ An interface is a collection of abstract members, bydefault interface members are public and abstract.

→ Syntax to define interface :-

```
Interface <Interface name>
{
    // abstract members
}
```

Note: Interface name should start with : I

Ex: IEmployee , IStudent . . .

→ Interface members we should implement with in derived class without using override keyword.

→ An interface cannot contain fields.

→ Interface cannot contain constructor.

→ We cannot implement a property (or) method in Interface.

→ Interface cannot contain static members.

→ Using interface we can implement multiple inheritence.

→ We cannot create an object for interface but we can create reference variable for interface.

→ Example to implement Single Inheritence by using interface.

```
Interface ImyInterface
{
  void Print();
  int x
  {
   get ;
   Set ;
  }
}
class dc : Imyinterface
{
   //Implementing Interface members

  Public void Print() .
   {
     Console.Writeline (" Print is calling");
   }
  int x;
  Public int x
   {
     get
       {return x ;
       }
     Set
       {
         x = value;
       }
   }
  }
```

```
class Program
{
    void Main()
    {
        Imyinterface  obj = new dc();
        obj.Print();
        obj.x = 100; Console.WL("x value is :"+obj.X);
        Console.ReadLine();
    }
```

o/p

Print is calling

x value is : 100

→ Example for Multiple inheritence by using interfaces.

```
namespace InterfaceEx2
{
  Interface Inokia1
  {
      void calling();
      void Receiving();
      void Sendmsg();
      void Endcall();
  }
  class Nokia1100:Inokia1
  {
      Public void calling
      {
```

```
    Console.WL(" Nokia1100 is calling");
    }
  Public void Receiving
    {
     Console.WL(" Nokia 1100 is receiving call");
    }
  Public void SendMsg
    {
     Console.WL(" Nokia 1100 is sending msg");
    }
  Public void Endcall
    {
     Console.WL(" Nokia 1100 is ending call");
    }
}
interface Inokia2
{
   void Wifi();
   Void Vediocalling();
}
class NokiaAsha : Inokia1, Inokia2
 {
    Public void calling
     {
      Console.WL(" NokiaAsha is calling");
     }
    Public void Receiving
     {
      Console.WL("NokiaAsha is receiving call");
     }
```

```
Public void SendMsg
{
    Console.WL("NokiaAsha is sending msg");
}
Public void Endcall
{
    Console.WL("NokiaAsha is ending call");
}
Public void Wifi
{
    Console.WL("NokiaAsha Wifi is on");
}
Public void VedioCalling
{
    Console.WL("NokiaAsha vediocall is supporting");
}
}
class Program
{
    void Main()
    {
        Inokia1 obj1 = new Nokia1100();
        Obj1.Calling();
        obj1.Receiving();
        Obj1.SendMsg();
        obj1.Endcall();
        Obj1 = new NokiaAsha();
        obj1.calling();
        Obj1.Receiving();
```

```
        Obj1.SendMsg();
        Obj1.EndCall();
        Inokia2 Obj2 = (Inokia2) Obj1;
        Obj2.WiFi();
        Obj2.VedioCalling();
        Console.ReadLine();
    }
```

Can we implement multiple inheritence with a combination of one class and multiple interfaces.

Yes. But first we have to write class name then we have to write interfaces names like below.

```
interface I1
{
}
interface I2
{
}
Class bc
{
}
class dc : bc, I1, I2
{
}
```

when we will go for interface?

→ whenever we want to implement all the behaviours in future derived classes. we will go for interface. As well as to implement multiple inheritence we will go for interface.

Realtime Example for interface.

```
namespace BankEx
{
    Interface Iaccount
    {
        Void OpenAccount(int acNO, String acName, double IntAmt);
        Void WithDraw(double amt);
    }
    class CurrentAccount:Iaccount
    {
        int acNO;
        String acName;
        double bal;
        internal CurrentAccount (int acNO, String acName)
        {
            this.acNO = acNO;
            this.acName = acName;
        }
        // implementing interface methods
        Public Void OpenAccount (double IntAmt)
        {
            this.bal = IntAmt;
            Console.WL(" your account is created successfully");
            Console.WL(" your account details are:");
            Console.WL("Acct NO is :"+acNO);
```

```
Console.WL("Acct holder name is:"+acName);
Consol.WL("Your acct current balance is:"+this.bal);
}

Public void WithDraw(double amt)
{
    if(this.bal >= amt)
      {  this.bal = this.bal - amt;
         Console.WL("Your withdrawn is Successful");
         Console.WL("Your current balance is:"+ this.bal);
      }
    else
      {
         Console.WL(" Insufficient funds");
      }
  }
}
class SavingAccount:Iaccount
{
    int acNO;
    String acName;
    double bal;
    Internal SavingAccount(int acNO, string acName)
     {
        Console.WL(" Account is created");
        Console.WL
        Console.WL
        }
```

```
Public void openAccount (double InAmt)
{       if (InAmt>=1000)
        {
    this.bal = InAmt;

    Console.WL(" Your Account is created");

    Console.WL(" Current bal is :" + this.bal);
        }
    else
        {
            Console.WL(" Your Account cannot be created, min bal
                                Should be 1000");
        }
    }
Public void withDraw (double amt)
                                                        1000
    {
        if(this.bal - amt >=1000)
            {
                this.bal = this.bal - amt;

                Console.WL(" withDraw is Successful");

                Console.WL(" Your balance is :" + this.bal);
            }
        else
            {
                Console.WL(" Insufficient funds");
            }
    }
}
    Class Program
        {
            Void Main ( )
                {
```

```
Iaccount Cust1 = new CurrentAccount(111, "Rama");
Cust1.OpenAccount(3000);
Cust1.WithDraw(2000);
cust1 = new SavingAccount(222, "Ravi");
cust1.OpenAccount(4000);
Cust1.WithDraw(2000);
Console.ReadLine();
    }
  }
}
```

Difference between Abstract class and interface.

Abstract class	interface
→ Abstract keyword.	→ Interface keyword.
→ Should use override keyword in derived classes.	→ No need of override keyword in derived classes.
→ Can have non-abstract members also.	→ By default all members are abstract
→ Cannot implement multiple inheritence	→ Used mainly to implement multiple inheritence.
→ In abstract class we give same access modifier in base and derived class	→ In interface no need of access modifier, by default it is public
→ Partially implemented.	→ No implementation.
→ Can contain field.	→ Cannot contain fields.
→ Can contain constructor	→ Cannot contain constructor.

→ Can implement methods which is non abstract.	→ Cannot implement methods.
→ Can implement a property which is non abstract.	→ Cannot implement a property
→ By default access modifier of abstract class members will be Private.	→ By default access modifier of interface members will be public.
→ By default abstract class abstract members will be virtual.	→ Bydefault interface members will be abstract.
→ can contain static members but it should be non abstract.	→ cannot contain static members.

Benifits of Oops

→ Modularity (achieving with class).

→ Reusability (achieving with inheritance).

→ Extensibility (achieving with inheritance).

→ Reimplimentation (achieving with polymorphism).

→ Security (Access modifiers)

Memory Management

What is memory management.

→ Memory management is a process of allocating the memory and deallocating the memory.

→ In .Net memory management is handling by garbage collector.

→ Garbage collector is an integral component of CLR.

→ CLR is an integral component of .Net framework.

Managed Heap

→ Heap is a data structure which can contain collection of objects

→ Garbage collector is using the heap data structure for memory management due to that reason we will call it as a managed heap.

→ To perform memory management garbage collector will do 2 duties

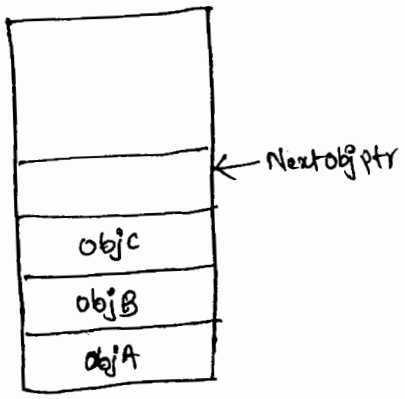
1. Allocating the memory
2. Deallocating the memory (or) releasing the memory.

1. Allocating the memory :

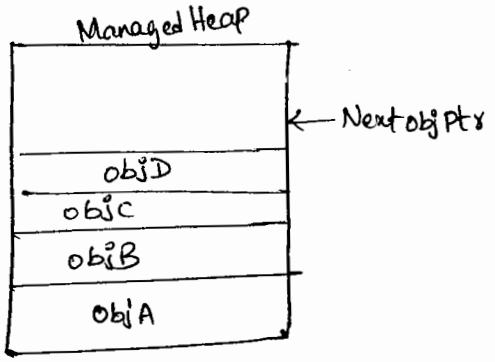
→ When a new object is created by the application, garbage collector will allocate memory for newly created object within the heap data structure where exactly next object pointer is pointing.

→ Next object pointer will be pointing to the next adjacent empty location within the heap.

→ Let us assume the heap will be occupied by couple of objects like below.



→ Let us assume our application is created a new object called objD, then garbage collector will allocate memory for objD within the heap data structure where exactly next object pointer is pointing and next object ptr will be moving to the next adjacent empty location like below.



→ In this way garbage collector will allocate the memory for newly created objects within the heap data structure.

→ In this way garbage collector allocating memory for objects within the heap continually, there is a chance of out of memory.

→ To overcome this garbage collector is doing the second duty called deallocating memory (or) releasing memory

2. Deallocating Memory :-

→ Deallocation is performing by the garbage collector in two steps

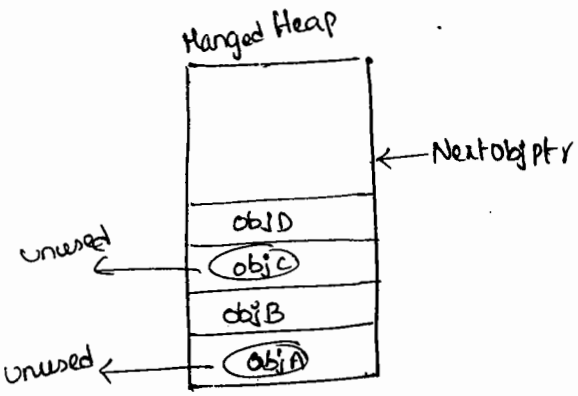
1. Recognizing the unused objects / unreferenced objects
2. Collection process.

1. Recognizing the unused objects.

What is unreferenced object?

→ An object which is not pointing by any reference variable will be called as unrefered (or) unused object.

→ As part of deallocation garbage collector will identify the unused objects within the heap and it will give a tag called unused object like below.



→ In the above heap diagram we have 4 objects. Among 4 objA, objC are unused objects and objB and objD are Used objects.

→ In this way garbage collector will recognize unused objects

2. Collection Process:-

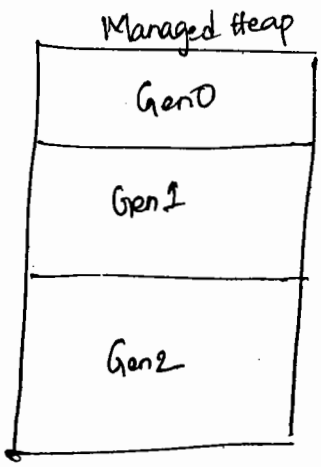
→ To implement collection process garbage collector will depend on generation algorithm.

→ According to generation algorithm, managed heap will be divided into 3 generations. They are

1. Generation 0: will contain just now created objects.
2. Generation 1: will contain just before created objects.
3. Generation 2: will contain long before created objects.

→ Managed heap will be divided in to 3 blocks.

→ Generation 0 block space < Generation 1 block space < Generation 2 block space.



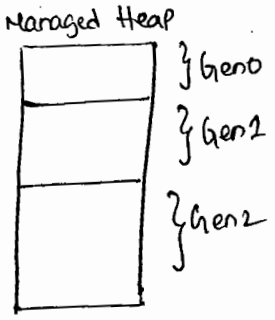
what is collection process?

→ collection process is nothing but destroying all unused objects with in current generation and moving the all used objects from current generation to next generation.

Ex: generation 0 to generation 1.

How garbage collector will implement collection process?

→ Whenever an application is initialized all the generations of the managed heap will be empty like below



→ Whenever new objects are created by the application garbage collector will allocate memory for newly created objects within the generation 0 continually till the generation 0 is full.

→ whenever generation 0 has no empty space then garbage collector will Perform the collection process with in the generation 0 is nothing but destroying the all unused objects of generation 0 and moving all used objects of generation 0 to generation 1.

→ Whenever generation 0 and generation 1 is full then garbage collector will Perform the collection process in generation 1 first (and gen0 second) means destroying all unused objects within the generation 1 and moving all used objects of generation 1 to generation 2.

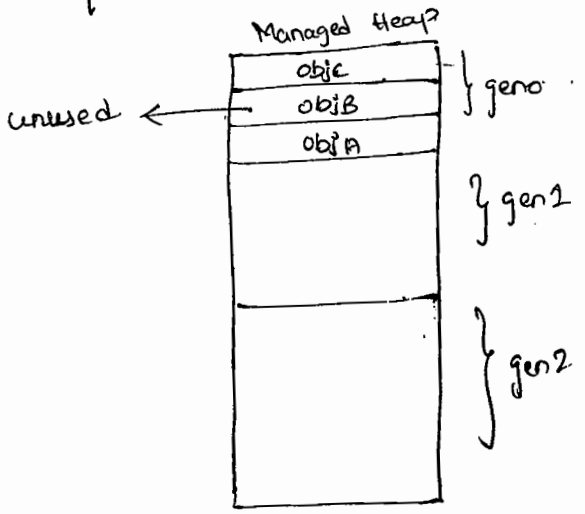
→ Whenever generation 0, generation 1 and generation 2 are full then garbage collector will perform the collection process within generation 2 first here most of the objects are unused objects due to that all unused objects of generation 2 will be destroyed.

→ After this it will perform collection process in generation 1 and generation 2.

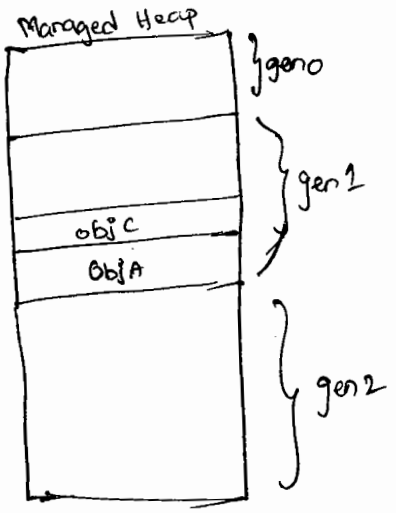
→ This collection process is a cycling process.

→ Finally with this we can say all new objects are allocating the memory within generation 0.

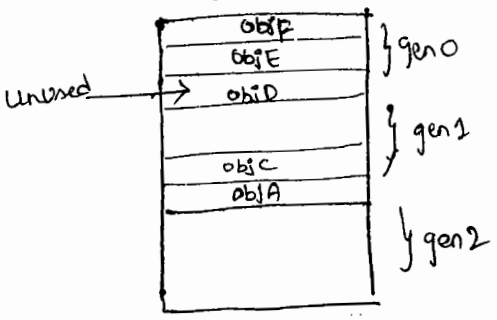
→ For example let us assume we have 3 objects like below in gen 0



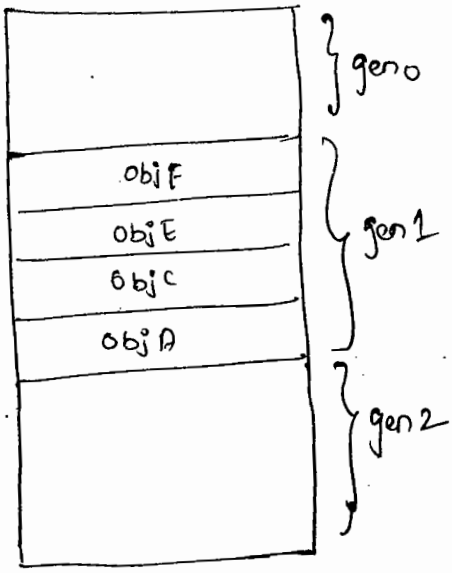
→ According to above diagram generation 0 has no space, due to that reason garbage collector will perform collection process within gen 0 like below.



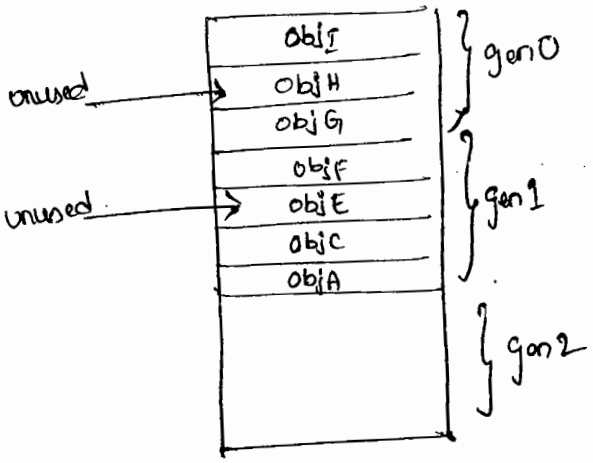
→ Now newly created objects will be allocated into generation 0 like below.



→ According to above diag gen 0 is full due to that reason again garbage Collector will perform collection process in gen 0



→ Let us assume newly created objects will be allocated within gen 0 like below



→ According to the above diagram gen 1 and gen 0 are full. due to that reason garbage collector will implement the collection process first in gen 1 and gen 0

→ Whenever 3 generations are full first garbage collector will implement collection process in generation 2, generation 1, generation 0.

→ It is a cycling Process

Why .Net memory management is called automatic memory management.

→ Memory management is performing by garbage collector internally. Here programmer will not involve in allocation and deallocation.

Is there any way to invoke the garbage collector by programmer.

→ Yes, like below

```
GC.Collect();
```

↳ class ↳ predefined method.

→ Collect method will invoke the garbage collector for collection process.

Is there any way to destroy an object by the programmer.

→ Yes, using dispose() method.

Ex: <objectname>.Dispose();

Delegates :-

→ Delegates are similar like C++ function pointers.

→ To define a delegate we have to use delegate keyword, delegates are reference types.

→ Delegate can hold the address of a function, using delegate object we can invoke a method without using ~~delee~~ method name.

→ Delegates are two type

1. Single caste delegate :- It can hold address of a single method.
2. Multi cast delegate :- It can hold addresses of multiple methods.

→ Using multi cast delegate we can invoke multiple methods with a single call.

Implementation of delegate concept can be divided into following steps

Step 1 : Defining a class

Step 2 : Defining a method.

Step 3 : Defining a delegate

<Syntax>

<accessmodifier> delegate <returntype> <delegate name>(<type> <arg1>, <type> <arg2>);

↓ Keyword.

Step 4 : creating object for delegate and initializing the method name.

<delegatename> <delegate objname> = new <delegatename>(<methodname>);

Step 5 : Invoking the method by using delegate object.

<delegateobjname ( )> ;

→ Delegates are called as type function pointers because delegate declaration should follow the method declaration which is holding by the delegate object. that means method return type and signature should be same for delegate.

Note : Delegate can hold the address of static methods as well as instance methods.

Example to invoke instance methods by using delegate object. (Single cast delegate)

```
namespace SingleCastDelegateEx1
{
    class calculate
    {
```

```
    internal int add(int a, int b)
      {
        return a+b;
      }
  }
internal delegate int MyDelegate (int x, int y);
  class Program
   {
      void Main( )
       {  Calculate obj = new calculate();
          MyDelegate delobj = new Mydelegate (obj.add);
  int res = delobj (10, 5);
      Console.WriteLine ("Addition result is:" + res);
      Console.ReadLine();
       }
   }
}
```

HW Implement single cast delegate to invoke static methods.

Multicast delegate

→ Multi cast delegate return type should be void because it can hold ~~to~~ only void methods.

Example for multicast delegate to invoke instance methods.

```
namespace multicastDelegateEx1
 {
    class Myclass
     {
        internal void Method1( )
         {
            Console.WriteLine(" Method1 is calling");
         }
```

```
Internal void Method2()
  {
    Console.WriteLine("Method2 is calling");
  }
}
Internal delegate void MultiDelegate();
class Program
{
    void Main()
    {
        Mydass obj = new Mydass();
        MultiDelegate delobj = new MultiDelegate(obj.Method1);
        delobj += new MultiDelegate(obj.Method2);
        delobj();
        delobj -= new MultiDelegate(obj.Method1);
        delobj();
        delobj += new MultiDelegate(obj.Method1);
        delobj();
        delobj -= new MultiDelegate(obj.Method2);
        delobj();
        Console.ReadLine();
    }
```

o/p

```
Method1 is calling
Method2 is calling
Method2 is calling
Method2 is calling
Method1 is calling
Method1 is calling
```

Advantages of delegates

→ Using delegates with a single call we can invoke multiple methods.

→ Implementation of delegates will improve the performance of the application.

Multiple class files

→ By default every project will come with single class file but it can contain multiple class files.

Example for multiple class files

Step 1: Open a Console application and rename it as multiple class files Example

Step 2: Add one more class file ~~to~~ the solution Explorer ~~the~~ i.e., class1.cs

Step 3: Write the below code within class1.cs

```
namespace MultipleClassFiles Example
{
  class Employee
  {
    internal void Induction()
    {
      Console.WriteLine("Every employee has to attend induction
    }                                  training");
    internal void Appraisal()
    {
      Console.WL("Every employee will have apprisal for an year");
    }
```

```
internal void Nop()
{
  Console.WL("Every Employee has to serve notice period while
                        leaving company");
}

}
}
```

<u>Step 4</u> : Write below code program.cs

```
namespace MultipleclassFilesExample
{
   class Program
   {
      Void Main( )
      {
         Employee obj = new employee();
         Obj.Induction();
         Obj.Appraisal();
         obj.Nop();
         Console.ReadLine();
      }
   }
}
```

Multi Threading :-

what is a thread?

→ Thread is an a independent execution path, it able to run simultaneously with other execution paths.

What is multi threading?

→ Executing multiple threads simultaneously is called as multi threading.

Purpose of multi threading

→ Whenever we want to execute multiple functionalities simultaneously we can go for multi threading.

Benifits of multi threading.

→ we can improve the performance of the application

What we can do using a thread.

→ Using thread we can start method execution, we can send method for sleep, we can suspend the method execution as well as we can call back the suspended method as well as we can terminate the method execution permanently.

What is the base class library for multi threading?

System.Threading.

How to create a user defined thread?

```
Thread thr1 = new Thread();
```

↓

thread object

(or)

user defined thread.

Thread

→ Thread is a predefined class which is part of System.Threading base class library.

→ Within Thread class microsoft defined all the Thread related required methods called.

1. Start()
2. Sleep()
3. Suspend()
4. abort()
5. resume() and so..on

How to initialize method to Thread

We can initialize method to Thread in two ways

1. With the help of ThreadStart delegate.

→ It can be in two steps

Step1: creating object for ThreadStart delegate and initializing method name

```
ThreadStart tstart1 = new ThreadStart (<methodname>);
```

↓ Predefined delegate

↓ delegate object

Step2: Creating object for Thread class and initializing delegate object like below.

```
Thread thr1 = new Thread (tstart1);
```

↓ delegate object

2. Without using ThreadStart delegate object.

Thread thr1 = new Thread(<methodname>);

How to invoke a thread?

thr1.Start();

Start()

→ It is a predefined member method of Thread class.

→ This method will invoke the given method.

Implementation of MultiThreading will be divided into following steps

Step 1 : Defining a class

Step 2 : Defining methods.

Step 3 : Creating ThreadStart delegate object.

Step 4 : Creating Thread object.

Step 5 : Invoking method by using Thread.

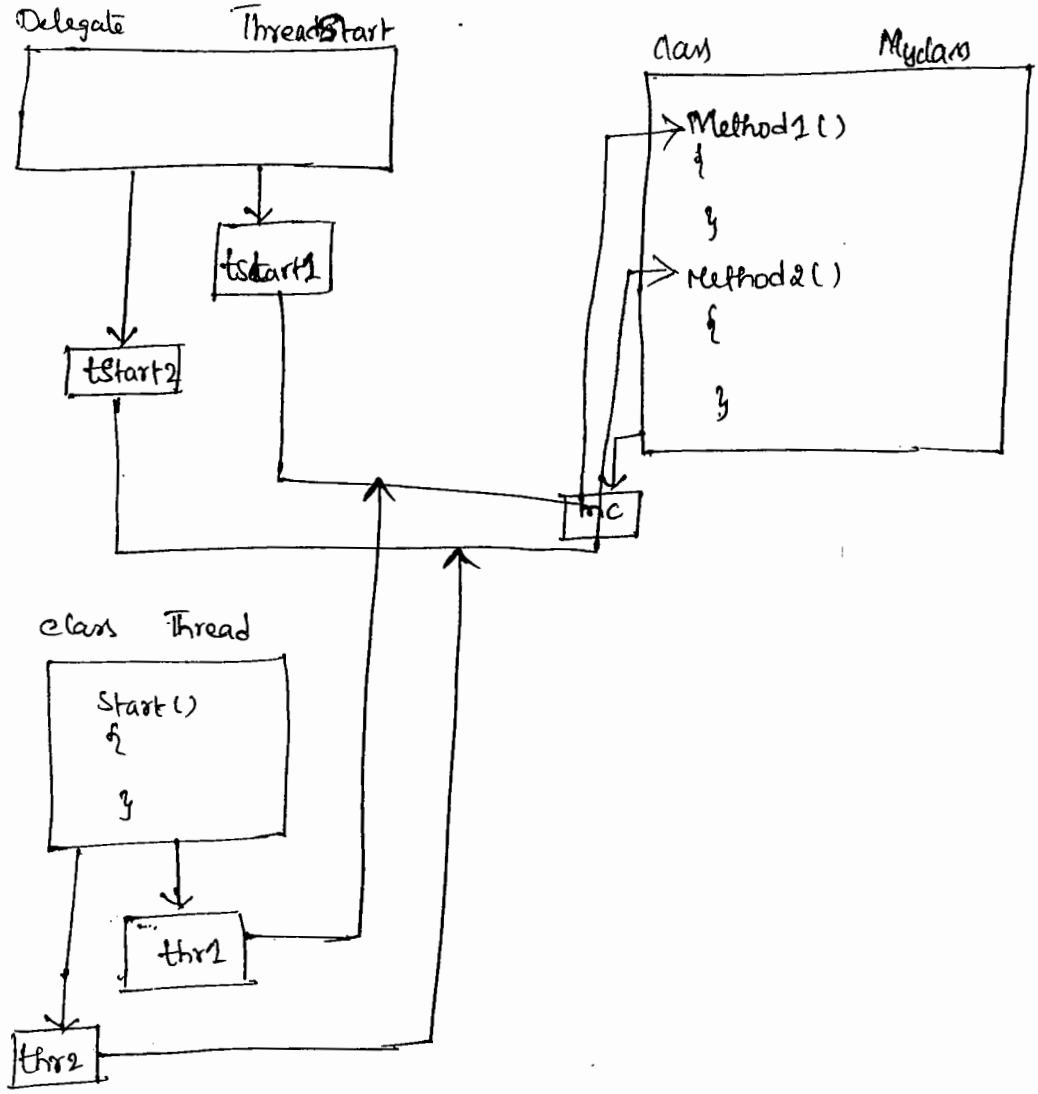
What is ThreadStart()?

→ ThreadStart is a predefined delegate which is part of System.Threading base class library.

Note: Using Thread we can control static methods as well as instance method.

Example to invoke a instance method by using Threads.

```
class Myclass
{
    internal void Method1()
    {
        for(int i=1; i<=10; i++)
        {
            Console.WL("Method1 = "+i);
        }
    }
    internal void Method2()
    {
        for(int i=1; i<=10; i++)
        {
            Console.WL("Method2 = "+i);
        }
    }
}
Class Program
{
    Void Main()
    {   Myclass obj = new Myclass();
        ThreadStart tstart1 = new ThreadStart(obj.Method1);
        Thread thr1 = new Thread(tstart1);
        thr1.Start();
        ThreadStart tstart2 = new ThreadStart(obj.Method2);
        Thread thr2 = new Thread(tstart2);
        thr2.Start();
    }   Console.ReadLine();
```



→ Implement the above prgm as static methods.

→ Example to invoke one method by using two threads

```
using System;
using System.Threading;
namespace ThreadEx2
{
    class Myclass
    {
        internal void Method()
        {
            for(int i=0; i<=10; i++)
            {
                Console.WL("Method = "+ i);
        }   }
```

```
class Program
{
    void main()
    {
        Myclass mc = new Myclass();

        ThreadStart tstart = new Threadstart(Obj.Method);

        Thread thr1 = new Thread(tstart);

        Thread thr2 = new Thread (tstart);

        thr1.Start();
        thr2.Start();

        Console.ReadLine();
    }
```

Name :-

→ It is a predefined member property of Thread class.

→ Using this property we can assign the user defined name to given Thread as well as we can retrive the user defined name of the given thread.

CurrentThread

→ It is a predefined static member property of Thread class ~~which~~.

→ CurrentThread property is representing the ~~curr~~ instance of the current executing Thread.

→ ~~Current executing Thread will return th~~

→ CurrentThread property will return the current executing Thread Object.

→ CurrentThread is a readonly property.

→ Example to assign a userdefined names to Threads as well as retriving the current executing Thread name

```
Using System.Threading;
namespace ThreadEx3
{
  class Myclass
  {
    internal void Method()
    {
      for(int i=1; i<=10; i++)
      {
        Console.WL(CurrentThread.Name+"="+i);
      }
    }
  }
class Program
  {
    Void Main()
    {
      Myclass obj=new Myclass();
      ThreadStart tStart = new ThreadStart(obj.Method);
      Thread thr1 = new Thread (tStart); thr1.Name="NTR";
      thr1.Start();
      Thread thr2 =new Thread (tStart);
      thr2.Name = "ANR";
      thr2.Start();
      Console.RL();
    }
```

Thread.Sleep() :-

→ Sleep is a static member method of Thread class.

→ We have two Sleep overloaded methods.

1. First Method

```
Thread.Sleep(int milliseconds);
```

→ This method will send a Thread for sleep according to given milliseconds.

2. Second Method

→ This method will take input as a TimeSpan object.

```
Thread.Sleep(TimeSpan obj);
```

→ We can consume in 5 ways.

1. Thread.Sleep(new Timespan());
2. Thread.Sleep(new TimeSpan(10));
3. Thread.Sleep(new TimeSpan(10, 10, 20));
4. Thread.Sleep(new TimeSpan(1, 2, 20, 10));
5. Thread.Sleep(new TimeSpan(10, 3, 10, 20, 30));

→ This Second sleep method will send a Thread for long sleep.

→ Example for first sleep method.

```
namespace firstsleepmethodEx.
{
  class Myclass
  {
     internal void Method1()
     {
        for(int i=1; i<=10; i++)
        {
```

```
        Console.WL (Thread.CurrentThread.Name + " = " + i);
        Thread.Sleep(20);
    }
  }
}
class Program
{
    void Main()
    {
        Myclass obj = new Myclass();
        ThreadStart tStart = new ThreadStart(obj.Method1 ~~Myclass~~);
        Thread thr1 = new Thread (tStart);
        Thread thr2 = new Thread (tstart);
        thr1.Name = "NTR";
        thr2.Name = "ANR";
        thr1.Start();
        thr2.Start();
        console.RL();
    }
```

→ Example for Second ~~ms~~ Sleep method

```
namespace ThreadEx5
{
  class Myclass
  {
    Internal void Method1()
    {
      for(int i=1 ;i<=10; i++)
      {
        Conode.WL (Thread.CurrentThread.Name + " = " + i);
        if(i=5)
        {
```

```
        Console.WriteLine(Thread.CurrentThread.Name + " is going for sleep");
        Thread.Sleep(new TimeSpan(0,0,20));
    }
  }
}
class Program
{
    Void Main()
    {
    // same as above prgm
    }
}
}
```

Why sleep() is static and why start() is instance ?

Suspend()

→ This method will suspend the given Thread temporarily.

Resume()

→ This method will call back the suspended Thread.

Abort()

→ This method will terminate the Thread Execution permanently.

Example for Suspend()

```
using System.Threading
namespace ThreadEx6
{
  class Myclass
  {
      internal void Method1()
      {
```

```
        for(int i=1 ; i<=10 ; i++)
          {
            Console.WL(Thread.CurrentThread.Name+" = "+i);
            if(i==5)
              {
                Console.WL(Thread.CurrentThread.Name+" is goint to Suspend");
                Thread.CurrentThread.Suspend();
              }
          }
      }
    }
    class Program
      {
        void Main()
          {
            // Same as above pogm
          }
      }
  }
```

→ Example to abort()

```
Using System.Threading;
namespace Thread Ex7.
{
    class Myclass
      {
        Internal void Method1()
          {
            for(int i=1 ; i<=10;i++)
              {
                Console.WL(Thread.CurrentThread.Name+" = "+i);
                if(i==5)
                  {
```

```
        Console.WL("Thread.CurrentThread.Name +" is going to Abort");
        Thread.CurrentThread.Abort();
    }
}
}
class program
{
    void Main()
    {
        // Same as above Prgm
    }
}
```

→ Example to call back Suspended Thread.

O/P

```
NTR = 1
  :
NTR = 5
NTR is going to Suspend
ANR = 1
  :
ANR = 5
ANR is going to Suspend
NTR = 6
  :
NTR = 10
ANR = 6
  :
ANR = 10
```

```
using System.Threading;
namespace ThreadEx8
{
    class Myclass
    {
        internal void Method1()
        {
            for(int i=0; i<=10; i++)
            {
                Console.WL("Thread.CurrentThread.Name+" = "+i);

                if(i==5)
                {
                    Console.WL("Thread.CurrentThread.Name+" is going to Suspend");

                    Thread.CurrentThread.Suspend();
                }
            }
        }
    }

    class Program
    {
        void Main()
        {
            Myclass obj = new Myclass();
            ThreadStart tstart = new ThreadStart(obj.Method1);
            Thread thr1 = new Thread(tstart);
            Thread thr2 = new Thread(tstart);
            thr1.Name = "NTR";
            thr2.Name = "ANR";
            thr1.Start();
            thr2.Start();

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

→ C#.Net will support following types of classes

1. Normal class
2. Abstract class
3. Static class
4. Sealed class
5. Partial class
6. genric class - - -

Static class

→ While defining a class if we have used Static Keyword which can be called as Static class.

→ Static class can contain only Static members.

→ Static class cannot be Inherited.

→ We cannot create an object for Static class because not required.

When we will go for static class?

→ whenever a class is having all static members then we can declare particular class as a Static class.

→ Example for Static class.

```
namespace Staticclassex.
{
    Static class Myclass
    {
        internal Static int a;
            Static Myclass()
            {
                a=10;
            }
```

```
        internal static void Show()
        {
            Console.WLC("a value is:"+a);
        }
    }
    class Program
    {
        void Main()
        {
            Myclass.Show();
            Console.RL();
        }
    }
}
```

Sealed class

→ While defining a class if we have used sealed keyword which is called as sealed class.

→ Sealed class cannot be inherited.

→ We can create an object for sealed class.

Example for sealed class : Thread

When can i go for sealed class?

→ Whenever a class required all the static and non-static members and cannot be inherited, then we will go for sealed class.

→ Example to implement sealed class.

```
Sealed class Myclass
{
}
class C2 : Myclass
{
}
```

→ Above code will generate an error bcoz sealed class cannot inherited.

Partial class

→ Partial class is introduced in 2.0 version.

→ Partial class will split a single class into multiple class files, but class name will be same but class files name should be differ.

→ for Example.

class1.cs

```
Partial class Myclass
{
    Method1()
    {
    }
}
```

class2.cs

```
Partial class Myclass
{
    Method2()
    {
    }
}
```

class3.cs

```
Partial class Myclass
{
    Method3()
    {
    }
}
```

When we will go for partial class?

→ Whenever multiple resources want to work on single class we will go for Partial class.

→ Example for Partial class

Step 1 : Open a Console application rename it as partial class Example.

Step 2 : Add 3 class files they are class1.cs, class2.cs, Class3.cs

Step 3 : class1.cs code.

```
namespace PartialclassEx
{
    partial class MultiMedia
    {
        internal void ~~Method~~ VedioRecording()
        {
            Console.WL("Vedio Recording is calling");
        }
    }
```

Step4: class2.cs code.

```
Partial class MultiMedia
{
    internal void AudioRecording()
    {
        C.WL("Audio Recording is calling");
    }
}
```

Step 5: class3.cs code

```
Partial class MultiMedia
{
    internal void ImageProcessing()
    {
        C.WL("Image Processing is calling");
    }
}
```

Step 6: program.cs code.

```
class Program
{
    void Main()
    {
        Multimedia obj = new Multimedia();

        obj.VedioRecording();
        obj.AudioRecording();
        obj.ImageProcessing();
        Console.RL();
    }
}
```

→ In ASP.Net every webpage class is a partial class

Ex: Webform1, Webform2 ... Webformn are partial classes.

→ In windowsForms every windowsForm class is a Partial class.

Ex: Form1, Form2 .... Formn are partial classes.

for each loop :-

→ It is one of the looping concept to fetch the elements from collection.

→ Using foreach we can fetch the elements from collection without using initialization, condition, increment/decrement.

Syntax :-

```
foreach(<datatype> <variable name> in <collection>)
{

}
```

→ Example to fetch the elements from array by using foreach.

```
class Program
{
    void Main()
    {
        int[] a = new int[] {10,20,30,40,50};

        foreach(int i int a)
        {
            Console.WL(i);
        }

        Console.RL();
    }
```

Advantage of foreach

→ Faster execution.

- Implement appropriate realtime examples for for loop, while loop, do while loop and foreach loop and identify the differences b/w each loop.

Array class:- Array is predefined class defined by microsoft.

Members of predefined class are of mainly two types 1. Instance 2. Static.

Instance Members

1. **length** : Number of elements in an array.
2. **Rank** : This property will return no. of dimensions within given array.
3. **CopyTo()** :- Using this method we can copy the elements from one array to another array. **Syntax:** <objectname>.CopyTo (otherarrayname, Index)
   eg: a.CopyTo(b,1);

Static Members

1. **Array.Reverse()** :- Reversing the elements. **Syntax**: Array.Reverse (Arrayname);
2. **Array.Sort()**;- Sort the elements. eg: Array.Sort(a);
3. **Array.clear()** :- Clear the elements eg: Array.clear(a);
4. **Array.BinarySearch()**:- This method will Search the given elements within the array.
   → If the element is identified it will return index value of the element
   → If the element is not identified it will return negative value.
   eg: Array.BinarySearch (a, "Marry");

Example for array class Members:

```
Using System;
namespace ArrayExample
{
  class Program
  {
     void Main()
      {
     string[] names = new string[] {"John", "David", "Marry","phillips",
                                    "francis"};
     Console.WriteLine("Array elements are:");
     foreach(String i in names)
      {
       Console.WL(i);
      }
     Array.Reverse(names);
     Console.WL("Reversed array elements");
     foreach(string i in names)
      {
       Console.WL(i);
      }
     Array.Sort();
     Console.WL("sorted array elements");
     int index = Array.BinarySearch(names, "marry");
     if(index >=0)
      {
       Console.WL("Searching elements name is:"+name[index]);
      }
     else
      {
       Console.WL("elements are not found");
      }
    // Copy to method.
     String [] Ename = new String[6];
      names.CopyTo(enames,1);
```

```
Console.WL("enames elements are:");
foreach (string i in ename)
{
  Console.WL(i);
}
Console.WL("no.of Dimensions are:"+ enames.Rank);
```

o/p

Array elements are:

John
David
Marry
phillps
francis

0	1	2	3	4
John	David	Marry	phillps	francis

Reversed array are:

francis
Phillips
Marry
David
John

0	1	2	3	4
francis	phillps	Marry	David	John

Sorted array are:

David
Francis
John
Marry
phillps

0	1	2	3	4
David	Francis	John	Marry	phillps

~~no. of Dimensions are : 1~~

Searching for an element in array : Marry

enames elements are:

David
Francis
John
Marry
phillips

0	1	2	3	4	5
	David	francis	John	Marry	phillps

no. of Dimensions are : 1.

Assemblies

Assemblies (class libraries): class libraries are the collection of classes.

→ Assemblies are of two types.

1. Predefined class assemblies (Base class libraries)
2. User defined class assemblies.

Predefined class assemblies :

→ The assemblies which are providing by microsoft and used by .Net Programmer is called as predefined. class assemblies.

eg: System.dll, System.Threading.dll, System.Web.UI. etc...

Userdefined class assemblies :

→ These class libraries are developed by .Net programmer as part of application development.

→ In .Net class libraries are called as assemblies. An assembly can be in the form of .Exe file (or) .dll file.

Diff b/w exe file and dll file.

exe file	dll file.
→ exe stands for executable file	→ dll stands for dynamic link library
→ extension of .exe file is .exe.	→ extension of dll file is .dll.
→ exe is self executable, exe is itself an application	→ dll is not self executable. It will depend on some other application (exe) for execution.
→ collection of classes which contains main method will produce exe file.	→ collection of classes which doesn't have main() method will produces a dll file.

Note :- In .Net if we want to develop a dll we have to go for class library project.

How to develop dll's :-

→ An assembly (dll) is a unit of code which provides versioning and deployment.

→ An assembly content can be divided into three parts.

Part 1 : Manifest: Manifest is representing complete info about the assembly in the form of metadata.

→ Metadata is representing in three ways:

1. Assembly Name
2. Assembly version
3. Public key (strong name)

Part 2 : MSIL code

Part 3 : Refrences: Refrences are representing the links to other files which are consuming by assemblies.

→ According to the scope of the assembly, assemblies are of two types.

1. Private assembly.
2. Shared assembly.

1. Private assembly An assembly which providing services to a single client application at a time is called as private assemblies.

Inside CLR

When we run dotnet application, .Net execution engine called CLR will load into RAM memory. As part of program execution CLR will maintain various blocks.

Among various blocks 4 are important. They are

1. Stack
2. Heap
3. Method Area
4. Execution Engine.

Stack: CLR will use stack to allocate memory for local variables. for every method within the stack one frame will be maintaining by CLR.

Heap: CLR will use Heap to allocate the memory for objects.

Method Area:- It will be divided into 3 parts like below.

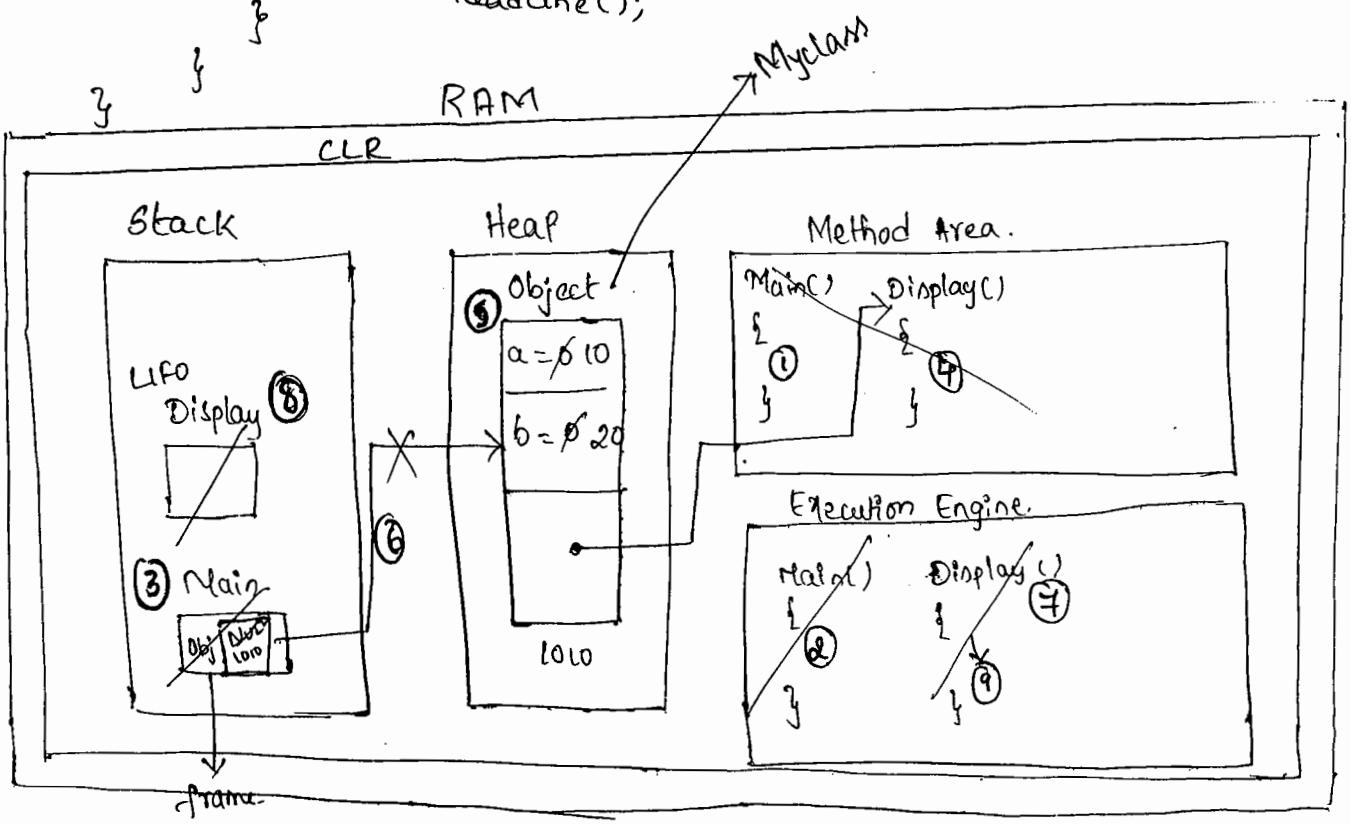
Static Variables	Methods	constructors

Note: In method area only the memory will be allocated for static variables. No memory will be allocated for methods & constructors i.e., when the class is loading concern class static variables will be allocating within method area & concern class methods & constructors will be loaded into method area.

Execution Engine: Execution engine will maintain the current executing block.

→ Example for memory allocations within CLR

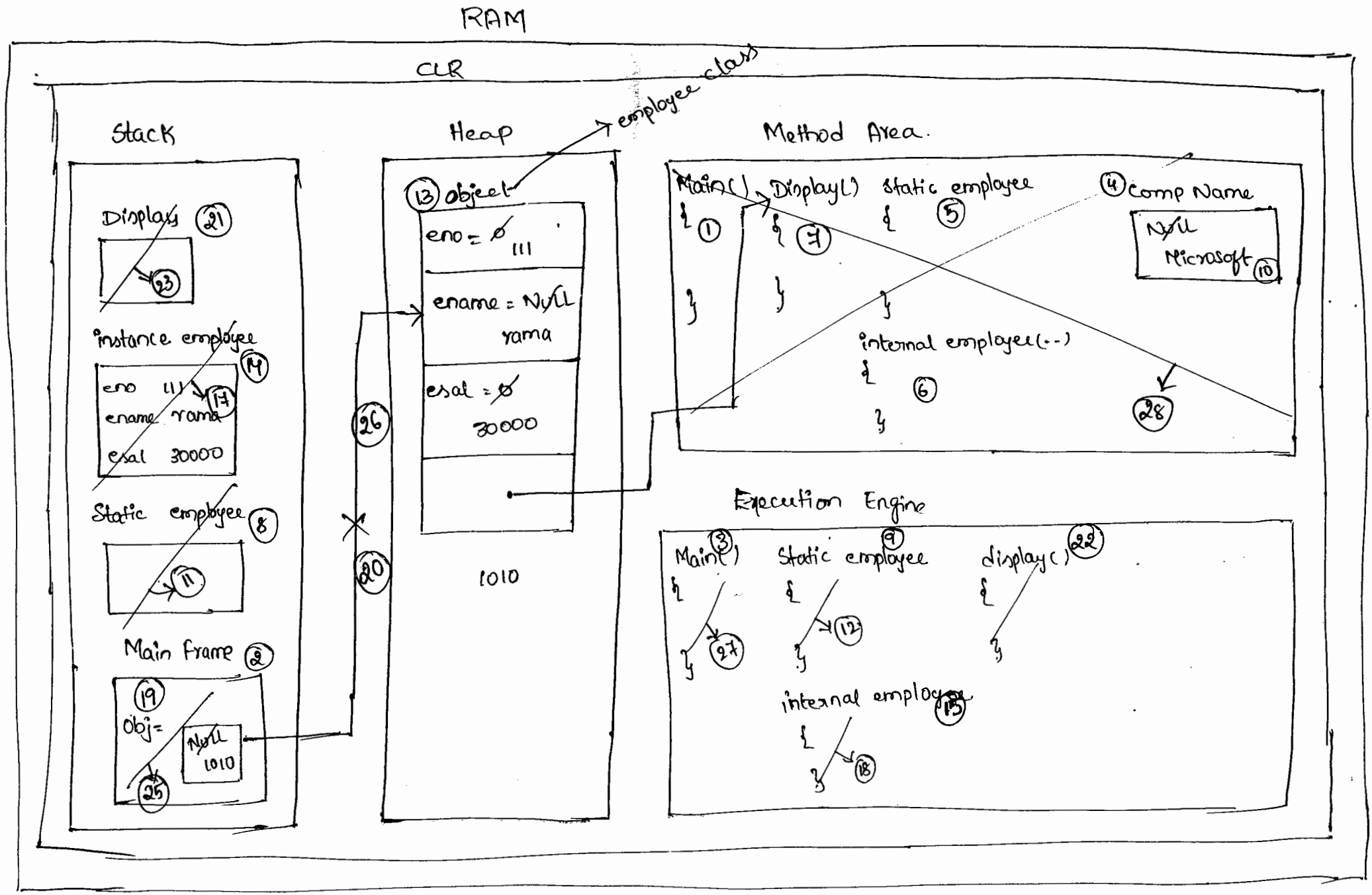
```
namespace MemoryAllocationEx
{
  class Myclass
  {
     int a = 10;
     int b = 20;
     internal void display()
     {
       Console.Writeline(a);
       Console.WriteLine(b);
     }
  }
  class Program
  {
     void Main()
     {
       Myclass obj = new Myclass();
       obj.Display();
       Console.ReadLine();
     }
  }
}
```



