About Author

The main aim of my book is to understand C#.Net (Microsoft's most powerful language) for new learners in a simple manner.

This book is dedicated to My PARENTS and My Students.

PREPARED BY

PUDI.VISWESWARA RAO

[illegible]

DOTNET APPLICATION LEAD DEVELOPER

About Book

Contents

Introduction to Visual C#

C# is type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework. You can use C# to create Windows client applications, XML Web services, distributed components, client-server applications, database applications, and much more. Visual C# provides an advanced code editor, convenient user interface designers, integrated debugger, and many other tools to make it easier to develop applications based on the C# language and the .NET Framework.

- C# .Net is the most powerful programming language among all programming language in .Net by Microsoft, because C#.Net will contain all the 3features of C++, vb6.0 and java.
- In c#.net, the symbol #must and should be pronounced by "sharp" only because Microsoft has taken the symbol from musical note, whose name is "sharp".
- Among all musical notes "sharp note" is most powerful note.
- So Microsoft expected that C#.Net will become most power full programming language among all programming languages.
- We can say that C#.net = c++ + vb6.0 + java + additional features

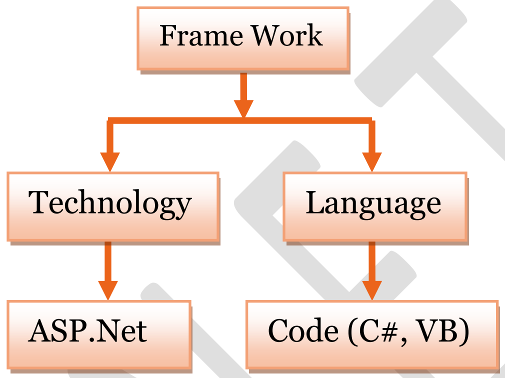
Differences between C++, VB6.0, JAVA and C#.Net

Sr no	**C++**	**VB6.0**	**JAVA**	**C#.NET**
01	Object oriented but not pure	Object Based programming language	pure object oriented programming language	Pure object oriented programming language
02	Supports multiple inheritance	Does not Supports multiple inheritance	Supports multiple inheritance using Interface	Supports multiple inheritance using Interface
03	Not platform independent	Not platform independent	platform independent	platform independent
04	Supports pointers	Does not Supports pointers	Does not Supports pointers	Supports pointers using unsafe code
05	Code reusability is achieved using header files.	Code reusability is achieved using COM and DCOM.	Code reusability is achieved using packages.	Code reusability is achieved using Assemblies.
06	Case sensitive PL	Not Case sensitive PL	Case sensitive PL	Case sensitive PL
07	Does not support XML.	Does not support XML.	Supports XML.	Supports XML.
08	Supports structures.	Supports structures.	Does not Support structures.	Supports structures.
09	Supports function pointers	Does not Supports function pointers	Supports Delegates	Supports Delegates

FRAME WORK

Frame work is the run time environment of.Net program or execution of.Net applications

- There are different versions in frame work like 1.0, 1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1 …4.8.
- Currently 4.8.1 frame work version is the latest that was introduced in 9 **Aug** 2022.
- Dotnet supports 64 programming languages within that 11 languages Introduced by Microsoft.
- Without frame work there will be no dot net.



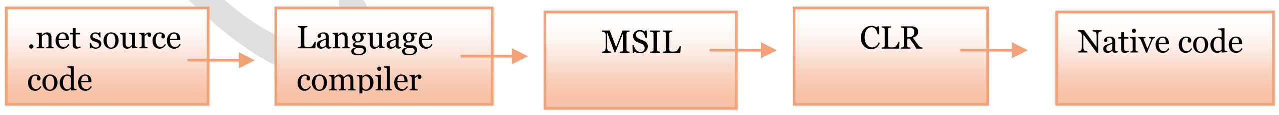
Using technology we can design websites/Web pages.
Using languages we can develop code /logic.

Platform

Platform means it's a combination of CPU and OS (operating system)

- Dot net is a platform independent language.
- What is Platform Independent .Language (PIL)?
  →PIL means we can run Dot net application in any OS and in any CPU.

Compilation & MSIL:-



MSIL→Microsoft Intermediate Language.

CLR→Common Language Run Time.

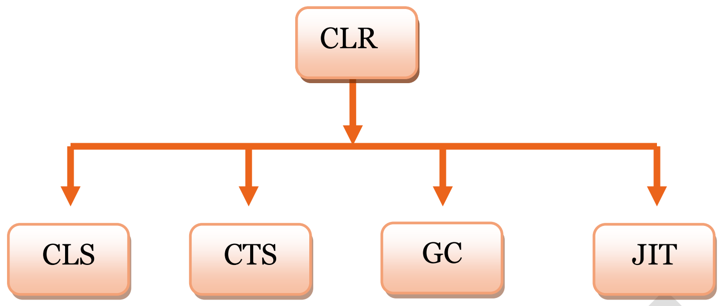
In dot net code is compiled twice. In first compilation source code is compiled by respective language. After that an intermediate code is generated which is called as MSIL by language compiler

..In second compilation MSIL is converted in to native code using CLR.

Note: Always first compilation is slow and second compilation is fast.

CLR

CLR means common language runtime. In that CLR we are having sub parts like



CLS: - CLS means Common Language Specification. CLS is responsible for language interoperability. This is achieved in two ways,

1. Managed code

2. Un Managed code

Managed code: -

Code for which MSIL is generated after Language Compiler Compilation is directly executed by CLR is known as Managed Code.

Unmanagedcode:-

Code that has written development of .Net for which MSIL is not available that is not executed by the CLR directly, rather CLR redirects the code to OS for execution, which is known as unmanaged code.

- CLR does not provide any facilities and features like CTS, memory management...etc.,

WHAT is CLS?

Every programming language will have some rules used to write the code which is known as Language specification.

As we know that managed code execution process of .Net supports many programming languages…

Every programming language has its own specification.

One programming language can’t understand other programming language specification.

But CLR can understand any programming Language, language specification rather CLR has its own language specification.

CTS: - (Common Type System)

AS we know that in managed code execution process .net supports many programming languages.

- Every programming language has its own specification and DATATYPES…
- One programming language can't understand other programming language Data types…
- But CLR will execute all data types, this is possible because of CLR contains its own set of data types.
- At the time of compilation all data types will convert into CLR data types…

Garbage Collector: - (GC)

GC is responsible to provide automatic memory management.

Why automatic memory management?

- There will be no problem of insufficient memory.
- Burden on the programmer will reduced i.e. programmer need not to write code to perform memory manage tasks.

What is Automatic Memory Management?

Garbage collector has its own engine called Optimization Engine, which runs when required and divides objects into two categories.

1. Objects in use→Objects in use are kept in the memory.
2. Idle objects → Idle objects are destroyed from the memory

JIT: - (Just-In-Time)

JIT is responsible to compile and MSIL code and to generate native code using CLR.

CLR will execute native code by providing all the facilities like Language interoperability…etc

********END OF FRAMEWORK ********

How to open visual studio

Click on start

$\downarrow$

Click on programs

$\downarrow$

Click on Microsoft visual studio 2022

$\downarrow$

Launch micro soft visualstudio2022

(Or)

Click on start

$\downarrow$

Click on Run

$\downarrow$

Type DEVENV (Development Environment)

$\downarrow$

Click on OK

Console Application:

1. Console applications do not provide any GUI facilities.
2. Console applications are similar to C, C++, programs.
3. In Console applications we completely work in CUI based environment.
4. CUI (character user interface) is a way to interact with computer programs by using commands (one or more lines of code).