```
namespace. CLRInside.
{
  class employee
  {
    int eno;
    String ename;
    double esal;
    Static String compName;
    Static employee()
    {
      compName = "Microsoft";
    }
    internal employee (int eno, string ename, double esal.)
    {
      this.eno = eno;
      this.ename = ename;
      this.esal = esal;
    }
    internal void Display()
    {
      Console.Writeline (" Emp No is" + eno);
      Console.Writeline("Emp Name is" + ename);
      Console.Writeline ("Emp Salary is" + esal);
      Console.Writeline (" Company name is" + compName);
    }
  }
  class program
  {
    void Main()
    {
      employee obj = new employee(111, "rama", 30000);
      obj.Display();
      Console.ReadLine();
    }
  }
}
```

Instance variables

Local variables


RAM
CLR
Stack
Display (21)
(23)
instance employee (14)
eno 111 (17)
ename rama
esal 30000
Static employee (8)
(11)
Main Frame (2)
(19)
Obj=
Null
1010
(25)
(26)
(20)
Heap
employee class
(13) object
eno = ∅
111
ename = Null
rama
esal = ∅
30000
1010
Method Area.
Main() (1)
Display() (7)
static employee (5)
internal employee(..) (6)
(4) comp Name
Null
Microsoft (10)
(28)
Execution Engine
Main() (3)
(27)
Static employee (9)
(12)
display() (22)
internal employee (15)
(18)

HW Implement Student class example for CLR inside Student class members are student id ; name, location, college name, totalmarks, avg marks. Behaviours are calculate totalmarks (m1, m2, m3) & avgmarks in same method & dont display. Display student info ()

Class libraries :-

class libraries are two types

1. Base class library. Ex: System
2. User defined class library Ex: MyconsoleApplication123

→ In .Net ~~Assen~~ class library is nothing but assembly.
→ An assembly can be in the form of either dll file or exe file.

Differences b/w exe and dll

exe	dll
→ stands for executable.	→ stands for dynamic link library.
→ Extension of the exe file will be .exe.	→ extension of the dll file will be .dll
→ Collection of classes which contain main method will produce exe file.	→ Collection of classes which doesnt have main method will produce dll file.
→ exe is a self executable which itself an a application.	→ dll is not a self executable which will depend on other application to execute.

Assembly

→ An assembly is a collection of classes.

→ An assembly is a unit of code which provides versioning and deployment.

→ Content of the assembly can be divided into 3 parts.

Part 1:

Manifest: → Manifest is representing the complete information about assembly in the form of metadata.

→ Metadata is representing like below.

1. Assembly name.
2. Assembly Version
3. Public Key (strong Name)

Part 2:

MSIL code:

Part 3:

References: Links to other assemblies which are consuming by this assembly.

How to develop a dll file by using .Net?

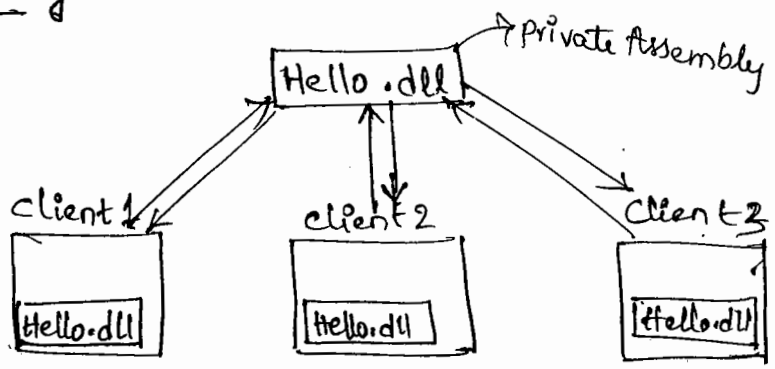
Ans With the help of class library project.

→ **Types of Assemblies**

→ According to scope of the assembly, assemblies are classified into 2 types.

1. Private Assembly
2. Shared Assembly.

1. Private Assembly:



→ An assembly which is providing services to single client application at a ~~ty~~ time is called as private assembly.

→ Private assembly will create a local copy with in client application folder, that local copy will provide the services to concern client

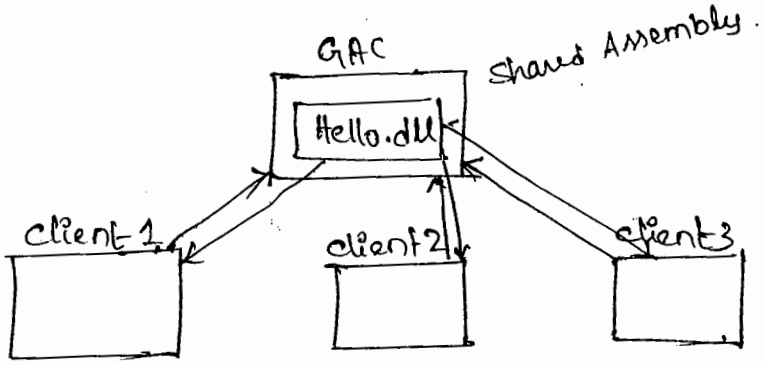
2. Shared Assemblies:-

→ An assembly which is providing services to multiple client applications at a time is called as shared assembly.

→ Shared assembly will not create any local copy within client folder.

→ Shared assembly will be providing services to multiple client applications from a common shared secured folder called GAC. (Global Assembly cache).

Diagram for Shared Assembly.



Implementation of private assembly divided into 4 steps:

Step 1: Creating private assembly (i.e., class library)

Step 2: Creating client appn (Console Appn)

Step 3: Adding private assembly reference to client appn.

Step 4: Consuming private assembly within client appn.

Example to implement private assembly in step by step process:

Step 1: Creating private assembly.

→ open class library project rename it as my assembly location as 'D' drive.

→ Rename class1.cs as calculate.cs write the code in calculate.cs.

```
namespace MyAssembly
{
  Public class calculate
  {
    public int add(int a, int b)
    {
      return a+b;
    }
    Public int sub(int a, int b)
    {
      return a-b;
    }
  }
}
```

→ Build Solution

→ with this process private assembly will create with in following Path :

D:\MyAssembly\MyAssembly\bin\Debug.

→ With in debug folder we can identify a file called Myassembly.dll is nothing but private assembly.

Step2 : Creating client appn.

→ open a console appn rename it has Myclient location it has D.

Step3 : Adding Myassembly Reference to Myclient .

→ open Myclient → select References → add → browse → D drive → My assembly → Myassembly → bin → debug → Myassembly.dll.

Step4 : Consuming private Assembly form client appn.

→ Program.cs code.

```
using MyAssembly
namespace Myclient
{
  class Program
  {
    Static void Main()
    {
      calculate obj = new calculate();
      int res = obj.add(10,5);
      Console.WriteL("addition result is : "+res);
      res = obj.Sub(10,5);
```

```
Console.WL(" Sub result is : "+res);
    C.RL();
    }
  }
}
```

→ Private assembly will create a local copy with in the every client folder. That we can check with in the following path.

D:\Myclient\Myclient\bin\Debug.

→ With in debug folder we can identify a file called Myassembly.dll is nothing but local copy of private assembly.

<u>Drawbacks</u> :

→ It will create multiple local copies.

<u>Implementing Shared Assemblies</u> :-

→ By default an assembly is a private assembly if we want to make our assembly as a shared assembly we have to install into "GAC" folder, then Particular assembly will be converting as shared Assembly.

→ Converting from private assembly to shared assembly will be divided in to 3 steps.

<u>Step 1</u> : Creating a strong name by using strong name utility.

Q) <u>What is strong name</u> ?

→ Strong name can be called as Public key.

→ Strong name will give the unique identity to assembly among collection of assemblies with in the GAC folder.

Q) What is strong name utility ?

→ It is one of the .Net framework utility using this utility we can create a strong name to given assembly like below.

Syntax to create strong name;

Note: we have to execute below command through visual Studio.Net command prompt.

```
D:\Sn -k MyKeys.snk
```

Sn → Strong name utility.
-k → Key
MyKeys → file name.
.snk → strong name key

→ The above command will create a strongname, that created strong name will be storing into MyKeys.snk.

Step 2 : Signing the assembly

→ Informing abt strong name to an assembly is called as signing the assembly.

Step 3 : Installing an assembly into GAC folder by using GAC utility.

Q) what is GAC utility?

→ It is one of the .Net framework utility.

→ using this utility we can install an assembly in to GAC folder.

→ GAC utility is representing with a file called ~~GACutility~~ gacutil.exe.

Syntax to install an assembly to GAC folder:

```
D:\ gacutil : MyAssembly.dll ↵
                installing.
```

Implementation of shared Assembly will be divided into following steps

Step 1 : Create private assembly

Step 2 : creating strong name.

Step 3 : signing the assembly

Step 4 : Installing an assembly into GAC.

Step 5 : creating client appn

Step 6 : Adding shared assembly Reference to client appn.

Step 7 : consuming shared assembly from client appn.

Example to implement shared assembly:

Step 1: creating private assembly.

→ open a class library prjt Rename it as "Mycalculate" location as "E" drive.

→ write the below code in class1.cs

```
namespace Mycalculate
{
  public class calculate
  {
    Public int Mul(int a, int b)
    {
      return a * b;
    }
    Public int Div(int a, int b)
    {
      while (b==0)
      {
        C.WL("please enter other than 0");
        b=int.Parse(C.RL());
      }
      return a/b;
    }
  }
}
```

→ Build Solution.

Step 2: Creating strong name.

E:\Mycalculate\Mycalculate > Sn - K Keys.Snk

Note: Strongname Keyfile we have to create beside bin folder of one assembly.

Step 3: Signing the Assembly.

→ write the below code in AssemblyInfo.cs file.

→ Goto the end ~~of the~~ and add below statement.

```
[assembly: AssemblyKeyFile("Keys.snk")]
```

Rebuild the Assembly (or) Solution.

Step 4 : Installing an assembly into GAC folder.

E:\cd Mycalculate\Mycalculate\bin\Debug ↵

E:\Mycalculate\Mycalculate\bin\Debug\gacutil\Mycalculate.dll ↵

Step 5 : creating client application.

→ open a console appn rename it as Myclient location as E drive

Step 6 : Adding stored assembly reference to client application

Same as above Ex.

Step 7 : consuming shared assembly from client appn.

Program.cs

```
using Mycalculate
namespace Myclient
{
  class Program
  {
  Void Main()
{
calculate obj = new calculate();
C.WL("Enter 1st number");
int x = int.parse(C.RL());
int y = int.Parse(C.RL());
int res = obj.Mul(x,y);
C.WL("Multiplication result is:"+res);
res = obj.Div(x,y);
C.WL("Division result is:"+res);
C.RL();
}
}
}
```

→ shared Assembly will not create local copy with in the client folder that we can check with below path.

E:\Myclient\Myclient\bin\Debug.

→ With debug folder there is no dll file becoz here Myclient Mycalculate.dll is shared assembly which will provide services to multiple clients from GAC folder.

Protected Internal :-

→ If we declare a class (or) class members access modifiers as protected Internal which can be accessed by all the classes of current prjt classes. and derived class of another prjt.

Ex : ① open a console application rename it as protected Internal Example.

② Add a class library project to solution Explorer. rename as MyProject.

③ class.cs code.

```
namespace MyProject
{
  Public class Myclass
  {
    Internal void Myfun()
    {
      C.WL("Internal method is calling");
    }
    Protected internal void Print()
    {
      C.WL("Protected internal method is calling");
    }
```

→ Build MyProject ↵

Step 4 : Adding MyProject reference to ~~projected Internate~~ Protected Internal Ex.

Step 5 : Consume Myproject from protected Internal Ex.

Program.cs code

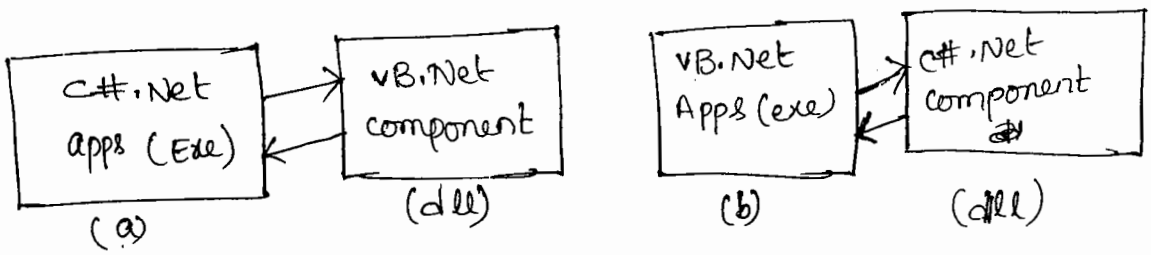
```
using MyProject;
Namespace ProtectedInternalEx
{class dc : Myclass
  { Internal void display()
    {
```

```
        base.Print(); //possible becoz of protected Internal
        base.MyFun(); // Not possible becoz of Internal
    }
}
class Program
{
    Void Main()
    {
        dc obj = new dc();
        obj.display();
        C.RL();
    } } }
```

Language Independency :-

One .Net Language code we can use in another .Net language application due to that reasons .Net can be called as language Independent platform.

→ That means while developing c#.Net application we can use VB.Net component as well as while developing VB.Net application we can consume c#.Net component.



→ Example to consume VB.Net component from C#.Net appn in step by step process.

Step 1 : Creating a VB.Net component.

→ open a class library project select language VB.Net and rename it has VBcomponent location as "E" drive.

→ Below code in class1.vb.

```
Public class vbclass
  Public sub Print()
      Console.writeline("Welcome to vB.net");
      Console.writeline("Welcome to vB.net");
  End Sub
    Public Function Add(Byval a As Integer, Byval b As Integer)
                                                        As Integer
    Return a+b
  End Function.
  End class.
```

→ Build Solution with this process vBcomponent.dll will generate.

Step 2 :- creating c#.Net client.

→ open a c#.Net console appn rename it as c# client location as "E" drive.

Step3 : Adding VB component reference to c# client.

Step 4 : Consuming VB component from c#client.

Program.cs code.

```
using vbcomponent;
namespace charpclient
{
  class program
  {
    Static void Main
    {
      vbclass objvb = new vbclass();
      objvb.print();
      Console.writeline("Enter 1st number;");
      int x = int.parse(Console.Readline());
      Console.Writeline("Enter 2nd number;");
      int y = int.parse(Console.Readline());
```

```
int res = objvb + Add (x,y);
Console.WriteLine ("addition result is: " + res);
Console.ReadLine ();
    }
  }
}
```

O/P

welcome to VB.net.
welcome to VB.net.

→ Example to consume a C#.net component from VB.net client.
→ Implement the all the above console program by using VB.net.
→ Implement all ASP.Net example by using VB.Net

Generics: (Introduced in 2.0)

→ Generics is similar like C++ templates.
→ Generics was introduced with .Net frame work .Net 2.0.
→ Generics will allow the programmer to decide parameter type at the time of consumption. i.e., in declaration we will declare it as a generic type & which can be consumed has int (or) float, char (or) double (or) string a/c to over requirement.
→ In C#.Net we can have Generic variable, Generic property, Generic constructor, generic method, Generic class.
→ To implement generics we will use place holder operator (< >) will type parameter like below <T>

Generic Function:- When we implement generics on method (or) functions then we call it as a Generic function.

Syntax:-

```
<am> <rt> <function name><T> (T <arg1>, - <arg2>)
{
  =
}
```

Note: We can declare static and instance members are generics members.

Ex for Generic function :-

```
namespace GenericExample.
{
  class Myclass //instance Method
  {
    internal void Display<T> (T u)
    {
      Console.Writeline(u);
    }
    internal static void Show<T> (Td,Tb)
    {
      C.WL(a+" , "+b);
    }
    internal void print <T,K> (Ta , Kb)
    {
      C.WL(a+" , " + b);
    }
  }
class Program
  {
    Myclass obj = new Myclass();
    obj.Display<int> (10);
    obj.Display<String>("satya");
    Myclass.Show<int>(10,20);
    Myclass.Show<String> ("satya","dotnet");
    obj.Print<int,string> (10,"satya");
    C.RL();
  }
}
}
```

Note : Arithmetic operation can not be perfomed generic function.

→ we can. declare function return type as T.

→ we can pass & different values to generic function.

```
// Non void generic function
internal int add<T> (Ta)
{
    C.WL(a);
    return 15;
}
// Generic return type
internal T Myfun <T> (Ta)
{ return a;
}
class program
{
static void Main()
{
    int res = adj add<int> (100);
    C.WL(res);
int res = obj.Myfun<int>(125);
    C.WL(res2);
}
}
}
```

→ we can pass different value to generic function.

→ we can declare function return type as generic type.

→ we can have non void generic function.

Generic class :

When we are implemented generics in class level which is called as generic class.

Syntax :

```
class <class Name> <T>
{
    // generic members
    // Non generic members.
}
```

→ In case of generic class at the time of creating object we will decide the type that object is called generic object.

→ we cannot create a normal object for generic class.

→ A/c for every time to requirement we have to create generic object.

→ using every generic object we can invoke non-generic methods.

Syntax to create generic object.

```
<classname> <Type> <object name> = new <class name><Type>();
```

Example for generic class with generic & non generic members ;

```
namespace Generic class Ex1.
{
 class Mydass <T>
  {
   Ta;
   internal Mydass (Ta)
   {
     this.a=a;
   }
   Tx;
   internal Tx
   {
     set
     {
       x= value;
     }
   }
```

```
Internal void display()
 {
   C.WL ("a value is:"+a);
   C.WL ("x value is:"+x);
 }
}
class Program
 {
   void Main()
   {
     Myclass<int> mcint = new Myclass<int> (10);
     mcint =100;
     mcint.display();
     Myclass<string> mcstring = new Myclass<string> ("satya");
     mcstring. = "dotnet";
     mcstring.Display();
     C.RLC();
   }
 }
}
```

o/p

```
10
100
satya
dotnet
```

Collections :

→ collections is representing collection of elements.

→ Every collection will be provided by Microsoft as predefined class.

→ Before .net 2.0 which have normal collections class.

→ Normall class all predefined within a base class called System.Collections.

Normall collections class: (up to 1.1)

1. Stack
2. Queue
3. ArrayList
4. Hashtable.

Generic collections:

→ Base class library is System.Collections.generics;

→ Stack<>
→ Queue<>
→ List<>
→ Dictionary<>

→ In .Net we have 2 types of collections
1. Normal collection (before 2.0)
2. Generic collection (After)

Example to implement stack generic collection:

```
class Program
{
  Void Main()
  {
   Stack<int> mystack = new Stack<int>();
   mystack.Push(10);
   mystack.Push(20);
   mystack.Push(30);
   mystack.Push(40);
   mystack.Push(50);
   C.WL(" Mystack elements where: ");
  foreach( int i in mystack)
     {
       C.WL(i);
     }
C.WL(" Pop element is : " + mystack.Pop());
C.WL(" Peak element is:" + mystack.peek());
C.WL(" mystack elements are: ");
```

return the top most element and remove it.

returns the topmost element without removing.

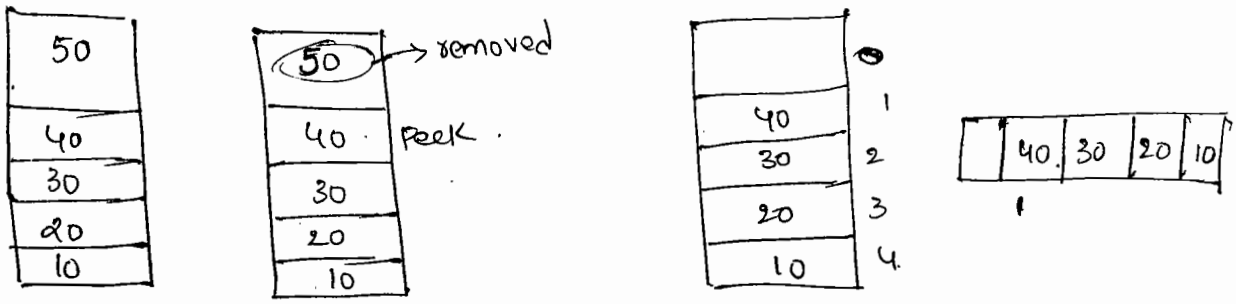
```
foreach(int i in mystack)
{
 C.WL(i);
}
int [] arr = new int[5];
mystack.copyTo (arr, 1);
```

↳ copies the stack to an existing one dimentional array.

```
foreach (int i in arr)
 {
   C.WL(i);
 }
mystack.clear();
```

→ removes all the objects from stack.

```
C.WL ("No. of mystack elements are :"+ mystack.count);
C.RL();
}
}
}
```



O/P : Mystack element is: 50
40
30
20
10

Pop element is: 50

Peak Stack element is: 40

Mystack elements are : 40
30
20
10

Myarray elements are : 40
30
20
10

Myarray elements are; 0
40
30
20
10

Generic List:

→ Generic list collection will be representing collection of elements, every element will be representing with one index value first element Index value zero, second Index value as 1 .....

Example

```
class Program
{
  Void Main()
  {
    List<int> mylist = new List<int>();
    mylist.Add(10);
    mylist.Add(20);
    mylist.Add(30);
    mylist.Add(40);
    mylist.Add(50);
    C.WL("Mylist elements are:");
    foreach(int i in mylist)
    {
      C.WL(i);
    }
  Mylist.Remove(10);   → Removes the object which is first item occurence
  Mylist.RemoveAt(1);  → remove the element with the specific index value.
  int[] arr = new int[5];
  MyList.CopyTo(arr,2);
  C.WL("My array elements are:");
  foreach(int i in arr)
  {
    C.WL(i);
  }
  C.WL("Mylist elements are:");
  foreach(int i in mylist)
  {
    C.WL(i);
  }
```

```
        Mylist.Clear();
        C.WL("Number of mylist elements are:"+Mylist.Count);
        C.RLC();
    }
  }
}
```

O/P

```
Mylist elements are.
10
20
30
40
50
Myarray elements are.
0
0
20
40
50
Mylist elements are
20
40
50
mylist of elements : 0
```

Dictionary Generic collection.

Dictionary is a collection of pairs each pair will have two values

1. Key value
2. Item value.

Example for generic Dictionary;

→ · EmpDic

Key	Item
111	Rama
222	John
333	Mary
444	David
555	philips

```
→ namespace genericDicEx
{
  class Program
  {
    void Main()
    {
      Dictionary<int, string> EmpDict = new Dictionary<int, string>();
      Empdict.Add(111, "rama");
      Empdict.Add(222, "John");
      Empdict.Add(333, "mary");
      Empdict.Add(444, "David");
      Empdict.Add(555, "philips");
    C.WL("empdict elements are : ");
    foreach(int i in Empdict.Keys)  → property
    {
      C.WL(i + " " + Empdict[i]);
    }
    bool res = Empdict.Remove(333);
    C.WL("Empdict elements after remove");
    foreach(int i in Empdict.Keys)
    {
      C.WL(i + " " + Emddict[i]);
    }
    Empdict.Clear();
    C.WL("No. of dictionary pairs are : " + Empdict.Count);
    C.RL();
  }
}
```

Indexer:

→ Indexer will treating an object as an array.
→ Indexer will be acting as a mediator b/w array and object.
→ Indexer definition will be similar like property.

Syntax:

```
<am> <returntype> this [<type> <arg1>] <int> [<index>]
                                  ↑ will represent index value of array
  ↓ current class object
{
  get
  {
    = //returning values from array.
  }
  Set
  {
    // assigning the values to array.
  }
}
```

Example for Indexer:

```
namespace IndexerEx
{
  // step1
  int [] EmpAges = new int[5];

  // step2
  internal int this[int i]
  {
    get
    {
      return EmpAges[i];
    }
    get
    {
      while (value<1 || value>120)
      {
        C.WL("Plz enter the age b/w 1 to 120");
```

```
        value = int.Parse(C.RL());
        }
        empAges[i] = value;
    }
}
class Program
{
    void Main()
    {
        Myclass obj = new myclass();
        C.WL("enter emp ages:");
        for (int i=0; i< obj.employes.length; i++)
        {
            obj[i] = int.Parse(C.RL());
        }
        C.WL("employee Ages are 6");
        for (int i=0; i < obj.empages.length; i++)
        {
            C.WL(obj[i]);
        }
        C.RL();
    }
}
}
```

When we will go for indexer?

→ wherever we want to assign the values to an array & retrieving values from an array from outside the class by implementing validations we will go for indexer.

What do you mean by smart array in C#.Net?

→ Indexers are called as smart array. which makes faster assigning to array faster retrivals from arrays

Can we have static indexers in C#.Net?

→ No, because indexer concept is depending on object.

Q) Can we implement indexer when a class is having two arrays?

→ No.

Q) Can we implement indexer on multidimensional array?

→ Yes

Q) Can we implement indexer on jagged array

→ Yes.

Enum (Enumerator) :-

→ Enum is value type.

→ To define enum we have to use a keyword called "enum".

→ Using enum we can create our own user defines types.

→ Enum is a collection of string constants which represents integer constants.

Example for enum

```
namespace EnumEx
{
  enum weekdays
  {
    Monday=1;
    Tuesday=2;
    wednesday=3;
    Thursday=4;
    Friday=5;
    Saturday=6;
    Sunday=7;
  }
```

```
class Myclass
{ internal static void Display (weekdays d)
                                  enum
  {
    Switch (d)
    {
      case weekdays.Monday:
        C.WL ("Today is first day of the week");
        break;

      case weekdays.Tuesday:
        C.WL ("Today is 2nd day of week");
        break;

      case weekdays.Wednesday:
        C.WL ("Today is 3rd day of week");
        break;
      case weekdays.Thursday:
        C.WL ("Today is 4th day of week");
        break;
      case weekdays.Friday:
        C.WL ("Today is 5th day of week");
        break;
      case weekdays.Saturday:
        C.WL ("Today is 6th day of week");
        break;
      Case weekdays.Sunday:
        C.WL ("Today is 7th day of week");
        break;
    }
  }
  class program
    {
```

```
void Main()
    {
        weekdays day = weekdays.wednesday;
        Myclass.Display(day);
        C.RL();
    }
}
}
```

→ Here enum will be loaded like a class.

→ enum doesnt have any returntype.

→ It will create memory in stack memory.

→ Default datatype of enum is int.

Q) Can we change the data type of Enum?

→ Yes, but it should be any one of the numerical data type like below.

```
enum <enumname> : Long
{
}
```

Structure :

→ Structure is value type, when we create an object for structure it will be allocated into stack memory.

→ To define structure we have to use Struct keyword.

→ Structure is a collection of members like variables, Properties, Constructors, methods and so... on..

Syntax :

```
<am> Struct <structurename>
{
    // members.
}
```

→ Default accessmodifier of structure is internal.

→ Default accessmodifier of structure members will be private.

→ we can create an object for structure without using new keyword only when the structure doesnot have instance variables.

→ We cannot define Explicit parameterless constructor with in the structure.

→ we cannot initialize instance variables of a structure by using instance field initializes for this we have to go for either parameterized constructor (or) property.

→ Structure can contain static members.

→ We cannot have static structure, we can not have abstract structure, we cannot have sealed structure.

→ But we can have partial structure and generic structure.

→ Structure will not support inheritance due to the reason using structure we cannot implement overriding but we can implement function overloading.

→ Access Modefier of structure and structure member can not be protected and protected internal.

Example

```
namespace structureEx.
{
  struct Employee
  {
    int eno;
    string ename;
    static string compName;
    static Employee ()
    {
      compName = "Satyam Computers";
    }
```

```
internal Employee (int eno, string ename)
{
  this.eno = eno;
  this.ename = ename;
}
internal void Display ()
{
  C.WL ("EmpNo is:" + empNo);
  C.WL ("EmpName is:" + ename);
  C.WL ("compName is:" + compName);
}
}
class Program
{
  Void Main ()
  {
    Employee emp1 = new Employee (111, "Rama");
    emp.display();
    C.RL();
  }
}
}
```

Differences b/w structure and class

Class	Structure
1. class Keyword	1. Struct Keyword.
2. object will be creating with in heap memory.	2. object will be creating with in stack memory.
3. Instance initializers are allowed.	3. Instance initializers not allowed.
4. To create object we should use new operator.	4. without using new operator also we can create object.
5. Explicit default constructor we can have.	5. we cannot have explicit default constructor.

6. we can have Static class, abstract class, sealed class.	6. We can not have static, abstract class, Sealed class
7. class will support inheritance	7. will not support inheritence
8. class will support function overriding.	8. will not Support function over-riding.
9. Access modifier of a class & class member can be any file.	9. Access Modifier of a structure cannot be protected & protected internal.
10. class is reference type.	10. Structure is value type.

Windows Forms

→ what is windows Applications?

→ windows Applications are Single user application

→ windows Application will not depend on internet connectivity.

→ windows Application will have a user Interface which can be called windows Form.

Ex:- MS-word, Super market billing application, mobile Shoppe day to daye transaction application.

→ when we will go for windows Applications?

→ According to the requirement whenever an application should be available to Single user we will develop a windows application for this type of requirement.

→ How to develop a windows Application by using .Net?

By using .Net if we want to develop a windows application we required a .Net language called C#.Net and we required a .Net windows technology called windows Forms.

→ Comparison between Console Application, windows Application, web Application.

Console Application	windows Application	web Application
1. Single user.	1. Single user.	1. Multi user.
2. No user Interface	2. will have user Interface (windows Forms)	2. Supports user Interface (web Page (or) web Form)
3. Not depending on internet	3. Not depending on internet	3. depending on internet
4. To develop Console Application we have to use C#.Net and console technology.	4. To develop windows Application we have to use C#.Net and windows Forms.	4. To develop web Application we have to use C#.Net and ASP.Net

5. In real-time we will not develop Console application	5. According to the requirement whenever that application should be available for only one user then we will go for windows Application.	5. According to the requirement whenever that application should be available for multiple users then we will go for web application.

→ windows Applications are called as windows Forms / winforms / Desktop applications / GUI based applications.

→ How to create a new windows application?

→ Start → Programs → Microsoft Visual Studio 2010 → Microsoft visual Studio 2012

→ It will open visual Studio .Net IDE

→ click one New project → it will open New Project window

→ Select c# language and type of application is <u>windows Forms</u>

rename application <u>myfirstwindowsApplication</u> click ok button.

→ with this process a new windows Application Development environment will opened.

→ windows Application development environment will having following windows.

1. Solution Explorer window
2. Designer window
3. Code window
4. Toolbox window
5. Properties window

1) <u>Solution Explorer window</u> :-

→ This window will display the complete project related files.

→ By default this window will have 2 important files.

a) Form1.cs

b) Program.cs

a) Form1.cs :-

→ It is a C#.Net class file

→ Here every windows Form is a userdefined class i.e Form1, Form2, Form3, - - - , Form n are userdefined classes.

→ This Form1.cs will support two modes

i) Design mode

ii) Source mode

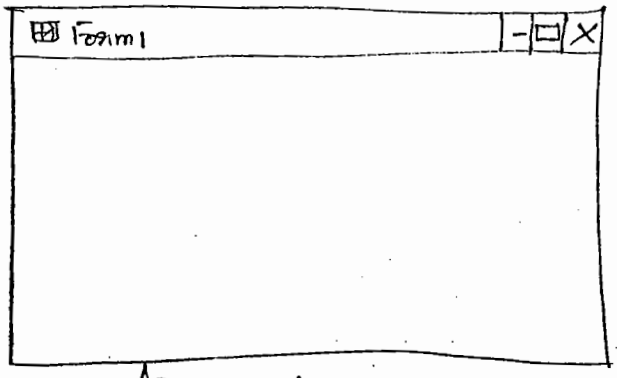
2) → i) Form1.cs/Design mode :-

→ In this mode, it will display the designer window of the concern windows Form.

→ In this mode, we can design the windows Form with the help of controls according to the requirement.

Structure of Form1.cs/Designer window

Form1.cs [Design]



3) → ii) Form1.cs/Source mode :-

→ In this mode it will display the concern form code window.

→ This window will display the Concern Form class file skelton (or) structure

→ For all windows Forms classes microsoft is providing a base class called "Form".

Form :-

→ Form is a Predefined class which is defined by microsoft within System.windows.Forms base class library.

→ Form class is Super class for all form related user defined classes like Form1, Form2, Form3, -- Formn.

→ within Form class microsoft defined all Forms related methods, properties and events.

Structure of System.windows.Forms base class library:-

```
namespace System.windows.Forms
{
    class Form
    {
        // Forms related
        // methods
        // properties
        // events
    }
}
```

Structure of Form1.cs class file :-

```
using System;
using System.windows.Forms;   } → BCL
namespace MyFirstwindowsApplication → Application name
{
    class Form1 : Form → Base class
    {                  → user defined class
        // methods
        // properties → All are inherited from the Form super class
        // events
    }   // How we will defining the functionality of the Form1.
}
```

Note:- To define the functionality of Form, we will define our own user-defined functions as well as we will use predefined methods, properties, events which are defined by microsoft within Form class.

4) Tool Box window :-

→ This window will display the collection of controls, every control will have its own functionality.

→ whenever we want to use a control we have to drag and drop from tool box to windows Form.

→ In windows Forms every control is providing the microsoft as a predefined class.

→ TextBox, Button, Label ... are predefined classes which are defined by the microsoft within a BCL called System.windows.Forms.

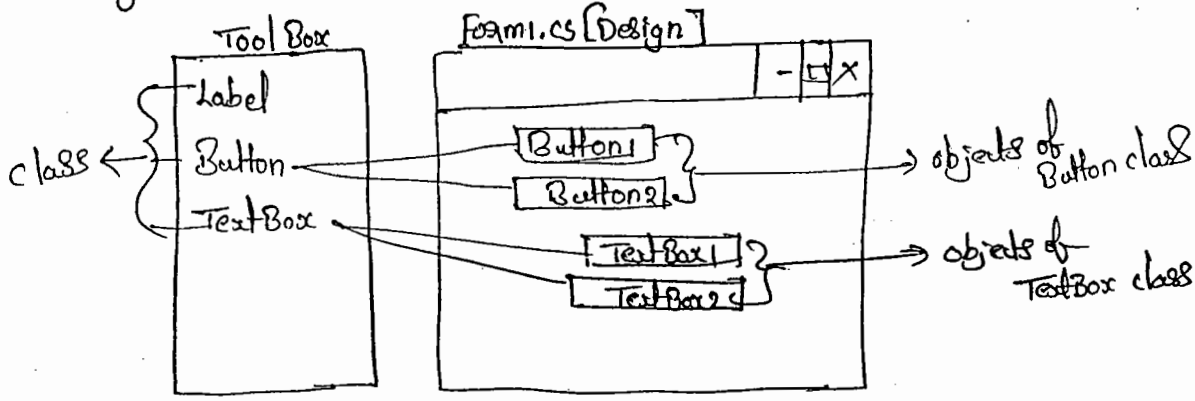
→ within every control microsoft defined concern control related defined methods, properties, events....

→ Structure of the System.windows.Forms BCL

```
namespace System.windows.Forms
{
    class Form
    {
    }
    class Button
    {
    }
    class Textbox
    {
    }
        :
}
```

→ whenever a control is representing within the toolbox window it is representing concern control predefined class.

→ whenever control is moving to the windows Form internally it is creating an object concern control class like below.



5) Properties window :-

→ This window will display the selected item properties and events.

Properties :-

→ Property is a member of a class which will define the shape of the object

→ A property we can access in two ways

a) In design time

b) In Run time

a) Design time :-

→ If we want to access a property in design time we can access by using properties window.

b) Run time :-

→ If we want to access a property in run time we can access with the help of code window like below.

Ex:- TextBox2.Text = "dotnet";

object of TextBox class

property TextBox class

value

Events :- (The reaction for some action)

→ Event is a member of a class

→ An event will not execute by default, event will execute for some action.

→ when user will do some action concern event will call (or) execute which is called event firing (or) event calling (or) event executing.

Default event :-

→ for every control we have collection of events, but one frequently used event is recognized as default event.

Ex:- Form default event is Load event

Button " " click event

TextBox " " Textchanged event

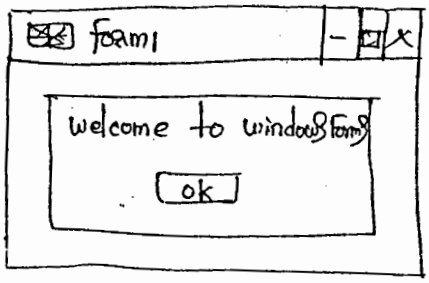
⇒ Example for Form Load event event

Load Event :-

→ Load event is the default event of the form.

→ This event will execute when the form is loading.

Requirement :-



Step1:- Open a windows Application rename it as LoadEvent Example.

Step2:- Double click on Form1 which will generate Load event within the form1.cs

Step3:- write the below code within Form1.cs

```
nameSpace LoadEventExample
{
    class Form1 : Form
    {
```

```
Public Form1()
{
    InitializeComponent();
}
```

default constructor

```
Private Void Form1_Load (object Sender, EventArgs e)
{
    MessageBox.Show ("welcome to windows Forms");
}
}
}
```

MessageBox.Show() → DialogResult → it's a dialog box

MessageBox.Show() :-

→ Here MessageBox is a Predefined class which is a Part of System.windows.Forms BCL.

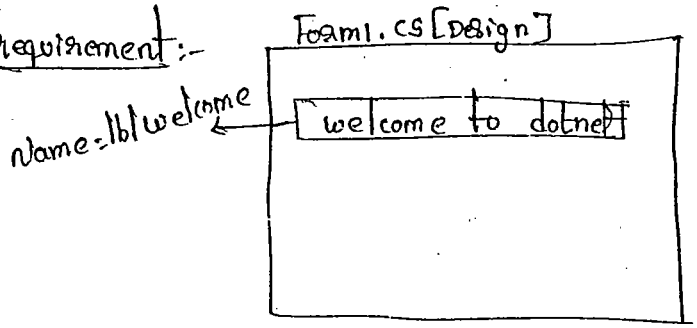
→ Show() Predefined static member method of MessageBox class. This method will display the message box with the given message.

) Label Control :-

→ Using this control we can display messages as well as we can use this control to describe about other controls.

→ Example to display welcome message by using label control.

requirement :-



Step1:- Open a windows Application rename it as label example.

Step2:- Drag and drop label control on Form1

Step3:- Open Properties window select label1 change the name Property as lblwelcome and remove the Text property.

Step 4:- write the below code within Form1.cs file

```
namespace LabelExample
{
        class Form1 : Form
        {
          Private void Form1_Load ( )
          {
                lblwelcome.Text = "welcome to dotnet";
          }
        }
}
```

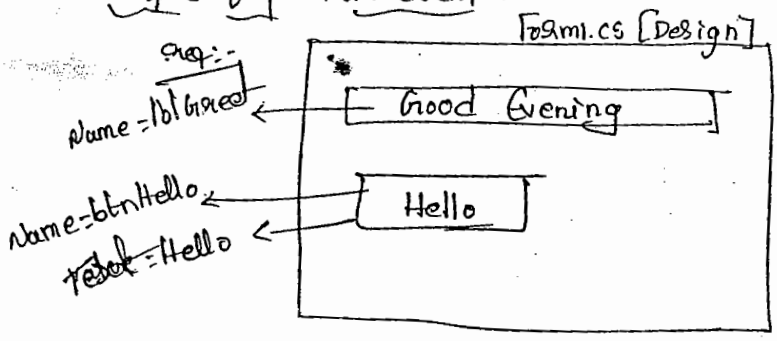
2) Button Control:-

→ This control we will use to give some command to the application.

Default Event:-

Click:- click event is the defaulet event of the button. this event will fire when user click the button control.

Example for click Event:-



Step 1:- Designing Form1

Step 2:- write the below code within Form1.cs file

```
namespace ClickEvent Example
{
        class Form1 : Form
        {
          Private void btnHello_click( )
          {
                lblGreet.Text = "Good Evening";
          }
        }
}
```

3) TextBox Control:-

→ whenever we want to accept input from user as well as whenever we want to display the output to user then we can use textBox control.

Properties:-

a) Name:- This property will give uniqueness to given control among collection of controls within the concern windowsForm.

b) Text:- it is representing content of the textbox (or) text of the textbox.

c) Backcolor:- It is " background color of the textbox. By default background color of the textbox is 'window'.

d) BorderStyle:- It is representing the style of the border. It have 3 options

i) None

ii) Fixed Single

iii) Fixed 3D (default)

e) CharacterCasing:-

i) Normal (default)

ii) Upper

iii) Lower

*f) Dock:- It is representing the position of the textbox with the windows Form.

options:- i) None (default)

ii) Left

iii) Right

iv) Top

v) Bottom

g) Enabled:-

a) True (default):- textbox will accept i/p from the user as well as it display output to the user.

b) False:- it will not accepting i/p from the user but it display output to the user.

h) Font:- using this property we can access Font attributes like Font Name, Font Size, Font Strikeout and soon.

i) Forecolor :- it is representing the text color.

j) Location :- it is representing x & y coordinate value of the text box with in windows form.

k) Locked :- it is boolean property.

i) True :- we can't move the textbox in design time.

ii) False (default) :- we can move the textbox in design time.

l) Multiline :-

i) True :- textbox will accept i/p from the user in multiple lines.
we can change the textbox width and height.

ii) False :- (default) :- textbox will accept i/p from the user in single line.
we can't change the textbox ~~width and~~ height, we can change the width of the textbox.

2GB

m) Maxlength :- it is representing the maximum number of character within textbox.
by defaut value is 32767

* Single line textbox will accept upto 2GB characters
* Multi line " " 4GB characters.

n) Passwordchar :- This property will read given text box as password textbox.

o) Readonly :-

i) True :- it will not accept i/p from user but it will display output

ii) False (default) :- it will accept i/p from the user as well as it display output to the user.

p) Size :-
it is representing the width & height of the textbox.

q) TabIndex :- this property is representing the cursor focus among all the text boxes within the windows form.
The textbox whose tabindex is '0' will get the first focus.
" " " '1' " Second focus & so on.

r) TextAlign :- it is representing the alignment of the text within textbox.

i) Left (default)
ii) Right
iii) Center

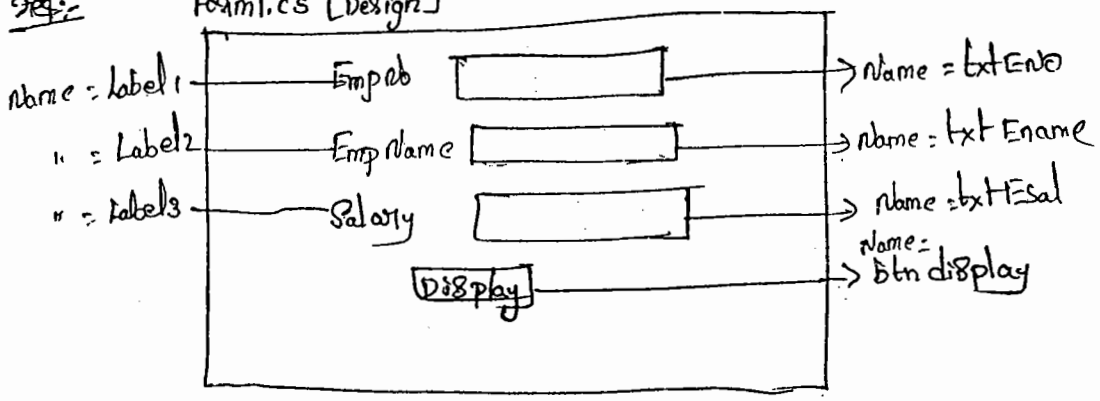
t) Visible :-

i) True (default) :- textbox will visible in design time as well as runtime.

ii) False :- textbox will visible in designtime but invisible in runtime

-> Example to accept display the Emp information ~~from the user~~ :-

Eg:- Form1.cs [Design]



Step 1 :- Design Form1.cs like above

-> Drag & drop 3 label control open properties window change

Label1 Text as EmpNO
Label2 " EmpName
Label3 " Salary

-> Drag & drop 3 textboxes open properties window change

Textbox1 Name as txtENO
" 2 " txtEname
" 3 " txtESal

-> Drag & drop 1 button open properties window change

Button1 name as btndisplay
~~Tes~~ " Text as Display

Step 2 :- Double click on Display button it will generate click event code with the Form1.cs

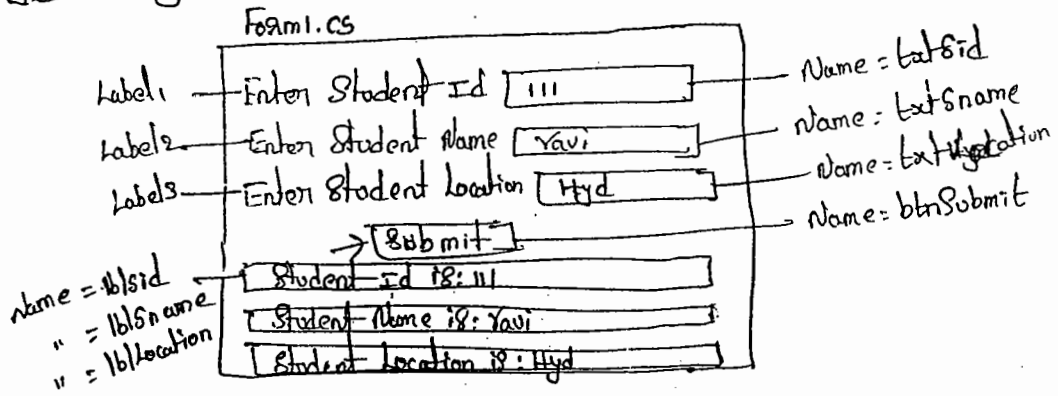
write the below code on Form1.cs