Steps to create Console application

Step1: Launch Visual studio

Step 2: Create new project

Step 3: Select below from the dropdown

Language: C#

Platform: Windows

Project type: Console

Step 4: Select **Console App (.Net Framework)**

Step 5: Click on next

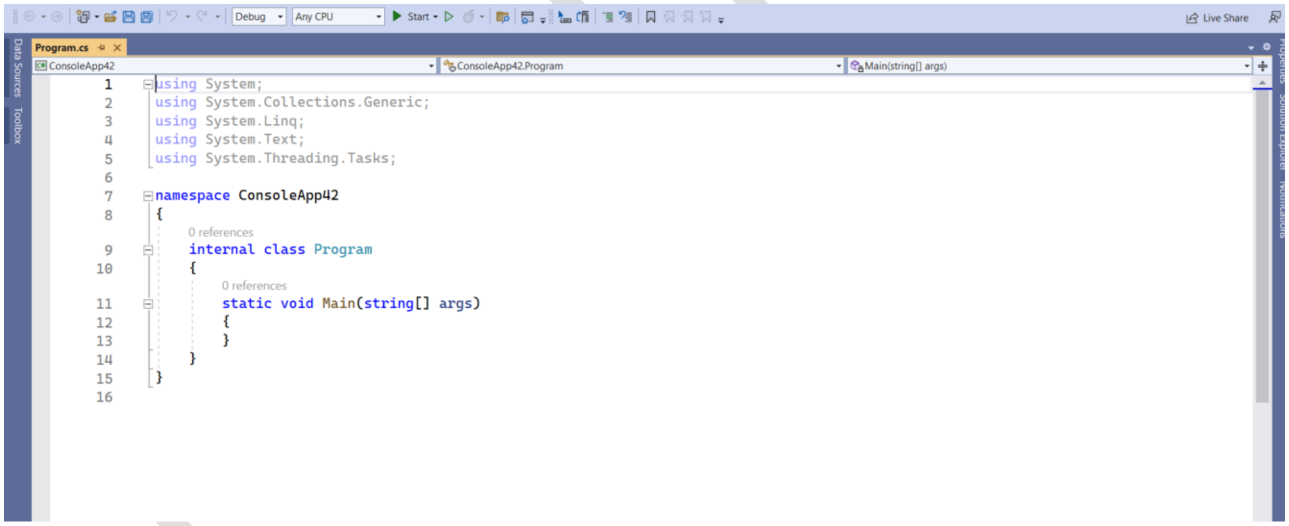
Step 6: Select below configurations

Project name: Specify the project name based on our requirement

Location: Specify the path of solution where we want to save the application

Step 7: Create

Step 8: The code will look like below.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp42
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
        }
    }
}
```

Name Space

Namespace is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace will not conflict with the same class names declared in another.

Namespace contains pre-defined classes, objects etc related to dot net.

Without namespace, we cannot write even a single line of code.

Ex: - using system;
using system .collections .generic;

Structure of C# .Net program

List of Class Libraries / Namespaces

NameSpace NameSpace name

{

Class Classname

{

Static void Main (string [ ] args)

{

Statements

}

}

}

Ex:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp42
{
    internal class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Points to Remember in C# program

- C# is case sensitive.
- All statements and expression must end with a semicolon (;).
- The program execution starts at the Main method and main is called as entry point of execution
- Program will generate output in command prompt
- Commented lines : Which will keep the code in program but it won't execute the code .**commented lines are 2 types**

Using //:

This is used to write a single-line comment, this is used to comment only single line

Using /* */:

Anything enclosed within /* and */, will treated as multi-line comments. This is used to comment more than one line.

Can a c# program run without the main () function?

No.

In the C# program, the main () function defines the **starting point of execution**. If a C# program **doesn't have a main () function** then no code statement will be executed in that program. In most cases the compiler gives an error, if it cannot find the main () function.

What will happen if we forgot to use a semicolon at the end of the statement in C#?

If we forgot to put the **semicolon at the end** of any **statement** in the C# then it will lead to a syntax error. In that case, an error message is issued by the compiler that a semicolon is expected.

What is a Compilation Error?

A compilation error, or compile-time error, is the error returned by the compiler if the syntax of the C# program is incorrect. When you try to compile a program with some syntax mistake, then the compiler will give an error, which is called a **Compile-time error**.

How to add comments in C#?

We can add **single-line** comments using the // at the starting of the comment text, and **multi-line** comment by enclosing the text within /* and */ in the C# program

Working with console class

Console class is used to work with input and output streams.
Console class is present in system namespace.
Methods/functions in console class

a) Write("message")
b) Write Line ("message")
c) Read( )
d) Read Line( )

Write ("message"): This method is used to write the message and keep the cursor in same line.
Write Line ("message"): This method is used to write the message and keep the cursor in next line.
Read ( ): This method is used read single character.
Read Line ( ): This method is used read group of characters from input stream.

Program to print Welcome on screen

```
Using System;
Using System.Collections.Generic;
Using System.Linq;
Using System.Text;

Namespace ConsoleApplication108
{
ClassProgram
    {

Staticvoid Main (string [] args)

        {

Console.WriteLine ("welcome to DotNet");

Console.ReadLine ();
        }

    }

}
```

Note: for checking output press F5

How to declare variables

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C# has a specific type, which determines the size and layout of the variable's memory. The range of values that can be stored within that memory. And the set of operations that can be applied to the variable.

Syntax:

Datatype variable name;

Ex: int a;

Ex: float b;

Ex: string data;

Integer accepts digits like 1, 2, 55, 999, and 10999

Float accepts decimals like 1.2f, 5.8f, 89.36f, 789.123f

In float we can declare up to 6 to 7 decimals

Char accepts single character like ‘c’, ‘1’,'*'

String accepts group of characters like “abc”, "12","12@","hfg12","naresh it"

Double accepts decimals like 1.2, 5.8, 89.36, and 789.123

In double we can declare up to 15 decimals

Int a, b, c;

a=40;

b=20;

c=55;

Int a=10, b=30, c=40;

String s1, s2;

s1="naresh it";

s2="Hyderabad";

String s1 ="naresh it ", s2="Hyderabad";

Ex: -

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
classProgram
    {
staticvoid Main(string[] args)
        {
int i = 13;

int j = 12;

int k = i + j;

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

Note:-

Here i ,j ,k are variables

DATA TYPES

Data types:-We are having two types of data types to assign values to variables.

1. **Value types**	2.**Reference types**
➢ Int	String
➢ Float	Object
➢ Char	
➢ Double	
➢ Byte	

Integer classified in below ways

Int 16 (2 bytes or 16 bits) int 16 is short int and it can store the values up to range **32,767**

Int 32 (32-bit integer) and it can store values up to **2,147,483,647**

Int 64 (8 bytes or 64 bits) it can store values up to **36,854,775,807**

Note: Values larger than the above mentioned maximum values will not be stored

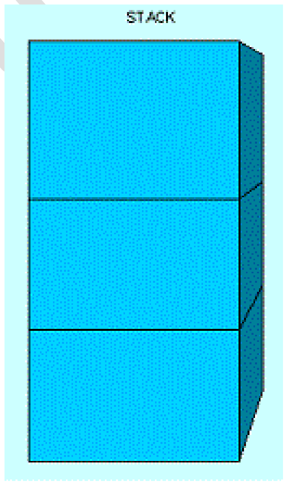
Variables of reference type store references to their data and Values stored in

Value type directly contain data in variables values stored in **stack**

Stack Memory:

- It is an array of memory.
- It is a LIFO (Last in First Out) data structure.
- In it data can be added to and deleted only from the top of it.

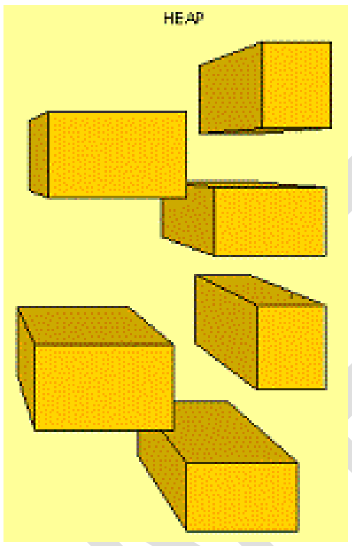
Memory Manages like below



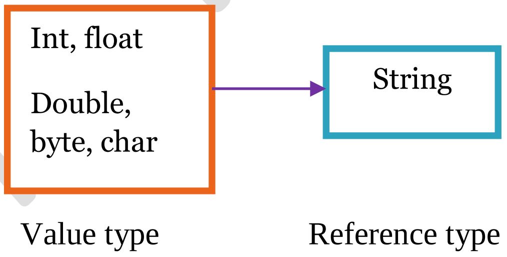
Heap Memory:

- It is an area of memory where chunks are allocated to store certain kinds of data objects.
- Data can be stored and removed in any order.

Memory Manages like below



Boxing: **-**It is a process of converting value type to reference type data type.



Syntax:. **ToString(); (dot.tostring)**

Ex: Value type variablename.tostring ()

Ex:

int i=10;
String s =i.tostring( );
Here i is value type and S is reference type variable

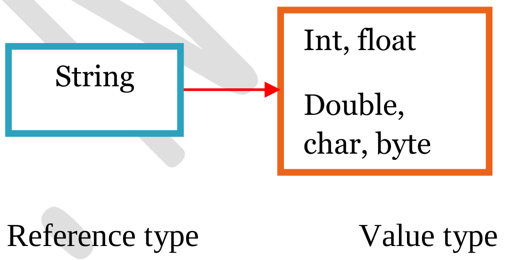
Ex:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
classProgram
    {
staticvoid Main(string[] args)
        {
double d = 12355.20;
string s;

            s = d.ToString ();

Console.WriteLine ("the value of d is" + d);
Console.WriteLine (" the value of is" + s);
Console.ReadLine ();
        }
    }
}
```