```
namespace TextboxExample
{
    class Form1 : Form
    {
        Private void btndisplay_click( )
        {
            txtENO.Text = "123";
```

```
txtEname.Text = "Jhon";
txtEsal.Text = "1000";
}
}
}
```

→ Example to accept student info from the user and display that accepted information

Step 1:- Design Form1.cs like below



Step 2:- write the below code in Form1.cs

```
Class Form1
{
  Private void btnSubmit_click ( )
  {
      txtsid.Text = Convert.ToString(txtsid.Text);
      txtsname.Text = Convert.ToString(txtsname.Text);
      txtLocation.Text = Convert.ToString(txtLocation.Text);
      lblsid.Text = "Student Id is:" + txtsid.Text;
      lblsname.Text = "Student Name is:" + txtsname.Text;
      lblLocation.Text = "Student Location is:" + txtLocation.Text;
        txtsid.clear();
        txtsname.clear();
        txtsloc.clear();
  }  txtsid.focus();
}
```

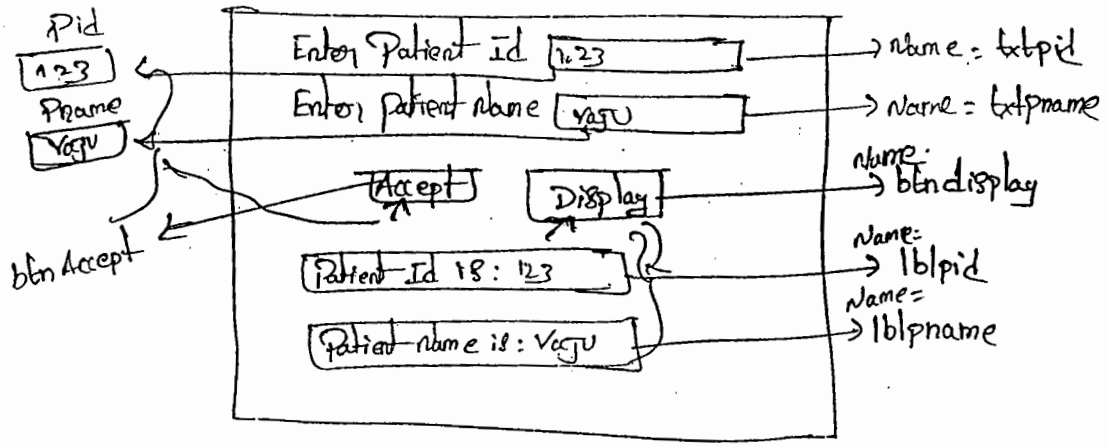
→ the cursor focus on 1st textbox

clear():- This method will clear the given textbox value.

Control class → focus():- This method will move the focus to concern control.

→ Example to accept patient information store it and display the stored information

Step 1:- Design Form1.cs like below



Step 2:- Form1.cs

```
class Form1
{
    public string Pid, Pname;
    private void btnAccept_click( )
    {
        Pid = Convert.ToString(txtpid.Text);
        Pname = Convert.ToString(txtPname.Text);
        txtpid.Text = "";
        txtPname.Text = "";
        txtpid.focus();
    }
    private void btndisplay_click( )
    {
        lblPid.Text = "Patient Id is : " + pid;
        lblPname.Text = "Patient name is : " + Pname;
    }
}
```

Event of TextBox :-

a) Leave :- This event is one of the focus related event. This event will fire when the focus is leaving from the TextBox.

b) Enter :- It is one of the focus related event, this event will fire when the focus is will enter into the textbox.

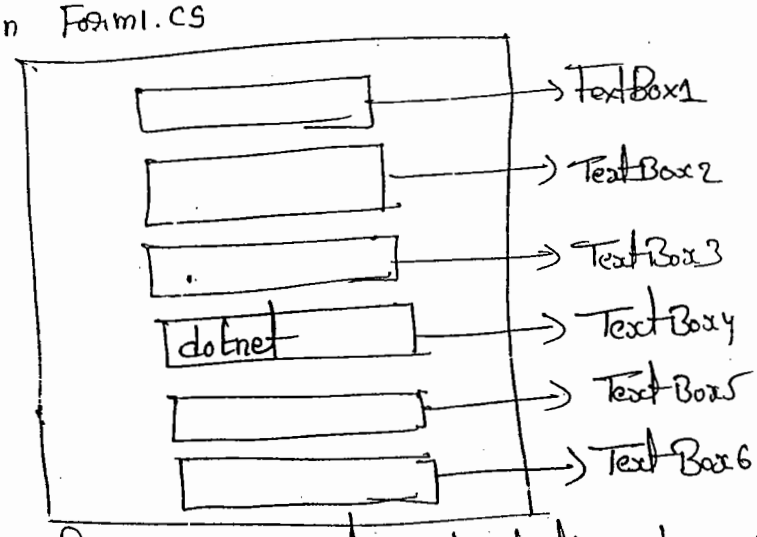
c) Textchanged :- It is default event of the textbox. This event will fire when user will change the text of the textbox.

d) click :- It is mouse related event, this event will fire when user will click mouse within the textbox.

e) keyPress :- It is keyboard related event, this event will fire when user will press any key from the keyboard within the textbox.

→ Example for textbox Events.

Step 1:- Design Form1.cs



→ when the focus is leaving from the textbox1 change the background color of textbox3 as ('Red').

→ when the focus is enter into textbox 3 initialize a text called "Sathya" within textbox5.

→ when user is trying to change the text of the textbox4 we have to display a messagebox like below

Your not permitted.

→ when user will click within the textbox2 change the background color of Form1 as blue,

→ when user entering some text into textbox6 display a messag box like below
"your entering some input".

Step 2:- Form1.cs in file write the below code.

```
class Form1
{
    Private void TextBox1_Leave( )
    {
        TextBox2.Backcolor = System.Drawing.color.Red;
    }
    Private void TextBox3_Enter( )
    {
        TextBox5.Text = "Sathya";
    }
    Private void TextBox4_Textchanged( )
    {
        MessageBox.Show("your not permitted");
    }
    Private void TextBox2_click( )
    {
        this.Backcolor = System.Drawing.color.Blue;
    }
    Private void TextBox6_keyPress( )
    {
        MessageBox.Show("your entering some input");
    }
}
```

→ Color.Red :-

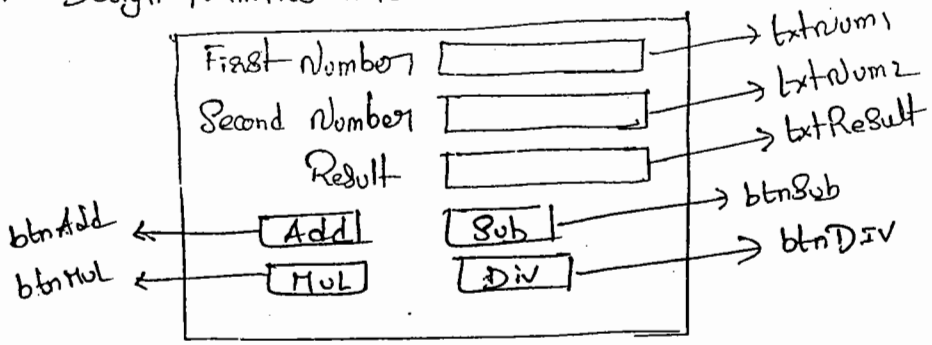
→ Color is a predefined structure which is part of System.Drawing BCL.

→ within this predefined structure microsoft re defined all the color related properties examples red, blue -- --Soon.

→ Example to implement calculation by using textboxes.

use this same problem try, catch & finally block bcz here we are getting divide by zero exception

Step1: Design Form1.cs like below



Step2:- write the below code within Form1.cs

```
class Form1
{
    Public int a,b,c;
    Private void btnAdd_click( )
    {
        a = int.parse (txtNum1.Text);
        b = int.parse (txtNum2.Text);
        c=a+b;
        txtResult.Text = Convert.Tostring(c);
    }
    Private void btnSub_click( )
    {
        a = int.parse (txtNum1.Text);
        b = int.parse (txtNum2.Text);
        c = a-b;
        txtResult.Text = convert.Tostring(c);
    }
    Private void btnMul_click( )
    {
        a = int.parse (txtNum1.Text);
        b = int.parse (txtNum2.Text);
        c=a*b;
        txtResult.Text = Convert.Tostring(c);
    }
    Private void btnDiv_click( )
    {
        a = int.parse (txtNum1.Text);
        b = int.parse (txtNum2.Text);
        c=a/b;
        txtResult.Text = Convert.Tostring(c);
    }
}
```

Validations :-> 1). txtNum1 should not be empty (leaving Cont) 3) -> txtNum2 should not be empty
2) -> txtNum1 should allow only digits 4) -> txtNum2 should allow only digits
5) while performing the division second number should not be zero.

-> KeyPress Event :-

-> This event will fire when user will press a key from the keyboard within the textbox.

-> Skelton of keyPress Event :-

```
Private void textBox1_KeyPress (Object Sender, KeyPressEventArgs e).
{

}
```

-> This keyPress Event is having to Argument
1.) Object Sender 2) KeyPressEventArgs e

1) Here 'Object' is a predefined class and 'Sender' is an object of Object predefined class

Here 'Sender' is presenting the control name which is raising the KeyPress event.

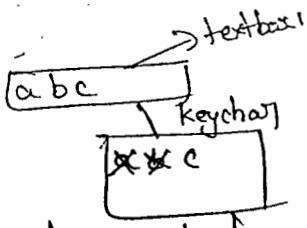
2) Here 'keyPressEventArgs' is a predefined class and 'e' is an object of keyPressEventArgs class

within KeyPressEventArgs class microsoft defined two important properties like below.

a) KeyChar
b) Handled

a) Keychar :- It is a predefined member property of KeyPressEventArgs class.

→ This property is representing character which is raising the keypress event.



keychar at a time it will take only one characters.

b) Handled :- It is a boolean property which can have either true (or) false.

By default Handled is "false".

After executing the keypress event, if u want to clear the character which is raising keypress event then we have to initialize

```
e.Handled = true;
```

Char :-

char is a predefined structure, within this predefined structure microsoft defined various methods to validate single characters.

members of 'char' predefined structure:-

char.IsDigit(char c) :-

This method will return 'true' value when input character is a 'digit' otherwise it will return as 'false' value.

Ex:- char.IsDigit('6') → true
char.IsDigit('a') → false

char.IsLetter(char c) :-

Ex:- char.IsLetter('a') → true
char.IsLetter('8') → false

This method will return 'true' value when i/p character is letter otherwise it will return as 'false'

char.IsSymbol(char c):-

ex:- char.IsSymbol('@') → true
char.IsSymbol('a') → false

char.ToLower( ) :- ex:- char.ToLower('A') → a

char.ToUpper( ) :- ex:- char.ToUpper('a') → A

Validations:-

1) txtNum1 Should not be empty:-

```
void txtNum1_Leave( )
{
    if(txtNum1.Text == "")
    {
        MessageBox.Show("please enter First number");
        txtNum1.Focus();
    }
}
```

2) txtNum1 Should allow only digits:-

```
void txtNum1_KeyPress( )
{
    if(char.IsDigit(e.KeyChar) == false)
    {
        MessageBox.Show("please enter only digits");
        e.Handled = true;
    }
```

3) txtNum2 Should not be empty:-

```
void txtNum2_Leave( )
{
    if(txtNum2.Text == "")
    {
        MessageBox.Show("please enter Second number");
        txtNum2.Focus();
    }
}
```

4) txtNum2 should allow only digits:-

```
void txtNum2_KeyPress( )
{
    if(Char.IsDigit(e.KeyChar) == False)
    {
        MessageBox.Show("please enter only digits");
        ~~txtNum2~~ e.Handled = true;
    }
}
```

5) while performing the division second number should not be zero:-

```
void btnDiv_click( )
{
    a = Convert.ToInt32(txtNum1.Text);
    b = Convert.ToInt32(txtNum2.Text);
    if(b==0)
    {
        MessageBox.Show("please enter second number");
        txtNum2.clear();
        txtNum2.Focus();
        return;
    }
    c = a/b;
    txtResult.Text = c.ToString();
}
```

Implement the above validation by using Exception Handling mechanism:-

```
void btnDiv_click( )
{
    a = Convert.ToInt32(txtNum1.Text);
    b = Convert.ToInt32(txtNum2.Text);
    try
    {
        c = a/b;
        txtResult.Text = c.ToString();
    }
    catch(Exception e)
    {
```

```
MessageBox.Show("please enter other than zero");
txtNum2.clear();
txtNum2.Focus();
return;
}
}
```

→ In above example keypress event should not fire for following keys.

Esc, BackSpace, Space, Tab, - - -

~~Link Button Control~~ :-

=> Link Label Control :- using this Control we can display an hyperlink with in the windows Forms. whenever we want to implement navigation from 1 windows Form to another windowsForm as well as navigation from windows Form to website then we can use this control.

Properties :-

a) Text :- this property will display given text as hyperlink.

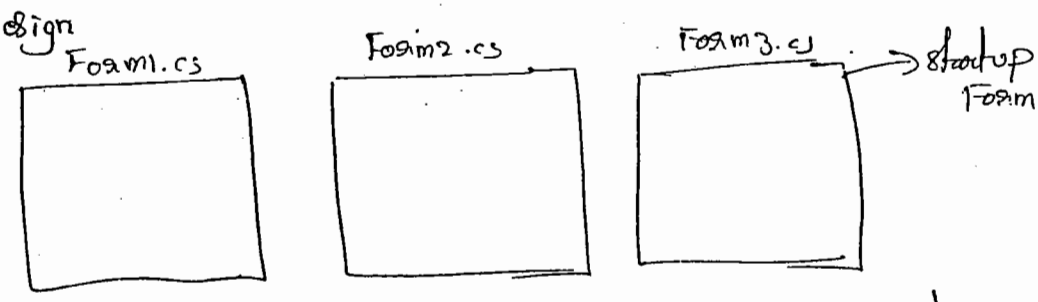
b) Link color :- " representing color of the hyperlink, by default color is blue.

Default Event :-

LinkClicked :- This is default event of the link Label. This event will fire when user will click the concern link label.

Example to Set as start Form

Step 1 :- Design



-> open a windows application rename it as StartupFormExample.

-> By default we will have one Form i.e Form1.cs.

-> Adding Form2 :-

↳ open Solution Explorer

↳ Select Application name rightclick add New item it will open add new item window

Select a predefined predefine template as windows Form

click add button

with this process a windows Form will add to the Solution Explorer i.e Form2.as

-> Adding Form3

Repeat the above step

Step 2:- Set Form3 as Startup Form

-> open Solution Explorer

-> Write the below code within Program.cs file

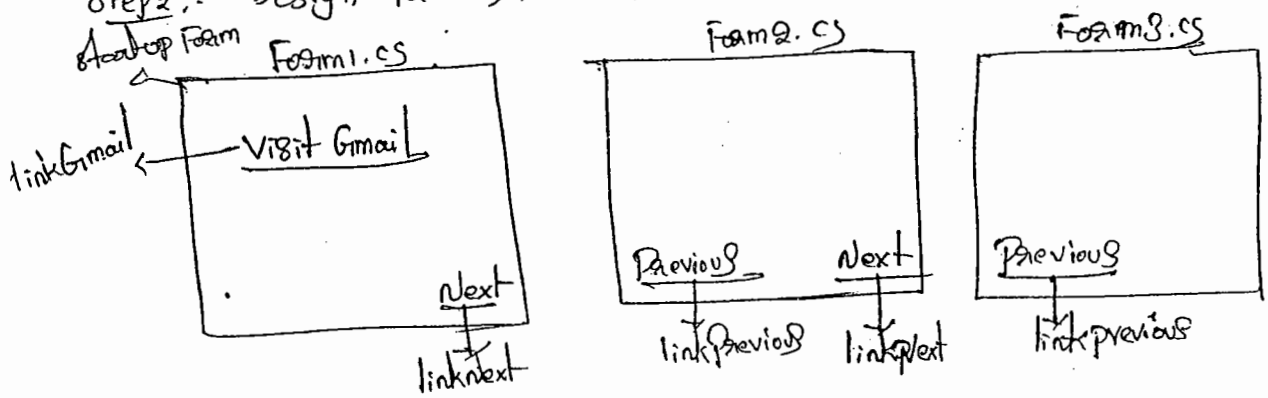
```
class program
{
    Void main()
    {
        Application.Run(new Form3());
    }
}
```

-> Example to implement internal navigations as well as External navigations

Step1:- ~~window~~ open a windows Application add two more windows Forms.

they are Form2.cs, Form3.cs

Step2:- Design Form1, Form2, Form3 like below



Step3:- write the below code within Form1.cs

```
using System.Diagnostics;
namespace LinkButtonExample
{
    class Form1
    {
        void linkGmail_linkclicked()
        {
            Process.Start("http://www.gmail.com");
        }
```

```
        void linkNext_Linkclicked( )
        {
            Form2 objF2 = new Form2();
            objF2.Show();
            this.Hide();
        }
    }
}
```

Form2.cs :-

```
class Form2
{
    void linkPrevious_linkclicked( )
    {
        Form1 objF1 = new Form1();
        objF1.Show();
        this.Hide();
    }
    void linkNext_linkclicked( )
    {
        Form 3 objF3 = new Form3();
        objF3.Show();
        this.Hide();
    }
}
```

Form3.cs :-

```
class Form3
{
    void linkPrevious_linkclicked( )
    {
        Form 2 objF2 = new Form2();
        objF2.Show();
        this.Hide();
    }
}
```

Process.Start():-

→ Process is a predefined class which is part of System.Diagnostics BCL

→ Start is predefined static member method of process class this method will open a webpage based on the given url.

Show() & Hide():-

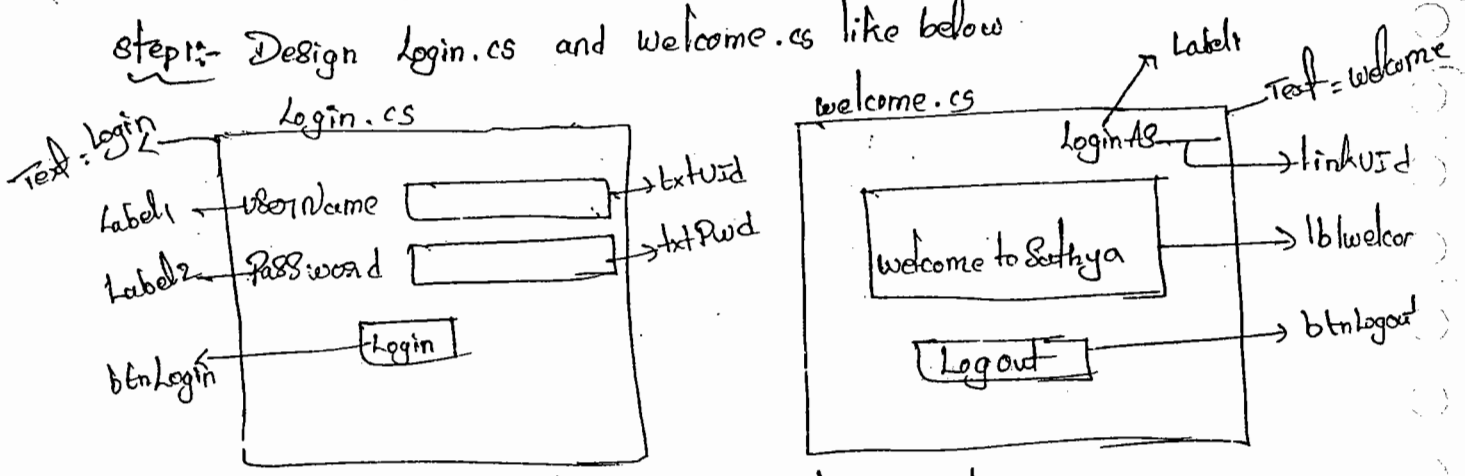
→ These two are predefined members of Form class.

→ Show() will open a concern windows Form.

→ Hide() will hide the concern windows Form.

→ Example for Login and welcome Forms.

Step1:- Design Login.cs and welcome.cs like below



Validations:- UserName textbox should not be empty.
Password textbox " " .

Login.cs Code:-

```
class Login
{ void btnlogin_click( )
  { if (txtUid.Text == " ")
    {
      MessageBox.Show("pls enter username");
      txtUid.Focus();
      return;
    }
    if (txtPwd.Text == " ")
    {
      MessageBox.Show("pls enter Password");
      txtPwd.Focus();
      return;
    }
```

```
        welcome objwelcome = new welcome();
            objwelcome.show();
        this.Hide();  objwelcome.linkUId.Text = txtUId.Text;
    }
}
```

welcome.cs code :-

```
class welcome
{
  Private void welcome_Load( )
  {
      lblwelcome.Text = "welcome to Sathya";
  }
  Private void btnLogout_click( )
  {
      Login objLogin = new login();
      objLogin.Show();
      this.Hide();
  }
}
```

Note :- Befor Executing the above Program change the linkUId access modifier as Public within welcome.Designer.cs file like below.

```
Public System.windows.Forms.LinkLabel linkUId;
```

⇒ Designer.cs file :-

Every windowsForm will have a file called .designer.cs, within this file we will have the Concern windowsForm autogenerated controls code.

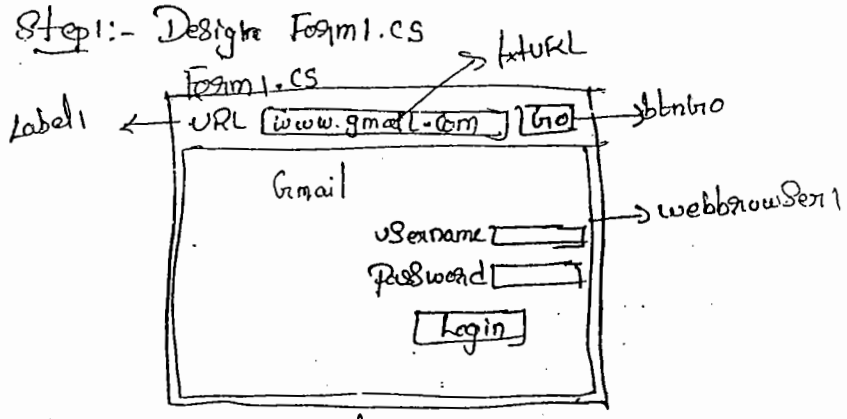
Imp IQ Web Browser Control :-

→ using this control we can display a web Page within the windows.Form.

→ whenever we want to display a webPage within the windows.Form then we can use a control called web browser

→ Example for web browser Control.

Step1:- Design Form1.cs



Step2:- write the below code within Form1.cs

```
namespace WebBrowserExample
{
  class Form1
  {
    Private void btnGo_click( )
    {
      webBrowser1.Navigate (txtURL.Text);
    }
  }
}
```

→ Note:-

Navigate():-

→ It is a Predefined member method of web Browser class.

⇒ This method will open a webpage based on given URL within the WebBrowser Control.

→ RadioButton Control:-

whenever we want to give Single Selection among multiple items then we go for radiobutton.

Properties:-

a) checked:- It is a boolean property. It can have either true value (or) false value.

i) True :- radio button is selected.

ii) False :- radio button is unselected. defaut value of the radio button is false.

b) Autosize :-

i) True :- (default)

we can't increase and decrease the width & height of the radio button.

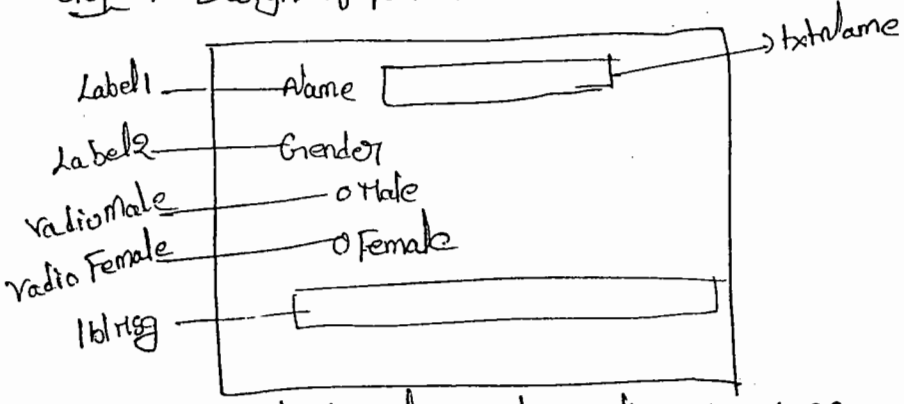
ii) False :- we can increase & decrease the width & height of the radio button

→ Events :-

a) checkedchanged :- It is default event of the radio button. This event will fire when user will select the radiobutton as well as unselect the radiobutton.

→ Sample for radio button

Step 1 :- Design of Form1.cs



Step 2 :- write the below code within Form1.cs

```
class Form1
{
    void radioMale_checkedchanged ( )
    {
        lblmsg.Text = ~~radio~~ txtName.Text + " is : " + radioMale.Text;
    }
    void radioMale_checkedchanged ( )
    {
        lblmsg.Text = txtName.Text + " is : " + radioFemale.Text;
    }
}
```

→ CheckBox Control:-

whenever we want to give multiple selection options among multiple options then we go for checkBox control.

Properties:-

a) checked:-

i) true:- checkBox is selected

ii) False (default):- checkBox is unselected.

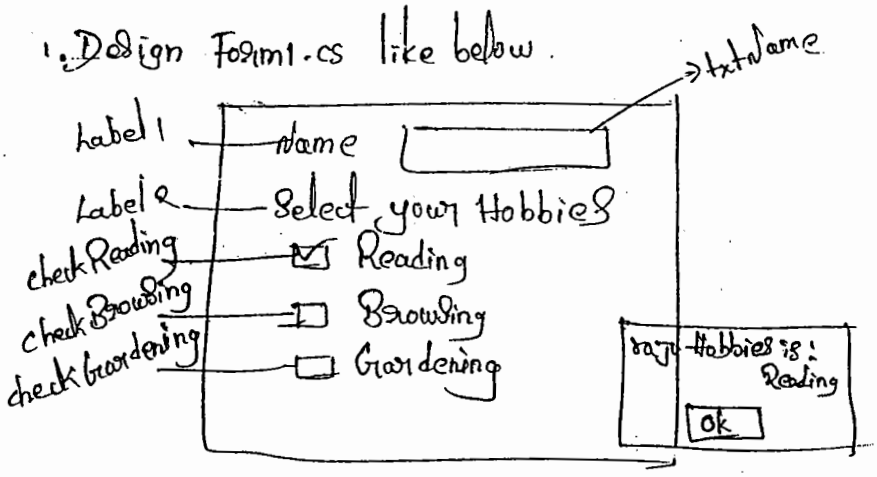
Event:-

a) checkedchanged:-

This event will fire when user will select the checkBox as well as unselect the checkBox.

→ Example for checkBox control.

1. Design Form1.cs like below.



2. write the below code within Form1.cs

```
class Form1
{
    void checkReading_checkedchanged( )
    {  if(checkReading.checked==true)
       MessageBox.Show (txtName.Text + "Hobbies is:" + checkReading.Text);
    }
    Void checkBrowsing_checkedchanged( )
    {  if(checkBrowsing.checked==true)
       MessageBox.Show (txtName.Text + "Hobbies is:" + checkBrowsing.Text);
    }
```

```
void checkGardening_checkedchanged ( )
{ if(checkGardening.checked)
    MessageBox.Show(txtName.Text + "Hobbies is: " + checkGardening.Text);
  }
}
```

⇒ **Grouping Controls:-**

→ Grouping Controls are Container controls, which can have a collection of other Controls.

→ In windows Forms we have 2 important Grouping controls

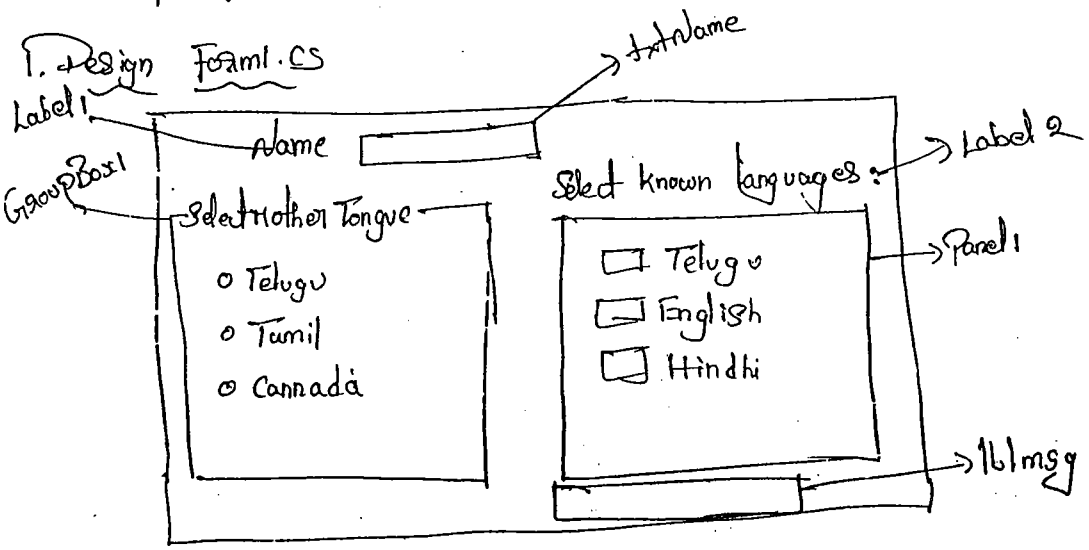
1. Group Box
2. Panel

Difference b/w GroupBox and Panel:-

Group Box	Panel
1. It has textarea	1. It has no textarea.
2. It will occupy more design Space.	2. It will occupy less design Space.
3. Group Box will Consume less memory.	3. Panel will Consume more memory.
4. When we have more design Space then we can go for GroupBox	4. When we have less design Space then we can go for Panel.

H.W → Example for GroupBox and Panel.

1. Design Form1.cs



ListBox Control:-

-> Using ListBox we will display multiple items to the user every item of the ListBox will be representing with index value.

1st item index will be '0'.

2nd " " 1 and so on.

-> ListBox will allow the user to select one item as well as multiple items as well as None items.

Design time properties of ListBox:-

a) Items:- it is a collection property this property is representing items of the listbox.

b) SelectionMode:-

i) one (default):- it allow the user to select one item

ii) None:- it will not allow the user to select any item

iii) MultiSimple:- listbox will allow the user to select multiple items

iv) MultiExtended:- "

c) Sorted:-

a) true:- items will displaying sorted order.

b) False (default):- listbox will display the item as it is.

Run time properties:-

a) SelectedItem:- This will return selected item with in the listBox

*b) SelectedIndex:- This will return index value of the selected item within the listbox.

c) SelectedItems:- This property will return selected items of the listbox.

d) SelectedIndices:- This property will return index values of the selected items within the listbox.

e) Items.count:- This property will return no. of items within the listbox.

Methods of ListBox class:-

a) Items.Add(): → object item

Note:- Item of the listbox datatype is object

This method will add given item to the listbox as a last item.

b) Items.Insert(int index, object item):-

This method will add an item to the listbox based on index value.

c) Items.Remove(Object Item)

This method will remove given item from the listbox.

d) Items.RemoveAt(int Index):-

This method will remove an item from the listbox based on index value.

e) Items.Clear():-

This method will clear all items from the listbox.

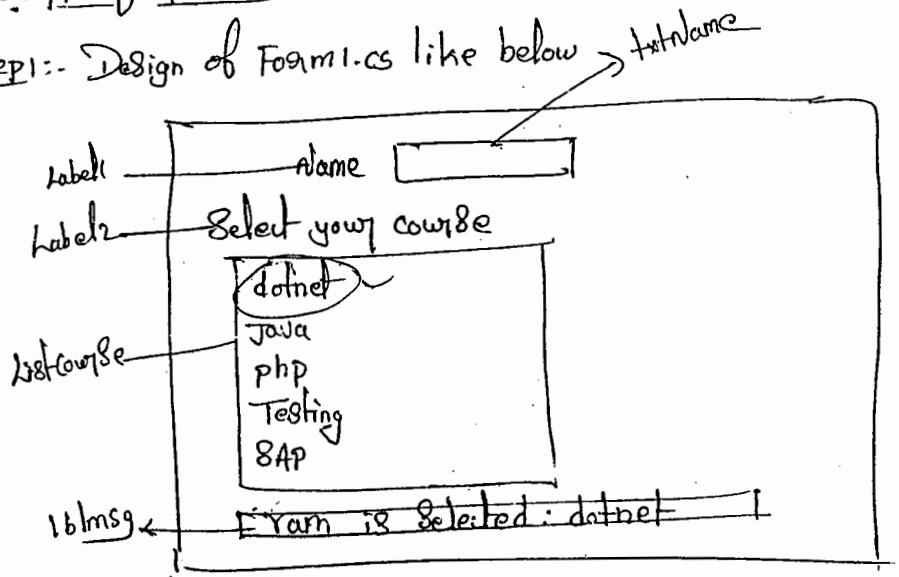
Events:-

a) SelectedIndexChanged:- (default event)

This event will fire when user will select an item within the listbox.

→ Example for ListBox

Step1:- Design of Form1.cs like below

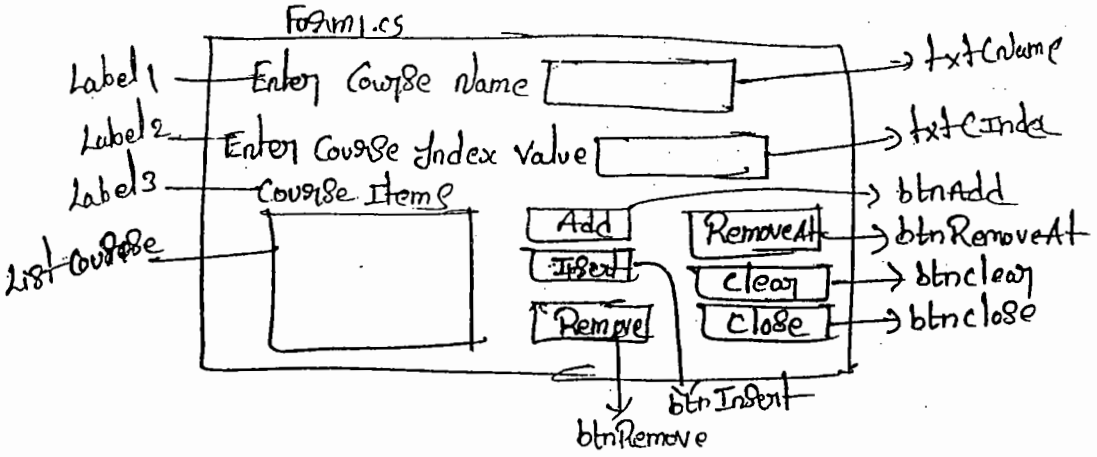


Add the items within the listbox in design time.

Step2:- write below code within Form1.cs

```
class Form1
{
    void listcourse_SelectedIndexChanged()
    {
        lblmsg.Text = txtName.Text + " is Selected: " + listcourse.SelectedItem;
    }
}
```

→ Example to Add an item to the ListBox in runtime



Form1.cs code:-

```
class Form1
{
    void btnAdd_click( )
    {
        ListCourse.Items.Add(txtCname.Text);
    }
    void btnInsert_click( )
    {
        ListCourse.Items.Insert((txtCIndex.Text).ToString(), txtCname.Text);
        Convert.ToInt32(txtCIndex.Text)
    }
    void btnRemove_click( )
    {
        ListCourse.Items.Remove(txtCname.Text);
    }
    void btnRemoveAt_click( )
    {
        ListCourse.Items.RemoveAt(Convert.ToInt32(txtCIndex.Text));
    }
    void btnClear_click( )
    {
        ListCourse.Items.Clear( );
    }
    void btnClose_click( )
    {
        this.close( );
    }
}
```

Validations:-

1) btnAdd click

→ txtCname should not be empty.

→ txtCname should not allow duplicate coursename (compare to listbox items)

2) btnInsert click:-

→ txtcname should not be empty.

→ txtcname should not allow duplicate coursenames.

→ txtcIndex should not be empty

→ txtcIndex should allow valid index value.

3) btnRemove click:-

→ txtcname should not be empty

→ txtcname should allow valid course name

4) btnRemoveAt click:-

→ txtcIndex should not be empty

→ txtcIndex should allow valid index values.

→ validations under clear button:-

→ when user will click the clear button if listbox is empty display a msgbox ("listbox is empty").

→ ListBox should not allow duplicate name like below

dotnet
Dotnet
DOTNET
dotNet

→ txtcIndex should allow only digits

1) btnAdd_click Validations:-

```
void btnAdd_click( )
{
  // if(txtCname.Text == " ")
  {
    MessageBox.Show("please enter course name");
    txtCname.Focus();
    return;
  }

  // if(listcourse.Items.Contains(txtCname.Text))
  {
    MessageBox.Show("please enter valid course name");
    txtCname.clear();
    txtCname.Focus();
    return;
  }
  listCourse.Items.Add(txtCname.Text);
}
```

2) btnInsert_click Validation:-

```
void btnInsert_click( )
{
  //1  if(txtCname.Text == "")
     {
       MessageBox.Show("please enter course name");
       txtCname.Focus();
       return;
     }
  //2  if(listcourse.Items.Contains(txtCname.Text))
     {
       MessageBox.Show("please enter valid course name");
       txtCname.clear();
       txtCname.Focus();
       return;
     }

  //3  if(txtCindex.Text == "")
     {
       MessageBox.Show("please enter course index value");
       txtCindex.Focus();
       return;
     }
  //4  if((Convert.ToInt32(txtCindex.Text) < 0)||(Convert.ToInt32(txtCindex.Text)
                                                  > listcourse.Items.count))
     {
       MessageBox.Show("please enter index value from 0 to" +
                                          listcourse.Items.count);

       txtCindex.clear();
       txtCindex.Focus();
       return;
     }
     listcourse.Items.Insert(Convert.ToInt32(txtCindex.Text), txtCname.Text);
}
```

3) btnRemove_click validations :-

```
void btnRemove_click ( )
{
  //1  if (txtCname.Text == "")
       {
         MessageBox.Show("please enter course name");
         txtcname.Focus();
         return;
       }
  //2  if(listcourse.Items.Contains(txtcname.Text) == false)
       {
         MessageBox.Show("please enter valid course name");
         txtCname.clear();
         txtCname.Focus();
         return;
       }
       else
       {
         listcourse.Items.Remove(txtcname.Text);
       }
```

4) btnRemoveAt_click Validations :-

```
void btnRemoveAt_click()
{
  //1  if(txtCindex.Text == "")
       {
         MessageBox.Show("please enter course index value");
         txtCindex.Focus();
         return;
       }
  //2  if((Convert.ToInt32(txtCindex.Text) < 0) || (Convert.ToInt32(txtCindex.Text)
                                                 > listcourse.Items.count ))
       {
         listcourse.Items.RemoveAt(Convert.ToInt32(txtCindex.Text));
       }
       else
       {
         MessageBox.Show("please enter Index value from 0 to " +
                                                 (listcourse.Items.count));
         txtcindex.clear();
         txtcindex.Focus();
       }
}
```

→ Combo Box Control :-

using ComboBox Control ~~multipl~~ can display multiple items to the user but it will display items to the user on demand.

comboBox will allow the user to select only one item. it will not allow to select multiple items.

(All methods of ListBox common to ComboBox)

Difference b/w ListBox and ComBo Box :-

ListBox	ComBo Box
→ List Box will display all the items by default	→ comBo Box will display all the items on default.
→ ListBox will allow user to select one item as well as multiple items.	→ ComBo Box will allow user to select one item.
→ ListBox will occupy more design space.	→ ComBo Box will occupy less design space
→ whenever we want to provide to provide multi selection we can go for ListBox.	→ whenever we want to provide single selection we can go for Combo Box.

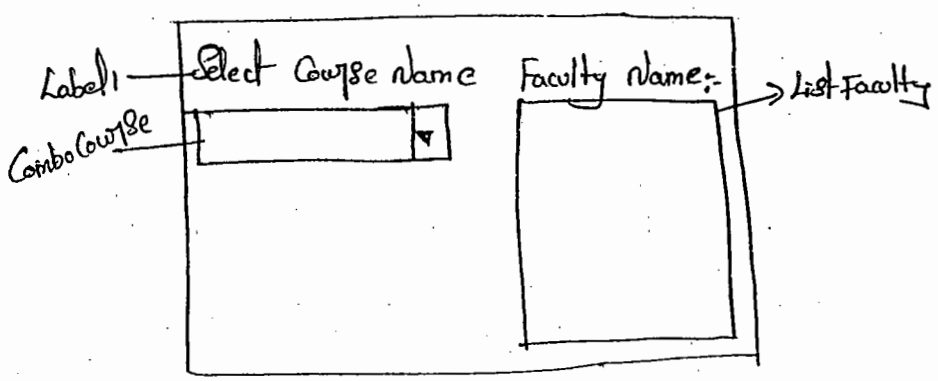
Default Event :-

SelectedIndexchanged :- (default event)

This event will fire when user will select an item within the combo Box.

→ Example for ComboBox and ListBox

Form1.cs



dotnet - revi, ramesh, rakesh

Java - kishore, sunil, anil

php - madhav, naresh, raju.

→ Adding the course Names to the Combo Box in design time and adding the Faculty Name to the ListBox in runtime.

Step1:- open a windows Application rename as ComboBox Example.

Step2:- Drag and Drop Labels, ComboBox, ListBox like above.

Step3:- Adding items to ComboBox

Select comboBox open Properties window double click on Items add items like dotnet, java, php.

Step4:- Adding items to ListBox

→ write the below code within Form1.cs

```
class Form1
{
    void ComboCourse_SelectedIndexchanged ()
    {
        ListFaculty.Items.Clear();
        Switch (ComboCourse.SelectedIndex)
        {
            case 0:  listFaculty.Items.Add ("ravi, ramesh, rakesh");
                     break;
            Case 1:  listFaculty.Items.Add ("kishore, Sunil, anil");
                     break;
            case 2:  listFaculty.Items.Add ("Madhav, Naresh, raju");
                     break;
        }
    }
}
```

→ Tab Control:-

→ Tab Control is a Container Control.

→ Tab control is a collection of Tab Page. Each tab Page will act like a window (or) Container. That container we can design with Some other controls.

→ But at a time within a Single wi tab Control we can display a Single tab Page.

-> By implementing tab control we can reduce the no. of windows forms.

Properties:

a) TabPages :- It is a collection property, which is representing the collection of tab pages within the tab control.

b) -> Every tabpage will be representing with one index value.

1st tab page will be '0'

2nd " " '1' and so on.

-> using this property we can add the tab pages to tab control.

-> Example for tab control in step by step process.

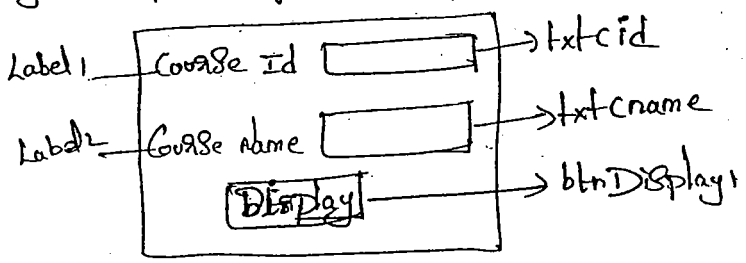
1) -> Drag and drop tab control and add tabpage 3 to tab control.

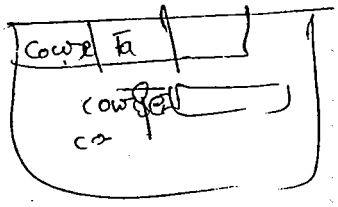
and change text of tabpage1 as 'Course',

" tabpage2 as 'Faculty',

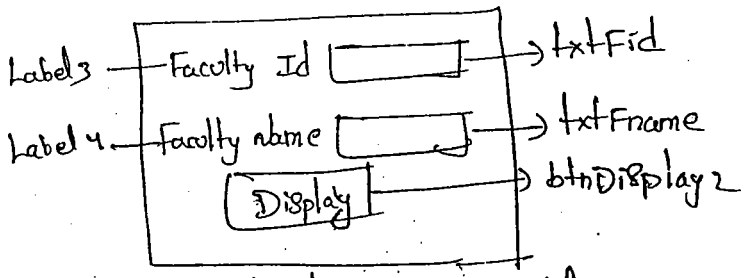
" tabpage3 as 'Student'.

2) -> Design Course tabpage like below

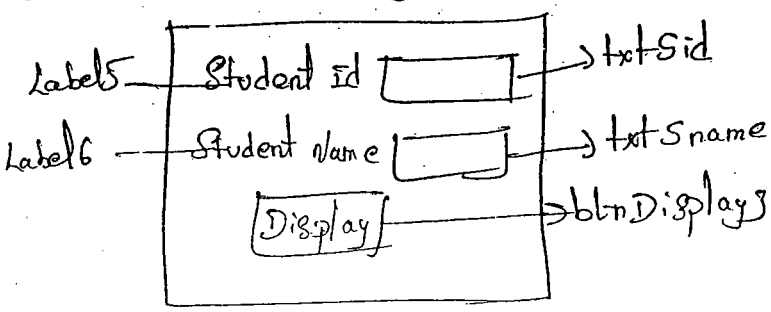




Design of Faculty tabpage like below



Design of Student tabpage like below



Step 8:- Form1.cs Code

```
class Form1
{
    void btnDisplay1_click( )
    {
        txtcid.Text = "10";
        txtcname.Text = "dotnet";
    }
    void btnDisplay2_click( )
    {
        txtFid.Text = "123";
        txtcname.Text = "ravi";
    }
    void btnDisplay3_click( )
    {
        txtSid.Text = "111";
        txtsname.Text = "Raju";
    }
}
```

Listview Control:-

→ Using Listview Control we can display the data in rows and column format,

- Every row of the listview will representing with one index value.
  1st row index value will be '0'
  2nd " " '1' and so on. as well as
- every column of the listview will be representing with one index value
  1st column index value will be '0'
  2nd " " '1' and so on.

→ whenever we want to add the rows to Listview Control in runtime. 1st we have to identify the width of the listview Control. like below

```
int width = listView1.Width;
```

↑ Variable ↑ Property

Then we have to decide the no. of columns which we want to display within the listview after that we have to decide the space of the each column

```
int Space = width/(no. of columns);
```

-> How to add columns to the listview - in runtime

listview1.Columns.Add(<Columnheader>, <Space>);

-> Example to adding the rows to the listview in runtime.

Step 1:- Design of Form1.cs

	C1	C2	C3
	Empno	Empname	Salary
r1	123	Jhon	1000
r2	124	Ravi	2000
r3	125	Mahesh	3000

Step 2:- Drag and drop listview control and go to Properties window

change View Property as Details and

'GridLines' " as True

Step 3: Form1.cs

```
class Form1
{
    void Form1_Load( )
    {
        int width = listview1.width;
        int Space = width/3;
        listview1.Columns.Add("Empno", Space);
        listview1.Columns.Add("Empname", Space);
        listview1.Columns.Add("Salary", Space);
        //1st row
        listview1.Items.Add("123");
        listview1.Items[0].SubItems.Add("Jhon");
        listview1.Items[0].SubItems.Add("1000");
        //2nd row
        listview1.Items.Add("124");
        listview1.Items[1].SubItems.Add("Ravi");
```

```
listview1.Items[1].SubItems.Add("2000");
//3rd row
listview1.Items.Add("125");
listview1.Items[2].SubItems.Add("rahesh");
listview1.Items[2].SubItems.Add("3000");
}
}
```

Note:- To add 1st item of the row we have to use items property, but to add 2nd item onwards we have to use index value of the row as well as we have to use SubItems SubItems property.

TreeView Control:-

→ using TreeStrip Control we can display collection of items in tree structure format.

→ Treview is a collection of nodes that means every item will be representing as a node.

→ These nodes are two types

1. Parent node
2. child node

A Parent node can contain multiple child node. as well as a child node can contain multiple sub child nodes.

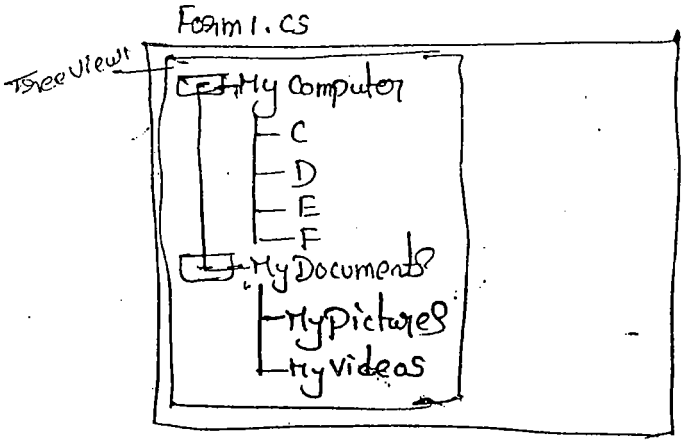
→ we can add the nodes to tree view Control in design time as well as run time.

Properties:-

a) nodes:- It is a collection property, this property is representing collection of the nodes within the tree control.

→ Example to add the nodes to the tree view in runtime.

Step1:- Design of Form1.cs.



Drag & drop Treeview Control

Step 2:- Form1.cs code:-

```
class Form1
{
    void Form1_Load( )
    {
        // parent nodes
        treeview1.Nodes.Add("My Computer");
        treeview1.Nodes.Add("My Documents");
        // child nodes under My computer
        treeview1.Nodes[0].Nodes.Add("C");
        treeview1.Nodes[0].Nodes.Add("D");
        treeview1.Nodes[0].Nodes.Add("E");
        treeview1.Nodes[0].Nodes.Add("F");
        // child nodes under My Documents
        treeview1.Nodes[1].Nodes.Add("My Pictures");
        treeview1.Nodes[1].Nodes.Add("My Videos");

        treeview1.ExpandAll();

    }
}
```

Note:-

→ To add Parent node to the treeview Control we have to use nodes collection property.

→ To add child node we have to use the ~~child~~ index value of the Parent node

ExpandAll():-

It is a predefined member method of treeview class, this method will expand all the nodes of treeview.

⇒ MenuStrip:-

→ Using MenuStrip control we can create a main menu within the windows Form.

→ Menu is a collection of items which are two types.

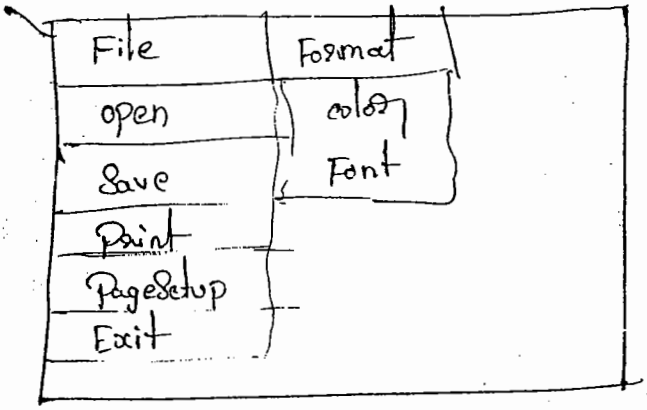
i) Parent Item

ii) child Item

→ A parent item can contain multiple child items and A child Item can contain multiple subchild items.

→ Example for menuStrip control

MenuStrip1 Design of Form1.cs



Step1:- Drag and drop menuStrip control and design like above.

To implement the functionality for above menu we have to depend on dialogue controls

Dialogue Controls:-

Using dialogue control we can open concern dialogue window, In .net windows Forms we have various dialogues control. List

Drag and drop following dialog controls:-

1. OpenFile dialog control
2. SaveFile dialog control
3. Print dialog control
4. PageSetup Dialog control
5. Color Dialog Control
6. Font Dialog Control.

ShowDialog():- It is a Predefined Common method for all dialog Controls This method will open Concern dialog ~~control~~ window.

Syntax:- DialogControlName.ShowDialog();

Default event of Menu Item:-

click:- It is the default of the menu Item, when user will click the Concern menu item this event will fire.

Step2:- Form1.cs Code:-

```
class Form1
{
  void openToolStripMenuItem_click( )
  {
      openFileDialog1.ShowDialog();
  }
  void SaveToolStripMenuItem_click( )
  {
      SaveFileDialog1.ShowDialog();
  }
  void PrintToolStripMenuItem_click( )
  {
      PrintDialog1.ShowDialog();
  }
  void PageSetupToolStripMenuItem_click( )
  {
      PageSetupDialog1.ShowDialog();
  }
  void ExitToolStripMenuItem_click( )
  {
      //this.close();   Application.Exit();
  }
  void ColorToolStripMenuItem_click( )
  {
      ColorDialog1.ShowDialog();
  }
```

```
void fontToolStripMenuItem_Click( )
{
    fontDialog1.ShowDialog();
}
}
}
```

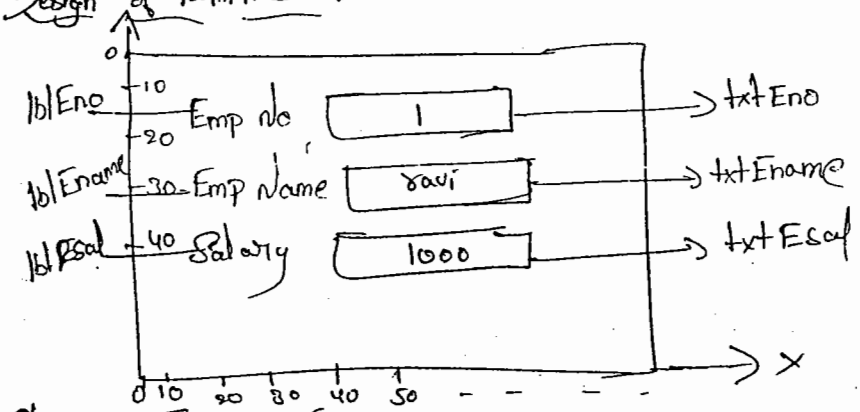
⇒ Application.Exit();

→ Application is predefined class which is part of System.Windows.Forms Base class Library.

→ Exit() is a static member method of Application class, this method will close the executable application.

→ Example to adding the controls to the windows Form in runtime.

Design of Form1.cs :-



Step 1 :- Form1.cs

```
class Form1
{
    void Form1_Load( )
    {
        Label lblEno = new Label();
        lblEno.Text = "Emp No";
        lblEno.Location = new Point(71, 71);
        this.Controls.Add(lblEno);

        TextBox txtEno = new TextBox();
        txtEno.Text = "1";
        txtEno.Location = new Point(141, 71);
        this.Controls.Add(txtEno)
```

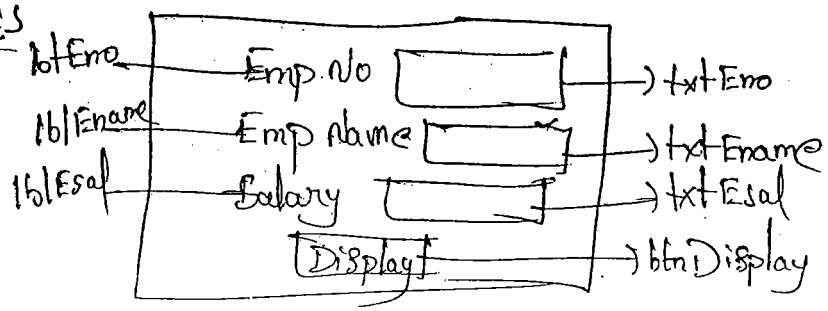
```
Label lblEmpname = new Label();
lblEmpname.Text = "Emp name";
lblEname.Location = new Point(71, 121);
this.controls.Add(lblEname);

TextBox txtEname = new TextBox();
txtEname.Text = "ravi";
txtEname.Location = new Point(141, 121);
this.controls.Add(txtEname);

Label lblEsal = new Label();
lblEsal.Text = "Salary";
lblEsal.Location = new Point(71, 181);
this.controls.Add(lblEsal);

TextBox txtEsal = new TextBox();
txtEsal.Text = "1000";
txtEsal.Location = new Point(141, 181);
this.controls.Add(Esal);
}
}
```

→ this :- It is a C#.net keyword which is representing the current windows Form and current class.

Controls :- It is a Collection Property which is representing collection of controls with in the windows Form.

→ Example to adding the controls in run time as well as defining the button Control click event in run time

Design Form1.cs



5. Add the above control to the windows Form in run time, when user will click the Display button display the textbox values. like above Program.

PictureBox Control :-

→ Using PictureBox control we can display an Image within the windowsForm.

Properties :-

a) **Image :-** for this Property we have to initialize the image which we want to display within the PictureBox. Control.

b) **SizeMode :-**

- Normal
- Strech Image
- AutoSize
- Center Image
- Zoom

→ How to add an Image to the PictureBox in runtime?

Syntax :- PictureBox1.Image = Image.FromFile(<Path of the image>)

Image → property, Image → class, FromFile → method

Image.FromFile( ) :-

→ Image is a Predefined class which is part of System.Drawing BCL

→ FromFile is a Predefined Static member method of Image class, this method will fetch the given Image from give Path and it will return that fetched image.

Windows Services :-

→ windows Service is one of the type of application.

→ This application is operating system dependent application that means windows service will work only on windows operating system.

→ windows Service will start when the windows operating system is booting as well as will run till the operating system is running and windows Service will stop when the windows operating system is shutting down.

→ we can start and stop the windows Service manually.

* If we want to develop a windows Service by using .Net we have to use a seperate project type called windows Service.

→ "System.ServiceProcess" is a base class library for complete windows Service Programming

→ By default windows Service development environment is coming with a class file called "Service1.cs".

→ "Service1.cs" class will come with a single user defined class called Service1, this Service1 class is coming with two override methods.

They are

1. OnStart ( )
2. OnStop ( )

→ 'Service1' is user defined class is coming with a super class called ServiceBase.

After Recording the file has to close.

Requirement 2:- After closing the Time_File.txt windows Service as to create another file called Loop_File.txt with in the D:\\ and it has to record a message continously with in that file till the windows Service is stopped.

D:\\ Loop_File.txt

This message is recorded by windows Service ---
This message is recorded by windows Service --
- - - - - - - - - -
- - - - - - - - -
- - - - - - - -

Requirement 3:- when the windows Service is stop it has to open Time_File.txt file and it has to record windows Service stopped Date & time and like below.

D:\\ Time_File.txt

windows Service is started at: 1-Feb-2013, 8:20pm
windows Service stoped at: 1-Feb-2013, 8:30pm

After recording the above time windows Service has to close the file called Time_File.txt.

Q) In this windows Service we are working with files concept due to that reason we have to import a Baseclass Library called "System.IO"

→ In this we are using thread Concept we have to import System.Threading Baseclass Library.

Step 1:- Developing a windows Service

→ open visual Studio .Net → click New Project → Select language as C# Select windows and right side Select windows Service rename it as mywindowsService location 'E:\' drive.

→ write the below code within Service1.cs file

```
using System.ServiceProcess;
using System.Io;
using System.Threading;
NameSpace mywindowsService
{
    class Service1 : ServiceBase
    {
        override void onStart();
        {
            StreamWriter str = new StreamWriter("D:\\Time_File.txt", true);
            str.writeline("Windows Service Started at :" + DateTime.Now.
                                                        ToString());

            str.close();
            thr1.Start();
```

2 arg Parameterized constructor.

```
        }
//reg task1  Public static void run()
        {
            while (true)
            {
                Streamwriter str = new Streamwriter ("D:\\Loop-File.txt", true);
                str.writeline("this message is recorded by windows Service
                                                    ");
                str.close();
            } //run
```

```
Thread thr1 = new Thread(new ThreadStart(run));
Override void OnStop()
{
    //req1
    //task2
    StreamWriter str = new Streamwrite("D:\\Time_File.txt",true);
    str.WriteLine("windows Service stoped at: "+DateTime.Now.ToString());
    str.close();
    //req2
    //task4
      thr1.Abort();
}
}
```

Build the Solution.

Step2:- Adding the Installers

Enable windows Service Designer window

right click Select Add Installer with this Process two Installer Controls will added. They are

1. ServiceProcessInstaller1
2. ServiceInstaller1

→ Select "ServiceProcessInstaller1" open Properties window change Account as 'LocalSystem'.

→ Select "ServiceInstaller1" open properties window change the StartType as 'Automatic', change Service name as 'MyDotnetService'.

" Display name as "MyDotnet windowsService".

" Description as "This is my Service"

→ ReBuild Solution

Step 3:- Installing windows Service into windows operating System by using Install utility. from Visual Studio Command prompt.

Open ~~cmd~~ VisualStudio Command prompt

change to E drive

C:\~~Program~~ Files\Microsoft.Net\Visual Studio.Net\vc > e: ↵

E:\> cd MyWindowsService\MywindowsService\bin\Debug ↵

E:\MywindowsService\MywindowsService\bin\Debug > installutil -i MywindowsService.exe ↵

The above Command will display following two messages

. The Commit phase Completed Successfully.

. The transacted install has Completed.

The above Command windows Service is installed into windows operating System.

Testing the windows Service:-

Start → Settings → Control Panel → Administrative → Services → double click Services it will display the list of available windows Services in this machine.

→ Select myDotnetwindowsService

double click it will open windowsService Properties window

change Startup Type as 'Manual' click Start Button.

With this process windows Service will start it will create two files within the 'D:\' drive. They are

1. Time_File.txt
2. Loop-File.txt

⇒ **StreamWriter :-**

→ It is a predefined class which is part of System.IO BCL.

→ within this class microsoft defined all the files related predefined constructors, predefined methods & properties ...

→ In the above program when we execute the below statement

```
StreamWriter str = new StreamWriter("D:\\Time_File.txt", true);
```

it will invoke a predefined two parameterized constructor like below

```
class StreamWriter
{
    StreamWriter(String s, bool b)
    {
        ⋮
    }
}
```

s → D:\\Time_File.txt, b → true

→ The above two parameterized constructor will create a file with the given name within the given path, after creating the file it will open the file.

→ If the file is already exist simply it will open the file.

WriteLine():- It is a predefined member method of StreamWriter class this method will write the given message, with text with the opened file

close():- This method will close the file.

⇒ **DateTime.Now :-**

→ Datatime is a predefined structure which is part of System Baseclasslibrary.

→ within this structure Data and time related properties and methods are defined.

Now :- 'Now' is a Predefined member property of DateTime Structure, this Property will return the current date and time.

3) ADO.Net :- C#

→ what is DataBase?

DataBase is a collection of Data, which is representing in table format. Finally DataBase is a collection of tables.

→ what is a table?

Representing the data in rows & columns format can be called as table. Table is a collection of rows, columns and constraints.

→ what is the purpose of DataBase?

To store the user information for future access every application is depend on database.

In General an application can be divided into 2 Parts

1. Front-End Part
2. Back-End Part

1.) Front End Part :-

→ Front end part of the application will accept input from the user as well as it will display output to the user within a application. front end part is directly interacting by the user.

→ Front end part of the application we can develop by using front-end technologies like .Net, Java, php and soon.

In .Net windows Application windows Form is front end as well as " web Application web Form nothing but front end part.

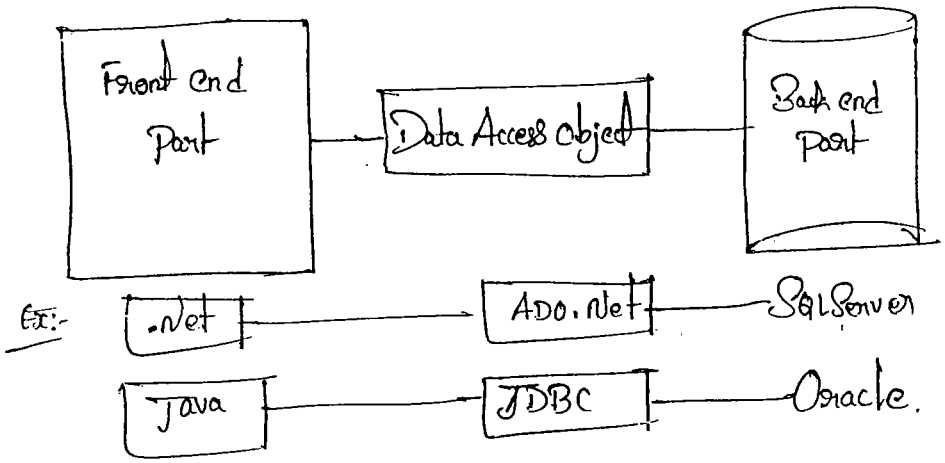
2) Back End Part :-

→ Back end part of the application will store the user information for future access.

→ Back end Part of the application is nothing but database, which we can develop by using back end technologies like SqlServer, Oracle, MySQL, MS-Access and so on.

→ These technologies can be called as RDBMS (Relational DataBase Management System).

→ Diagram for Front end and Back end Part of the application :-



→ Data Access Object :-

→ whenever front end part of the application wants to communicate backend part it requires a mediator that is nothing but Data Access object.

→ Data Access object allows communication b/w front end and back end part of the application.

→ Finally we can say that Data access object is a mediator (or) Middle man between front end part and Back end part.

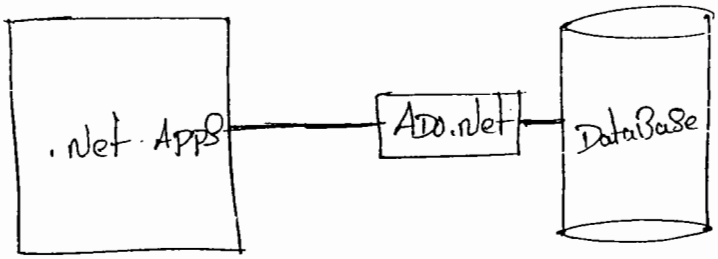
→ Examples for Data Access object are - ADO.Net, JDBC and so on.

ADO.NET :-

→ ADO.Net is part of .Net technology, but it is not a .Net language, but it is .Net Data Access technology.

→ ADO.Net is a Data Access object, which is allowing communication between .Net Applications and DataBases.

→ Finally we can say that ADO.Net is acting as a mediator between .Net Application and DataBase like below.



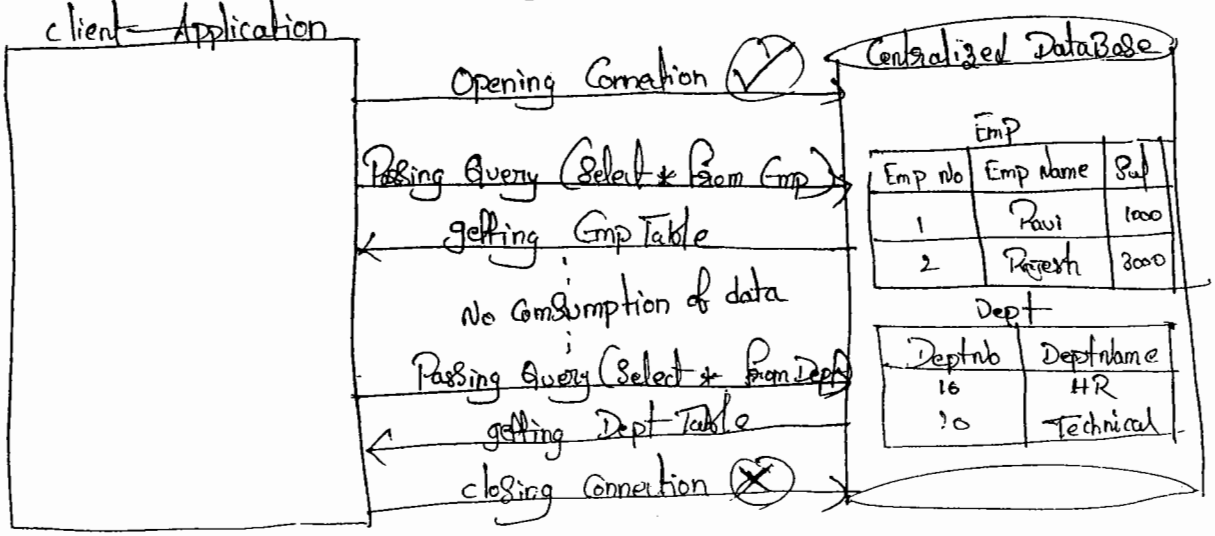
Data Access Architectures :-

→ The way how client application is communicating database is representing by data access architectures.

→ Data access architectures are two types

1. Connected Oriented Architecture
2. Disconnected Oriented Architecture

1) Connected Oriented Architecture :-

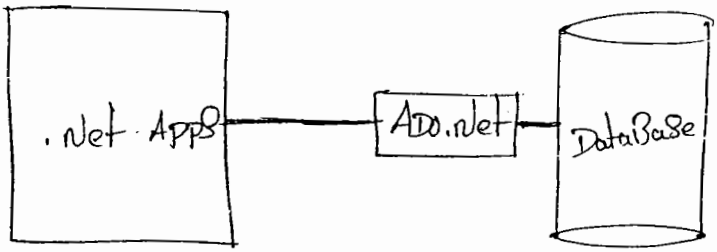


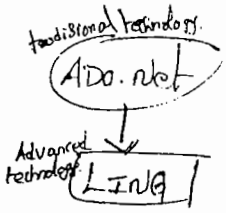
ADO.NET :-

→ ADO.Net is part of .Net technology, but it is not a .Net language, but it is .Net Data Access technology.

→ ADO.Net is a Data Access object, which is allowing communication between .Net Applications and DataBases.

→ Finally we can say that ADO.Net is acting as a mediator between .Net Application and DataBase like below.





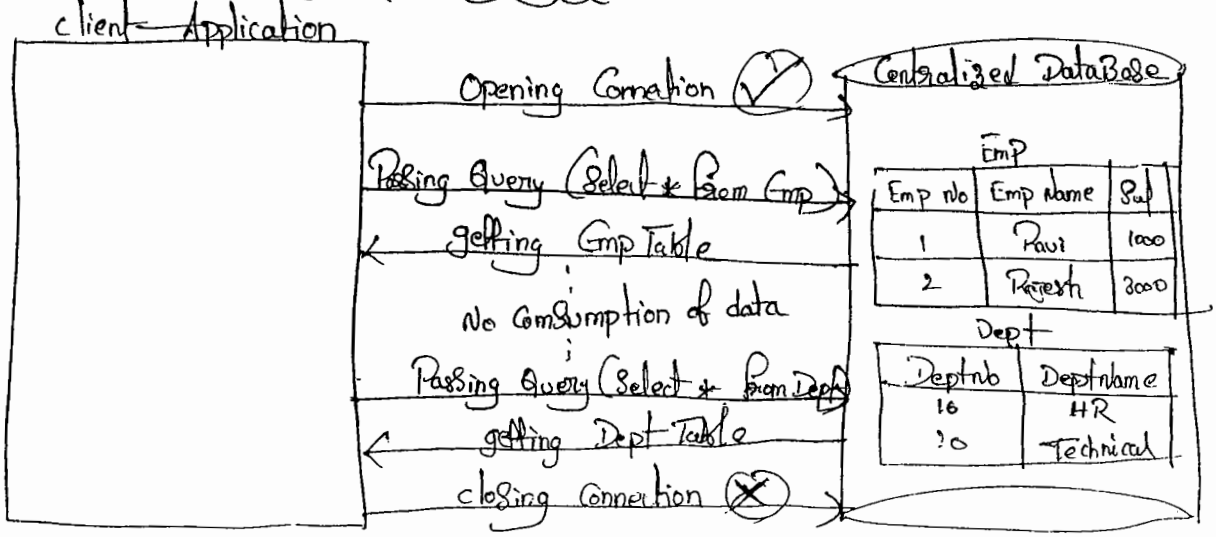
Data Access Architectures :-

→ The way how client application is communicating database is representing by data access architectures.

→ Data access architectures are two types

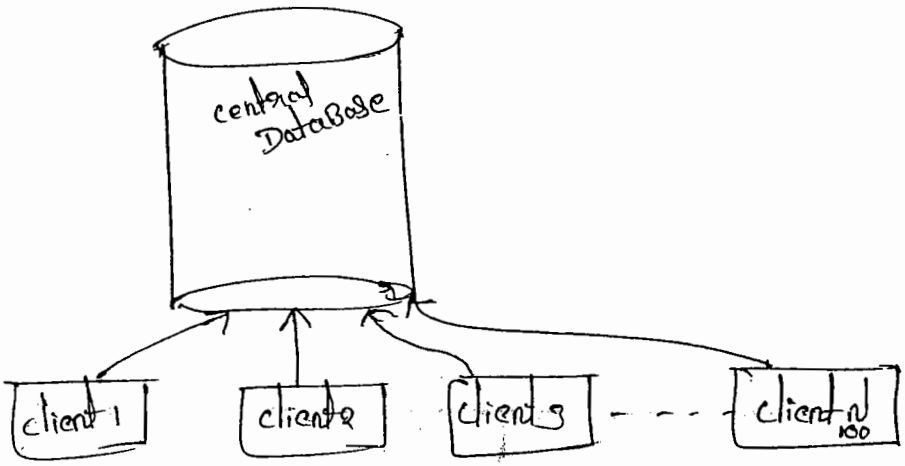
1. Connected Oriented Architecture
2. Disconnected Oriented Architecture

1) Connected Oriented Architecture :-



→ In connected oriented architecture client application will maintain the connection to central database irrespective of data consumption that means it will open the connection at the begining of the application & it will close the connection at the end of the application in b/w eventhough data consumption is not there but still will maintain connection to the central database.

→ Drawbacks of Connected Oriented Architecture:-



→ According to the above diagram 100 clients are connecting to the central database, but in these 100 clients may be 10-20 clients are consuming the data at a time which are called as used connections, remaining 80 clients are not consuming the data but still they are maintaining connection to central database which are called as unused connections.

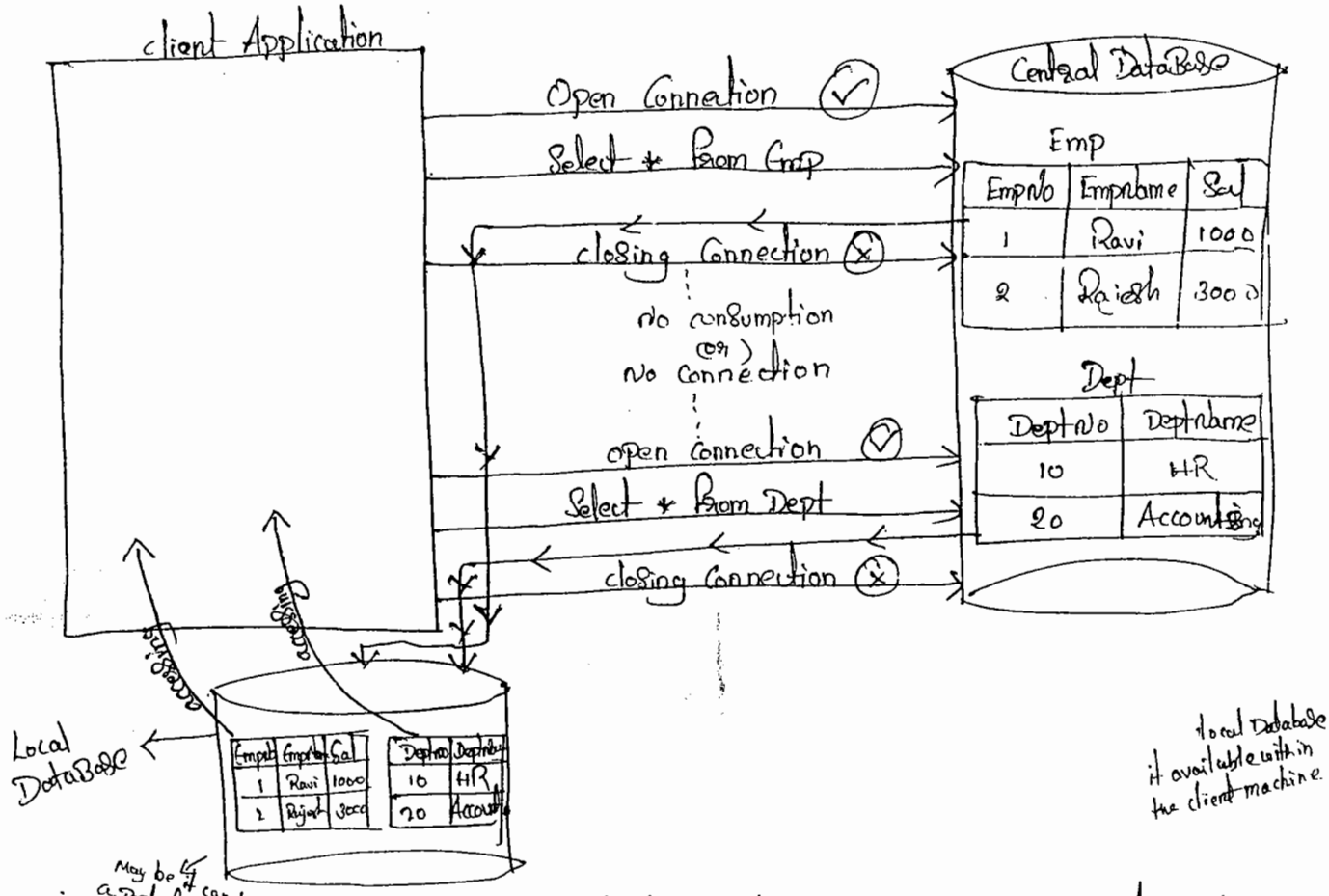
→ Because of these unused connections we may have following problems

1. which will increase burden to database server.
2. which will increase the network traffic.

Finally which will effect the performance of database Server as well as client Application.

To overcome the above drawbacks of connected oriented Architecture we can go for disconnected oriented Architecture.

1 DisConnected Oriented Architecture:-



→ In Disconnected oriented architecture client application will maintain the Connection to the Central Database according to the Data Consumption.

That means when client application required some data it will open connection to central Database and it will get the data from central Database and that fetched data it will fill into local database then immediatly it will close the connection to central Database.

Now client Application will access the data from the local database.

* whenever client Application require some other data from the central database then only it will open the connection to central database to get the data.

→ Local Database will be within the client machine, But central DataBase will be available within the Remote machines but sometimes it will be available within the client machine also in case of Desktop applications (or) windows applications

Advantages:-

→ In Disconnected oriented Architecture only the clients which require the data they will maintain the connection to the central database. By this in disconnected oriented Architecture we can avoid the unused connections. which will reduce the burdden to database server as well as which will reduce the network traffic. By this we can improve the Performance of the client Application.

⇒ Evaluation of ADO.Net:-

→ Before .Net under microsoft family we have 3 popular technologies they are VB, VC++, ASP

⇒ for these technologies microsoft has introduced Data access objects like below

They are DAO (Data Access object)
RDO (Remote Data Access object)
ADO (ActiveX Data access object)

-> These dataaccess objects will support only connected oriented architecture.

-> Due to the drawbacks of connected oriented Architecture microsoft has introduced a new data access object called ADO.Net for .Net technology.

-> ADO.Net will support connected oriented Architecture as well as Disconnected oriented Architecture.

ADO.Net will depend on mainly two Components.

1. Data Provider
2. Data Set

-> 1) Data Provider:-

* -> (1) Data Provider is responsible for Providing the Connection, (2) Executing the Sql Commands within DataBase, (3) it is responsible for to fetch the Sql command results and forward that results towards to the client application to bind to the client application user Interface (or) (4) to fill that fetched data into Local database.

-> Microsoft is providing Various Data Providers to communicate Various DataBases.

-> Every Data Provider Providing by the microsoft as a baseclass Library like below.

System.Data.Sqlclient;
System.Data.oledb;
System.Data.odbc;

→ To Communicate SqlServer Database we can use a data Provider called System.Data.Sqlclient.

→ whenever our application is communicating SqlServer DataBase we have to import data provider like below.

```
using System.Data.Sqlclient;
```

Every → Data Provider will Support 4 objects.

1. Connection object
2. Command object
3. DataReader object
4. DataAdapter object

1). Connection object:-

→ Using Connection object we can open connection, we can maintain the connection as well as we can close connection to the Central database.

→ How to Create a Connection Object:-

whenever we want to create a connection object we have to create an object for a Predefined class called SqlConnection like below

```
SqlConnection Conn = new SqlConnection( );
```

connection object

• SqlConnection:-

Here SqlConnection is a Predefined class which a part of a BCL called System.Data.Sqlclient.

within SqlConnection class microsoft defined all the connection related methods & properties.

-> How to open a connection:-

```
conn.Open();
```

Open():- It is a predefined member method of SqlConnection class. This method will open a connection to Central Database.

-> How to close a Connection:-

```
Conn.close();
```

close():- It is a predefined member method of SqlConnection class. This method will close the connection to Central Database.

Step 2) Command Object:-

-> Using Command object we can execute SqlCommands like select, insert, update, delete and so on within Central DataBase.

-> How to create a command object:-

If we want to create Command object, we have to create an object for SqlCommand predefined class like below.

```
SqlCommand cmd = new SqlCommand();
```

Command object

• SqlCommand:- It is a predefined class which is part of System.Data.SqlClient Base class Library.

within this class microsoft defined all the Command Executing related methods and properties.

3) DataReader object:-

→ using DataReader we can fetch the SqlCommand resulted data and we can bind to the client application user Interface.

→ DataReader we will use in only "Connected Oriented Architecture".

→ DataReader is Read Only, Forward Only and Connected Record Set.

Read only ← it read the only one record
forward only ← next record

How to Create a DataReader object:-

→ whenever we want to Create a DataReader object, we want to create an object for SqlDataReader predefined class like below.

```
SqlDataReader dr = Cmd.ExecuteReader();
```

dr → DataReader object

• SqlDataReader:-

It is a predefined class which is part of System.Data.SqlClient within this class microsoft defined all the data reading related methods & properties.

4) DataAdapter object:-

→ using DataAdapter object we can fetch the SqlCommand resulted data from Central DataBase and can forward that data toward client application finally we can fill that data into local database.

→ DataAdapter is allowing the communication b/w Central Database and local DataBase. Finally we can Data Adapter is mediator b/w CentralDataBase and Local DataBase.

→ DataAdapter object we will in disconnected Oriented Architecture

-) How to create a DataAdapter object:-

whenever we want to create a DataAdapter object we have to create an object for SqlDataAdapter predefined class like below.

```
SqlDataAdapter da = new SqlDataAdapter();
```

da → DataAdapter object.

• SqlDataAdapter:-

It is predefined class which is defined by the microsoft with in a Bcl System.Data.Sqlclient.

within this class microsoft defined all the billing related methods and Properties.

⇒ 2) DataSet:-

-> DataSet is nothing but local Database which will be creating within the client machine.

-> DataSet we will use in only Disconnected Oriented Architecture.

How to create a DataSet:-

whenever we want to create a DataSet we have to create an object for a predefined class DataSet like below.

```
DataSet ds = new DataSet();
```

ds → DataSet object

• DataSet:-

DataSet is a predefined class which is part of System.Data Bcl. That means DataSet is not part of any DataProvider.

-> DataSet is a Disconnected In-memory representation of fetched data from Central DataBase.

→ DataSet will be representing with XML format.

⇒ What are the Components we required to implement Connected Oriented Architecture :-

I. Data Provider
- Connection object
- Command object
- DataReader object

→ what are the Components we required to implement Disconnected oriented Architecture :-

I. Data Provider
- Connection Object
- Command object
- DataAdapter object

II. DataSet.

Imp: when we will implemented Connected Oriented Architecture :-

According to the requirement when we want to fetch the data from Central database and we want to display to user directly then we implement Connected oriented Architecture.

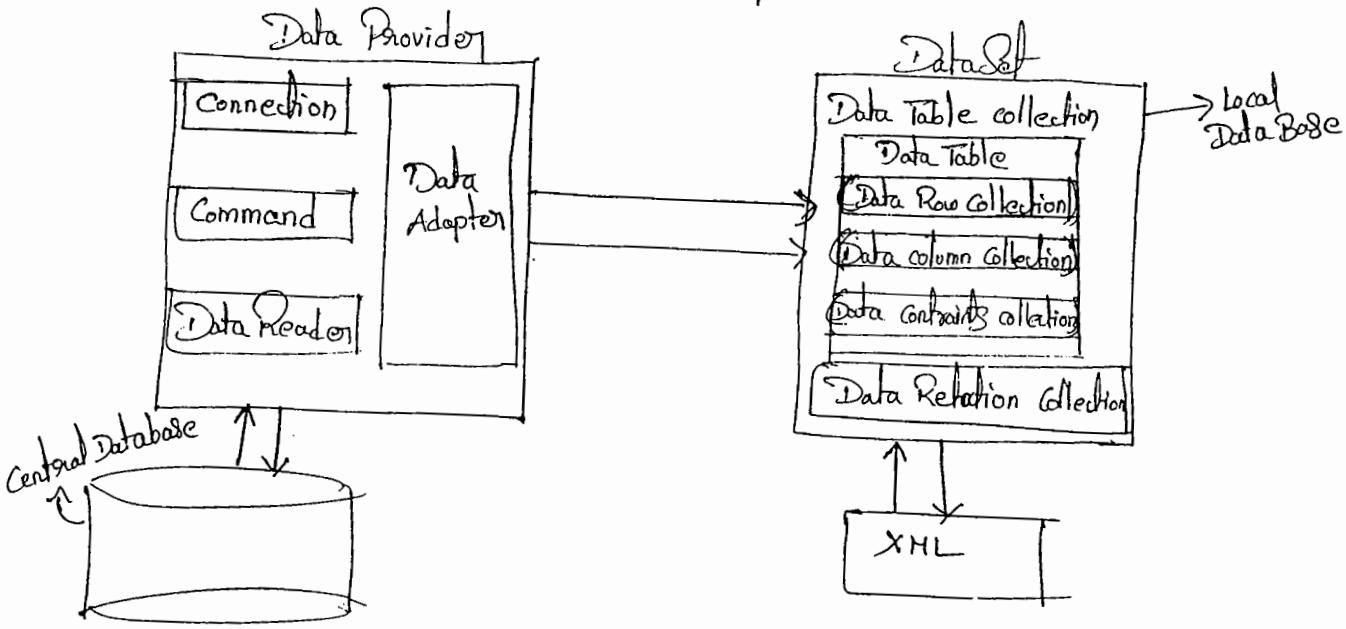
when we will go for Disconnected Oriented Architecture :-

According to the requirement whenever we want to fetch the data from Central database and if we want to perform modifications like Insert, Update, Delete and so on and then we can go for Disconnected oriented Architecture.

ADO.Net Architecture:-

-> ADO.Net architecture is representing the components of .Net

-> Diagram for ADO.Net Architecture.



⇒ DataSet:-

-> DataSet is a collection of Data tables and collection of Data Relations

-> DataTable is a collection of Data rows, Data columns and collection of Data constraints.

⇒ Accessing the SQLServer DataBase by using Disconnected Oriented Architecture in step by step process :-

Step 1:- Importing DataProvider

```
using System.Data.Sqlclient;
```

Step 2:- Defining Connection String

```
String cs = "server = Sathya; database = mydatabase; Uid = Sa;
                                                  Pwd = abc";
```

cs → Connection String Variable

Sathya → Server name (or) System name (or) Machine name

mydatabase → database name

Connection String will have 4 attributes

1.) Server → Here we have to initialize the Server name (or) machine name which is having the SqlServer database

2) DataBase → we have to initialize the database name which we are going to access.

3) Uid → we have to initialize the UId of the SqlServer. By default uid of SqlServer is Sa.

4) Pwd → we have to initialize the password of the SqlServer database.

Step 3:- Creating Connection object using Connection String.

```
SqlConnection Conn = new SqlConnection(cs);

Conn.Open();
```

Step 4:- Defining Command object by initializing the command String and connection object.

```
SqlCommand Cmd = new SqlCommand("Select * from Emp", Conn);
```

↑ Command String

Step 5:- Defining DataAdapter by initializing Command object.

```
SqlDataAdapter da = new SqlDataAdapter(Cmd);
```

Step 6:- Creating local DataBase (or) DataSet object

```
DataSet ds = new DataSet();
```

Step 7:- Filling the fetched data from central DataBase into the local DataBase by using DataAdapter object.

```
da.Fill(ds, "empnew");
Conn.Close();
```

Step 8:- According to the requirement we will implement Insert, update, delete operations to local DataBase.

Step 9:- Binding the Data from DataSet to User Interface Controls. like TextBox, DataGrid View, List-Box and So on.

⇒ Accessing the SqlServer DataBase by using Connected oriented Architecture.

Step 1:- Importing Data provider

```
using System.Data.Sqlclient;
```

Step 2:- Defining Connection String

```
String cs = "Server = Sathya; DataBase = mydatabase; Uid = sa; Pwd = abc";
```

Step 3:- Creating Connection object using Connection String.

```
SqlConnection Conn = new SqlConnection (cs);
Conn.Open();
```

Step 4:- Defining Command object

```
SqlCommand cmd = new SqlCommand ("Select * from Emp", Conn);
```

Command String

Step 5:- Creating DataReader object

```
SqlDataReader dr = cmd.ExecuteReader();
```

Step 6:- Binding the Data from DataReader object to client application User Interface

Step 7:- closing the Connection to Central DataBase.