Unboxing:-It is a process of converting reference data type to values type .

Syntax:-

Data type. Parse (ref type varaiblename);



Ex:-

```
staticvoid Main(string[] args)0
        {
int i;
string s = "20";
            i = int. Parse(s);

Console.WriteLine ("the value of s is" + s);
Console.WriteLine (" the value of i is:" + i);
Console.ReadLine ();
        }
    }
}
```

Type casting: Converting any data type to value type.

Syntax:-

Convert .**To**DataType();

Ex: - double to int

```
staticvoid Main(string[] args)
        {
 double d = 101.222;
 int i;
 i = Convert.ToInt32(d);
 Console.WriteLine("the value of d is :" + d);
 Console.WriteLine(" the value of i is :" + i);
 Console.ReadLine();
        }
    }
}
```

Value type: - Here values are stored directly in the variables.

Ex; - int, float, double, char, long, byte

Reference type: - Here Values are stored in another address location.

Ex: - string, object

Conditional Statements

Conditional statements can use to execute different actions depending on a given condition.

1. Simple if
2. if else
3. else if
4. nested if
5. nested if else
6. Switch Case

Simple if:

Syntax for simple if:

```
If (condition)
{
Statements;
}
```

```
staticvoid Main(string[] args)
        {

Console.WriteLine("enter a vaalue");
int a = int. Parse(Console.ReadLine());
Console.WriteLine("enter b value");
int b = int. Parse(Console.ReadLine());
if (a > b)
            {
Console.WriteLine(" a is grater");

            }
Console.ReadLine();

        }
```

Note: If condition is true it executes the If block. Otherwise it will come out of the loop.

If else:

```
    if (condition)
{
  Statement
}
else
{
Statement
}
```

Ex:-

```
staticvoid Main(string[] args)
        {

Console.WriteLine("enter i vaalue");
int i = int. Parse(Console.ReadLine());
Console.WriteLine("enter j vslue");
int j = int. Parse(Console.ReadLine());

if (i > j)
            {

Console.WriteLine("i is grater");
            }
else
            {

Console.WriteLine("j is grater");
            }
Console.ReadLine();


        }
```

Note:- If condition is true it executes the If block. Otherwise it goes to else block.

Else if:

If (condition)
{
Statement:
}
Else if(cond)
{
Statement;
}
Else
{
Statements;
}

Note:- If condition is true it executes the If block. Otherwise it goes to else if block. Otherwise it goes to else block.

Nested if

In c#.Net nested if or if else or else if works one inside another if or else if statement(s).

Syntax:

```
if (condition)
{

if (condition)
{
Statements;
}
}
```

Nested if else syntax

```
if (condition1)
{
/* code to be executed if condition1 is true */
if (condition2)
 {
/* code to be executed if condition2 is true */
}
else
{
/* code to be executed if condition2 is false */
}
} else
{
/* code to be executed if condition1 is false */
}
```

Ex:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 100;
            int b = 200;
            if (a == 100)
            {
                //condition inside condtion
                if (b == 200)
                {
                    Console.WriteLine("Value of a is 100 and b is 200");
                }
            }
            Console.ReadLine();
        }
    }
}
```

Ex 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int value1 = 7, value2 = -23, value3 = 13;
            if (value1 > value2)
            {
                if (value1 > value3)
                {
                    Console.WriteLine("largest is"+ value1);
                }
                else
                {
                    Console.WriteLine("largest is"+ value3);
                }
            }
            else
            {
                if (value2 > value3)
                {
                    Console.WriteLine("largest is"+ value2);
                }
                else
                {
                    Console.WriteLine("largest is"+ value3);
                }
            }
        }
    }
}
```

Switch:-

This is a condition based statement. It used to check only one condition.

Note: Performance wise switch case is better compare with if Condition.

Syntax:

Switch (condition)
{
Case 1:
Statements;
Break;
Case 2:
Statements;
Break;
-
Default:
        Statements;
Break;
}

Ex;-

```
staticvoid Main(string[] args)
        {
Console.WriteLine("enter a number");
int i = 10, j = 10;
int n = int.Parse(Console.ReadLine());
switch (n)
            {
case 1:
Console.WriteLine("sum" + (i + j));
break;

case 2:
Console.WriteLine("sub" + (i - j));
break;
case 3:
Console.WriteLine("mul" + (i * j));
break;
case 4:
Console.WriteLine("div" + (i / j));
break;
default:
Console.WriteLine("enter the number between 1-4");
break;
            }
Console.ReadLine();

}
```

LOOPS

There may be a situation, when you need to execute a block of code several numbers of times. A loop statement allows us to execute a statement or group of statements multiple times. While we have many loop statements.

While loop:-

It is a condition based loop

It should be used when we don't have the range.

Syntax: -

Initialization;

While (condition)

{

Statements;

Increment /decrement;

}

```
staticvoid Main(string[] args)
        {

int i = 1;
while (i <= 10)
            {
Console.WriteLine(i);
i++;
            }
Console.ReadLine();
        }
```

Do while loop:-

Mostly if we don't know the exact range we will go for while and **do-while** loop.

Syntax:-

do

{

Increment /decrement;

}

While (condition);

Ex:-

```
staticvoid Main(string[] args)
        {

int i = 1;
do
            {

Console.WriteLine(i);
i++;
            }
while (i <= 10);

Console.ReadLine();
        }
```

Note:-In while loop first it will check for condition and then execute statements.

In do**-while** loop first it will execute statements and then it will check the condition.

For Loop

For loop includes all three characteristics as initialization, termination and increment/decrement in a single line.

Syntax:-

for (initialization; condition; increment or decrement)

{

Statements;

}

Execution Flow.

1. Initialization
2. Condition.
3. Statements.
4. Inc or dec.
5. Condition
6. Statements.
7. Inc or dec.
8. It will repeat until condition fails and then loop will terminate.

Ex:-

```
staticvoid Main(string[] args)
        {
int i;
for (i = 0; i <= 10; i = i + 2)
Console.WriteLine("the value of i is :" + i);
Console.ReadLine();

        }
```

Ex 2:-

```
staticvoid Main(string[] args)
        {
int i, sum = 0;
for (i = 1; i <= 5; i++)
            {
sum = sum + i;
            }
Console.WriteLine("sum value of  is :" + sum);
Console.ReadLine();

        }
```

Ex 3:-

```
staticvoid Main(string[] args)
        {
int i;
for (i = 9; i >= 0; i--)
            {

Console.WriteLine ("sum value of is" + sum);


            }
Console.ReadLine ();

        }
```

ARRAY

Array is collection of homogeneous data items. Basically we have three types of arrays.

1. **Single dimension**
2. **Multi dimension**
3. **Jagged array**

- By using array we can easily perform searching and sorting.
- We can store more than one value temporarily.
- Array is a user defined data type which is used to store multiple values of single variable of same data type.
- Index of the array will always, with 0 and ends with (size -1).
- Array value can be identified by array index value.

Syntax: -

Single dimension: -

Datatype [ ] arrayname = new datatype [size]

Multi dimension: -

Datatype [,] arrayname= new datatype [Row Size, Column Size]

Jagged array:-

Datatype [ ][ ] arrayname = new Datatype [rows][ ];

Single Dimension Array example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication29
{
    class Program
    {
        static void Main(string[] args)
        {
            int r, i;
            Console.WriteLine("enter row size of array");
            r = int.Parse(Console.ReadLine());
            int[] a = new int[r];
            //int[,] a = new int[3];
            Console.WriteLine("enter array elements");
            for (i = 0; i < r; i++) // 3
            //for (i = 0; i < 3; i++) //After entering row size this loop will
look like this
```

```
            { //a[0] a[1]

                a[i] = int.Parse(Console.ReadLine());
            }
            //int[] arr =  { 1, 2, 3 }; here arary values contructed
            Console.WriteLine("entered array elements are");
            for (i = 0; i < r; i++)
            {
                Console.Write(a[i] + " "); //for printing rows
            }

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

Multi Dimension Array example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication29
{
    class Program
    {
        static void Main(string[] args)
        {
            int r, c, i, j;
            Console.WriteLine("enter row size,column size values");
            r = int.Parse(Console.ReadLine());
            c  = int.Parse(Console.ReadLine());
            int[,] a = new int[r, c];
            //int[,] a = new int[3, 3];
            Console.WriteLine("enter array elements");
            for (i = 0; i < r; i++) // 3 , 3
            //for (i = 0; i < 3; i++) //After entering row size this loop will
look like this
            { //a[0,0] a[0,1]
                for (j = 0; j < c; j++)
                // for (j = 0; j < 3; j++) //After entering column size this
loop will look like this
                {
                    a[i, j] = int.Parse(Console.ReadLine());
                }
            }
            //int[,] arr = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } }; here
arary values contructed
            Console.WriteLine("entered array elements are");
            for (i = 0; i < r; i++)
            {
                for (j = 0; j < c; j++)
                {
                    Console.Write(a[i, j] + " "); //for printing rows
                }
                Console.WriteLine(); // new line at each row
            }
            Console.ReadLine();
        }
    }}
```

Jagged Array

A jagged array (also known as a ragged array) is an array of arrays in which the member arrays in turn can be of different dimensions and sizes

What is the difference between array and jagged array in C#?

In a multidimensional array, each element in each dimension has the same, fixed size as the other elements in that dimension. In a jagged array, which is an array of arrays, each inner array can be of a different size.

Syntax: Datatype [ ][ ] arrayname = new Datatype [rows][ ];

Ex:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp42
{
    internal class Program
    {
        public static void Main()
        {
            int[][] arr = new int[2][];// Declare the array

            arr[0] = new int[] { 11, 21, 56, 78 };// Initialize the array
            arr[1] = new int[] { 42, 61, 37, 41, 59, 63 };

            // Traverse array elements
            for (int i = 0; i < arr.Length; i++)
            {
                for (int j = 0; j < arr[i].Length; j++)
                {
                    Console.Write(arr[i][j]+" ");
                }
                Console.WriteLine();
            }
            Console.ReadLine();
        }
    }
    }
```

OOPS

Oops means Object Oriented Programming System.

- Oops is an approach which is used to design computer program by using classes and object.

Class

Class is **closed container** where we can **bind variables and methods**. In C#.Net, class is declared by **class** keyword.

Points remember

- In C#.Net, we will have variables and methods inside a class.
- The purpose of variable is for accessing a value but not storing a value.
- Object is used to call the method and syntax is objectname.methodname();
- Whenever we define a class, there is no memory allocated for variables **Object** will allocate sufficient memory to variables **without object and memory we can't work with variables and class**.

Syntax declaring a class: -

class class-name

{

Variables;

Methods ()

}

Object: - Object is Instance of class.

Syntax: - classname objectname =new classname ();

Ex: - class A (Here A is a classname)

A obj =new A ();

Whenever we create an object, memory is allocated to its base class &its derived class.

What is method in C#?
A method is a group of statements which helps to perform a specific task.

Methods in object class:-

1. Equals ( )
2. GetHashCode ( )
3. ToString ( )
4. GetType ( )

Ex:-

```
classstudent
    {
intstno;
stringstname;
voidsetvalues()
        {
stno = 101;
stname = "NareshIT";
        }

voiddisplayvalues()
        {
Console.WriteLine("student numis" + stno);
Console.WriteLine("student name is" + stname);
        }

staticvoid Main(string[] args)
        {

student s = newstudent();
s.setvalues();
s.displayvalues();
Console.ReadLine();
        }
    }
```

Main concepts in oops:-

We have four concepts in OOPS

- Encapsulation
- Data abstraction
- Inheritance
- Polymorphism

Encapsulation: -

Encapsulation means adding or binding of data members with in a class.

Data members can be methods or variables

Data abstraction: -

Data Abstraction is a mechanism to show only relevant data to the user. It shows required data and hides unwanted data

Inheritance: -

Inheritance is a mechanism of getting or accessing data from base class to derived class.

Inheritance should access between base class and derived class.

Base class: -The class which gives variables and methods,

Derived class:-The class which takes variables or methods from base class.

Syntax: - For inheritance process we use ":" (colon) for derived class.

```
class A       ( Here A is base class/ super class/ parent class)
  {
  Variable;
  Methods( )
  }
  Class B: A     (Here B is derived class/sub class/child class)
  {         → B inheriting from A
  Variables;
  Methods( )
   }
```

Types of inheritance:- Basically, we have four types of inheritance.