```
conn.close();
```

⇒ Ado.Net will Support two data access models

1. Wizard Model
2. Programming Model

1) Wizard Model :-

-> To Communicate the database by using wizard model programmer need not to write a single line of code.

-> In Real-time we will not use this wizard model. This mode we will use to test the database connectivity.

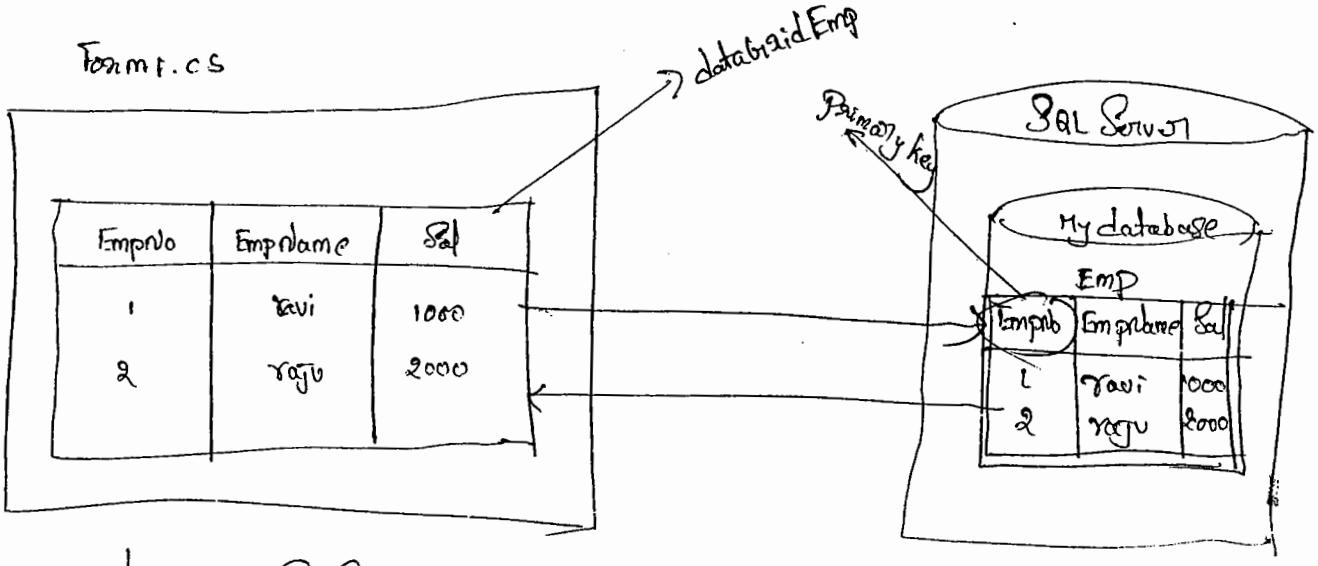
2) Programming Model :-

-> To Communicate the database by using programming model. Programmer has to write some code.

-> This programming model we can implement in 2 ways.

1. By using Disconnected oriented Architecture
2. By using Connected oriented Architecture.

-> Example to Communicate the SqlServer Database by using wizard model.



How to open SqlServer :-

Start -> Programs -> SqlServer 2008 -> SqlServer Management Studio

it will open SqlServer Login Screen like below.

Server type : DataBase Engine

Server Name : Sathya -> Computer Name

Authentication : SqlServer Authentication

run
inetmgr

Login : sa

Password : abc

click → Connect

→ It will open SqlServer Development Environment.

→ This development environment will have mainly two windows.

1. Object Explorer window
2. New Query window

Creating a DataBase :-

→ Open New Query window and execute the below command. and press F5.

```
Create database mydatabase
```

How to enter into our database ?

```
use mydatabase
```

Creating a table :-

```
Create table Emp (Empno int , primary key , EmpName varchar(10) ,
                  Sal int)
```

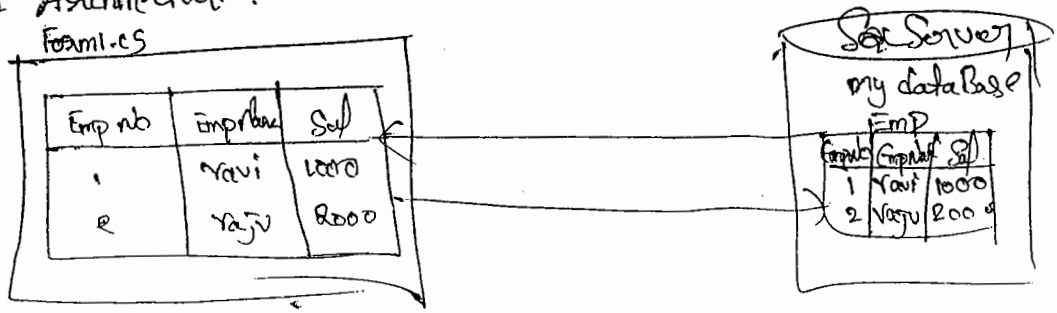
Inserting records into Emp table :-

```
Insert into Emp values (1, 'ravi', 1000)
Insert into Emp values (2, 'raju', 2000)
```

Displaying (or) retrieving the Emp table :-

```
Select * from Emp
```

→ Example to Communicate the SqlServer Database by using Disconnected orientated Architecture.



Form1.cs code :-

```
// Step1
using System.Data.Sqlclient;
namespace Disconnected Example
{
  class Form1
  {
    void Form1_Load (    )
    {
      //Step2
      String cs = "Server = Sathya ; Database = mydatabas ; uid = sa;
                                                pwd = abc";
      // Step 3
    Sqlconnection conn = new Sqlconnection(cs);
      Conn.open ( );
      //Step 4
    Sqlcommand cmd = new Sqlcommand ("Select * from Emp",conn);
      //Step 5
    SqlDataAdapter da = new SqlDataAdapter(cmd);
      //Step 6
    DataSet ds = new DataSet( );
      * // step 7
      da.Fill (ds,"EmpNew");
      Conn.close( );
      // Step 8 Binding Emp table from dataSet to dataGridView
        Control
      dataGridEmp.DataSource = ds.Tables["EmpNew"];
    }
  }
}
```

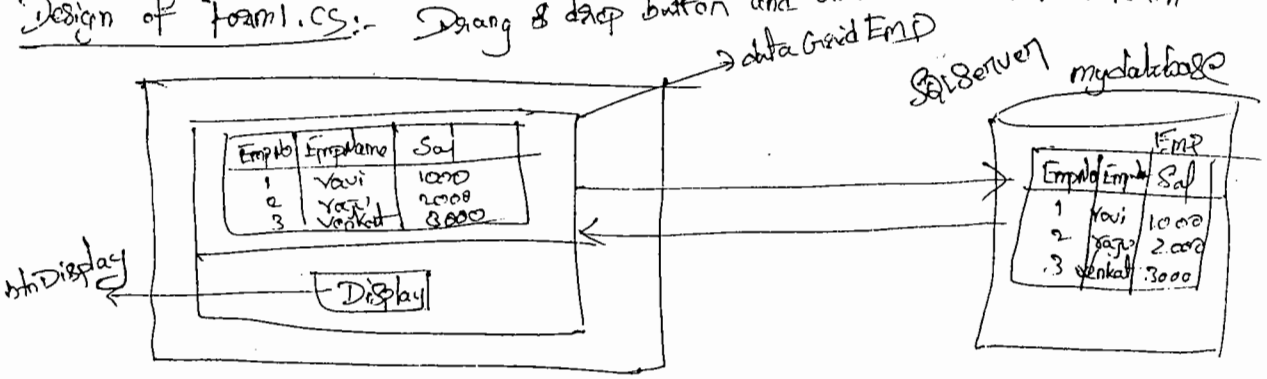
Control db Emp
Local db Emp
Access no problem this is optional

) new table name

) here we can also give of index value of the table '0'

→ Example to communicate the SqL Server DataBase by using Connected oriented Architecture.

Design of Form1.cs:- Drang & drop button and Gridview control on Form



Form1.cs code:-

```
//Step1   using System.Data.SqlClient;
        class NameSpace ConnectedExample
        {
            class Form1
            {   void btnDisplay_click( )
                {
//Step2           String cs = "Server=Sathya;Database=mydatabase;Uid=sa;
                                                    pwd=abc";
//Step3           SqlConnection conn = new SqlConnection(cs);
                Conn.open();
//Step4           SqlCommand cmd = new SqlCommand("Select * from Emp", conn);
//Step5           SqlDataReader dr = cmd.ExecuteReader();
                DataTable dt = new DataTable();
                  dt.Load(dr);
//Step6           dataGridEmp.DataSource = dt;
//Step7           Conn.close();
                }
            }
        }
```

Note:- 1 DataTable:-

→ DataTable is a predefined class which is part of System.Data BCL.

→ Here dt is object of DataTable class which can hold one table data

Load():-

-> Load is a predefined member method of predefined DataTable class.

-> using this method we can load the data from datareader object to ~~data~~ table object.

DataGridView:-

DataSource:- It is predefined member property of DataGridview class for this property we have to initialize the dataSource name from where we are binding the data.

=> Sathya Project:- windows Application day to day transaction:-

Title : Sathya Project
client : Sathya Technologies
Languages : C#.Net 4.0
Tools : Visual Studio.net 2010
Technologies : Windows Forms
RDBMS : SQLServer 2008
Team Size : 1

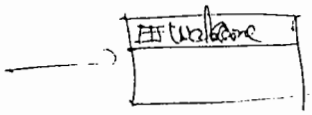
Description:- This is windows application which is computerizing Sathya technologies day to day transactions.

In this application we implemented following modules.

1. Login
2. New User registration
3. welcome
4. Course
5. Faculty
6. Student
7. Details

Step 1:- Open a windows Application rename it as SathyaApp

Step 2:- Rename Form1.cs as welcome.cs

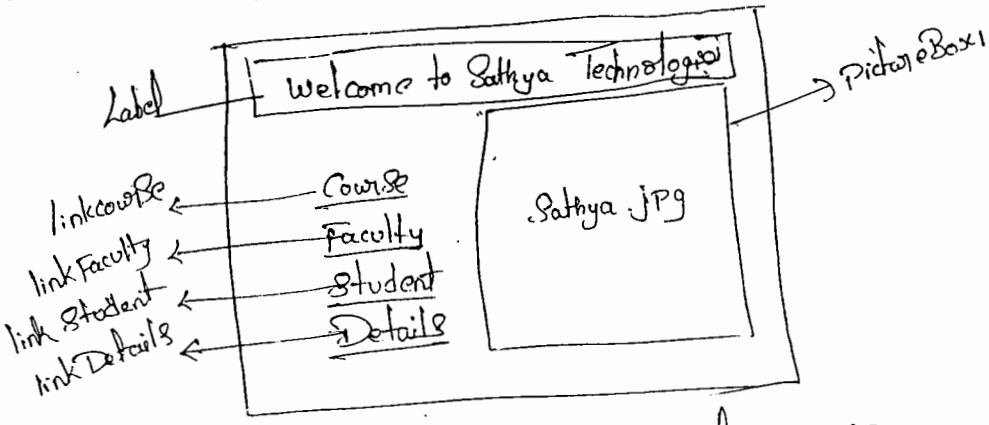


Step 3:- Add Four windows Forms rename them as like below

- Course.cs
- Faculty.cs
- Student.cs
- Details.cs

Welcome Module:-

Step 4:- Design welcome.cs like below



write the below code with within welcome.cs

```
namespace SathyaApp
{
    class welcome
    {
        void linkcourse_click( )
        {
            Course objcourse = new Course();
            objcourse.Show();
            this.Hide();
        }
        void linkFaculty_click( )
        {
            Faculty objFaculty = new Faculty();
            objFaculty.Show();
            this.Hide();
        }
```

```
Void linkStudent_click ( )
{
    Student objStudent = new Student();
    objStudent.Show();
    this.Hide();
}
Void linkDetails_click()
{
    Details objDetails = new Details();
    objDetails.Show();
    this.Hide();
}
}
}
```

Course Module :-

This module is representing the available courses information. In this module we are implementing the following requirements

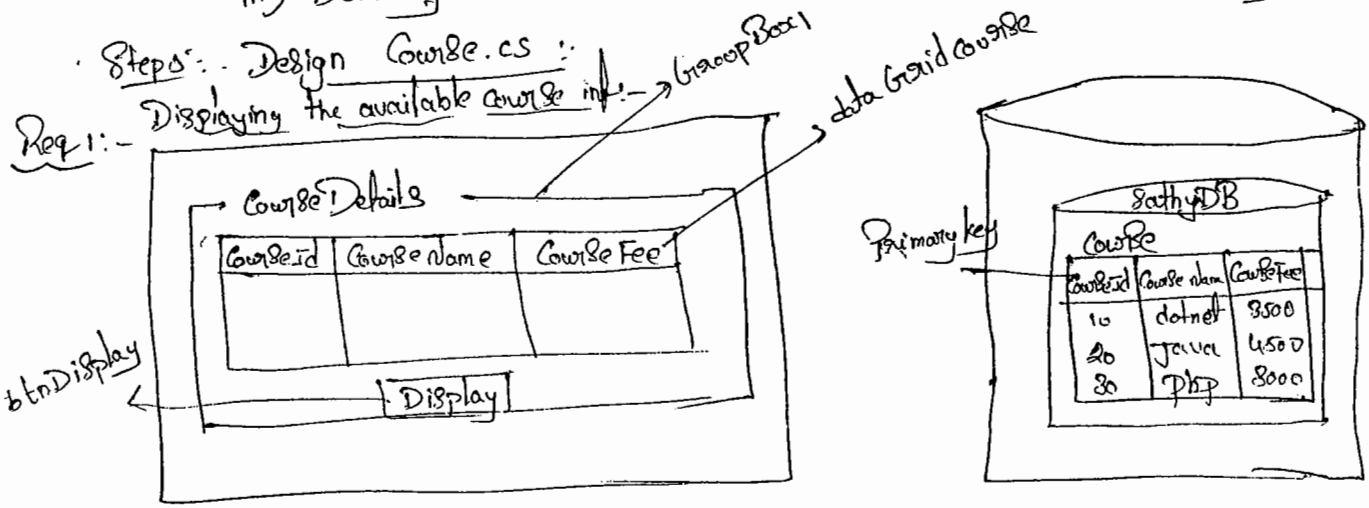
i) Displaying the available course information.

ii) Registering a new course.

iii) Deleting the existing course information.

Steps :- Design Course.cs :

Req 1 :- Displaying the available course info :-



In SQLServer :-

```
Create database SathyDB;
use SathyDB;
Create table Course(CourseId int Primary key, Coursename varchar(10),
                    CourseFee int);
```

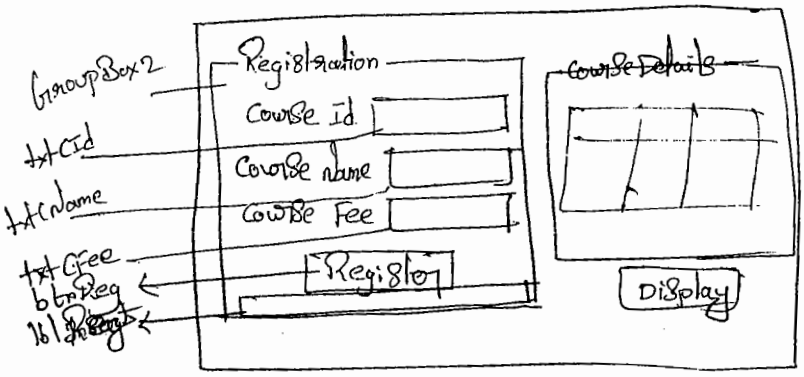
```
insert into Course values (10, 'dotnet', 3500);
insert into Course Values (20, 'Java', 4500);
insert into Course Values (30, 'Php', 3000);
```

→ write the below code within Course.cs

```
//Step1:- using System.Data.SqlClient;
namespace SathyaApp
{
    class Course
    {
        Void btnDisplay_click( )
        {
//Step 2      string cs = "Server=Sathya; DataBase="SathyaDB"; Uid=sa; Pwd=abc;";
//Step3       SqlConnection conn = new SqlConnection(cs);
            Conn.open();
//Step4       SqlCommand cmd = new SqlCommand("Select * from course", conn);
//Step5       SqlDataAdapter da = new SqlDataAdapter(Cmd);
//Step 6      DataSet ds = new DataSet();
//Step 7      ds.Fill(ds, "CourseNew");
            conn.close();
//Step 8      dataGridCourse.DataSource = ds.Tables["courseNew"];
        }
    }
}
```

Server = .

Req.2:- Registering a new course / Design of Course.cs for req 2.




```
Void btnReg_click( )
{
```

```
String cs = "Server = Sathya; database = Course; uid = sa; Pwd = abc;";
SqlConnection conn = new SqlConnection(cs);
Conn.open();
int CId = Convert.ToInt32(txtCId.Text);
String Cname = txtCname.Text;
int CFee = txtCFee.Text;
SqlCommand cmd = new SqlCommand("insert into course values (@CId, @Cname,
                                                            @CFee)", conn);
// assigning the values from variables to parameters
    Cmd.Parameters.AddWithValue("@CId", CId);
    Cmd.Parameters.AddWithValue("@Cname", Cname);
    cmd.Parameters.AddWithValue("@CFee", CFee);
    cmd.ExecuteNonQuery();
    Conn.close();
    txtCId.clear();
    txtCname.clear();
    txtCFee.clear();
    txtCId.Focus();
    lblReg.Text = "Course Registration is Completed Successfully";
}
```

Parameters
@CId
@Cname
@CFee
Variables
CId
Cname
CFee

- Parameters :-
  It is a collection property which is representing the parameters of the command object SqlCommand.

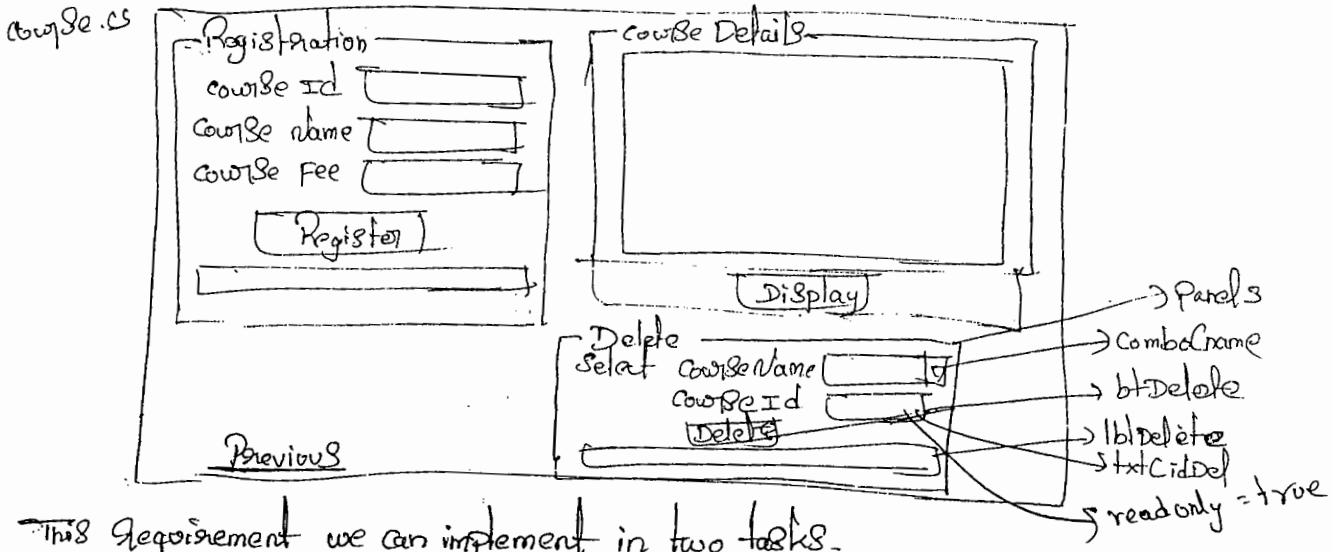
- AddWithValue();
  This method will assigning the value from controls to parameters.

⇒ ExecuteNonQuery() :-
  It is a predefined member method of SqlCommand, this method will execute the command object Non-Query Command.

Query Command → Select | NonQuery Commands are Insert, update, Delete, Create ...

Req3:- Deleting Existing Course Record from the Central database

Design of Course.cs for req3



This Requirement we can implement in two tasks.

→ 1. Binding the CourseName column to ComboBox and Course Id column to TextBox

→ Delete selected course record from the course table.

→ write the below code within Course.cs

```
void Course_Load( )
{
    String cs = "Server = Sathya; Database = SathyaDB; Uid = sa; Pwd = abc";
    SqlConnection Conn = new SqlConnection(cs);
    Conn.Open();
    SqlCommand Cmd = new SqlCommand("select * from Course", conn);
    SqlDataAdapter da = new SqlDataAdapter(Cmd);
    DataSet ds = new DataSet();
    da.Fill(ds, "courseNew");
    Conn.close();
    // binding Coursename column in ComboBox
    ComboCname.DataSource = ds.Tables["courseNew"];
    ComboCname.DisplayMember = "courseName";
    // binding Courseid to textbox
    txtcidDel.DataBindings.Add("Text", ds.Tables["courseNew"], "courseid");
}
```

↓ property of textbox

↓ datasource name

↓ column of the Course table

Note:- change txtcidDel textbox readonly property as (true). (check with enabled)

```
void btnDelete_click ( )
{
    String cs = "Server = Sathya; database = SathyaDB; uid = sa; Pwd = abc; ";
    SqlConnection conn = new SqlConnection(cs);
    Conn.open();
    int cid = Convert.ToInt32(txtcidDel.Text);
    SqlCommand Cmd = new SqlCommand ("Delete Course where CourseId = @cid", Conn);
    Cmd.Parameters.Addwithvalue ("@cid", cid);
    Cmd.ExecuteNonQuery();
    Conn.close();
    lblDelete.Text = "Course record is deleted";
}
Void linkPrevious_linkclicked ( )
{
        welcome objwelcome = new welcome();
        objwelcome.Show();
        this.Hide();
}
```

Validations for Course Module:-

→ txtcid, txtCname, txtCFee should not be empty.

→ txtcid should not allow the duplicate values / Course id should allow digits

★ Handling SqlException (Primary key) from .net Application.

Dotnet
Dotnet
Dotnet
DOTNET
.Net
.NET

→ txtCname should not allow the duplicate Course name

→ txtCFee should allow only digits.

→ DataGridCourse should display the Course details by default when the form is loading with updated (or) latested course information. (DataGridview)

→ Course Name ComboBox and Course id TextBox should display the lastest Course information.

txtcid should not allow the duplicate course id's :-

update the below code within course.cs

```
class course
{
  void btnReg_click( )
  {
      ⋮ // Same as above

    try
    {
      cmd.ExecuteNonQuery();
      lblReg.Text = "Course Registration is Completed Successfully";

    }
    catch(SqlException se)
    {
      messageShow.Show("please enter valid course id");
    }
    finally
    {
      conn.close();
      txtcid.clear();
      txtcname.clear();
      txtcfee.clear();
      txtcid.focus();
    }
}
```

Faculty Module:-

In this module we are maintaining the faculty related information in this module we are implementing the following functionalities.

req:-1 Displaying faculty information.

req:-2 Registering new faculty information.

req:-3 Deleting existing faculty information.

req:-4 updating the faculty contact information.

⇒ Constraints:-

→ Constraint is a Condition which we can assign of single column (or) multiple columns of the table.

→ SqlServer will support various constraints, every constraint will follow some rules and regulations.

Purpose of constraint:-

→ To maintain consistency data we will implement constraints.

Some of the constraints are

① **Primary key:-**

→ Primary key constraint will not allow duplicate values and null values.

② **Foreign key:-**

→ whenever we want to establish relation between two tables we required a column.

→ whenever we are taking the common column it should be the primary key column of the parent table.

→ To establish strong relationship b/w parent table and child table we have to assign foreign key constraint for that child table common column like below.

Course → Parent table

Primary key

CourseId	Coursename	Course Fee
10	dotnet	3500
20	Java	4500
30	Php	3727

child table ← Faculty

Foreign key

Primary key

FacId	Facname	Mobilenumber	Mail Id	CourseId
1	ravi	1234	ravi@gmail.com	10
2	raju	9123	raju@gmail.com	20

Creating child table by assigning the foreign key constraint :-

→ create table Faculty(FacId int primary key, Facname varchar(10), Mobile number int, MailId varchar(10), CourseId int foreign key references course(courseId));

constraint — parent table — column of course table

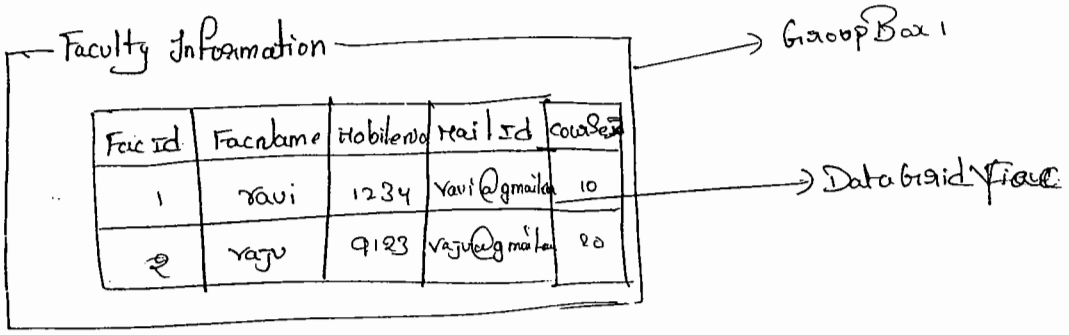
→ insert into Faculty values(1,'ravi', 1234, 'ravi@gmail.com', 10)
insert into Faculty values (2,'raju', 9123, 'raju@gmail.com', 20)

delete course where courseId = 10

The above command will through an error from foreign key constraint because we cannot delete a parent record which is having child records.

req1 :- Displaying faculty information

Design of Faculty.cs :-



FacId	Facname	Mobileno	MailId	courseId
1	ravi	1234	ravi@gmail.c	10
2	raju	9123	raju@gmail.c	20

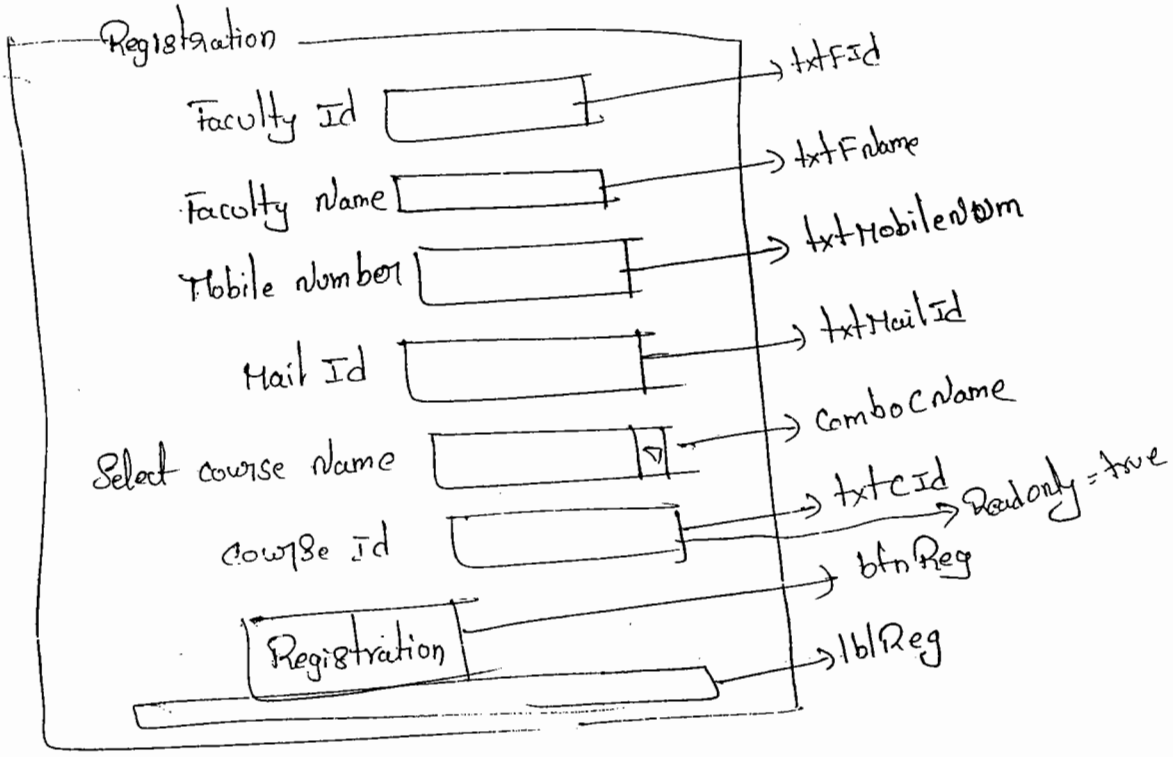
write the below code within Faculty.cs

```
class Faculty
{
    void Faculty_Load( )
    {
        String cs = "Server = Sathya; database = myfacdatabase; uid = sa; pwd=abc";
        SqlConnection Conn = new SqlConnection(cs);

        Conn.open();
        SqlCommand Cmd = new SqlCommand("Select * from Faculty", Conn)
        SqlDataAdapter da = new SqlDataAdapter(Cmd);
        DataSet ds = new DataSet();
```

```
da.Fill(ds,"Faculty new");
Conn.close();
dataGridFac.DataSource = ds.Tables["Faculty new"];
}
```

Req 2:- Registering new faculty information into Faculty table.

Design for Req 2:- of Faculty.cs:-



This we can implement in two tasks

task 1:- Binding the course name column to Combo Box and Course Id Column to text Box.

task 2:- Inserting Faculty information into Faculty table.

write the below within Faculty.cs

```
class Faculty
{
    Void Faculty_Load( )
    {                                   //Req 2 task 1
        String cs
        :
        :
        Conn.Open();
```

```
// Add the below code to the existing code.

    conn.Open();
    SqlCommand cmd~~course~~ = new SqlCommand("Select * from Course", conn);
    da = new SqlDataAdapter(cmd);
    da.Fill(ds, "coursenew");
    Conn.close();
// binding courseName
    combo Cname.DataSource = ds.Tables["coursenew"];
    Combo Cname.DisplayMember = "courseName";
// binding courseId
    txtc id.Databindings.Add("Text", ds.Tables["coursenew"];"courseId");
}

void btnReg_click()
{
    :
    conn.open();
    int fid = convert.ToInt32(txtFid.Text);
    String fname = txtFname.Text;
    int mobinum = Convert.ToInt32(txtMobileNum.Text);
    String MailId = txtMailId.Text;
    int cid = Convert.ToInt32(txtcId.Text);
    SqlCommand cmd = new SqlCommand("insert into Faculty values(@fid, @fname,
                                    @mobinum, @MailId, @cId)", conn);

    cmd.Parameters.Addwithvalue("@fid", fid);
    cmd.Parameters.Addwithvalue("@fname", fname);
    cmd.Parameters.Addwithvalue("@mobinum", mobinum);
    cmd.Parameters.Addwithvalue("@MailId", MailId);
    cmd.Parameters.Addwithvalue("@cId", cId);
    try
    {
        cmd.ExecuteNonQuery();
        lblReg.Text = "Faculty registration is Successfully Completed";
    }
```

```
catch (SqlException se)
{
    MessageBox.Show("please enter valid Faculty Id ");

}
finally
{
    conn.close();
    txtfid.clear();
    txtfname.clear();
    txtMobilenum.clear();
    txtMailId.clear();
    txtfid.Focus();
    Faculty_Load(Sender,e);
}
}
```

Faculty(

Faculty_Load(Sender,)

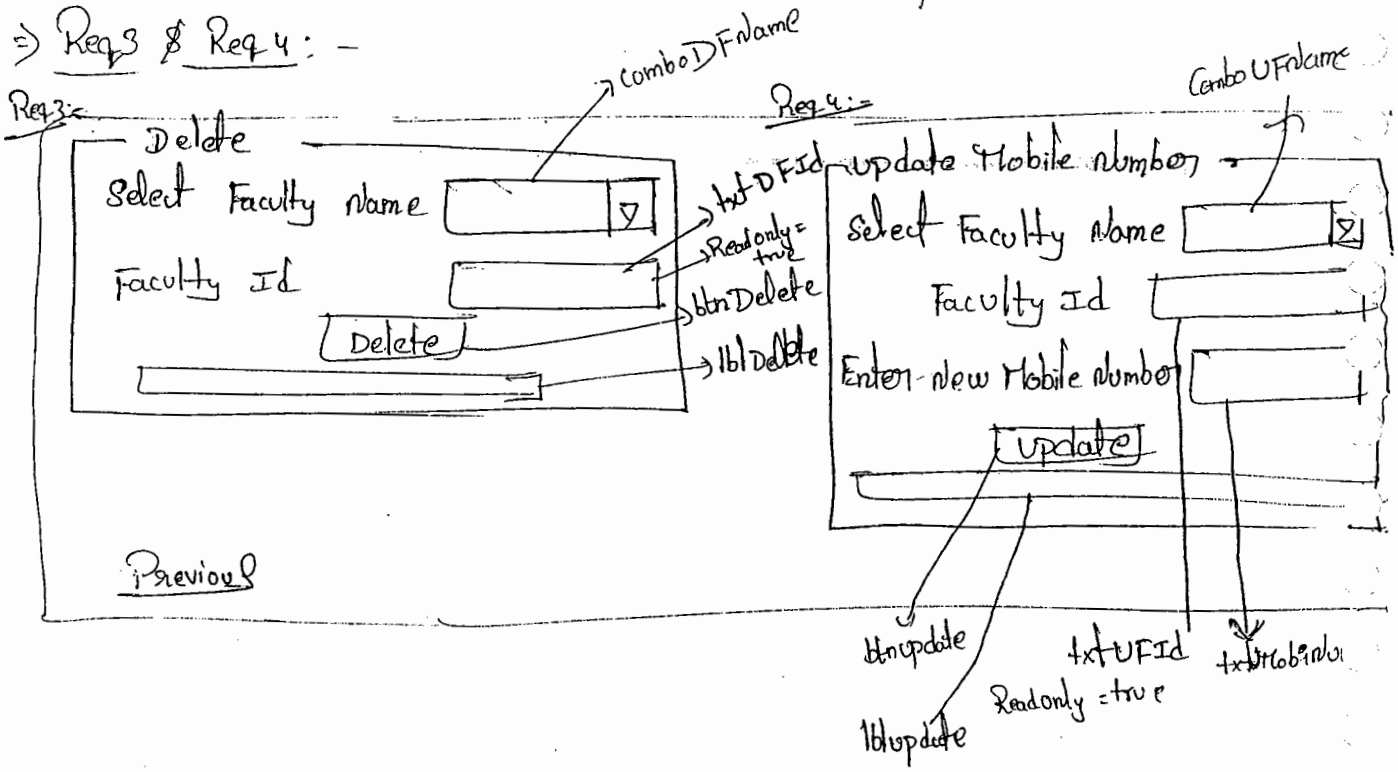
Invoking through the load event.

Note:- Invoking the Load Event

Faculty_Load(Sender, e); ⟶ Programmer invoking

load event parameters name

⇒ Req3 & Req4 :-




```
class Faculty
{
    Void Faculty_Load( )
    {
        Add the below code to Existing code.
```

```
~~Conn.Open();~~ // binding FacultyName in Delete
ComboDFName.DataSource = ds.Tables["Facultynew"];
ComboDFName.DisplayMember = "FacultyName";
// binding FacultyId in Delete
txtDFId.Databindings.Add("Text", ds.Tables["Facultynew"], "FacultyId");
// binding Facultyname in update
comboUFName.DataSource = ds.Tables["Facultynew"];
comboUFName.DisplayMember = "Facultyname";
// binding Faculty Id in update
txtUFId.Databindings.Add("Text", ds.Tables["FacultyNew"], "FacultyId");
}
Void btnDelete_click ( )
{
    :
    Conn.open();
    int fid = Convert.ToInt32 (txtDFId.text);
    Sqlcommand cmd = new Sqlcommand ("Delete Faculty where FId = @fid", conn);
    cmd.parameters.AddwithValue ("@fid", fid);
    cmd.ExecuteNonQuery ();
    conn.close();
    lblDelete.Text = "Faculty name is deleted Successfully";
    Faculty_Load (Sender, e);
}
void btnupdate_click ( )
{
    :
    conn.open();
    int fid = Convert.ToInt32 (txtUFId.Text);
    int Mobinum = Convert.ToInt32 (txtUMobilenum.Text);
    Sqlcommand cmd = new Sqlcommand ("update Faculty set Mobilenum = @mobinum
                                      where FacId = @fid", conn);
    cmd.Parameters.AddwithValue ("@mobinom", mobinom);
```

```
cmd.parameters.AddWithValue("@fid", fid);
cmd.ExecuteNonQuery();
conn.close();
lblupdate.Text = "Faculty Mobile number is updated Successfully";
Faculty_Load(sender, e);
}
}

void linkprevious_linkclicked()
{
    welcome objwelcome = new welcome();
    objwelcome.Show();
    this.Hide();
}
}
```

(In course module when user is deleting a parent record which is having child records program execution will terminate but if exception, handle this exception by displaying a user friendly message)

Validations for faculty module:-

→ txtfid, txtfname, txtMobileNumber, txtMailId textboxes should not be empty

→ txtfid should not allow the duplicate faculty id's..

→ txtfname should allow the digits & should not allow the special character

→ txtMailId should allow the valid expression

→ txtMobileNo should allow only digits and txtMobileNum should allow minimum digits and it should allow maximum 12 digits.

→ DataGridview binding, Delete comboBox bind, update comboBox binding code has to execute for every Load, for every registration, for every delete and update.

→ New Cell Number textbox should not be empty, it should allow only digits, it should allow minimum 10 digits and maximum 12 digits.

Note:- In the above project we have declare connection string multiple times but it is not the right way, whenever we are developing an application we can declare connection string only once globally within the app.config file, which can access by all the windows forms within that application.

Student Module:-

In this module we will maintain the Student information.

-> Creating Student table

Create table Student (Sid int primary key, Sname varchar(20) not null, mobilenum int, MailId varchar(20), CourseId int foreign key reference course(CourseId))

insert into Students values (111, 'John', 1234, 'John@gmail.com', 10);

insert into Student values (222, 'David', 5678, 'David@gmail.com', 20);

Student

Sid	Sname	Mobilenum	MailId	CourseId
111	John	1234	John@gmail.com	10
222	David	5678	David@gmail.com	20

In this module we implementing the following functionalities

Req1:- Displaying the Student details

Req2:- Registering the Student Details

Req3:- Updating Student Mobile number and Mail Id

Implement Student Module with validations.

-> Details Module:-

-> In this module we are implementing 1 to many relations.

-> In SathyaDB database we have 3 tables like below

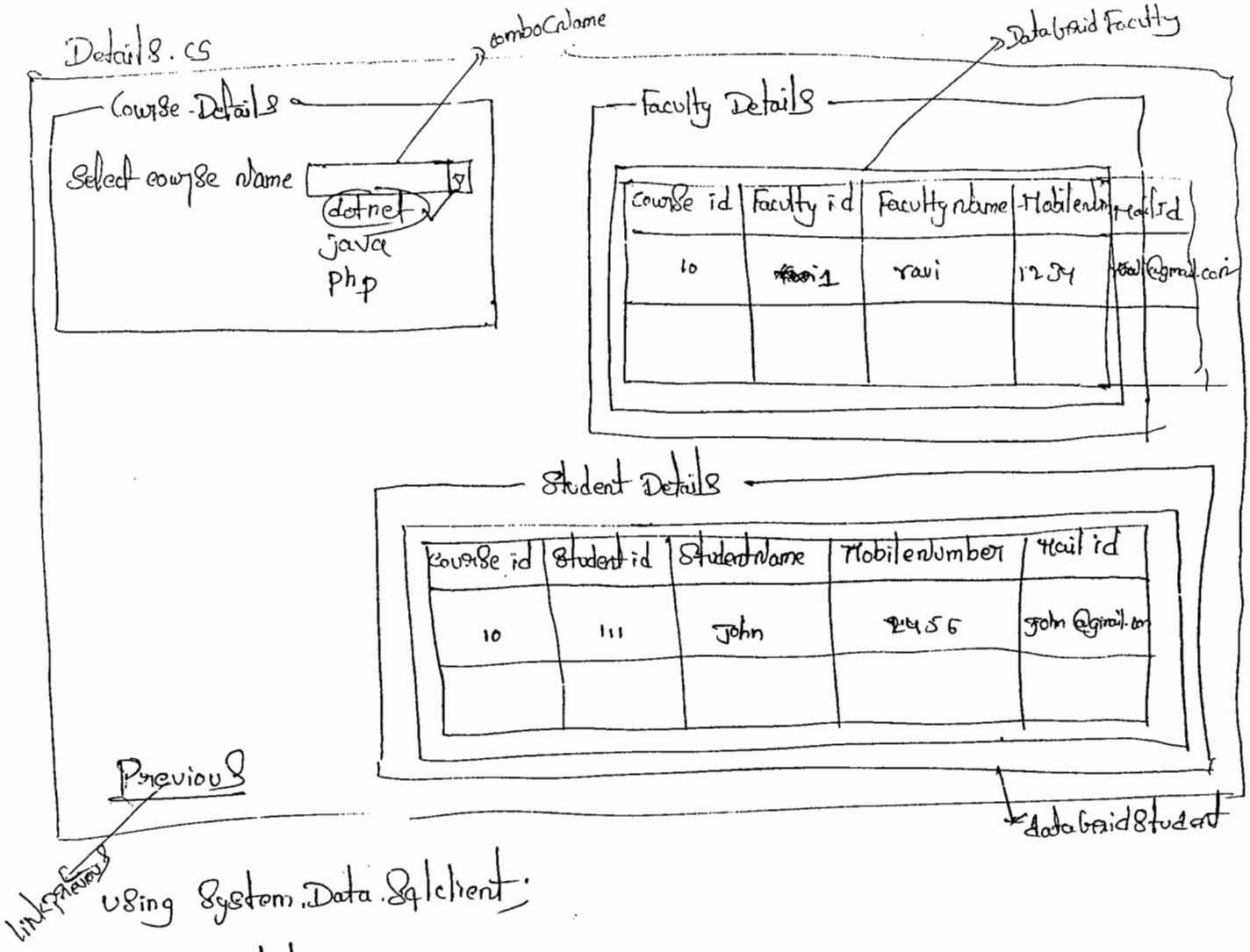
Parent table <- Course

1

child tables <- Faculty

Student -> child table

-> Design of Details.cs



linkPrevious

```
using System.Data.SqlClient;

class Details
{
    void Details_Load( )
    {
        String cs = "Server=Sathya; database=SathyaDB; uid=sa; Pwd=abc";
        Sqlconnection conn = new Sqlconnection (cs);
        conn.open();
        SqlCommand cmdcourse = new Sqlcommand("Select * from course", conn);
        SqlCommand cmdFac = new Sqlcommand("Select * from Faculty", conn);
        SqlCommand cmdStud = new Sqlcommand("Select * from Student", conn);
        SqlDataAdapter dacourse = new SqlDataAdapter (cmd_Course);
        SqlDataAdapter daFac = new SqlDataAdapter (cmdFac);
        SqlDataAdapter daStud = new SqlDataAdapter (cmdStud);
        DataSet ds = new DataSet();
```

→ Implement Hospital DataBase with 3 tables

→ Parent table is Department

→ child tables are 1) Doctors 2) Patients

Department

Deptno	Deptname

Doctors

Did	DName	Mobilenum	Deptno

Patients

Pid	PName	Mobilenum	Deptno

write the code to implement above Database and display the data within the windowsForm based on relations that means when user will select deptname then we have to display the concern dept doctors information and concern Department patient information.

→ Create DataBase Hospital

```
use Hospital
Create table department (deptno int primary key , deptname varchar(20)
                                                    not null);

insert into department values (11, 'Cardiology');
insert into department values (22, 'Nurology');
insert into department values (33, 'orthophidics');

Create table doctors (Doctorid int primary key, Doctorname varchar(20)
                        not null, Mobile num int, Deptno int
                        foreign key reference department(deptno));

insert into doctors values (101, 'Ramesh', 9432, 11);
insert into doctors values (102, 'Rajesh', 9841, 22);
insert into doctors values (103, 'Shekar', 4532, 33);

create table patients (Patient id, ~~Patient name~~ int primary key,
    Patient name varchar(20) not null, Mobilenum int, Deptno int
                foreign key reference department(deptno));
```

```
daCourse.Fill(ds, "Course new");
daFac.Fill(ds, "Faculty new");
daStud.Fill(ds, "Student new");
conn.close();
// binding Course table
// reestablishing relations between Parent and child tables with in dataset.
ds.Relations.Add("courseFac", ds.Tables["course new"].columns["courseId"], ds.Tables["Faculty new"].columns["courseId"])
```

relation name — Parent table — Parent table common column — child table common column

```
ds.Relations.Add("courseStud", ds.Tables["course new"].colums["courseId"], ds.Tables["Student new"].columns["courseId"])
// binding parent table
comboCname.DataSource = ds.Tables["course new"];
comboCname.Displayname = "Course Name";
// binding child table faculty
dataGridFaculty.DataSource = ds.Tables["course new"]; ~~["courseFac"]~~;
dataGridFaculty.~~Displayname~~ DataMember = "courseFac";
```

Relation name

```
// binding child table Student
dataGridStudent.DataSource = ds.Tables["course new"];
dataGridStudent.~~Displayname~~ DataMember = "courseStud";
}
void linkprevious_linkclicked( )
{
    welcome objwelcome = new welcome();
    objwelcome.Show();
    this.Hide();
}
}
```

whenever we are binding the data based on relations from dataset, to bind the Parent table data we have to initialize Parent table name only, to bind child table data we have to initialize for DataSource Property Parent table name and for Datamember property we have to initialize relationname.