1. Single level inheritance
2. Multi-level inheritance
3. Multiple inheritance
4. Hybrid inheritance
5. Hierarchical inheritance

1) Single level inheritance:-It is a process of getting or accessing properties from one base class to another class. (Derived class)

Syntax: -

class A

{

Variables;

Methods( )

}

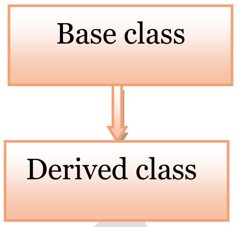
Class B: A

{

Variables;

Methods( )

}



Ex:-

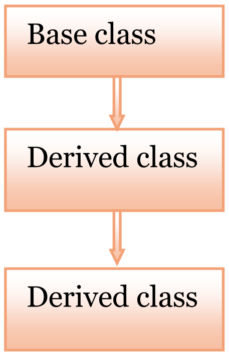
```
classbranches
    {
intbcode;
stringbname, baddress;
publicvoidgetdata()
        {
Console.WriteLine("enter branch code");
bcode = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter branchname");
bname = Console.ReadLine();
Console.WriteLine("enter branchaddress");
baddress = Console.ReadLine();
        }
publicvoiddisplaydata()
        {
Console.WriteLine("branch code is" + bcode);
Console.WriteLine("branch name is" + bname);
Console.WriteLine("branch address is" + baddress);
        }
    }
classemployee : branches
    {
intempid, eage;
stringenmae, eaddress;
publicvoidgetempdata()
        {
Console.WriteLine("enter employee details");
empid = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter employee age");
eage = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter employee name");
enmae = Console.ReadLine();
Console.WriteLine("enter employee address");
eaddress = Console.ReadLine();
        }

publicvoiddisplayempdata()
        {
Console.WriteLine("employee id is" + empid);
Console.WriteLine("employee age is" + eage);
Console.WriteLine("employee name is" + enmae);
Console.WriteLine("employee address is" + eaddress);
        }
    }
classinheritance
    {
staticvoid Main(string[] args)
        {
employee e1 = newemployee();
e1.getdata();
e1.getempdata();
e1.displaydata();
e1.displayempdata();
Console.ReadLine();
        }
    }
```

2) Multi level inheritance:-

It is a Process of getting properties from one base class to more than one derived class

Syntax: -

Class A
{
Variables;
Methods( )
}
Class B: A
{
Variables;
Methods( )
}
Class C: B
{
 Variables;
Methods()
}



Ex:-

```
classbranches
    {
intbcode;
stringbname, baddress;
publicvoidgetdata()
        {
Console.WriteLine("enter branch code");
bcode = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter branch name");
bname = Console.ReadLine();
Console.WriteLine("enter branch address");
baddress = Console.ReadLine();
        }
publicvoiddisplaydata()
        {
Console.WriteLine("branch code is" + bcode);
Console.WriteLine("branch name is" + bname);
Console.WriteLine("branch address is" + baddress);
        }
    }
classemployee : branches
    {
intempid, eage;
stringenmae, eaddress;
publicvoidgetempdata()
        {
Console.WriteLine("enter employee details");
empid = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter employee age");
eage = Convert.ToInt32(Console.ReadLine());
enmae = Console.ReadLine();
eaddress = Console.ReadLine();
        }

publicvoiddisplayempdata()
        {
Console.WriteLine("employee id is" + empid);
Console.WriteLine("employee age is" + eage);
Console.WriteLine("employee name is" + enmae);
Console.WriteLine("employee address is" + eaddress);
        }
    }

classsalary : employee
    {
double basic, da, hra, gross;

publicvoidgetsaldetails()
        {
Console.WriteLine("enter basicsal");
basic = Convert.ToDouble(Console.ReadLine());
        }

publicvoidcalcuate()
        {
da = 0.4 * basic;
hra = 0.3 * basic;
gross = basic + da + hra;
        }
publicvoiddisplaysaldetails()
        {
Console.WriteLine("employee basic is" + basic);
Console.WriteLine("employee da is" + da);
Console.WriteLine("employee hra is" + hra);
Console.WriteLine("employee gross is" + gross);

        }
    }

classinheritance
    {
staticvoid Main(string[] args)
        {
salary e1 = newsalary();
e1.getdata();
e1.getempdata();
e1.displaydata();
e1.displayempdata();
e1.getsaldetails();
e1.calcuate();
e1.displaysaldetails();

Console.ReadLine();
        }
    }
```

3) Multiple inheritances: -

It is a process of getting properties from more than one base class to single derived class.

Note: - class does not support multiple inheritances

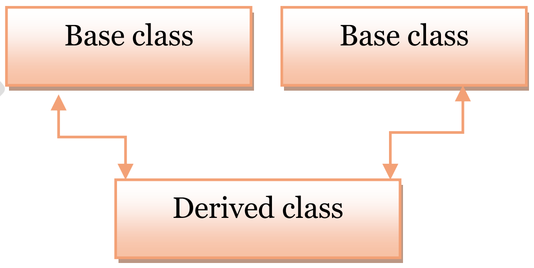
Ex:-

Class A

{

Void add ( )

}



Class B

{

Void add ( )

Void sub ( )

}

Class C: A, B

{

}

Public main string args [ ]

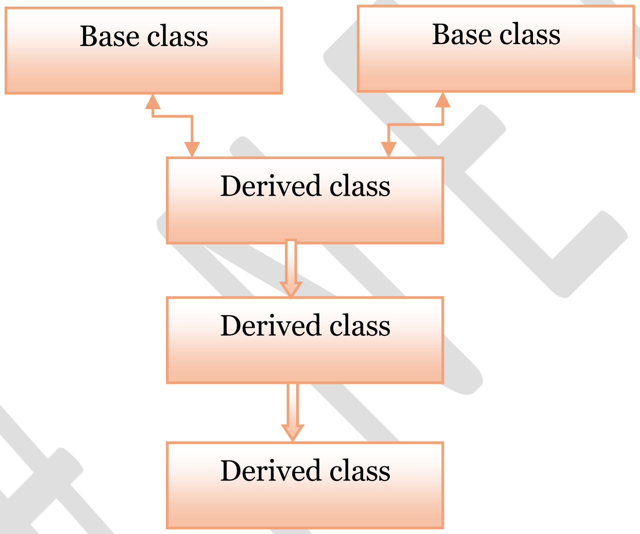
{

```
C  C1 =new C ( );
C1.add ();
 }
}
```

Explanation: -Here method add ( ) is declared in two classes, so the **main class** will get confused to take methods of which class and it may give error output. Hence class does not support multiple inheritances and that situation is called **ambiguity**.

4) Hybrid inheritance:-

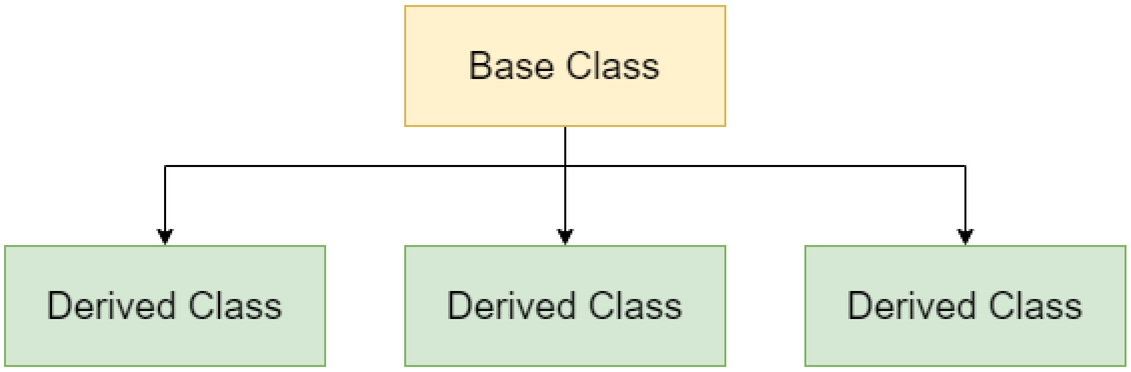
It is a combination of Multiple and Multi level inheritance.