```
insert into Patients values (1111, 'dinesh', 1234, 11);
insert into Patients values (2222, 'vijay', 5678, 22);
insert into Patients values (3333, 'naresh', 3489, 33);
```

Department

deptno	deptname
11	Cardiology
22	Nurology
33	Orthophidics

Doctors

Doctorid	Doctorname	Mobile Num	deptno
101	Ramesh	9342	11
102	Rajesh	9841	22
103	Shekhar	4582	33

Patients

Patientid	Patientname	Mobilenum	deptno
1111	dinesh	1234	11
2222	vijay	5678	22
3333	naresh	3489	33

Design of Hospital.cs :-

Department Details

Select Department Name [▽]

Cardiology ✓
Nurology
orthophidics

Doctors Details

Doctorid	Doctorname	Mobile Num	deptno
101	Ramesh	9342	11

Patients Details

Patient id	Patient name	Mobilenum	deptno
1111	dinesh	1234	11

Hospital.cs code:-

```
using System.Data.Sqlclient;
class Hospital
{
  void Hospital_Load( )
  {
    String cs = "Server:Sathya; database=Hospital; uid=sa; Pwd=abc";
    Sqlconnection conn = new Sqlconnection(cs);
    Conn.open( );
    Sqlcommand cmdDept = new Sqlcommand("Select * from department", conn);
    sqlcommand cmdDoctors = new Sqlcommand("Select * from Doctors", conn);
    Sqlcommand cmdpatients = new Sqlcommand("Select * from patients", conn);
    SqlDataAdapter daDept = new SqlDataAdapter(cmdDept);
    SqlDataAdapter daDoctors = new SqlDataAdapter(cmdDoctors);
    SqlDataAdapter dapatients = new SqlDataAdapter(cmdpatients);
    DataSet ds = new DataSet( );
    daDept.Fill [ds, "departmentnew"];
    daDoctors.Fill [ds, "Doctorsnew"];
    dapatients.Fill [ds, "patientsnew"];
    conn.close( );
    // restablishing relations b/w Department table to Doctors and
                                  patients table.

    ds.Relations.Add("deptDoct", ds.Tables["departmentnew"].columns["deptno"],
                        ds.Tables["Doctors new"].columns["deptno"];
    ds.Relations.Add("deptpatients", ds.Tables["departmentnew"].columns["deptno"],
                        ds.Tables["Doctors new"].columns["deptno"];
```

```
ComboDname.DataSource = ds.Tables["departmentnew"];
ComboDname.DisplayMember = "DeptName";

dataGridDoctors.DataSource = ds.Tables["Doctorsnew"];
dataGridDoctors.DataMember = "deptDoct";
dataGridPatients.DataSource = ds.Tables["patients new"];
dataGridPatients.DataMember = "deptPatients";
```

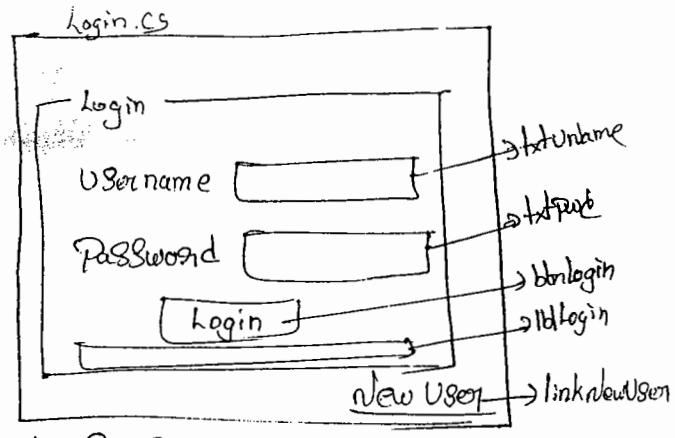
}

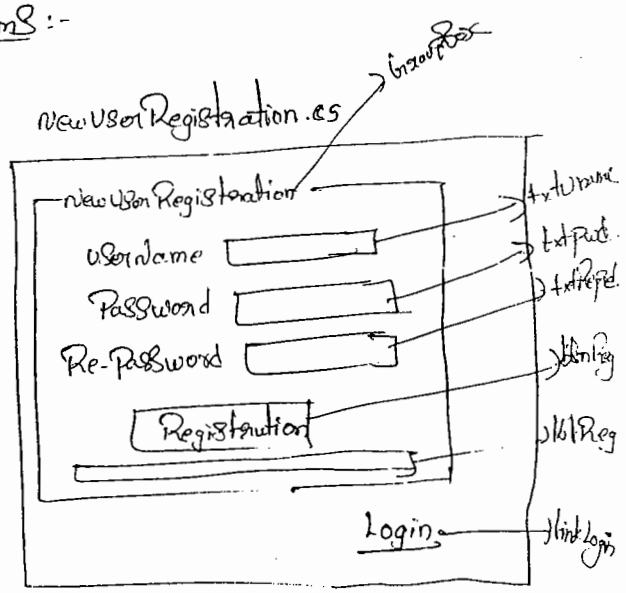
Login Module and New User Registration Module :-

-> Add two New windows Forms rename them as like below.

1. Login.cs
2. NewUserRegistration.cs

Design of Login and New User Registration Forms :-





In SqlServer :-

```
use SathyaDB;
Create table usersList (UserName varchar (20) primary key, Password
                        varchar (20) not null )
```

Write the below code within NewRegistration.cs :-

```
using System.Data.SqlClient;

class NewUserRegistration
{
```

```
void btnReg_click( )
{
    String cs = "Server = Sathya; database = SathyDB; uid = sa; Pwd = abc";
    SqlConnection conn = new SqlConnection(cs);
    conn.open();
    String uid = txtUname.Text;
    String pwd = txtpwd.Text;
    SqlCommand cmd = new SqlCommand("insert into usersList values (@uid,@pwd)",
                                    conn);
    cmd.parameters.AddWithValue("@uid", uid);
    cmd.parameters.AddWithValue("@pwd", pwd);
    cmd.ExecuteNonQuery();
    conn.close();
    txtuid.clear();
    txtPwd.clear();
    txtRepwd.clear();
    txtUid.Focus();
    lblReg.Text = "User Registration completed Successfully";
}
void linkLogin_linkclicked( )
{
    Login objLogin = new Login();
    ObjLogin.Show();
    this.Hide();
}
}
```

→ Select * from usersList

usersList

username	Password
anr	Vanisri
chiru	radha
dotnet	Satya
mahesh	Samantha
ntr	Savithri

→ write a Query to display the no. of records within the userslist table.

Select count(*) from userslist; ⇒ 5

⇒ Count(*):-

→ Count(*) is a predefined Sql function, it will return no. of records within the given table.

→ query to identify username as dotnet and password as Sathya within userslist table.

Select Count(*) from userslist where ~~dotnet~~ username = 'dotnet' and password = 'Sathya'.

Login.cs Code:-

```
using System.Data.Sqlclient;
class Login
{
  void btnLogin_click( )
  {
    :
    conn.open( );
    String uid = txtUname.Text;
    String Pwd = txtpwd.Text;
    Sqlcommand cmd = new Sqlcommand ("Select count(*) from userslist where
                                      username = @uid and password = @pwd ", conn);
    cmd.parameters.AddwithValue ("@uid", uid);
    Cmd.Parameters.AddwithValue ("@pwd", Pwd);
    int i = (int) cmd.ExecuteScalar ( );
    if (i==0)
    {
      lblLogin.Text = "Invalid User";
      txtuid.clear ( );
      txtpwd.clear ( );
      txtuid.Focus ( );
    }
```

C++ style of conversion ref. type to value type i.e unboxing

```
else
{
    welcome objwelcome = new welcome();
    objwelcome.Show();
    this.Hide();
}
conn.close();
void linkNewUser_linkclicked( )
{
    NewUserRegistration objNewUser = new NewUserRegistration();
    objNewUser.Show();
    this.Hide();
}
}
```

→ Design Logout link within following windows Forms

1) welcome.cs
2) Course.cs
3) Faculty.cs
4) Student.cs
5) Details.cs

write the below code within all the above Forms

Validations under NewUserRegistration.cs :-

→ txtUname, txtPwd, txtRepwd should not be empty.

→ txtUname should not allow duplicate usernames.

→ Username should start with letter

→ Pwd should have minimum six character. & one special character.

→ both pwd should match.

→ Validations under login.cs :-

→ txtUname, txtpwd should not be empty.

→

Execute Methods:-

1) ExecuteNonQuery():-

→ ExecuteNonQuery is a predefined member method of SqlCommand class, using this method we can execute the commandObject non query commands like Insert, update, delete and soon.

2) ExecuteReader():-

→ ExecuteReader() is a predefined member method of SqlCommand class.

→ using this method we can create a DataReader object.

→ ~~Execute Sto~~

3) ExecuteScalar():-

→ ExecuteScalar() is a predefined member method of SqlCommand class.

→ This method will execute the command object command till 1st match.

→ whenever we want to execute command object command string till 1st match. as well as whenever we want to skip the unnecessary scanning of the table we can execute particular ~~obje~~ command object command string with the help of ExecuteScalar() method.

10 → Difference b/w DataReader and DataSet?

DataReader	DataSet
→ DataReader we will use in connected oriented architecture	→ DataSet we will use in disconnected oriented architecture.
→ using dataReader we can Read the data.	→ using dataSet we can Store the data.
→ ~~At a~~ time dataReader can represent single record.	→ DataSet can contain multiple tables
→ DataReader can communicate the central database directly.	→ DataSet cannot communicate the central database directly, if ~~yet~~ wants to communicate it will communicate with the help of DataAdapter.

4.) DataReader representing with a predefined class

→ DataReader is a part of every Data-Provider.

→ For example:-

```
NameSpace System.Data.SqlClient
{
    class SqlDataReader
    {

    }
}
```

5.) DataSet is not part of any Data Provider.

→ It is representing with a predefined class DataSet which is part of System.Data Bil.

Ex:-

```
NameSpace System.Dat
{
    class DataSet
    {
    }
}
```

→ Difference b/w ADO and ADO.Net:-

ADO	ADO.Net
1) ADO is a data access object for microsoft traditional technologies like VB, VC++, ASP	1) ADO.Net is a data access object by microsoft Advanced tenchnology called .Net.
2) ADO will support only connected oriented Architecture.	2) ADO.Net with support both connected oriented Architecture and disconnected oriented Architecture.

Deployment :-

→ Deployment is a Process of converting the Complete Source code into Single exe file (or) Setup file and installing that Setup file into the client machine Successfully.

→ Deployment is one of the SDLC phase.

SDLC (Software Development life cycle) :-

→ Software application development process will be called as SDLC.

→ This SDLC will be divided into six phases. They are

Phase 1 :- Requirements Gathering

Phase 2 :- Analysis and Designing

Phase 3 :- Coding (or) Development

Phase 4 :- Testing

Phase 5 :- Deployment

Phase 6 :- User Acceptance Testing (UAT)

Note :- UAT will be performing by the client side people.

→ To create a Setup file for .Net Application .Net Programmer doesn't require to depend on any deployment tools like Install Sheet.

→ Bcz, For .Net developers microsoft is providing a Seperate Project type called Setup and Deployment, using this .Net programmer can create a Setup file for .Net Application.

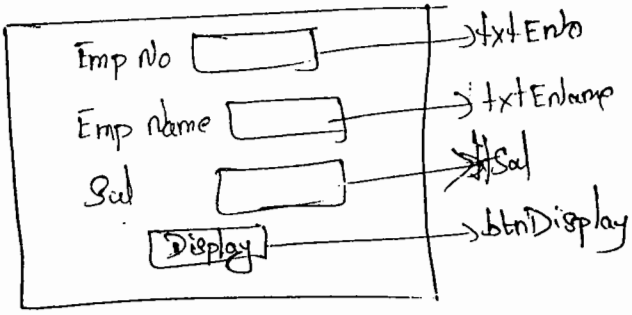
→ Example to Create a Setup file for .Net windows Application and installing that Setup file into client machine in Step by Step Process

Setup

Step 1:- Open a windows application rename it as my windowsclient

Location as E:\ drive.

Step 2:- Design Form1.cs like below.



Form1.cs code

```
class Form1
{
    void btnDisplay_click( )
    {
        txtEmpNo.Text = "123";
        txtEmpname.Text = "Jhon";
        txtSal.Text = "1000";
    }
}
```

Build the Solution and run the application

Step 3:- File → New Project

Expand Other Project Types

- Select Setup and Deployment and Expand it
  - Select Visual Studio Installer
    - Select Setup wizard

rename it as EmpSetup

Select Solution:- Add to Solution

click OK

with this process Setup wizard window will open

→ click Next button it will open Setup Wizard 2 here select like below

✓ ⊙ Create a setup for a windows application

→ click Next button it will open Setup wizard 3 here

click [Add] button and Navigate towards to the My windowsclient Application Path below Path

E:\Mywindowsclient\Mywindowsclient

Note:- here we have Select other than .cs file

→ Select Form1 and Mywindowsclient

click [ok] button

→ click Next

→ click Finish

Step 1: with this process FileSystem on Target Machine window will open

(1) → Here Select Application Folder right click

Add → Assembly

it will open a Select component window

Using Browse Tab navigate towards the windowsclient Application Folder Path like below.

E:\Mywindowsclient\Mywindowsclient\bin\Debug

within this debug Folder Select a file called Mywindowsclient.exe

(2) → Here Select User Desktop Folder right click

Add → Assembly

repeat the above Setup

③ → Select User's Program Menu right click

Add → Assembly

repeat the above Step

Step 5:- open Solution Explorer Select EmpSetup project right click Build. (wait for Build Succeeded. at bottom)

with this process one Setup file will create for our client Application called MywindowsClient within the below path.

E:\EmpSetup\EmpSetup\Debug

within the Debug we can identify a Setup file called ~~like below~~ EmpSetup likebelow.

EmpSetup (windows Installer Package)

Step6:- Double click the above Setup file

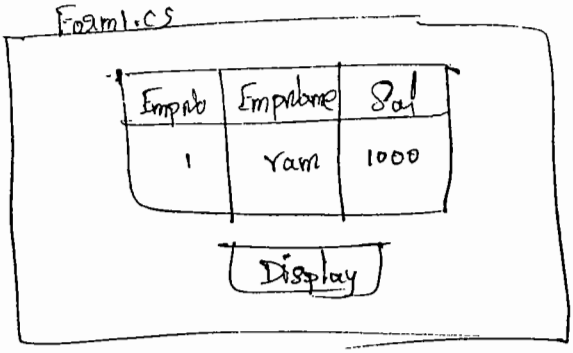
next → Next → Next → close

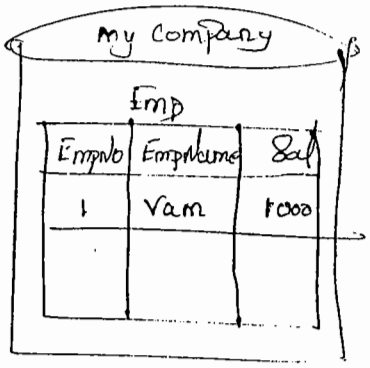
with this process our client Application will be installing.

⇒ Note:- Before implementing this process we have to take the SQLServer Backup.

⇒ Example to Create a Setup for windows application & taking the Backup to database & Restoring that database into client machine, finally

Step 1:- Open a windows Application and design Form1.cs like below and Create the database like below.





Step 2:- write the code for Form1.cs

Step 3:- Open a Setup wizard Project and add to the current client application.

Step 4:- Creating backup file for my company Database.

→ open SqlServer

→ Select my company database rightclick Select tasks and backup it will open backup database window.

→ first click remove button then click add button it will open Select backup destination window.

Here click Browse Button

Here select 'E:\' drive (destination place where we want to keep the backup file) Enter filename as mycompany.bak and next click [ok] button

then [ok]

↳ ok button

→ with this process we will get a message like below that is the backup of database mycompany completed successfully.

-> After this we can check C drive there we can identify a file called mycompany.bak.

Note:- Before installing the Setup file into the client machine we can install SqlServer Express Edition which is free download from microsoft website but if it is not a licensed version.

If we client is a big client then he will go for a licensed version.

After installing the SqlServer Expressed Edition within the client machine we have to restore the Database backup like below.

Steps:- Restoring the database backup:-

-> open SqlServer Express Edition within the client machine.

-> open object Explorer then select databases right click

select Restore DataBase

it will open a restore DataBase window Here select from device

then click Browse button

- it will open Specify Backup window
- click Add button
- It will open a window called Locate Backup File.

Here select the mycompany.bak file from the location.

click ok button

then again click ok button.

-> Again it will open restore database window.

Here select the Restore checkBox then write

To DataBase Name as [mycompany]

then click [ok] button

with this process it will display a message

"Restore database mycompany is completed successfully".

Step 6: Installing the setup file ~~into the~~ client machine.

note:- client machine doesnot require .net s/w but we have to copy the setup file and database backup file towards click location.

LINQ:-

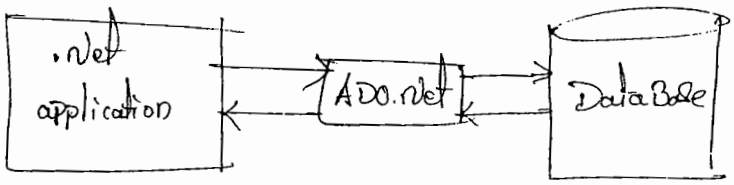
→ LINQ Stands for Language Integrated Query.

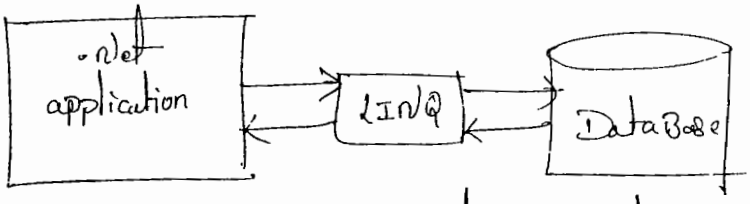
→ LINQ advanced data access object in .NET which was introduced by the microsoft with .NET framework 3.5

→ ADO.NET is traditional Data access object in .NET

LINQ is Advanced Data access object in .NET.

→ Using ADO.NET, a .Net application can Communicate the databases, Similarly using LINQ, we can also a .Net application can Communicate the DataBases.





→ Using ADO.NET to Communicate the databases within the .NET application we will write SQL Commands like Select, Insert, Update, Delete, Create and Soon.

→ Using LINQ to Communicate the databases within the .NET application we don't require to write SQL Commands, these SQL commands we can implement by using any one of .Net language like C#.NET (or) VB.NET

Imp:

→ To retrieve the Data from datasources LINQ is providing one Query which can Called as LINQ query.

Syntax for LINQ Query

from <variable name> ~~int~~ in <datasource> [clauses] select <variablename>

from <variable-name> in <datasource> [clauses] select <variablename>;

- from → keyword
- <variable-name> → user defined
- in → keyword
- <datasource> → where we are getting
- [clauses] → where (or) Having (or) order by
- select → keyword
- <variablename> → user defined

→ To communicate the datasources by using LINQ microsoft is providing one predefined tool called <u>ORTool</u> (Object Relational Tool).

→ This ORTool will be providing one predefined class, that predefined class is having various predefined methods to implement remaining operations on datasources like insert, update, delete and soon.

→ whenever we want to implement insert, update and delete and soon operations within the datasources we have to create an object for ~~ki~~ concern LINQ predefined class and we have to access the concern LINQ class predefined methods and properties.

→ ORTool is treating table as a class and columns as properties.

→ Using LINQ to access a data table we have to create an object for Table class. and using that object we can access the columns.

→ Using LINQ we can communicate the datasources like collections, Databases and XML.

⇒ Example to retrieve the Data from List Collections

In Console Application

emp List

4	20
3	50
2	10
1	30
0	40

```
Void main()
{
    List<int> Emplist = new List<int>(){40,30,10,50,20};
    Var v1 = from eno in emplist Select eno;
    foreach (int id in v1)
    {
        Console.writeline (id);
    }
    Console.Readline();
}
```

op:-
20
50
10
30
40

⇒ Example to retaieve elements from Emplist which are >20.

```
void main()
{
    List<int> Emplist = new List<int>() {40,30,10,50,20};
    Var v1 = from eno in emplist where eno > 20 select eno;
    foreach (int id in v1)
    {   Console.writeline(id);
    }
    Console.Readline();
}
```

⇒ Example to retaieve the elements from Emplist collection which are >20 in order.

```
Void main()
{   List<int> Emplist = new List<int>(){40,30,10,50,20};
    Var v1 = from eno in emplist where eno>20 order by eno
                                        select eno;
    foreach (int id in v1)
    {   Console.writeline(id);
    }
    Console.Readline();
}
```

-> Example to retrieve ~~from~~ elements from Emplist collection which are greater than 20 in descending order.

```
Void main ()
{
    List<int> emplist = new List() {40,30,10,50,20};
    var v1 = from eno in emplist where eno>20 order by eno descending
                                        select eno;
    foreach (int id in v1)
    {   Console.WriteLine(id);
    }
    Console.ReadLine();
}
```

H.W -> Implement all the above programs on Stack collection, Queue collection and dictionary collection by using Linq.

=> Example to communicate the SQLServer DataBase by using LINQ in Step by Step process.

Step①:- Open a windows application rename it as LINQExample1

Step②:- Creating Database and table

Menu => View -> Server Explorer -> Select Data Connections

right click select create new SQLServer DataBase

it will open a window Here write

Server name as Sathya

Select SQLServer Authentication

User name [sa]

Password [abc]

Enter DataBase Name as [MyCompany]

click [ok]

with this process my Company database will created

Select mycompany and Expand Select tables

right click Select add New Table

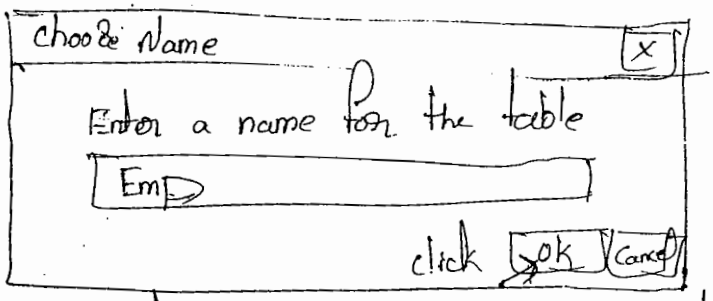
it will open a window Here write like below

primary key ←

ColumnName	Data Type
EmpNo	int
EmpName	Varchar(20)
Sal	int

Note:- Select EmpNo column right-click Set primary key.

click on Save icon it will open a window like below



with this process within mycompany database Emp table is created.

Step (3) :- Adding OR Tool :-

→ Open Solution Explorer Select Project Name (LINQSample1)

right click → add New Item → it will open Add New Item windo.

Here Select a predefined Template called LINQ to SQL classes

rename it as LINQ-toSQL.dbml (dbml → database markup Language)

click Add

→ with this process OR Tool will add to the Solution Explorer

OR Tool is representing with mainly two files.

1) .dbml.layout

2) .designer.cs

1.) .dbml.layout file :-

→ It is representing the designer window of the ORTool..

2) .designer.cs file :-

→ It is representing the C#.net class file

→ This class file will have LINQ related predefined class like below

Structure of .designer.cs :-

Note:→ System.Linq is the main BCL for LINQ Programming.

```
class LinqToSqlDataContext : System.Data.Linq.DataContext
{
}
```

User defined class — Baseclass library — predefined class

Note:→ Here DataContext is a predefined class within this class microsoft defined all the required methods and properties to implement insert, update and delete operations and soon

→ This DataContext is a part of System.Data.Linq BCL.

→ Here LINQ to SQL dataContex is a user defined class which is derived class of DataContext predefined class.

Step 4) Adding Emp table to ORTool designer window (:

→ open Server Explorer. Select Emp table and (Enable ORTool designer window) then drag & drop Emp table on ORTool designer window.

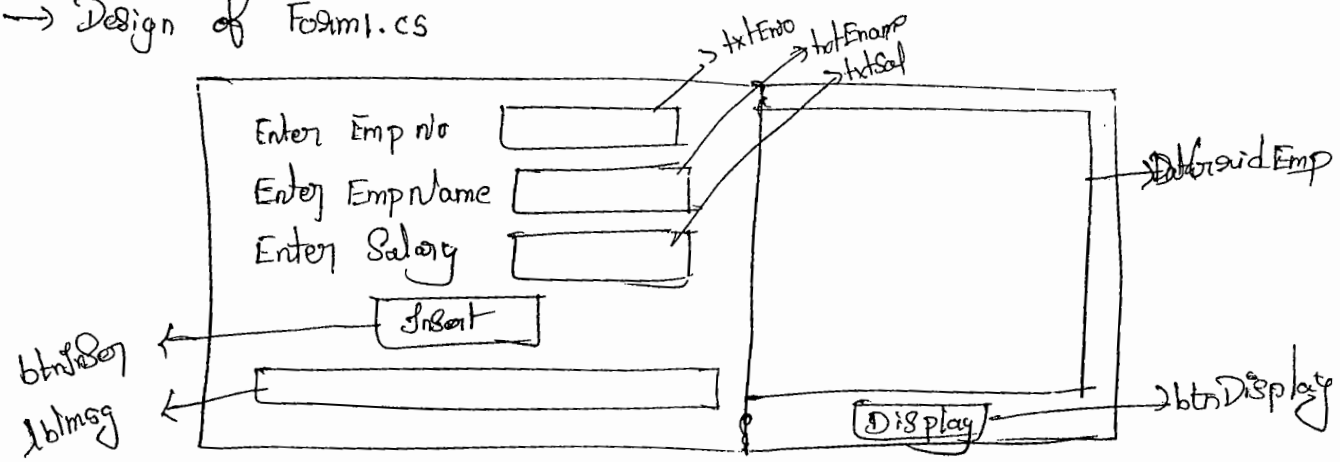
Select [Yes]

→ with this process Emp table image will add on ORTool designer window

After implementing the above the below code will Generate linq to Sql .designer.cs file

```
class ~~LinqToSqlDataContext~~
{
    class Emp
    {
        EmpNo
        {
            get
            {
            }
            set
            {
            }
        }
        EmpName
        {
            get
            {
            }
            set
            {
            }
        }
        Sal
        {
            get
            {
            }
            set
            {
            }
        }
    }
}
```

→ Design of Form1.cs



→ write the below code within Form1.cs

```
NameSpace LINQExample1
{
    class Form1
    {
        LinqToSqlDataContext objLinq = new LinqToSqlDataContext();
        Emp objEmp = new Emp();
        Void btnInsert_click()
        {
            objEmp.EmpNo = Convert.ToInt32(txtEno.Text);
            objEmp.EmpName = txtEname.Text;
            objEmp.Sal = Convert.ToInt32(txtSal.Text);
            objLinq.Emps.InsertOnSubmit(objEmp);
                    ↓
                    property which is representing the Emp table
            objLinq.SubmitChanges();
            txtEmpNo.clear();
            txtEmpName.clear();
            txtSal.clear();
            lblmsg.Text="Record is inserted Successfully";
        }
        Void btnDisplay_click()
        {
            Var v1 = from empnew in objLinq.Emps Select empnew;
            DataGridViewEmp.DataSource = v1;
        }
    }
}
```

Hw → Implement Sathya Project by using LINQ Concept.

→ Currency Manager :-

→ whenever we want to navigate from one record to another record within the table we can use Currency Manager Concept.

→ Currency Manager is a predefined class which is part of System.Windows.Forms Base class Library.

→ whenever we want to implement navigations within a table, we have to initialize Concern table into currency Manager object.

→ Currency Manager class has two important properties like below

a) count b) Position

Note:- whenever currency manager object is pointing to a table, Concern table records will be representing with index values.

1st record index value will be '0'

2nd " " " '1' and so on.

a) Count :-

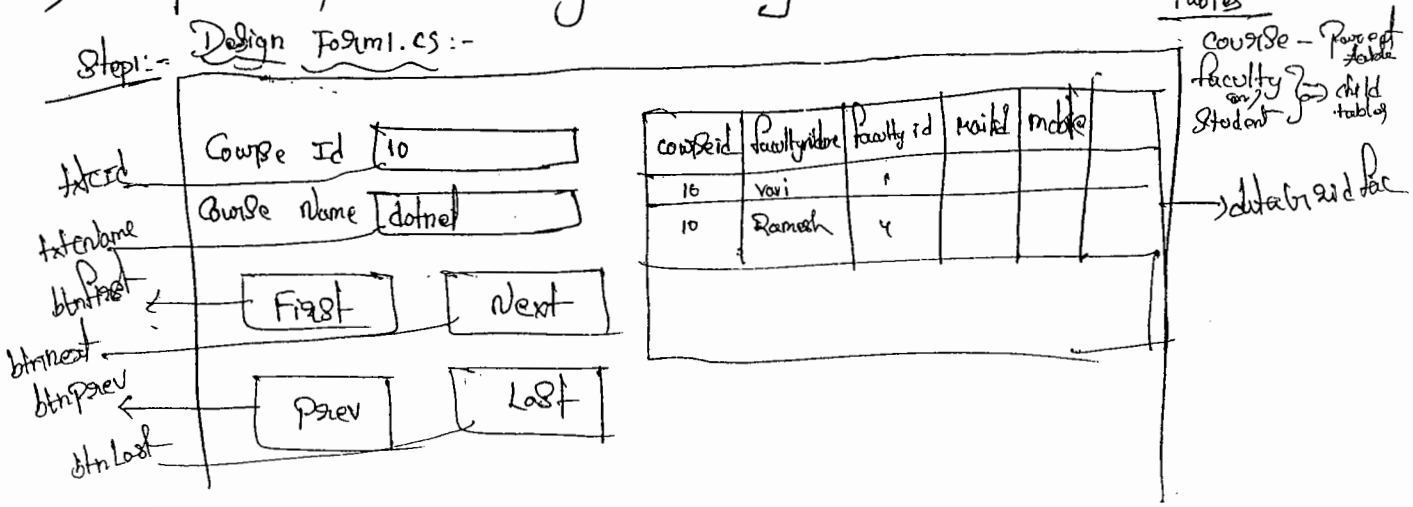
This property is representing the no. of records within the targeted table.

b) Position :-

This property is representing the index value of the record which is pointing by the Currency Manager object.

⇒ Example to Provide a Navigation facility to the user within a table.

Step1:- Design Form1.cs :-



P.Pk

CourseId	CourseName	CourseFee
10	dotnet	8500
20	Java	4500
30	php	3727
40	Sap	1000

P.K F.K

FacId	FacName	MobileNum	MailId	courseId
1	ravi	99123	ravi@gmail.com	10
2	ramu	12345	ramu@gmail.com	20
3	kishore	94234	kishore@gmail.com	30
4	ramesh	83421	ramesh@gmail.com	40

⇒ write the below within Form1.cs

```
NameSpace CurrencyManagerExample
{
    class Form1
    {
        CurrencyManager cm;
        Void Form1_Load( )
        {
            SqlConnection conn = new SqlConnection ("Server = Satya; database = SathyaDB;
                                                    Uid = sa; Pwd = abc; ");

            conn.open();
            SqlCommand cmdcourse=new SqlCommand ("Select * from course ", conn);
            SqlCommand cmdfac = new SqlCommand ("Select * from faculty", conn);
            SqlDataAdapter dacourse = new SqlDataAdapter (cmdcourse);
            SqlDataAdapter dafac = new SqlDataAdapter (cmdfac);
            DataSet ds = new DataSet();
            dacourse.Fill (ds, "course new");
            dafac.Fill (ds, "faculty new");
            conn.close();
//Reestablishing relations in dataset ds.Relations.Add ("courseFaculty", ds.Tables["courseNew"].columns["courseId"],
                                                    ds.Tables ["facultynew"].columns["courseId"]);
//binding (Parent table) course txtId.DataBindings.Add("Text", ds.Tables ["courseNew"], "course id");
```

Parent table primary key column

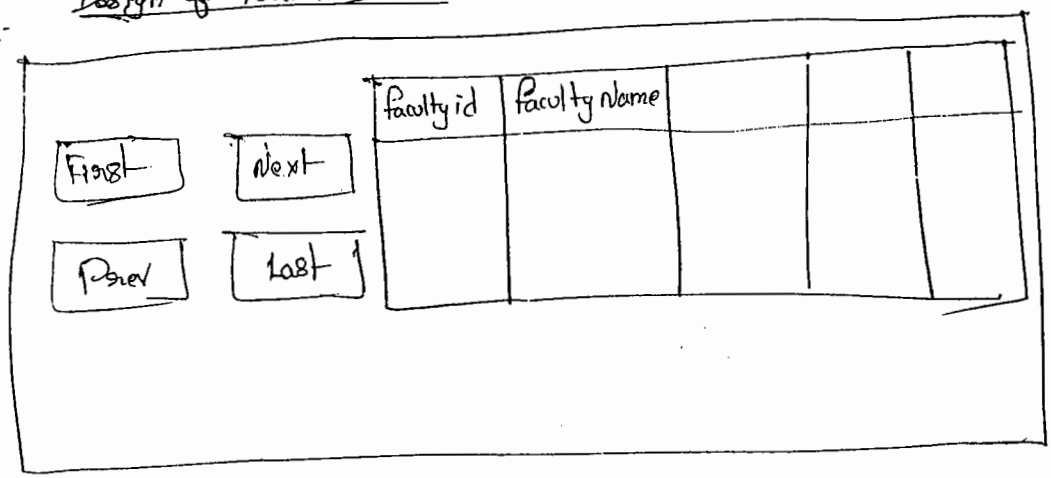
relation name

child table foreign key column

```
txtcname.DataBindings.Add ("Text", ds.Tables["courseenew"], "courseename");
// binding faculty (child table)
dataGridFac.DataSource = ds.Tables["courseenew"];
dataGridFac.DataMember = "CourseFac";
// Initializing course table to CurrencyManager object.
cm=(CurrencyManager)dataGridFac.BindingContext[ds.Tables["courseenew"]];

}
void btnfirst_click( )
{
        cm.position = 0;
}
void btnNext_click( )
{
        cm.position += 1;
}
void btnPrev_click( )
{
        cm.position -= 1;
}
void btnlast_click( )
{
        cm.position = cm.count - 1;
}
}
```

→ Example to implement navigations on faculty table by using DataGridView.

Step:- Design of Form1.cs

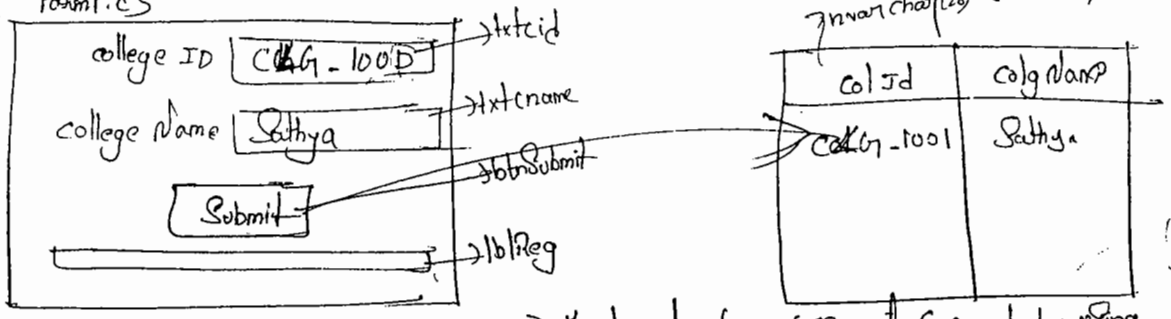


Task: (1).

→ Example to autoGenerate value within the textbox.

College ID automatically to increment don't given chance to user)

Step1:- Form1.cs



⇒ Implement above increment Concept by using SqlServer without using c#.net logic.

```
using System.Data.SqlClient;

class Form1
{
    Sqlconnection Conn = new Sqlconnection("Server = ASHok; database = college; uid = Sa;
                                            pwd = abc;");

    Void Form1_Load( )
    {
        int num = 100;
        conn.open();
        SqlCommand cmd = new Sqlcommand("select count(*) from coollege", conn);
        int i = (int) cmd.ExecuteScalar();
        conn.close();
        i++;
        num = num + i;
        txtcid.Text = "CLG-" + Convert.ToInt32(num);
    }
    Void btnSubmit_click( )
    {
        conn.open( );
        String cid = txtcid.Text;
        String cname = txtcname.Text;
        Sqlcommand cmd = new Sqlcommand("insert into college values(@cid,
                                        @cname)", conn);
            cmd.parameters.Addwithvalue("@cid", cid);
            cmd.parameters.Add withvalue("@cname", cname);
            cmd.ExecuteNonQuery();
            conn.close();
            txtcname.clear();
            lblmsg.Text = "College Submitted Successfully";
            Form1_Load(sender, e);
    }
}
```

Task:-2:-

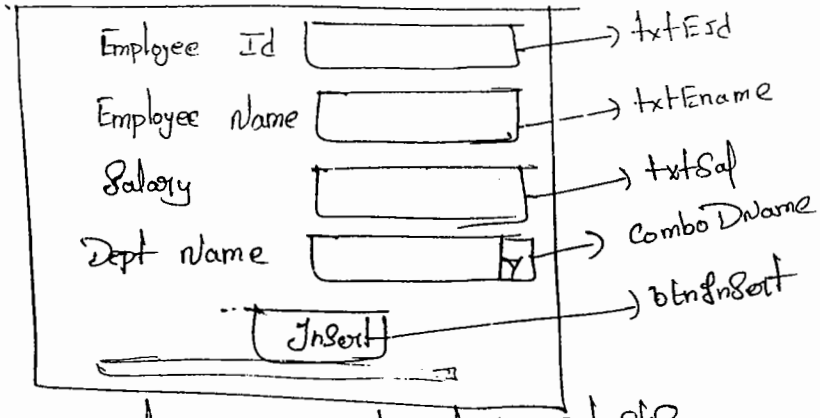
→ Get the DeptId based on DeptName and Store the values into Emp table with DeptId

Dept (varchar)

DeptId	DeptName
d-10	HR
d-11	Technical
d-12	Marketing

Emp (varchar(20)) (money)

EmpId	EmpName	Salary	DeptId
'e-10'	ram	1000	'd-11
e-11	raci	2000	d-12



- This requirement we can implement in 3 tasks

Task 1:- Binding the deptName colomn of the dept table to comboBox.

Task 2:- Getting the deptId from the dept table based on selecting deptname.

Task 3:- Inserting Employee Information i.e EmpId, EmpName, Salary, DeptId into Emp table.

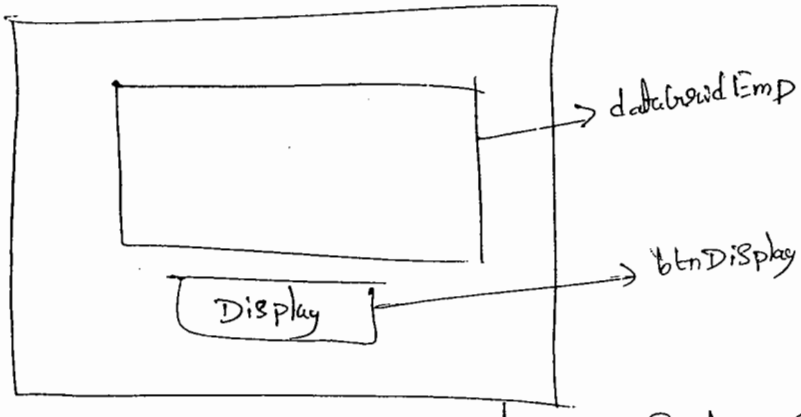
```
class Form1
{
    Void Form1_Load( )
    {
        SqlConnection Conn = new SqlConnection("Server = sathya; database = mydatabase;
                                                uid = sa; pwd = abc;");

        Conn.Open();
        SqlCommand cmd = new SqlCommand("Select * from Dept", conn);
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        DataSet ds = new DataSet();
        da.Fill(ds, "deptnew");
```

```
conn.close();
ComboDname.DataSource = ds.Tables["deptnew"];
ComboDname.Displaymember = "deptname"; (or) ComboDname.DisplayMember = "deptname";
                                            ValueMember = "deptno";
}

Void btnInsert_click()
{
    String cs = "Server = Sathya; database = mydatabase", uid = sa, Pwd = abc";
    SqlConnection conn = new SqlConnection(cs);
    conn.open();
    String dname = ComboDname.Text;
    SqlCommand cmd = new SqlCommand("select deptid from Dept where
                                        deptname = @dname", conn);
    String deptid = Convert.ToString(cmd.ExecuteScalar());
    String eid = Convert.ToString(txtEId.Text);
    String ename = txtEname.Text;
    int sal = convert.ToInt32(txtSal.Text);
    SqlCommand cmd1 = new SqlCommand("insert into Dept values (@eid, @ename,
                                        @sal, @deptid", conn);
        cmd1.Parameters.AddWithValue("@eid", eid);
        cmd1.Parameters.AddWithValue("@ename", ename);
        cmd1.Parameters.AddWithValue("@sal", sal);
        cmd1.Parameters.AddWithValue("@deptid", deptid);
    cmd1.ExecuteNonQuery();
    conn.close();
    Form1.Load(sender, e);
    lblInsert.Text = "Inserted Record";
    txtEId.clear();
    txtEname.clear();
    txtSal.clear();
}
```

```
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
da.Fill(ds, "deptnew");
                 Tablename
!
String deptid = ds.Tables["deptnew"].Rows[0][0];
                                      rowindex value   columnindex value
```

Task 3:- Binding the two tables data into data Grid View Control (the two tables are dept, emp)

--> Example to declaring Connection String within the app.Config.



Step 2:- Add a application Configuration file to Solution Explorer.

Step 3:- write the below code within app.config file.

```
<?xml version="1.0"
<configuration>
    <appSettings>
        <add key="cs" value="Server=.;database=mydatabase;uid=sa;
                                                pwd=abc;" />

    </appSettings>
</configuration>
```

Step 4:- webForm1.cs

```
using System.Configuration;
using System.Data.SqlClient;
NameSpace GlobalconnectionExample
{
    class webForm1
    {
        void btnDisplay_click( )
        {
            SqlConnection conn = new SqlConnection(ConfigurationSettings.AppSettings
                                                    ["cs"].ToString());

            Conn.open();
            SqlCommand cmd = new SqlCommand("Select * from Emp", conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
```

```
da.Fill(ds,"Empnew");
conn.close();
dataGridEmp.DataSource = ds.Tables["Empnew"];
}
}
```

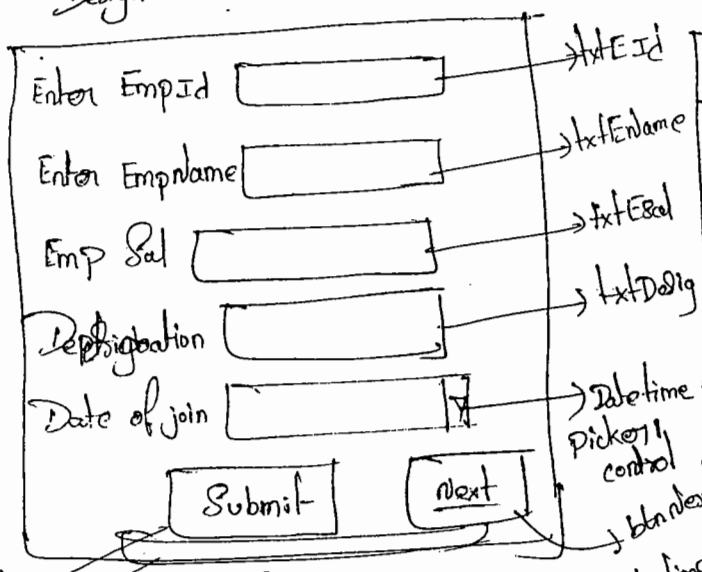
ConfigurationSettings:-

It is a predefined class which is part of System.Configuration base class Library.

AppSettings:- It is a Property member of configuration Settings class.

→ Example to Insert Employee Information into Emp Table with Joining date and Display the Employee Information based on Selected Joining date implement with the help of DateTimePicker Control

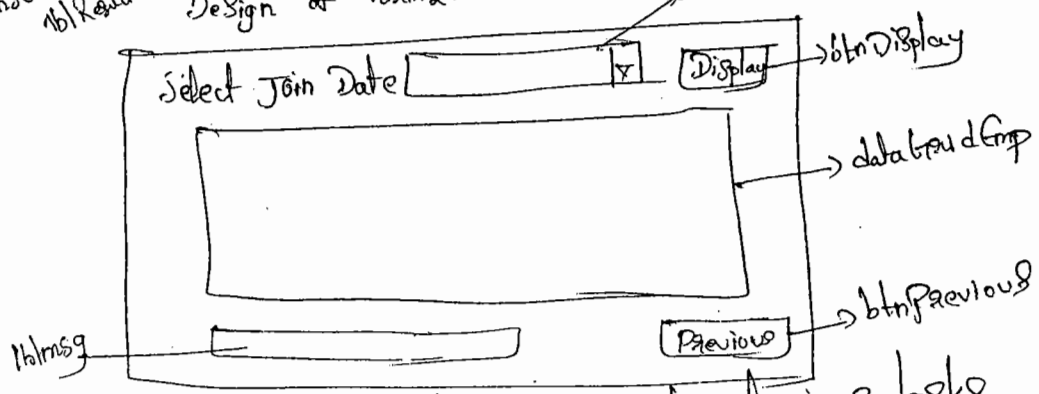
Design of Form1.cs



tbl_Empdetails

EmpId	Empname	EmpSal	Designation	DOJ
101	Sairam	5000.00	HR	..

Design of Form2.cs



This requirement we can implement in 3 tasks

→ Task1 :- Insert the Employee information into tbl_Empdetails table

Task2 :- when the Form2 loading bind the tbl_Empdetails data into datagrid View.

mreddyfech Suppo

Task 3:- Display the selected Date joining Employee Information.

Form1.cs :-

```
using System.Data.SqlClient;
namespace DateTimeExample
{
  class Form1
  {
    void btnSubmit_click( )
    {
      SqlConnection Conn = new SqlConnection("Server=Satya;database=Sathyadatabase;Uid=sa;
                                              Pwd=abc;");

        Conn.open();
      int eno = Convert.ToInt32(txtEmpId.Text);
      string ename = txtEname.Text;
      double esal = Convert.Todouble(txtEsal.Text);
      string desig = txtDesig.Text;
      string Doj = dateTimePicker1.Text;
      SqlCommand cmd = new SqlCommand("insert into tbl_Empdetails values(@eno,@ename,
                                        @esal,@desig,@Doj)", conn);

        cmd.Parameters.AddwithValue("@eno", eno);
        cmd.Parameters.AddwithValue("@ename",ename);
        cmd.Parameters.AddwithValue("@esal", esal);
        cmd.Parameters.AddwithValue("@desig", desig);
        cmd.Parameters.AddwithValue("@Doj", Doj);

      cmd.ExecuteNonQuery();
      conn.close();
      lblResult.Text = "Record is inserted";
    }
    void btnnext_click( )
    {
        Form2 objF2 = new Form2();
        objF2.Show();
        this.Hide();
    }
  }
}
```

Form2.cs:-

```
using System.Data.SqlClient;
NameSpace DateTimeExample
{
    class ~~WebForm2~~
    {  ~~Form2_Load()~~
       SqlConnection conn = new SqlConnection("Server=Satya; database=Sathyadatabase; uid=sa;
                                                Pwd=abc;");

       SqlCommand cmd;
       SqlDataAdapter da;
       DataSet ds;

       Form2_Load()
       {
           conn.open()
           cmd = new SqlCommand("Select * from tbl_empdetails emp", conn);
           da = new SqlDataAdapter(cmd);
           ds = new DataSet();
           da.Fill(ds, "emp");
           conn.close();
           dataGridEmp.DataSource = ds.Tables["emp"];
       }
       btnDisplay_click()
       {
           conn.open()
           String Date = Datetimepicker1.Text;
           cmd = new SqlCommand("Select * from tbl_empdetails ~~where DOJ~~ = @Date", conn);
           ~~da = new SqlDataAdapter(cmd);~~ cmd.parameters.~~Add~~WithValue("@Date", Date)
           ~~ds = new DataSet();~~
           ~~da.Fill(ds,~~ "
           SqlDataReader dr = cmd.ExecuteReader();
           if(dr.HasRows)
           {   while(dr.Read())
               {   dataGridEmp.DataSource = dr;
               }
           }
           else
           {   dataGridEmp.Visible = "false";
               lblmsg.Text = "No Records available";
           }
           conn.close();
       }
    }
```

correct

```
da = new SqlDataAdapter(cmd);
ds = new DataSet();
da.Fill(ds, "empnew");
if(ds.Tables["empnew"].Rowcount > 0)
{
    dataGridEmp.DataSource = ds.Tables["empnew"];
}
else
{
    lblmsg.Text = "No Employees joined in this Date";
}
```