Hierarchical Inheritance:

Hierarchical inheritance is a type of inheritance in which **multiple classes** inherit from a **single base class**. The **base class** shares **many of the same** properties as the parent class.

A single base class generates multiple derived classes like shown below.



Ex

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    public class Program
    {
        public class Parent
        {
            public void DisplayParent()
            {
                Console.WriteLine("I am from parent class");
            }
        }
        public class Child1 : Parent
        {
            public void DisplayChild1()
            {
                Console.WriteLine("I am from Child1 class");
            }
        }
        public class Child2 : Parent
        {
            public void DisplayChild2()
            {
                Console.WriteLine("I am from Child2 class");
            }
        }
        public static void Main(string[] args)
        {
            Child1 obj1 = new Child1();
            Child2 obj2 = new Child2();
            obj1.DisplayChild1();
            obj1.DisplayParent();  // accessing parent class
            obj2.DisplayChild2();
            obj2.DisplayParent();   // accessing parent class
            Console.ReadLine();
        }
    }
}
```

Polymorphism:

Poly means **many** Morphing means **behaviors**. If we change the values then the behavior of methods will change.

Here we have two types which occur at Compile Time and Run Time respectively.

1. Method Overloading.
2. Method Overriding.

Method: - Method is a sub-program in a class that operate through a designated class. A method is a group of statements that together perform a specific task or action.

Syntax: -

Return type method name ( )

{

}

For accessing the method we should give

Objectname.method name ( );

Here the return type may Void or Data Type

Note: -

1. If we write Void as return type then no need to return value
2. If we write Data type as return type we should return that particular data type value.

Method overloading: -

Same method name but different parameters. This occurs at compile time.

Ex:-

```
Public add ()
{
C=a+b;
}
Public add(float a ,int b)
{
C=a+b;
}
Public add (int a, float b)
{
C=a+b;
}
```

Method overriding: -

Same method name and same parameters. It occurs at runtime.

Ex:-

```
Public virtual void add (int a, int b) //base class method

{

C=a+b;

}

Public override void add (int a, int b) //derived class method

{

}
```

Differences between overloading and overriding

Overloading	Overriding
1. Same method name but different parameters.	1.Same method name and same parameters
2.It occurs with same class only	2.It occurs with base class and derived class
3. It occurs at compile time or static or early binding	3.It occurs at runtime or dynamic or late binding

Note:-

For overriding process we have to use **virtual** keyword in base class and **override** keyword in derived class.

VARIABLES

Variable: -It is the name given for a particular memory location.

Purpose of variable is to access the value.

We can change the variable value at the time of execution.

Types of variables:-We have three types of variables.

1. Instance variables
2. Static variables
3. Local variables

Instance variables: - Variables are declared inside the class and outside the method.

It is not required to declare with static keyword.

Memory for instance variable is allocated after creating an object.

Syntax: -

```
Class A
{
int A;
int B;
Void add ()
{
}
}
```

Local variable:-
Variables are declared inside method.

The memory is allocated at the time of execution of method

The scope is within the method i.e., if we declare a variable inside the method, we can’t access that variable anywhere else.

Syntax: -

```
class A
{
Void add()
{
int a;
int b;
}
}
```

Note: -
1) If the variable required to declare in method, then that variable should declared as a local variable.
2) If variable is required for more than one method, then declare it as **instance variable**

Static variable:-

1. Static means single copy of value and When we create static variable heap memory will be allocated
2. If you we want make a variable or method as static we must use static keyword.
3. For accessing static variables or methods no need to create an object.
4. We can't use non static variables in static methods, but we can use static variables in non-static methods also.

Points to remember

- Can we use static variable in non-static method? --> **Yes**
- Can we use static variable in static method? --> **Yes**
- Can we use non static variable in static method? --> **No**
- Can we use object for calling static method? --> **No**
- How to access a static method? --> **Classname.MethodName ()**
- How to call non static method? --> **objectname.methodname ()**

Ex:-

Static int varaiblename;

Static void add ()

{

}

For accessing Static Methods:-

We have to use **Classname . Methodname ();**

Write a program to understand other data types in real time ?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    class program
    {
        static void Main(string[] args)
        {
            decimal dec = 389.5m;
            bool isCSharpFun = true;
            bool isFishTasty = false;
            DateTime today = DateTime.Now;
            Guid obj = Guid.NewGuid();
            var a = 'f';
            var b = "test";
            var c = 30.67d;
            var d = false;
            var e = 54544;
            //e="45";
            // Dynamic variables
            dynamic val1 = "test";
            val1 = 3234;
            // Get the actual type of dynamic variables Using GetType() method
            Console.WriteLine("Get the actual type of val1: {0}",
val1.GetType().ToString());
            // Display the type
            Console.WriteLine("Type of 'a' is : {0} ", a.GetType());
            Console.WriteLine(e);
            Console.WriteLine("New Guid is " + obj.ToString());
            // DateTime date1 = new DateTime(2015, 12, 25);
            // Console.WriteLine(date1.ToString()); // 12/25/2015 12:00:00 AM
            Console.WriteLine(dec);
            Console.WriteLine(isCSharpFun);   // Outputs True
            Console.WriteLine(isFishTasty);   // Outputs False
            Console.WriteLine(today);
            Console.ReadLine();
        }
    }
}
```

Enum Variable

An enumeration is a set of named integer constants. An enumerated type is declared using the enum keyword.

C# enumerations are value type data type. In other words, **enumeration** contains its **own values**.

Syntax

enum <enum_name> {
enumeration list
};

Ex: enum Days {Sun, Mon, Tue, Wed, Thu, Fri, Sat};

Where,

- The **enum_name** specifies the enumeration type name.
- The enumeration list is a **comma-separated** list of identifiers.

Each of the symbols in the enumeration list stands for an integer value, one greater than the symbol that precedes it. By default, the value of the first enumeration symbol is 0.

Ex

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
   public class Program
    {
        enum Days { Sun, Mon, tue, Wed, thu, Fri, Sat };
        static void Main(string[] args)
        {
            int WeekdayStart = Convert.ToInt32( Days.Mon);
            int WeekdayEnd = Convert.ToInt32(Days.Fri);