```
void linkPrev_linkClicked()
{
        Form1 objF1 = new Form1();
        objF1.Show();
        this.Hide();
}
```

→ Example to update the Salary within emp table and insert the given Salary into Salary table.

Deptno	Deptname
10	HR
11	Technical

Dept

Empno	Empname	Sal	Deptno
111	Ravi	10000	10

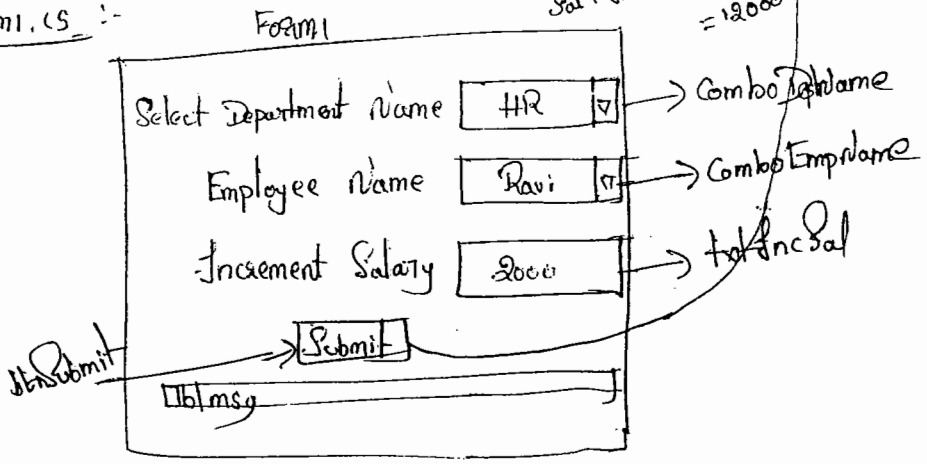
Emp

SalId	IncSal	Empno
Sal-01	2000	111

Sal

Update Sal to Sal + IncSal = 10000 + 2000 = 12000

Design Form1.cs :-



code for Form1.cs :-

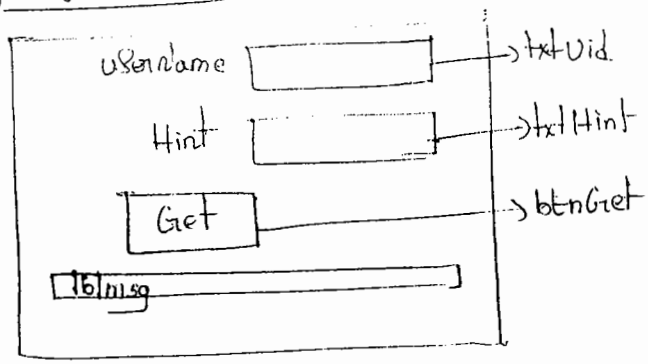
```
class Form1
{
public void binder()
{
class Form1
{
    SqlConnection Conn = new SqlConnection("Server=.; database=Employee;
                                            Uid=sa; Pwd=abc");
```

```
Public Void bind()
{
    Conn.Open();
  SqlCommand Cmd=new SqlCommand("Select Deptname from Dept", Conn);
  SqlDataReader dr=cmd.ExecuteReader();
  while(dr.Read())
  {
    ComboDeptname.Items.Add(dr[0].ToString());
  }
  Conn.close();
}

Void Form1_Load()
{
  bind();
}
Public String AutoGenerateSalId()
{
    Conn.open();
  SqlCommand cmd=new SqlCommand("Select count(*) from Sal", Conn);
  int i=(int) cmd.ExecuteScalar();
  Conn.close();
  i++;
  String SalId="Sal"+i;
                (sal-)
  return SalId;
}

Void ComboDeptname_SelectedIndexchanged()
{
  ComboEmpname.Items.clear();
  Conn.Open();
  String dname=ComboDeptname.Text;
  SqlCommand cmd=new SqlCommand("Select DeptId from Dept where Deptname=
                                  '"+dname"'", Conn);
    int dno=(int) Cmd.ExecuteReader();
  SqlCommand cmd1=new SqlCommand("Select Empname from Emp where
                                  Deptno=@dno", Conn);
    Cmd1.Parameters.AddWithValue("@dno", dno);
  SqlDataReader dr=cmd1.ExecuteReader(cmd1);
```

```
    while (dr.Read())
    {
        ComboEmpname.Items.Add(dr[0].ToString());
    }
    conn.close();
}

    int eno;
void ComboEmpname_SelectedIndexchanged()
{
    Conn.open();
    String ename = ComboEmpname.Text;
    Sqlcommand Cmd = new Sqlcommand("Select eno from Emp where Empname = @ename",
                                    conn);
    int eno = (int)Cmd.ExecuteScalar();
    Conn.close();
}

Void btnSubmit_click().
{ String Salid = AutoGenerateSalId();
  Conn.open(); intIncSal = Convert.ToInt32(txtIncSal.Text);
  Sqlcommand Cmd = new Sqlcommand("Insert into Sal values (@Salid,(@IncSal,
    (@eno); update Emp Sal = Sal + @IncSal where Empno = @eno ", conn);
    Cmd.Parameters.Addwithvalue("@Salid", Salid);
    Cmd.Parameters.Addwithvalue("@IncSal", IncSal);
    Cmd.Parameters.Addwithvalue("@eno", eno);
    Cmd.ExecuteNonQuery();
    Conn.close();
    comboDeptname.Text = " ";
    comboEmpName.Text = " ";
    txtIncSal.Text = " ";
    lblmsg.Text = "Salary Inserted Successfully";
}
```

→ Example for Forgot Password.

Design of Form1.cs :-



→ Primary key

tbl_Pwd

Uname	Pwd	Hint
Kalyan	pulsar	bike
Praveen	Hp	Laptop
Suresh	Sureshi23	myname

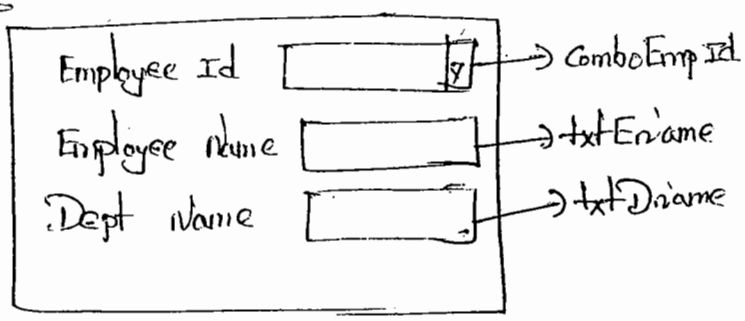
Code for Form1.cs :-

```
class Form1
{
  SqlConnection Conn = new SqlConnection("Server=Sakya; database = Sakyadb; uid = sa;
                                          Pwd = abc ");

  void btnGet_click()
  {
    String Uname = txtUid.Text;
    String hint = txtHint.Text;
    conn.open();
    SqlCommand cmd = new SqlCommand("Select pwd from tbl_pwd where uname =@uname
                                     and Hint =@hint ", conn);
      cmd.Parameters.AddwithValue ("@uname", uname);
      cmd.Parameters.AddwithValue ("@Hint", hint);
    SqlDataReader dr = cmd.ExecuteReader();
      String Pwd = " ";
      int i=0;
    if (dr.HasRows)
    {
        while (dr.Read())
        {
          Pwd = dr[0].ToString();
          i++;
        }
    }
    if (i==1)
    {
      lblmsg.Text = "your Password is " + Pwd;
    }
```

```
else
{
    lblmsg.Text = "please enter valid username and hint ";
}
Conn.close();
}
```

→ Example to get the DeptName, Employee name based on Selected EmployeeId of Employees.

Design Form1.cs :-



Task1:- Binding EmpId Column to the ComboBox.

Task2:- Getting Employee name and DeptId columns of Emp table Dept table based on Selected Employee id.

Task3:- Assign the collected Deptid into variable called Deptid.

Task4:- Getting Deptid.

Primary key Dept

DeptId	DeptName
10	HR
20	Technical
30	Marketing

Emp

EmpId	EmpName	DeptId
101	Ashok	10
102	Kumar	10
103	Rajesh	20
104	Ramu	10
105	Ganesh	30

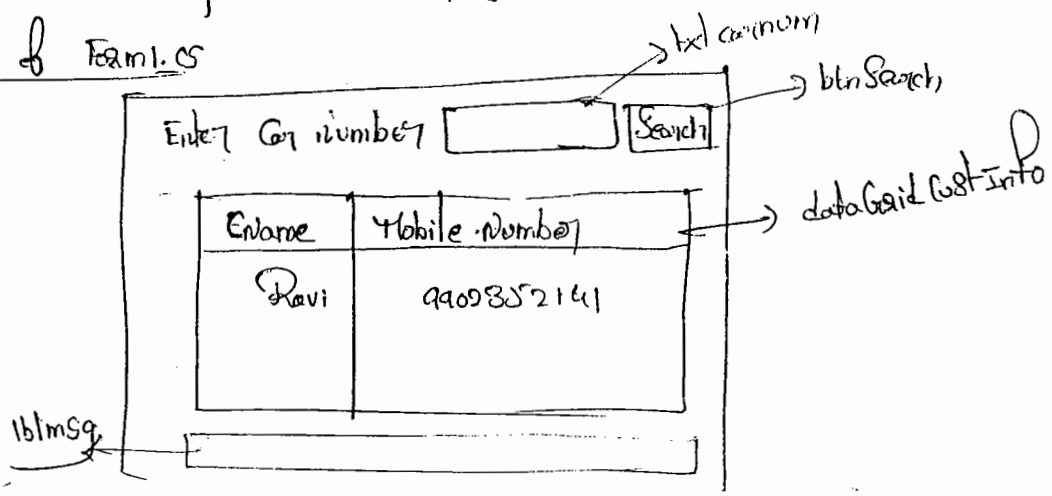
Code For Form1.cs :-

```
class Form1
{
    SqlConnection Conn = new SqlConnection("server=Satya; database=Company;
                                            uid=sa; Pwd=abc");

    void Form1_load()
    {
        Conn.Open();
```

```
SqlCommand cmd = new SqlCommand("Select Empid from Emp ", conn);
SqlDataReader dr = cmd.ExecuteReader();
while(dr.Read())
{
    comboEmpid.Items.Add(dr[0].ToString());
}
dr.close();
conn.close();
}
Void ComboEmpid_SelectedIndexchanged()
{
    Conn.open();
    int eid = Convert.ToInt32(ComboEmpid.SelectedItem);
    SqlCommand cmd = new SqlCommand("Select e.Empname, d.Deptname from Dept d
                        inner join Emp e on d.Deptid = e.Deptid where e.Empid =
                        @eid ", conn);
        cmd.Parameters.Addwithvalue("@eid",Eid);
    SqlDataReader dr = cmd.ExecuteReader();
    while(dr.Read())
    {
        txtEname.Text = dr[0].ToString();
        txtDname.Text = dr[1].ToString();
    }
    dr.close();
    conn.close();
}
}
```

→ Example to Search the customer Information.

Design of Form1.cs



Customer

CustId	CustName	Mobilenum
1	Ravi	9502342121
2	Raju	9848121320

Car

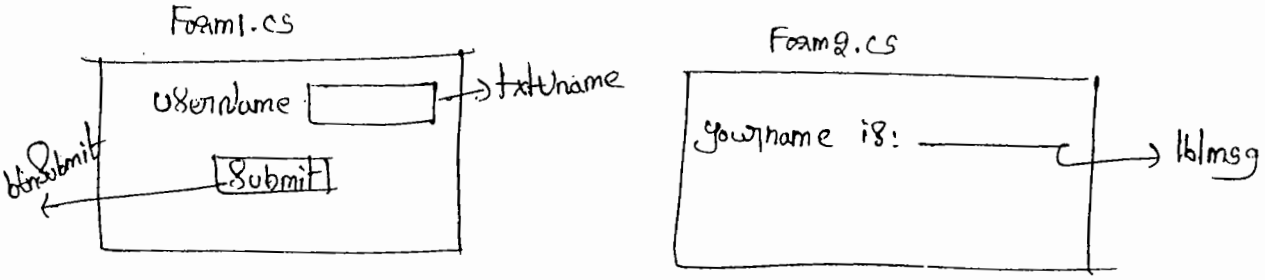
CarId	CarNumber	CustId
10	AP9132	1
20	AP10162	2

code for Form1.cs :-

```
class Form1
{
    void linkSearch_LinkClicked
    {
        SqlConnection Conn = new SqlConnection ("Server = . ; database = Cardb ; uid = sa ;
                                                 Pwd = abc");
        Conn.Open();
        String Carno = txtCarnum.Text;
        SqlCommand cmd = new SqlCommand ("Select cust.CustName , cust.Mobilenum
                from Customer cust inner join car c on c.CustId = cust.CustId
                where CarNumber = @carno", conn);
        SqlDataAdapter da = new SqlDataAdapter (cmd);
        DataSet ds = new DataSet();
        da.Fill (ds, "CustInfo");
        conn.Close();
        if (ds.Tables["CustInfo"].Rows.Count > 0)
        {
            dataGridCustInfo.DataSource = ds.Tables["CustInfo"];
        }
        else
        {
            lblmsg.Text = "No customers are found";
        }
    }
}
```

→ Example to access one windows Form Control value within the another windows Form.

Step 1:- Design Form1.cs and Form2.cs



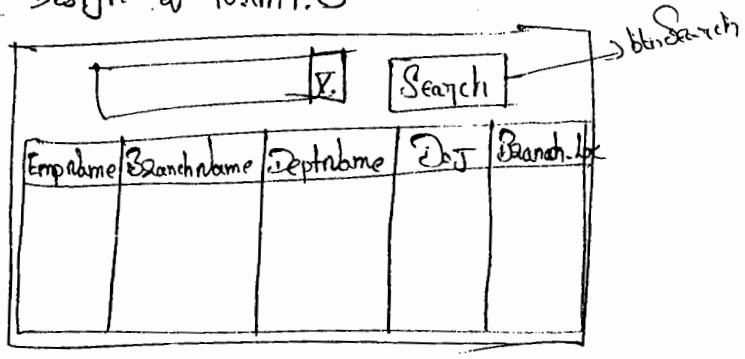
Step2:- Code for Form1.cs

```
class Form1
{
    void btnSubmit_click()
    {
        Form2 objF2 = new Form2();
        objF2.lblmsg.Text = txtUname.Text;
        objF2.Show();
        this.Hide();
    }
}
```

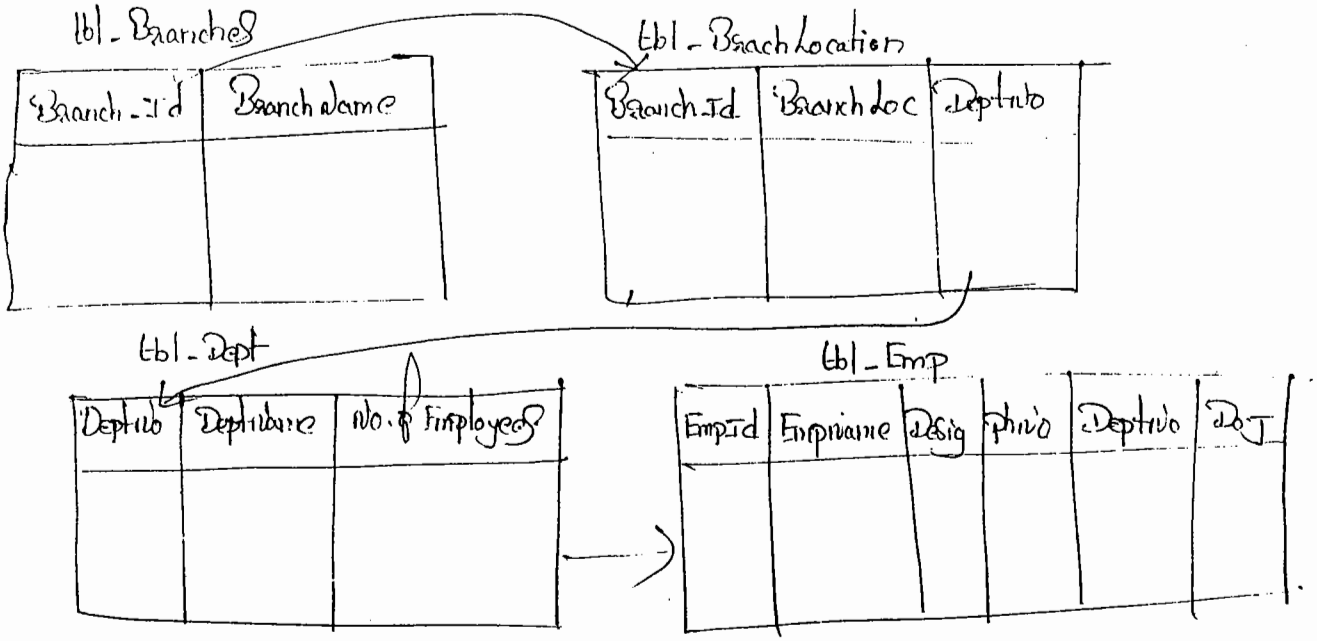
Note:- update the access modifier of lblmsg as public within Form2.Designer.cs file.

→ Example to display the Employee Information from 4 tables based on Single date.

Step 1:- Design of Form1.cs



Step2:- Code for Form1.cs

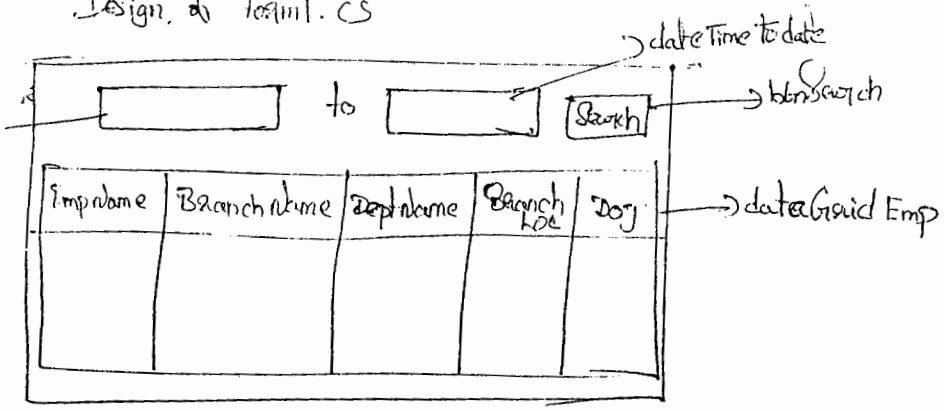



```
class Form1
{
    void btnSearch_click()
    {
        String date = datetimepicker.Text;
        SqlConnection Conn = new SqlConnection("Server = .; database = datedb; uid = sa;
                                                pwd = abc");
        SqlDataAdapter da = new SqlDataAdapter("Select e.Empname, bl.Branchname,
        d.Deptname, bl.BranchLoc, e.DoJ from tbl_Branches b inner join
        tbl_BranchLocation bl on b.BranchId = bl.BranchId innerjoin tbl_Dept d on
        bl.Deptno = d.Deptno innerjoin tbl_Emp e on d.Deptno = e.Deptno where e.DoJ =
        '" + date + "'", Conn);
        DataSet ds = new DataSet();
        da.Fill(ds, "EmpInf");
        dataGridEmp.DataSource = ds.Tables["EmpInf"];
    }
}
```

→ Example to display the Employee Information for 4 tables between 2 dates.

Same 4 tables from above example

Design of Form1.cs



Step2:- Code For Form1.cs

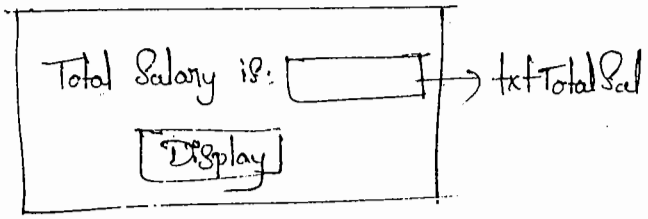
```
class Form1.
{
    void btnSearch_click()
    {
        String fromdate = dateTimefromdate.Text;
        String todate = dateTimetodate.Text;
        SqlConnection Conn = new SqlConnection("Server=.;database=datedb;uid=sa;
                                                Pwd=abc");
        SqlDataAdapter da = new SqlDataAdapter("Select e.Empname, bl.Branchname,
        d.Deptname, bl.BranchLoc, e.Doj from tbl_Branches b innerjoin tbl_BranchLocation
        bl on b.BranchId = bl.BranchId innerjoin tbl_Dept d on bl.DeptNo = d.DeptNo
        innerjoin tbl_Emp e on d.DeptNo = e.DeptNo where e.Doj between '"+fromdate+"'
        and '"+todate+"'", conn);
        DataSet ds = new DataSet();
        da.Fill(ds, "EmpInf");
        dataGraidEmp.DataSource = ds.Tables["EmpInfo"];
    }
}
```

→ Example to display the Total Employees Salary

Emp

EmpNo	EmpName	Sal
10	ram	1000
20	kishore	2000

Step 1:- Form1.cs



Step 2:- Code for Form1.cs

```
class Form1
{
    Void btnDisplay_click()
    {
        Conn.open();
        SqlCommand Cmd = new SqlCommand("Select Sum(Sal) from Emp",
                                        Conn);
        int Salary = Convert.ToInt32(cmd.ExecuteScalar());
        txtTotalSal.Text = Salary.ToString();
    }
```

14/05/14

Mobile Shoppe Project

Mobile Shoppe is a windows application which is computerizing mobile shop day to day transaction

This application will be consuming by two types of users

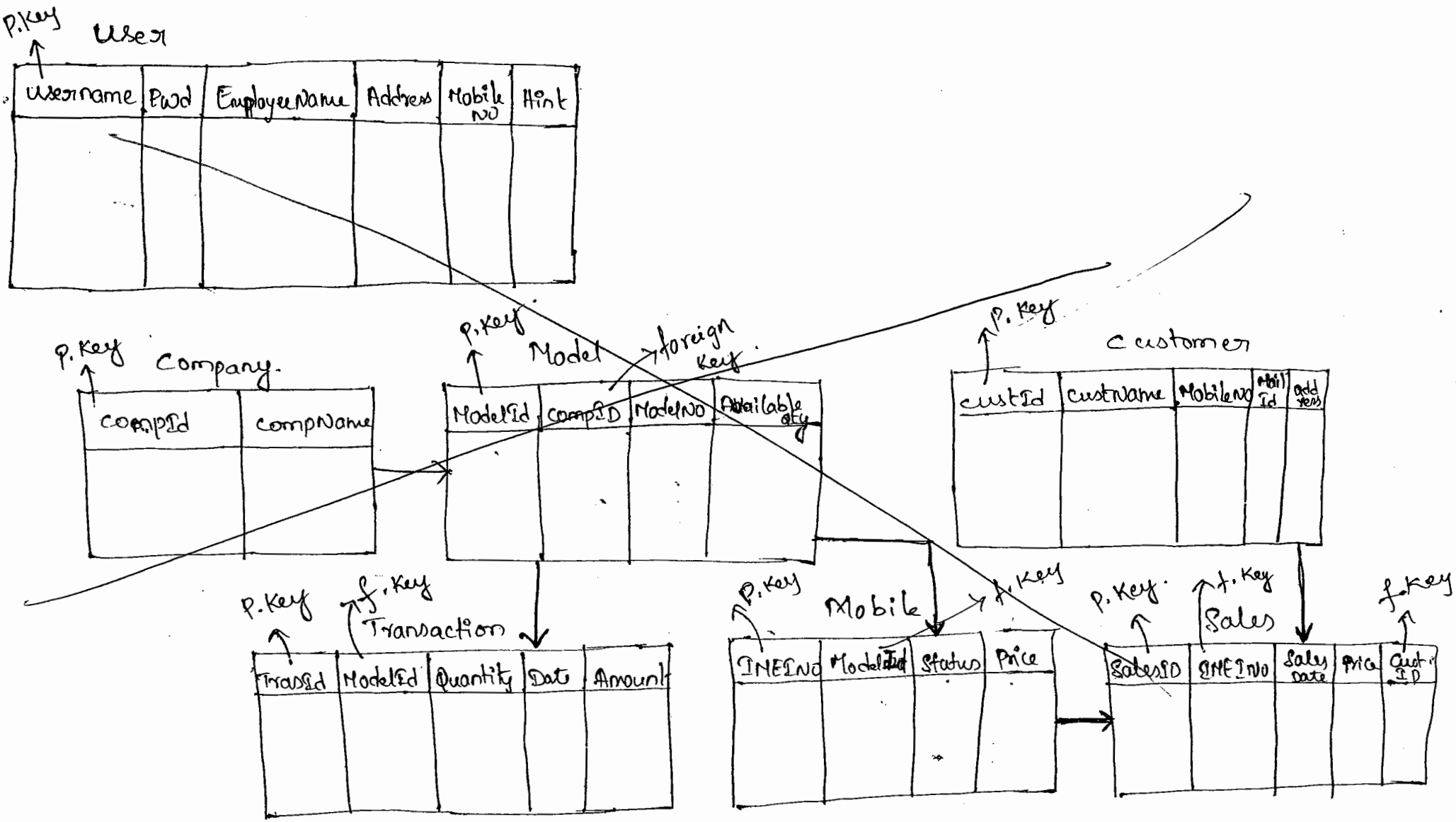
1. Admin (owner of the shop)
2. Employee (Sales men).

Admin Privilages:

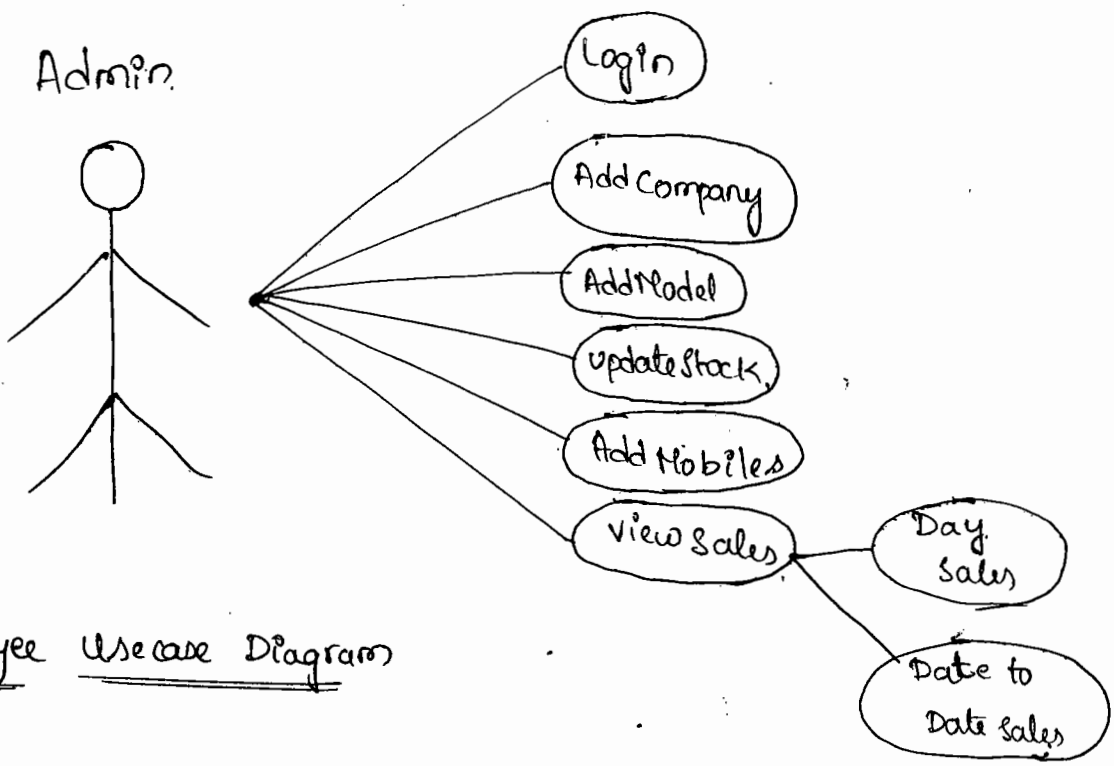
→ Adding company Names.
→ Adding mobile models
→ updating stock
→ Adding mobiles with IMEI number
→ Adding employees.
→ view Sales details (date wise & date to date)

Employees (Sales men) Privilage:

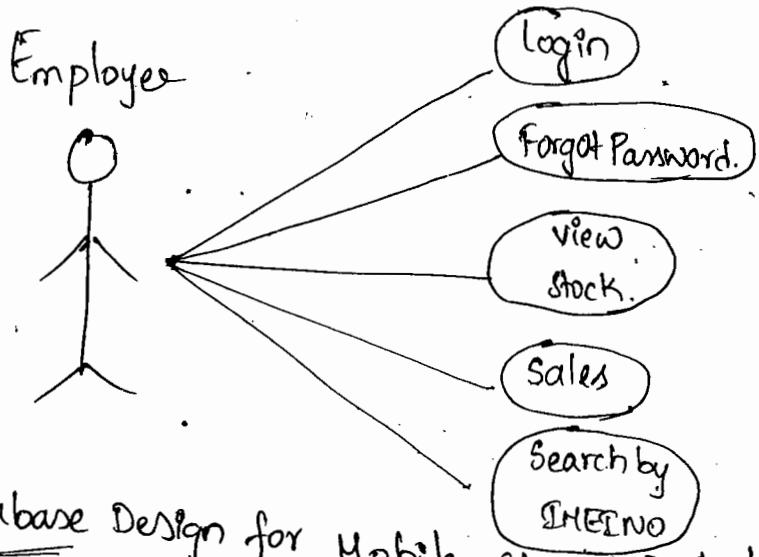
→ login
→ Forgot Password.
→ checking stock
→ Adding sales details
→ Searching Customer Information based on IMEI no.


P.Key
User
Username
Pwd
EmployeeName
Address
Mobile NO
Hint
P.Key
Company.
CompId
CompName
P.Key
Model
foreign Key
ModelId
CompID
ModelNO
Available qty
P.Key
Customer
CustId
CustName
MobileNO
Mobil Id
address
P.Key
f.Key
Transaction
TransId
ModelId
Quantity
Date
Amount
P.Key
f.Key
Mobile
IMEINO
ModelId
Status
Price
P.Key
f.Key
Sales
SalesID
IMEINO
Sales Date
Price
Cust ID
f.Key

Admin Usecase Diagram



Employee Usecase Diagram



Database Design for Mobile Shope Project

P.Key User.

username	Pwd	EmployeeName	Address	Mobile NO	Hint

P.Key Company.

CompID	CompName

P.Key Model F.Key

ModID	ModNum	Available qty	Comp ID

P.Key Customer

custID	custName	MobileNO	Mail ID	Address

P.Key F.Key Transaction

TransID	ModelID	Quantity	Date	Amount

P.Key F.Key Mobile

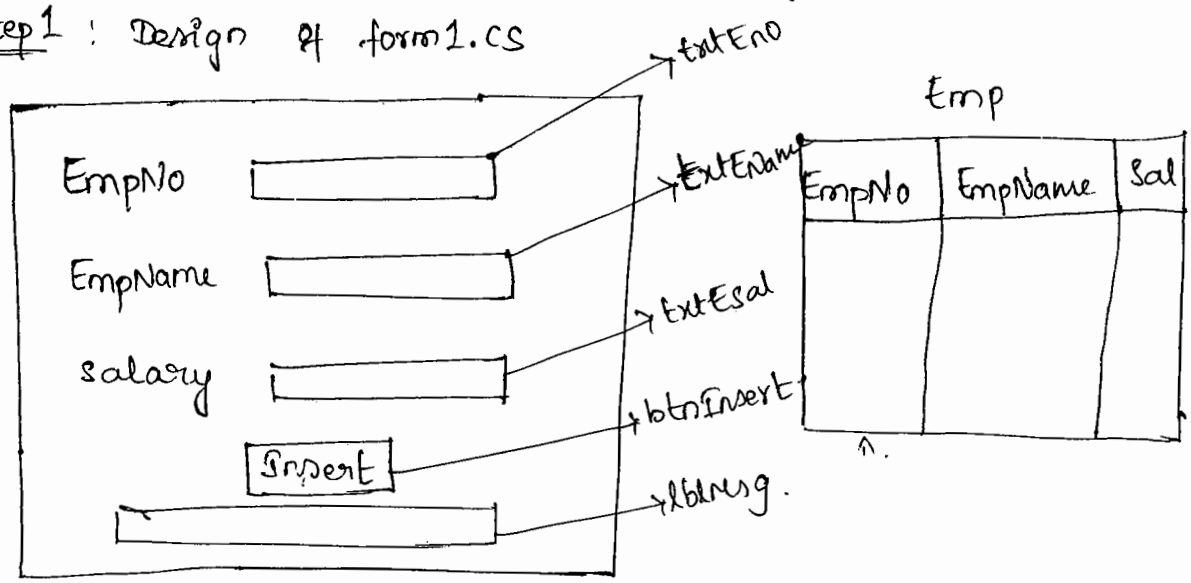
IMEINO	ModelID	Status	Price

P.Key F.Key Sales F.K

SalesID	IMEINO	Sales Date	Price	custID

Example to create AutogeneratedID by using C#.Net code.

Step 1 : Design of form1.cs




```
using System.Data.SqlClient;
namespace AutogenerateIDEx
{
  class Form1
  {
    void
    SqlConnection conn = new SqlConnection("Server=.; database = mydatabase;
                                            uid=sa; Pwd=abc;");
    SqlCommand cmd;
    internal void AutoGenId()
    {
      cmd = new SqlCommand("Select ~~MAX(EmpNO)~~ ISNULL(MAX(EmpNO),0) from Emp", conn);
      conn.Open();
      int i = cmd.ExecuteScalar(); i++;
      conn.Close();
      txtEno.Text = i.ToString();
    }
    void Form_Load()
    {
      AutoGenId();
    }
```

```
void btnInsert_Click( )
{
    int eno = int.Parse(txtEno.Text);
    String ename = txtEname.Text;
double esal = double.Parse(txtEsal.Text);
    cmd = new SqlCommand("Insert into Emp values(@eno,
                                @ename, @esal)", conn);
cmd.Parameters.AddWithValue("@eno", eno);
cmd.Parameters.AddWithValue("@ename", ename);
cmd.Parameters.AddWithValue("@esal", esal);
conn.open();
int i = cmd.ExecuteNonQuery();
conn.close();
AutoGenID();
txtEname.clear();
txtEsal.clear();
txtEname.Focus();
if (i==1)
{ lblMsg.Text = "Record Inserted";
}
else
{ lblMsg.Text = "Record not inserted";
}
}
```

Example to Generate the ID's automatically with the help of identity concept within SqlServer.

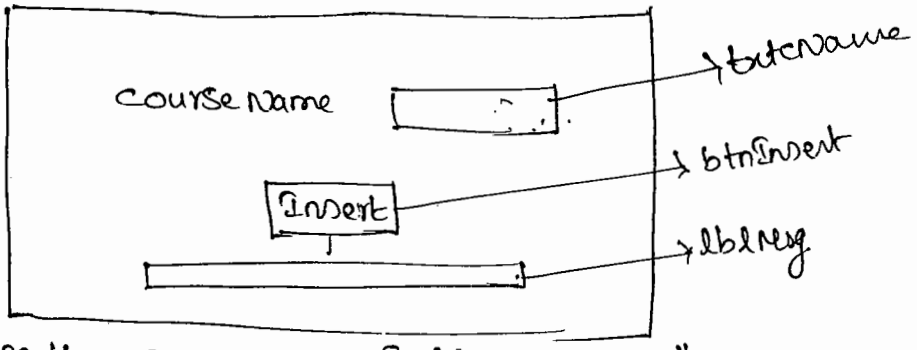
Create table course(cid int Primary key identity(1, 1), cname varchar(20))

insert into course values('dotnet')
insert into course values('java')

cid	cname
1	dotnet
2	java

→ Giving course names from front end.

Step 1 Design




```
SqlConnection conn = new SqlConnection ("Server=.; database=mydatabas;
                                         Uid=sa; Pwd = abc;");
Void btnInsert_click( )
{
    String cname = txtcname.Text;
    SqlCommand cmd = new SqlCommand("Insert into course
                                     values(@cname)", conn);
    cmd.
    conn.open();
    int i= cmd.ExecuteNonQuery();
    conn.close();
    if(i==1)
    {
      lblmsg.Text = "Record inserted";
    }
    else
    {
      lblmsg.Text = "Record not inserted";
    }
}
```

Example to generate AutoId's like below

cid (varchar(10))	cname (varchar(30))
c1	dotnet
c2	php

Implementation of Mobile Shope Project

Step 1 : Create a new windows application rename it as Mobile shope Project.

Step 2 : Add six windows forms to the Solution Explorer they are

1. UserLogin.cs
2. Adminlogin.cs
3. AdminHomepage.cs
4. UserHomepage.cs
5. ForgotPassword.cs
6. ConfirmDetails.cs

Step 3 : Set userLogin.cs as StartupPage.

Step 4 : Create a database within SQL Server that is "MobileShopedb"

Step 5 : Create 7 tables within database like below a/c to design.

1. tbl_company
2. tbl_Model
3. tbl_Transaction
4. tbl_Mobile
5. tbl_Customer
6. tbl_Sales
7. tbl_user

Step 6 : Declare the Global ConnectionString within app.config file.

Admin Module

UserLogin.cs

```
class UserLogin
{
    void linklabel_linkClick()
    {
        AdminLogin objLogin = new AdminLogin();
        objLogin.Show();
        this.Hide();
    }
}
```

AdminLogin.cs

```
class AdminLogin
{
    void linkBack_linkClick()
    {
        UserLogin objLogin = new UserLogin();
        objLogin.Show();
        this.Hide();
    }
    void btnlogin_click()
    {
        if (txtUid.Text == "admin" && txtpwd.Text == "admin")
        {
            AdminHomePage objAdminHome = new AdminHomePage();
            objAdminHome.Show();
            this.Hide();
        }
```

True.

```
else
{
    lblMsg.Text= " User is not valid";
}
}}
```

Adding Companies

→ This requirement we can implement in 2 tasks

1. Generating CompanyId automatically & binding generated Id within txtbox.
2. Inserting the record into company table.

Below code in AdminHomePage.CS

```
class AdminHomePage
{
SqlConnection Conn = new SqlConnection(ConfigurationManager.ConnectionStrings
                                        ["cs"].ToString());
SqlCommand cmd;
void AutoGenId()
{
cmd = new SqlCommand("Select ISNULL(MAX(CompID), 0) from Company",
                                                        conn);
Conn.open();
int i = cmd.ExecuteScalar();
i++;
Conn.close();
txtCompID.Text = i.ToString();
}
AdminHomePage
void Page_Load()
{
    AutoGenId();
}
```

```
void btnAdd_Click()
{
    int compID = int.Parse(txtCompID.Text);
        String compname = txtcompname.Text;
cmd = new SqlCommand("Insert into Company values(@compID,
                        @compname)", conn);
cmd.Parameters.AddWithValue("@compID", compID);
cmd.Parameters.AddWithValue("@compname", compname);
conn.open();
cmd.ExecuteNonQuery();
    AutoGenID();
    conn.close();
    txtcompname.clear();
    }
}
```

Adding Models

```
using System.Data.SqlClient;
using System.Data;
```

Updating Stock

This requirement we can implement by using following tasks

1. Trasaction ID generation automatically.
2. Binding company names to combobox1 ie., ComboCname.
3. Binding the model numbers based on selected company name companyId.
4. Inserting transaction details ie., transactionID, ModelID, Qty, current date, amount into transaction table. and update the model table. availableQty based on modelID

```
class AdminHomePage
{
    internal void AutoTransId()
    {
     // write the code to generate transactionId automatically. &
        assign to txtTranId control.
    }
    internal void BindingCompanyNames()
     {
      // same as above
     }
    void ComboCname_SelectedIndexChanged()
    {
      int compID = Int.Parse(txtcombocname.Text);

      Grid.Row cmd = new SqlCommand("Select mid, mobnum,
                                    where. compid = @cid", com);

        conn.open(); c

        cmd.ExecuteScalar();
```

internal void btnupdate_click()
{

Adding Mobiles

1. Binding Company Names to combobox.
2. Binding Modelnumbers to Combobox based on Selected companyname companyId
3. Inserting into mobile table & assign by default status as not sold.

Add Employee

User Module.

below code in userlogin.cs

```
SqlConnection conn = new SqlConnection(con.
sqlcommand cmd;
String.        select count(*) from user where username = @1
                                  Password = @2", conn)
```

below code in forgotPassword.cs.

```
class ForgotPassword
// Select Pwd. from tbl_User where. username = @uid. and
   hint = @hint.

  String. st = txtUname.Text;
  String hint = txthint.Text;
  cmd = new Sqlcommand("Select Pwd from tbl_User where
     username = @st and hint = @hint", conn);
// passing values to parameters
  Conn.open();
  int i = cmd.ExecuteScalar();
  conn.close();
  if(i=21
```

Sales Module

This requirement we can impement in 2 steps

Step 1: user homepage.cs

1. Binding companynames
2. Binding Model numbers based on selected companyname.
3. Binding IMEINo, Price based on selected modelno whose status is notsold. ~~and price~~.
4. When user click Submit button redirect to confirmdetails

Step 2 Below code within usershomepage.CS

```
class UserHomePage
{
  //task1
   BindingCompanies()
   {
        * from, com.
        // same as above
   }
//task 2
   Binding modelNum()
   {
        model.
        // same as above
   }
//task 3
      Binding IMEINOPrice()
      {
        cmd = new SqlCommand(" select * from tbl_
```

```
void combo_SelectedIndexchanged()
{
    double cost = double.Parse(~~txtcomp~~ comboIMEINo.Text);
    txtPrice.Text = cost;
}
void btnSubmit_Click()
{
    ConfirmDetails.objconfirm = new confirmDetails();
    objconfirm.Show();
    this.Hide();
}
```

assign customer name, Company name, modulno, mobileno, address, IMEIno, EmailId, price from user home page.cs controls to confirm details.cs label controls.

Step 2 : Confirm details (confirmdetalls.cs)

1. Autogen CustId, autogenSalesId.
2. Inserting Customer details into customer table & Sales details into Sales table.
3. Updating. Status column of mobile table as Sold based on IMEINo and update available qty in model table. by reducing 1 based on. modelID
4. When you click cancel button hide confirmdetails

below code within confirm details.cs

```
class Confirmdetails
{
    // 1
    autocustID()
    {
    }
    // 2
    autoSalesID()
    {
    }
```

```
void btnOK_click()
{
    int    custid = int.Parse(~~lblcust id.Text~~ t.Text);
    String  custname = lblcustname.Text;
    long  Mobilenum=long.Parse(lblMobilenum.Text);
    String. custmailId = lblMailId.Text;
    String Address = lblAdress.Text;
    ~~cmd~~ =new SqlCommand("insert
    int  Salesid = int.Parse(t.Text);
    long IMEINo = Long.Parse.
```

Below code within userhomePage.cs

```
class userhomePage
{   // task 1
    Binding company
    {
    }
    // task 2
      Binding ModelNo()
      {
      }
    user Homepage_Load()
      {
          Bindingcompany();
          {
          }
      }
```

void comboModelNum_SelectedIndexchanged()

{ mid - - -

cmd = new SqlCommand("Select AvailableQty. from tbl_Model where ModelId = @mid", conn);

cmd.Parameters.AddWithValue("@mid", mid);

conn.open();

SqlDataReader dr = cmd.ExecuteReader()

~~DataSet ds = DataTable dt = new DataTable(d~~

Searching customer based on IMEINO

write the below code within userhomepage.cs.

```
class UserHomePage
{
    void linkSearch_Click()
    {
        long ImeiNo = long.Parse(txtIMEINo.Text);
    SqlCommand cmd = new SqlCommand("Select c1.CustomerName,
        c1.MobileNumber, C1.EmailId, C1.Address from tbl_Sales S1
        Inner Join tbl_Customer c1 on S1.CustId = C1.CustId
        where S1.IMEINO = @ImeiNo)", Conn);
    cmd.Parameters.AddWithValue("@ImeiNo", ImeiNo);
    conn.open();
    SqlDataReader dr = cmd.ExecuteReader();
    GridCustomer.DataSource = dr;
    conn.close();
    }
}
```

Sales Report (Admin)

Write a Query to get Sales details based on single date.

```
Select  S1.SalesId, C1.CompanyName, Md1.ModelNum, S1.IMEINO,
        S1.Price from tbl_Sales S1 inner join tbl_Mobile Mb1
        on  S1.IMEINO = Mb1.IMEINO inner join tbl_Model
        Md1  on  Mb1.ModelId = Md1.ModelId inner join
        tbl_Company C1 on Md1.CompId = C1.CompId where
        S1.IMEINO = @IMEINO.   S1.date = @d.
```

```
class AdminHomePage
{
    void DateTimePicker_ValueChanged()
    {
        DateTime Date = Convert.ToDateTime(DateTimePicker1.Text);

        Cmd = new SqlCommand("Select S1.SalesId, C1.ComanyName,
                Md1.ModelNum, S1.IMEINO, S1.Price from tbl_Sales S1
                inner join tbl_Mobile Mb1 on S1.IMEINo = Mb1.IMEINo
                inner join tbl_Model. Md1 on Mb1.ModelId = Md1.ModelId.
                inner join tbl_company C1 on Md1.compId = C1.CompId
                where S1.date = @date'
```

Validations for Mobile Shope.

→ Textboxes should not be empty

Adding companies

→ Company textbox should not allow duplicate values.

Adding Models

→ Company name should be selected

→ Model number should not be duplicated.

Update Stock

→ Company Name, Model Number should be Selected

→ Quantity, Amount should allow only numerical values

Adding Mobile

→ Company name, ModelNO should be selected

→ IMEINO, Price should allow only numerical values.

→ IMEINO should not allow duplicate values.

Sales Report

Day

→ Dont allow the user to select future date

→ When user click Sunday click holiday msg.

→ Let us assume our client has established business on jan 2013, avoid user to select previous date of estabishment date.

Date to Date

→ User Should select valid. date for starting & ending date.

→ Starting date should not be before establishing date.

→ ending date should not exceed the current date.

Add Employee

→ Employee user name should not be duplicate

→ Mobile no should not allow other than digits

→ Mobile no's should be (min 10 & max 12)

→ username should have atleast 6 chars, atleast 1 char & 1 number.

→ Password should have 8 chars & Start with capital letter.

→ password should have 1 digit & 1 Spl char.

→ Both passwords should match.

→ Hint Should be min 5 chars.

<u>Userlogin and forgot Password</u>

→ Textboxes should not be empty

<u>Sales</u>

→ Mobile no should be digits & should have 12 digits

→ customer name should be string.

→ Address Should be string.

→ Email ID Expression should be valid

<u>View Stock</u>

→ Should Select company name & model number.

<u>Search customer</u>

→ IMEINO txtbox Should not be empty

→ IMEINO should have 16 digits

SATHYA TECHNOLOGIES

MOBILE SHOPPE PROJECT

(WINDOWS FORMS(C#.NET))

Sathya

Address:

2nd Floor, Sri Sai Arcade,

Beside: Adithya Trade Center,

Ameerpet, Hyderabad-500038.

Ph:040-65538958/65538968

SATHYA TECHNOLOGIES-Mobile Shoppe Project Scenario

1.	Name of the Project	MOBILE SHOPPE
2.	Objective/ Vision	Mobile Shoppe is a windows application which is computerizing a mobile shop's day to day transactions. The objective of this project is to maintain the details of the mobile store, like maintaining product details available in the store.
3.	Users of the System	1. Administrator 2. Employee
4.	Functional Requirements	• Administrator can manage the company and model details • He can view the Sales details, Add the Employees • Employee can view the Stock Details and make the sales • Employee can recover the password through Hint Question • Employee can search the customer • Only Authenticated users can access the system
5.	Reports	• Day Wise Reports • Weekly Reports • Monthly Reports
6.	Languages and Technologies to be used	• C#.Net • ADO.NET
7.	Backend	• MS SqlServer-2008
8.	Tools to be Used	• Microsoft Visual Studio Editor-2010 • .NET Framework-4.0 • Unified Modeling Language

MOBILE SHOPPE

Description:

Mobile Shoppe is a windows application which is computerizing a mobile shop's day to day transactions. The objective of this project is to maintain the details of the mobile store, like maintaining product details available in the store.

Admin:

Admin will have following privileges.

- Adding Company Names
- Adding Mobile Models
- Updating Stock
- Adding Mobile Details with IMEI number
- Adding Employees
- Viewing Sales Details(Date Wise and Date to Date)

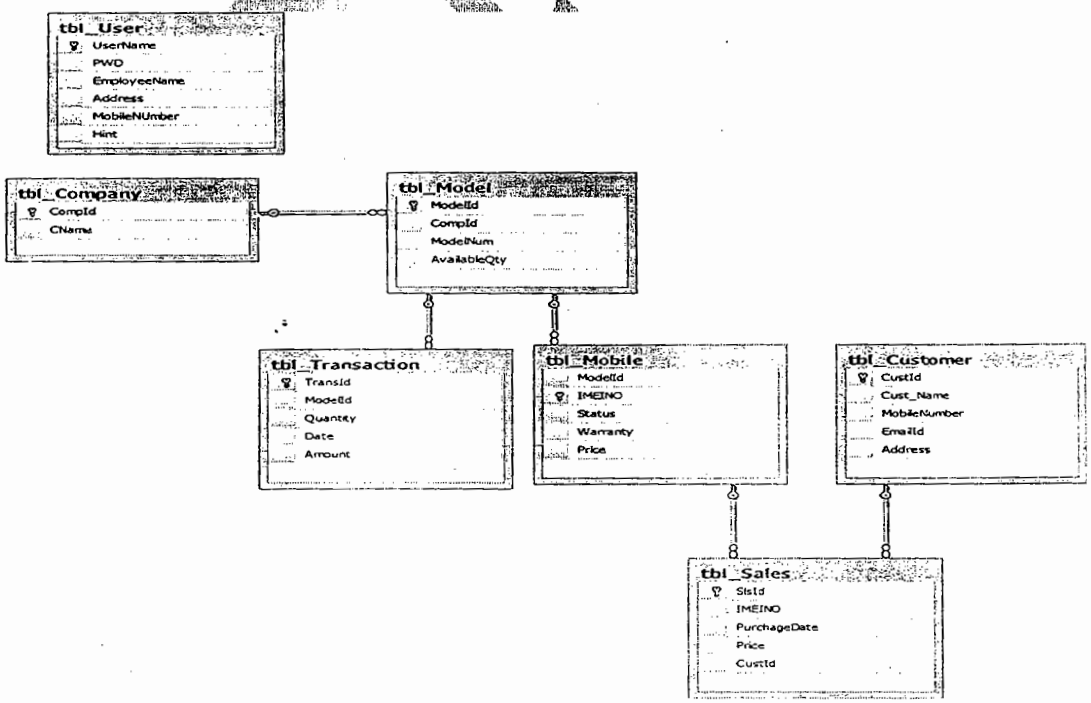
Employee:

Employee will have following privileges.

- Adding Sales Details
- Viewing Stock Details
- Searching Customer Information by using mobile IMEI number

Database Design

ER Diagrams



UseCase Diagrams:

Use case is used to define the core elements and process that make up a system.

It contains two things those are

1. Actor
2. UseCase

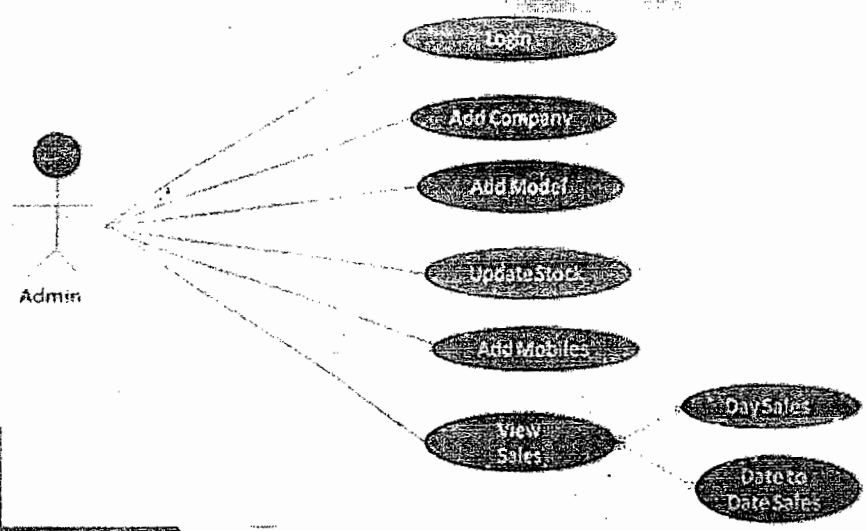
Actor: The actors are also called as key elements.

UseCase: The Use cases are also called as Processes.

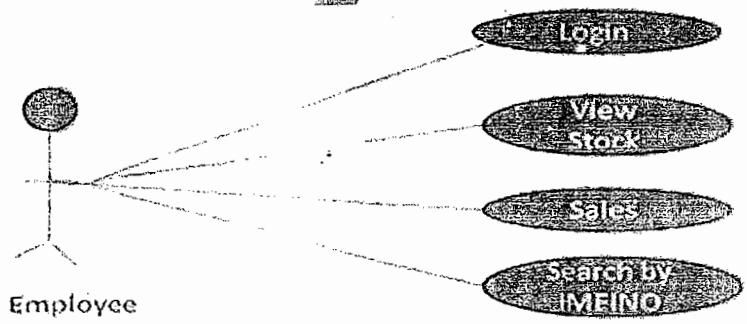
It always gives the relationship between actor and usecases or activities.

It is used for understanding the behavior of the application. It gives the clarity about project development code.

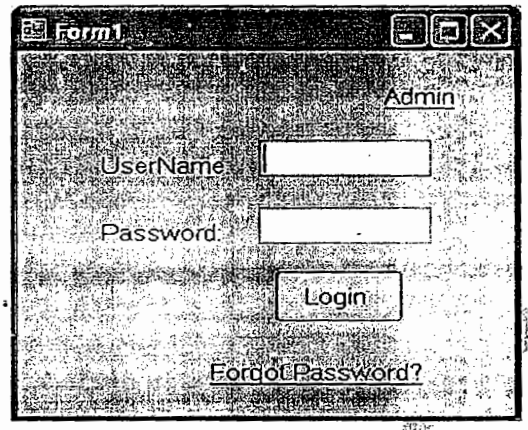
Admin UseCase Diagram:



Employee Use Case Diagram:

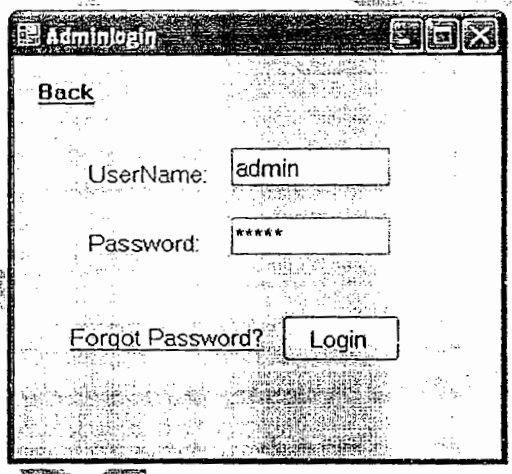


Home Form



- when the user clicks on the admin button it should redirect to "Admin Login" Form

1. **Admin Login Form**



- Give the username and password as "admin".
- When the admin clicks on login button if it is valid it should redirect to "Admin Home" form otherwise it should display an error message.

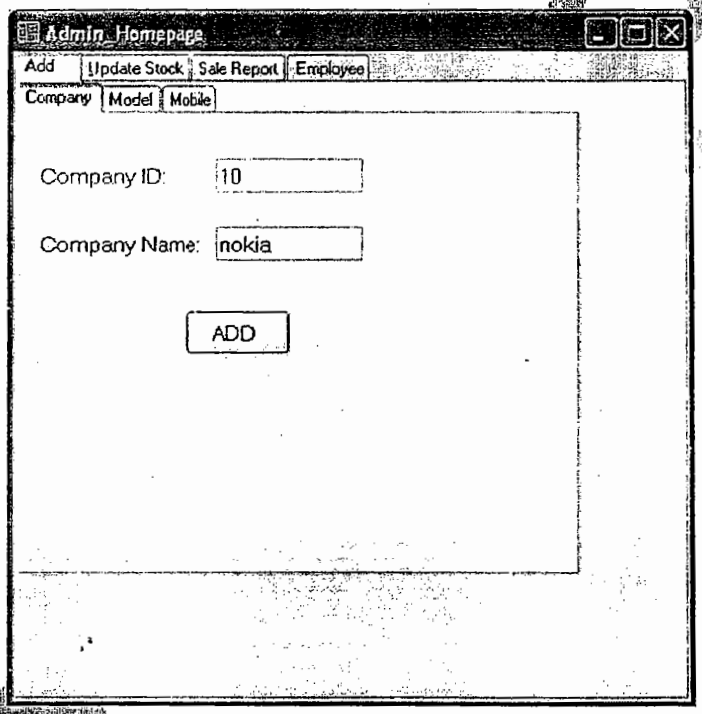
2. Admin Homepage

AddComapany:

Create a table with name "tbl_Company" as shown below.

tbl_Company

Column Name	Data Type	Allow Nulls
CompID	varchar(20)	☐
CName	varchar(20)	☑
		☐



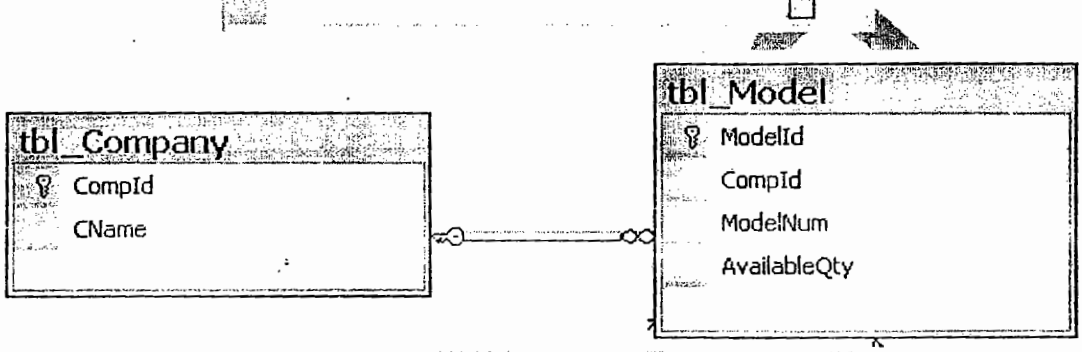
- Company ID should be generated automatically from the code and it should not be editable.
- When the user enters company name and click on add button it must store in to "tbl_Company" table in the database.

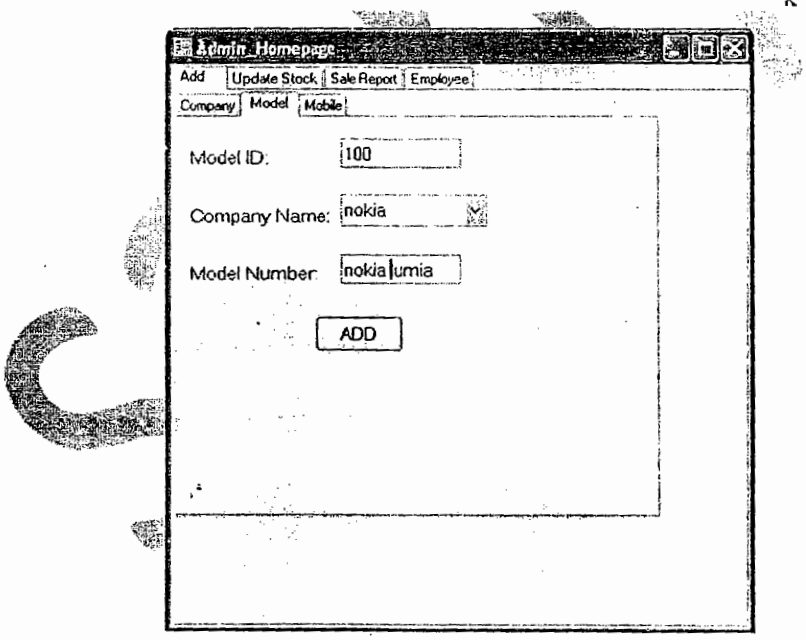
Add Model:

Create a table with name "tbl_Model" as show below.

tbl_Model

Column Name	Data Type	Allow Nulls
ModelId	varchar(20)	☐
CompId	varchar(20)	☑
ModelNum	varchar(20)	☑
AvailableQty	int	☑
		☐



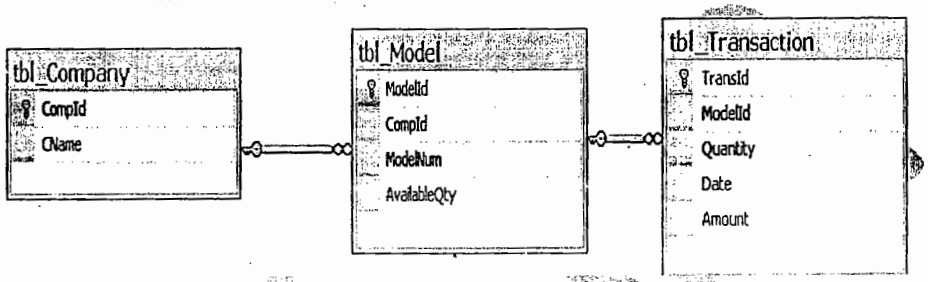


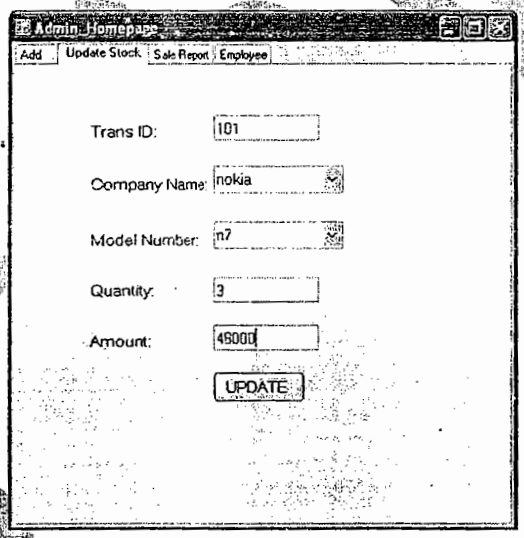
- Model Id should be generated automatically from the code and it should **not** be editable.
- Get the company names from "tbl_Company" table.
- When the user clicks on add button the model ID,company ID,model name should be stored in the "tbl_Model" table and quantity should be zero.

Update Stock:

tbl_Transaction

Column Name	Data Type	Allow Nulls
TransId	varchar(20)	☐
ModelId	varchar(20)	☑
Quantity	int	☑
Date	date	☑
Amount	money	☑
		☐



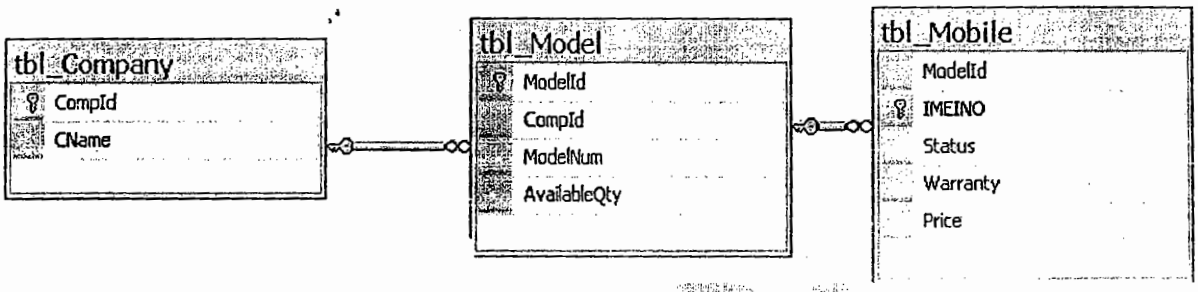


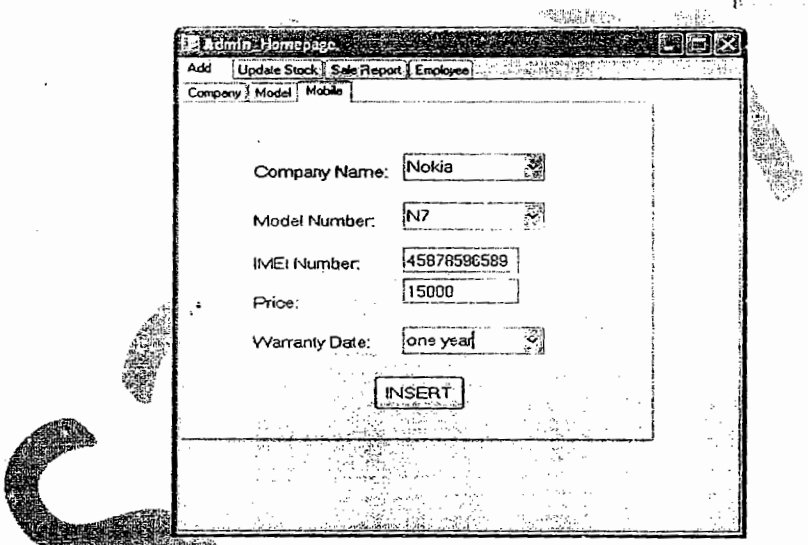
- Trans ID should be generated automatically from the code and it should not be editable.
- Get the company names from "tbl_Company" table and bind in the company name combobox.
- When the user selects company names all the models of that company should be displayed in the model combo box from "tbl_Model" table.
- When the user clicks on update button all transaction details must be stored in "tbl_Transaction" table and 'quantity' must be updated in "tbl_Model" based on modelID.

AddMobile:

tbl_Mobile

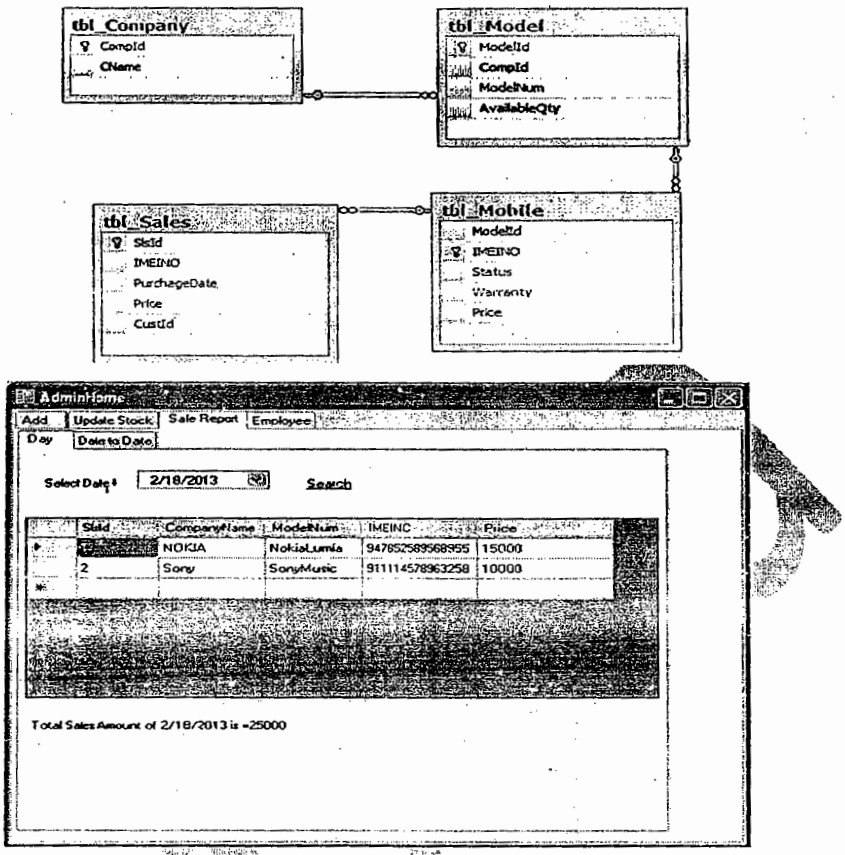
Column Name	Data Type	Allow Nulls
ModelId	varchar(20)	☑
IMEINO	varchar(50)	☑
Status	varchar(20)	☑
Warranty	date	☑
Price	money	☑
		☐





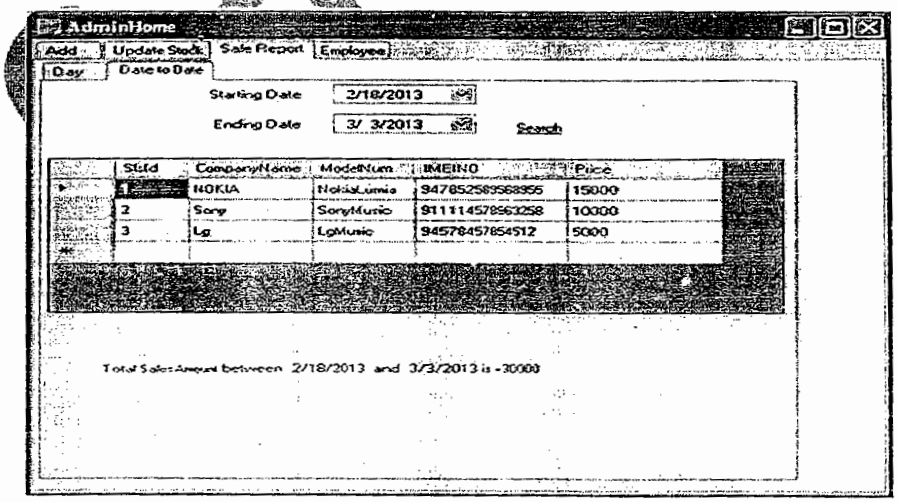
- Get the company name from "tbl_Company" table.
- When the user select company names all the models of that company should be displayed in the model combo box from "tbl_Model" table.
- When the user clicks on insert button modelID, model number, IMEI number, price, warranty date should be inserted in to the "tbl_Mobile" table. By default status must be "Not sold".

Sales Report:



- When user selects the date all sales details of that day should be displayed in the form from three tables displayed above.

Date to Date Report:



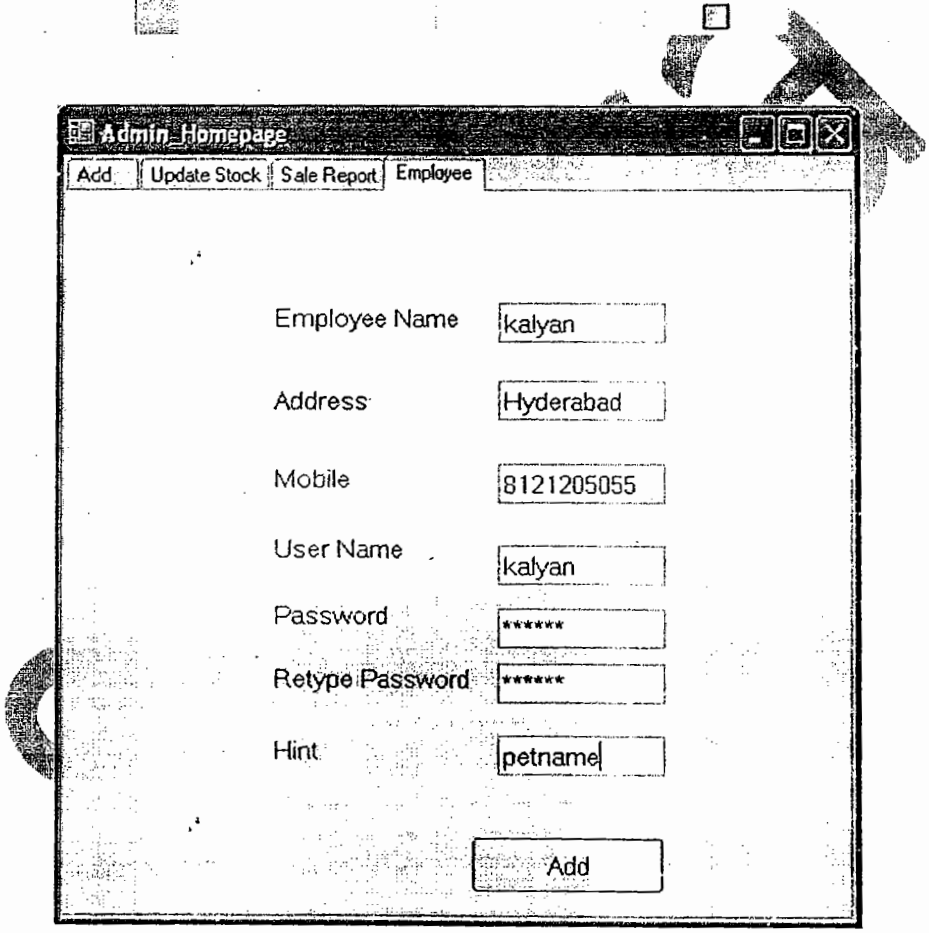
- When user selects the start date and end date all sales details between these days should be displayed in the form from three tables displayed above.
- The total sales amount of the selected dates should be displayed in the bottom of the grid view.

AddEmployee:

Create a table with name "tbl_user".

tbl_User

Column Name	Data Type	Allow Nulls
UserName	varchar(20)	☐
PWD	varchar(20)	☑
EmployeeName	varchar(20)	☑
Address	varchar(MAX)	☑
MobileNUmber	varchar(20)	☑
Hint	varchar(50)	☑
		☐



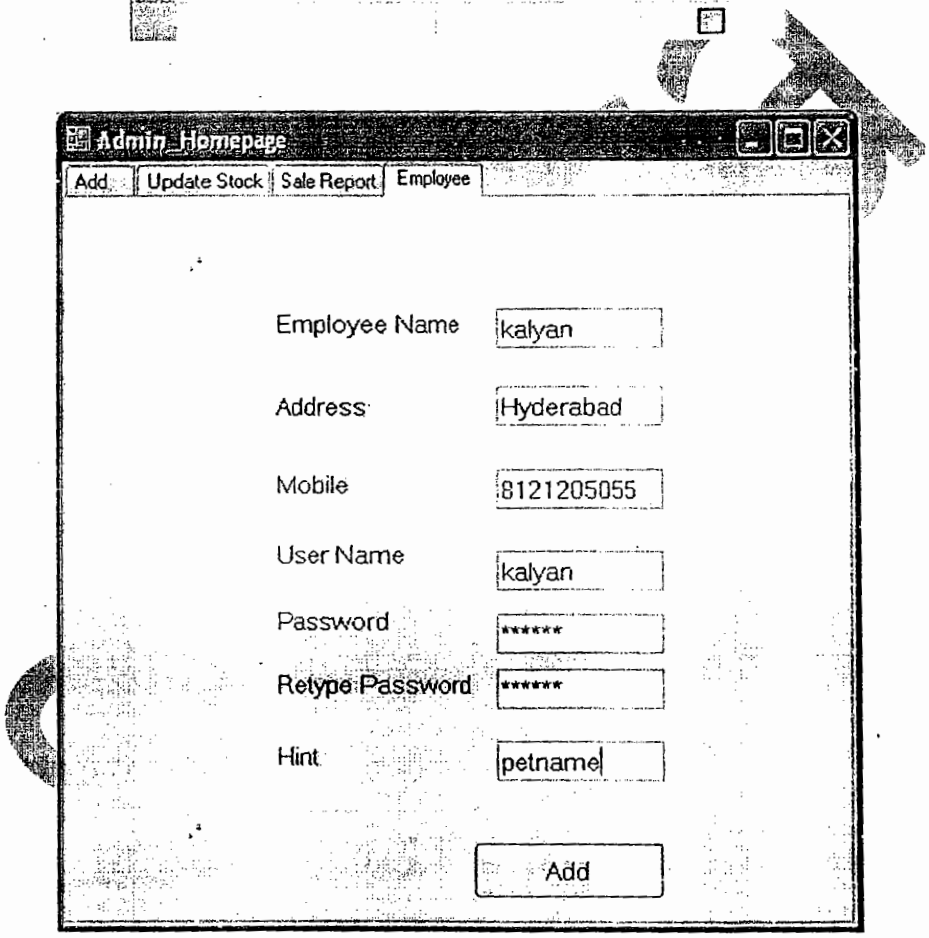
- When user clicks on add button all the details must be stored in "tbl_user" table.

AddEmployee:

Create a table with name "tbl_user".

tbl_User

Column Name	Data Type	Allow Nulls
UserName	varchar(20)	☐
PWD	varchar(20)	☑
EmployeeName	varchar(20)	☑
Address	varchar(MAX)	☑
MobileNUmber	varchar(20)	☑
Hint	varchar(50)	☑
		☐

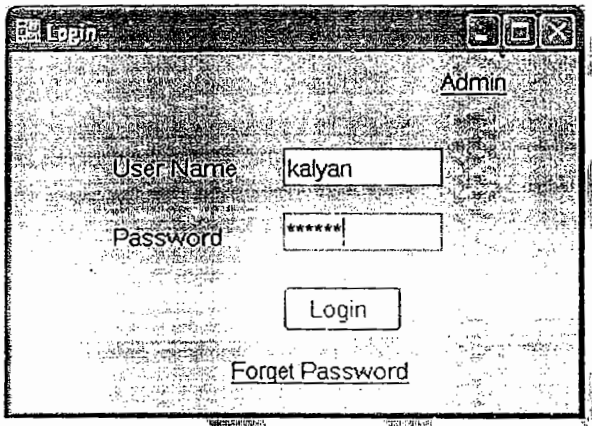


- When user clicks on add button all the details must be stored in "tbl_user" table.

Home Form:

- Create a table with the name "tbl_User".

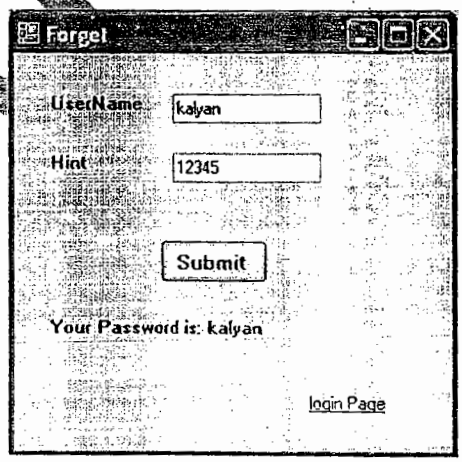
Column Name	Data Type	Allow Nulls
UserName	varchar(20)	☐
PWD	varchar(20)	☑
EmployeeName	varchar(20)	☑
Address	varchar(MAX)	☑
MobileNUmber	varchar(20)	☑
Hint	varchar(50)	☑
		☐



- The user enter his credentials and clicks on Login button if it is valid redirect to UserHomePage.
- If it is invalid it will display one error message to user.

Forgot Password Form:

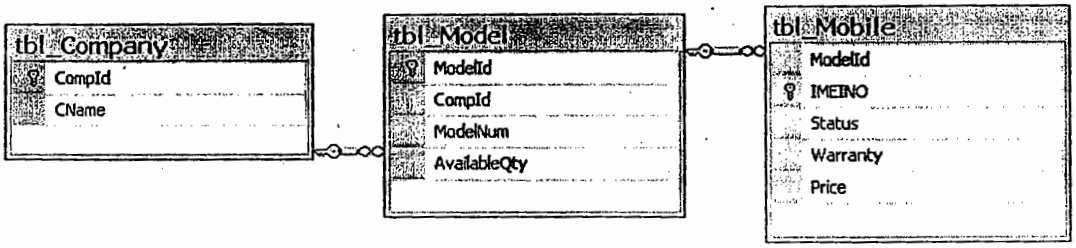
- The User clicks on Forget Password linkbutton, it should redirect to Forget Password Form.

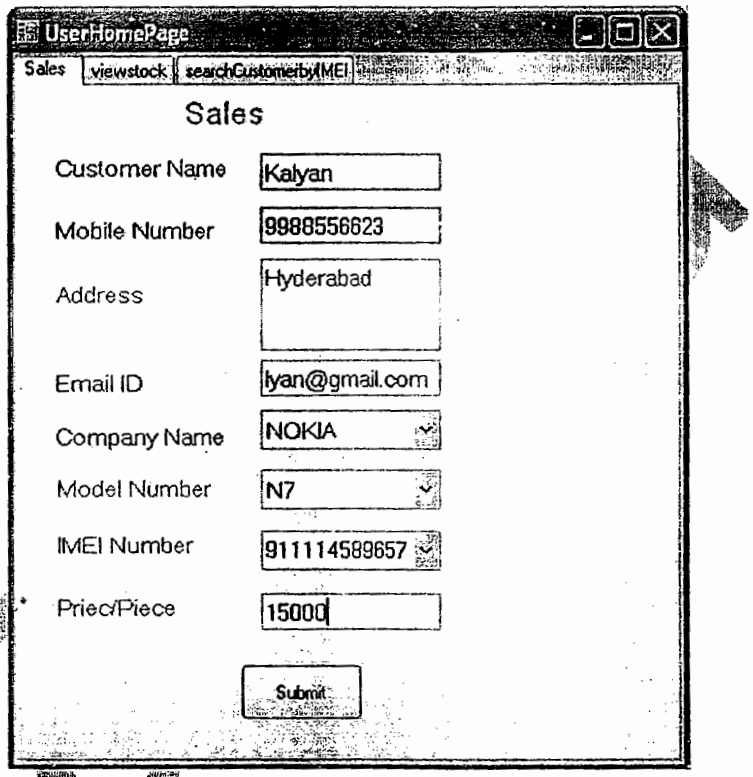


- When the user enter the UserName and Hint, and clicks on the submit button.
- If the UserName and Hint is valid it should display the password in below label, Other wise it display one error message to user.
- Based on UserName and Hint it will get password from "tbl_User" table.

Sales Form:

- Create a table with the name "tbl_Customer".



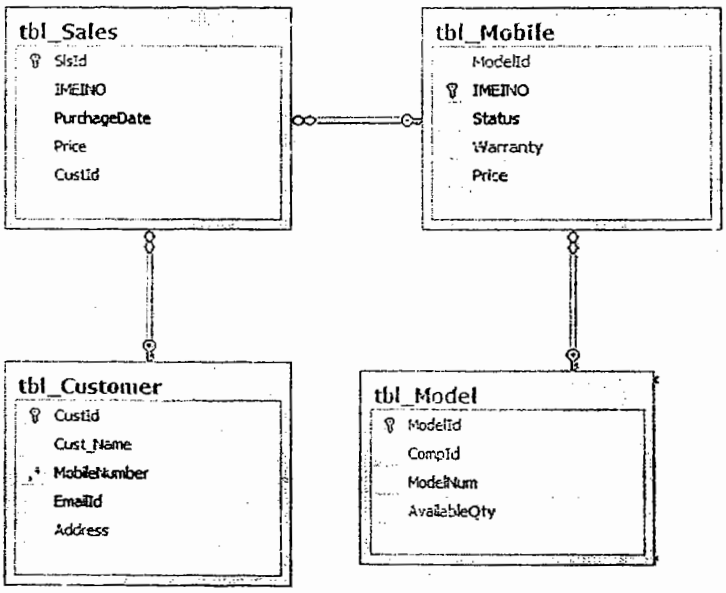


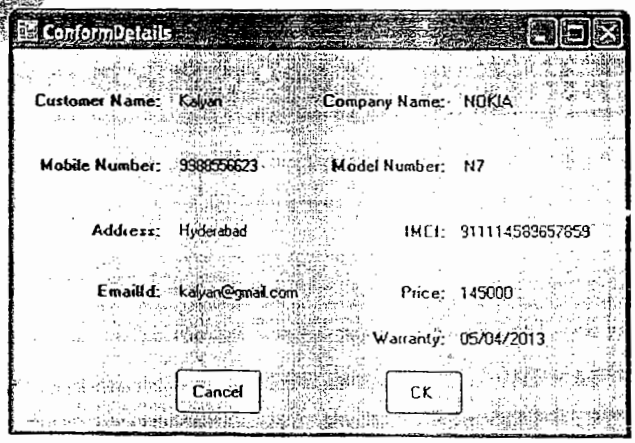
- User enter the details into these fields, Here When the user select the company name in the combo box. The company related models automatically shown in below combo box(Model Number).
- Based on model number that related IMEI numbers also displayed in below (IMEI number)combo box.
- These two combo boxes are should read only.
- When user clicks on submit button it will redirect to conform details Form. and show all the details in labels as shown below.

Column Name	Data Type	Allow Nulls
Sid	varchar(20)	
Cust_Name	varchar(20)	☑
MobileNumber	varchar(20)	☑
EmailId	varchar(20)	☑
Address	varchar(MAX)	☑

DOTNET-PC.MobileS...e - dbo.tbl_Sales

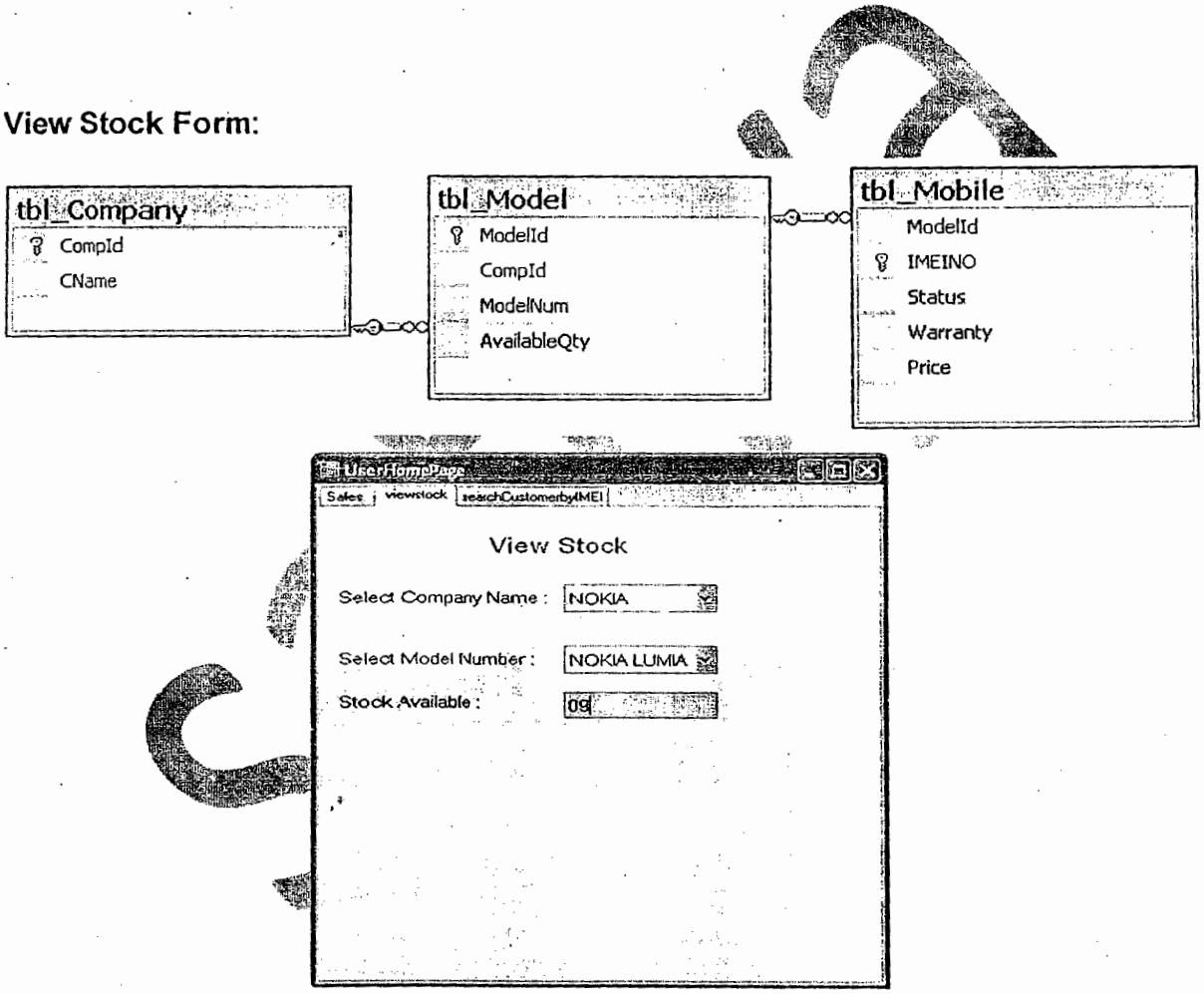
Column Name	Data Type	Allow Nulls
Sid	varchar(20)	
IMEINO	varchar(50)	☑
PurchageDate	date	☑
Price	money	☑
CustId	varchar(20)	☑





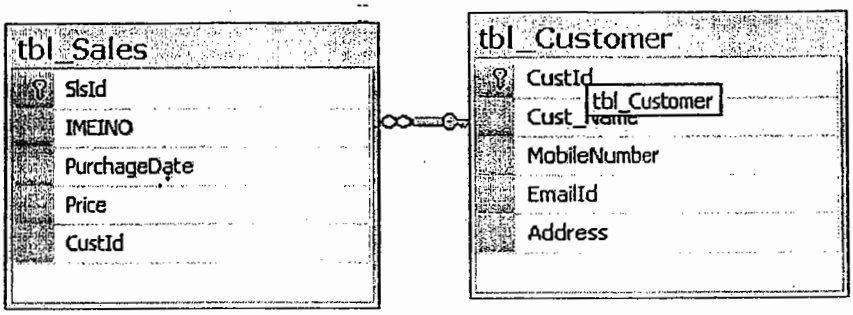
- The sales id and customer id should be generated automatically.
- When the form is loaded based on IMEI number warranty should display
- Here the user clicks on Ok button the Customerid CustomerName, MobileNumber, Address and EmailId should be stored/Inserted in "tbl_Customer" table and Salesid, .IMEI Number, Price and customerid should be inserted into Sales Table
- Based on the IMEI number the status in the mobile table should be changed to "sold"
- The Available Quantity should be updated in model table based on Modelid.
- The user clicks on Cancel button it should not be stored into the database.

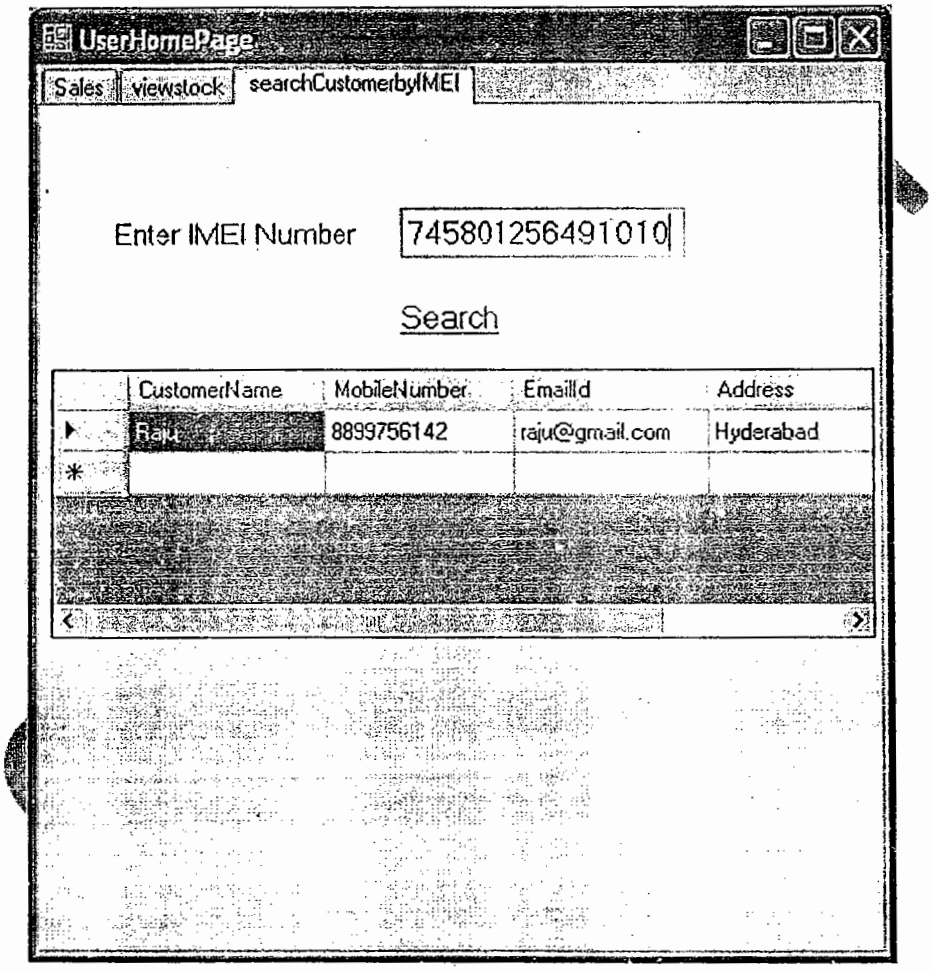
View Stock Form:



- In this Form User can View the Stock Availability.
- When the user select the company name in the combo box. The company related models automatically shown in below combo box.
- Based on the model number it will get the ModelId. Based on ModelId it will show Stock availability from tbl_Model table.
- Here the stock availability textbox should not editable.

Search Customer by IMEI Form:





- Here user can enter the IMEI number and click on submit button.
- If the IMEI number is valid, based on IMEI number it will get the data from "tbl_Sales" and "tbl_Customer" tables for the required fields in dataGridView.
- If the IMEI number is invalid it will display one error message to the User.