            Console.WriteLine("Monday "+ WeekdayStart);
            Console.WriteLine("Friday "+ WeekdayEnd);
            Console.ReadKey();
        }
    }
}
```

Different between instance and local variable?

Instance variable	Local variable
The scope of instance variable is within the class i.e., we can access anywhere in the class	The scope of local variable is within the method i.e., we can't access outside method
Memory is allocated after creating the an object	Memory is allocated at the time of method execution

Access modifiers:-

It tells the range of variables where we can use in program.

Access modifiers	Same Assembly			Other Assembly	
	Same class	Derived class	Non-derived class	Derived class	Non-derived class
Private	✓	x	x	x	x
Protected	✓	✓	x	x	x
Internal	✓	✓	✓	x	x
Public	✓	✓	✓	✓	✓

1) Public:-

If we declare a variable as public, then we can use those variables in same project and also in new project also.

2) Private: -

If we declare a variable as private, then we can use those variables only in same class.

3) Internal:-

If we declare a variable as internal, then we can use those variables in same assembly only

4) Protected:-

If we declare a variable as protected, then we can use those variables in same class and derived class

CONSTRUCTOR

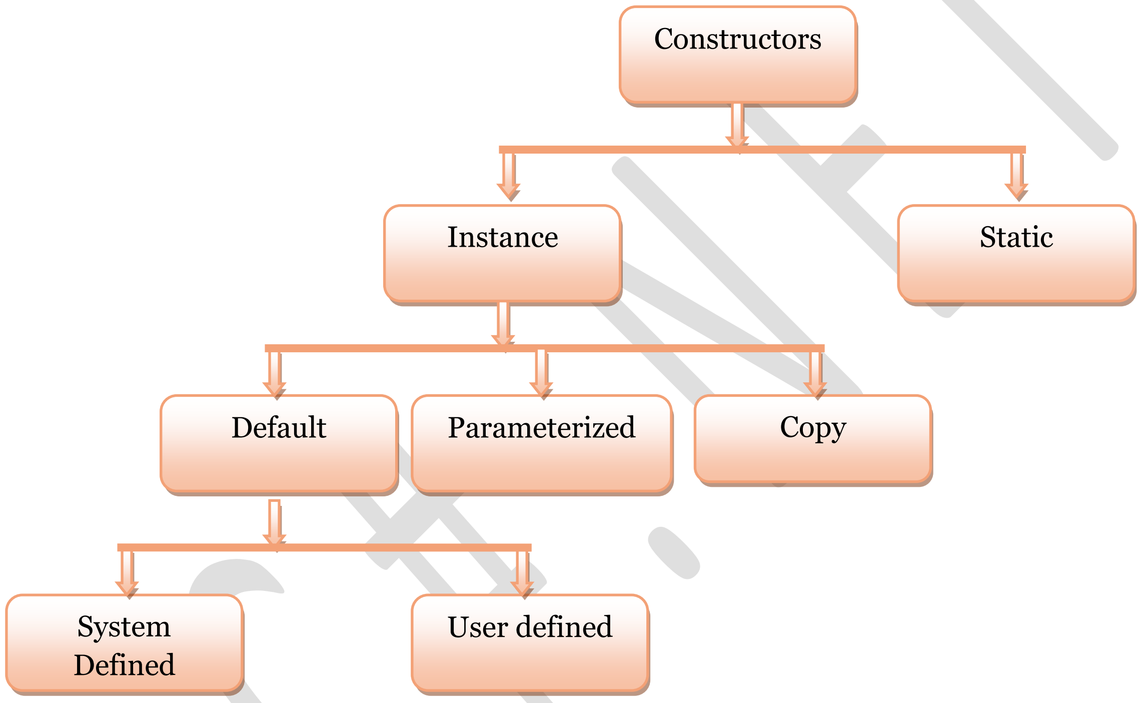
Constructor is used to initialize the values to variables in a class.

It is used to pass the values.

Syntax:-

```
Access-modifier classname ( )
{
}
```

Types of constructors:-



System defined default constructor:-

In this constructor default values will be initialized to variables. That means for integer Zero will be initialized and for string variables Null will be initialized. If we create "N" number of objects we will pass single value for "N" number times.

Ex:-

```
classemployee
    {
intempno, emid;
stringemname, eaddress;

publicvoiddislayempdata()
        {
Console.WriteLine("empno is" + empno);
Console.WriteLine("emid is" + emid);
Console.WriteLine("empname is" + emname);
Console.WriteLine("emp address is" + eaddress);
        }
    }
classconstructor
    {
staticvoid Main(string[] args)
        {
employee obj = newemployee();
obj.dislayempdata();
Console.ReadLine();

        }
    }
```

User defined default constructor:-

In this constructor user defined values will be initialized to variables. That means here user will initialize the value.If we create "N" number of objects we will pass single values for "N" number times.

Ex:-

```
classemployee
    {
intempno, emid;
stringemname, eaddress;
public employee()
        {
empno = 101;
emid = 102;
emname = "Dotnet";
eaddress = "hyd";
        }
publicvoiddislayempdata()
        {
Console.WriteLine("empno is" + empno);
Console.WriteLine("emid is" + emid);
Console.WriteLine("empname is" + emname);
Console.WriteLine("emp address is" + eaddress);
        }
    }
classconstructor
    {
staticvoid Main(string[] args)
        {
employee obj = newemployee();

obj.dislayempdata();
employee obj2 = newemployee();

obj2.dislayempdata();
Console.ReadLine();


        }
    }
```

Parameterized constructor:-

In this constructor the parameters will be passed while declaring the constructor.
And the values will be passed while creating an object. If we create “N” number of objects we can pass “N” number values. This is the main advantage while comparing with default constructor.

Ex:-

```
classemployee
    {
inteno, eage;
stringename, eaddress;
public employee(int id, string s1, string s2, int age)
        {
eno = id;
eage = age;
ename = s1;
eaddress = s2;
        }

publicvoiddislayempdata()
        {
Console.WriteLine("empno is" + eno);
Console.WriteLine("emid is" + eage);
Console.WriteLine("empname is" + ename);
Console.WriteLine("emp address is" + eaddress);
        }
    }
classconstructor
    {
staticvoid Main(string[] args)
        {
employee e1 = newemployee(101, "dotnet", "hyd", 25);
e1.dislayempdata();


employee e2 = newemployee(102, "dotnet2", "vsp", 58);
e2.dislayempdata();

Console.ReadLine();


        }
    }
```

Copy constructor:-

In copy constructor we will copy parameters from one object to another object.

```
classstudent
    {
intstno;
stringstname;
public student()
        {
stno = 101;
stname = "dotnet";
        }
student(student obj)
        {
stno = obj.stno;
stname = obj.stname;
        }
voiddisplaydata()
        {
Console.WriteLine("student name is" + stname);
Console.WriteLine("student number is" + stno);
        }
staticvoid Main(string[] args)
        {
student s1 = newstudent();
s1.displaydata();
student s2 = newstudent();
s2.displaydata();
Console.ReadLine();
        }
    }
```

Static constructor:-

In Static constructor we have to use **Static key word.** We have to declare variables with Static key word.

Ex:-

```
classa
    {
static a()
        {
Console.WriteLine("i am static constructor");
        }
staticvoid Main(string[] args)
        {
Console.WriteLine("i am static main method constructor");
Console.ReadLine();
        }
    }
```

Solid Principles

Single dependency principle

Every module should implement for one purpose only.

Ex: suppose if we have a validation class that should contain information or methods related to validations only and no other implementations allowed in that particular class.

Open close principle

Open for extension and close for modifications.

Ex: we must design a class open for modification that means if we have new requirement we should ready to implement that and similarly if unit testing or testing completed for all the classes then we should close modifications until and un less we found new some bug..

Liskov substution principle

Always use derived class instead of parent class that means we shouldn't change parent class features or Skelton in child class.

Interface segregation principle

User shouldn't be forced to implement long or big interfaces instead of it they can implement small interfaces with required methods.

Dependency inversion principle

High level classes shouldn't depend on low level classes

Ex: If high level classes implemented business logic and low level classes implemented database related logic so here high level classes shouldn't depend on low level classes.

Abstract class

A class which contains one or more abstract function or non-abstract functions is known as abstract class.

- To make any class as abstract use **abstract** keyword
- It is compulsory to create new derived class from abstract class.
- Abstract METHODS contains only method heading but not method body
- Abstract METHOD Tells **"what to do"** but not "**how to do ".**
- Abstract class is derived in order to provide functionality to its abstract functions
- It contain both abstract and non-abstract function

Abstract function:

A function which contains only declaration but not implementation and body is known as abstract function.

To make any function as abstract we use keyword "abstract". **Overriding** is compulsory for abstract function.

Ex:-

```
abstractclassemployee
    {
protectedinteid, eage;
protectedstringename, eaddress;
publicabstractvoidgetempdata();
publicvirtualvoiddislayempdata()
        {
Console.WriteLine("employee id is" + eid);
Console.WriteLine("emloyee name is" + ename);
Console.WriteLine("emplyee address" + eaddress);
Console.WriteLine("employee age is" + eage);
        }
    }
classsalary : employee
    {
doublebonous, ca;
publicoverridevoidgetempdata()
        {
Console.WriteLine("entermanager id details");
eid = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter ename");
ename = Console.ReadLine();
Console.WriteLine("bonous is");
bonous = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("enter CA");
ca = Convert.ToDouble(Console.ReadLine());
        }
publicoverridevoiddislayempdata()
        {
Console.WriteLine("manager id is" + eid);
Console.WriteLine("manager name is" + ename);
Console.WriteLine("manager bonous is" + bonous);
Console.WriteLine("manager ca is" + ca);
        }
    }
classclsabstract
    {
staticvoid Main(string[] args)
        {
salary s1 = newsalary();
s1.getempdata();
s1.dislayempdata();
Console.ReadLine();

        }
    }
```

Interface: -

It is an agreement between itself and its implemented class.

- Class does not support multiple inheritance to overcome this problem we are going for interface.
- We can perform multiple inheritance using interface.
- We can implement abstract Functions using interface concept.

Ex: -

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    public class program
    {
        interface nokia
        {
            void sendsms();
            void call();
        }
        interface newnokia
        {
            void bluetooth();
            void wifi();
        }
        public class nokia1100 : nokia //interface1
        {
            public void sendsms()
            {
                Console.WriteLine("we can send sms");
            }
            public void call()
            {
                Console.WriteLine("we can make a call from  this");
            }
        }
        // two interfaces inherited here
        public class nokia3500 : nokia, newnokia
        {
            public void sendsms()
            {
                Console.WriteLine("we can send sms and images");
            }
            public void call()
            {
                Console.WriteLine("we can make a call and conference call from 
this");
            }
            public void bluetooth()
            {
                Console.WriteLine("we can use bluetooth option hre");
            }
            public void wifi()
            {
                Console.WriteLine("we connect to internet using wifi");
            }
        }

        //one class and interface
        public class nokia35000 : nokia3500, newnokia
        {
            //stattements
        }
        static void Main(string[] args)
        {
            Console.WriteLine("nokia-1100");
            Console.WriteLine("nokia 3500");
            nokia3500 n = new nokia3500();
            n.sendsms();
            n.call();
            n.bluetooth();
            n.bluetooth();
            n.wifi();
            Console.ReadLine();
        }
    }
}
```

1) We performed multiple in heritance here using Interface.
2) Here interfaces (**nokia, newnokia**) are fixing the rules the implementing classes (nokia1100**, nokia7210**) are following the rules.

Q) Difference between abstract class and interface?

Abstract class	Interface
1.A class which contains abstract function and non-abstract functions called abstract class	1. which contains all abstract functions is known as interface
2.A class may contain non-abstract functions	2. Interface can't contain non-abstract functions.
3.It contains all members of class	3. It can contain only abstract functions, properties, indexes, events and cannot contain non-abstract functions, data fields. Constructors.
4.Here we have to use keyword “abstract”	4.Here we have to use keyword ‘interface”
5.Abstract class can't be used to implement multiple inheritance	5.Interface can be used to implement multiple inheritance
6.By default, abstract class members are not public & not abstract	6.By default, interface members are public

Properties

1. Property is a member of class used to write the data in the data field and read the data from the data fields
2. A property can never store a value but used to transfer a value.
3. Properties are members that provide a flexible mechanism to read & write, the values of private fields
4. Properties enable a class to explore a public way of getting and setting values while hiding implementation of verification code.
5. To perform read & write operations, property can contain two accessors or methods.
6. Properties provide the convenience or public data members without the risk that come with unprotected, uncontrolled &unverified access to an object data.

Types of properties: -

We have three types of properties like,

1. Write only

2. Read only

3. Read write

Write only

Syntax:-

Access modifier data type property name
{
set
{
Variablename=value;
}

}

Read only

Syntax:-

Access modifier data type property name
{

Get

{
return variablename;
}
}

Ex:-

```
classstudent
    {
int num1, num2, result;
publicint pnum1
        {
set { num1 = value; }
        }

publicint pnum2
        {
set { num2 = value; }
        }
publicintpresult
        {
get { return result; }
        }
publicvoid add()
        {
result = num1 + num2;
        }
publicvoidmultily()
        {
result = num1 * num2;
        }
    }
classproperty2
    {
staticvoid Main(string[] args)
        {
student s1 = newstudent();
Console.WriteLine("enter any two number");
            s1.pnum1 = Convert.ToInt32(Console.ReadLine());
            s1.pnum2 = Convert.ToInt32(Console.ReadLine());
s1.add();
Console.WriteLine("sum is" + s1.presult);
s1.multily();
Console.WriteLine("multiplication is" + s1.presult);
Console.ReadLine();
        }
    }
```

Read-write

Syntax:-

Access modifies data type property name

{

set {variablename =value ;}

get {return variablename;}

}

Ex:-

```
classemployee
        {
intempid, eage;
publicintpempid
            {
set
                {
empid = value;
                }
get
                {
returnempid;
                }
            }
publicintpeage
            {
set
                {
```

```
eage = value;
                }
get
                {
returneage;
                }
            }
        }
classproperty1
        {
staticvoid Main(string[] args)
            {
employee e1 = newemployee();
Console.WriteLine("enter emloyee number");
                e1.pempid = Convert.ToInt32(Console.ReadLine());
                e1.peage = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("emloyee id is" + e1.pempid);
Console.WriteLine("emp name is" + e1.peage);
Console.ReadLine();
            }
        }
    }
```

Sealed Class

A class which cannot allow a new class to derive is called as sealed class

- To make a class as sealed we must use key word **Sealed**
- Sealed class is completely opposite to abstract class
- Sealed class must write bottom class within the inheritance process
- Sealed class cannot contains abstract function
- Sealed class never used as base class
- It is mainly used to avoid inheritance

Ex:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace sealedclassprgrm
{
    public class Program
    {
        public class employee
        {
            protected int eid, eage;
            public virtual void getemdata()
            {
                Console.WriteLine("enter empid" + eid);
                eid = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("enter empage" + eage);
                eage = Convert.ToInt32(Console.ReadLine());
            }
            public virtual void displayempdata()
            {
                Console.WriteLine(" empid is" + eid);
                Console.WriteLine(" eage is" + eage);
            }
        }
        public sealed class manager : employee
        {
            double bonous, ca;
            public override void getemdata()
            {
                Console.WriteLine("enter eid details");
                eid = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("enter bonous details");
                bonous = Convert.ToDouble(Console.ReadLine());
                Console.WriteLine("enter CA details");
                ca = Convert.ToDouble(Console.ReadLine());
            }
            public override void displayempdata()
            {
                Console.WriteLine(" empid is" + eid);
                Console.WriteLine(" manager bonous is" +bonous);
                Console.WriteLine(" ca is" +ca);
            }
        }
        static void Main(string[] args)
        {
            manager m1 = new manager();
            m1.getemdata();
            m1.displayempdata();
            Console.ReadLine();
        }
    }
}
```

Difference between abstract class and sealed class

Abstract class	Sealed class
1.A class contain one or more abstract functions or non-abstract functions	1.A class which can't derive a new class
2.It contains abstract functions and non-abstract functions	2.a class contains non-abstract functions
3.we have to use a keyword "**abstract** "	3.we have to use a keyword "**sealed** "
4. Creating a new class from abstract class is compulsory to consume it.	4. It is not possible to derive or create a new class.
5. We never create object to abstract class.	5. We always needs to create object to sealed class.
6.Abstract class cannot be bottom most class within the inheritance	6.It is a compulsory bottom class
7.It should be base class only	7.it should not be base class

Partial Class

Partial class is used to divide huge amount of logic into simple parts

- It must be declared with keyword "partial"
- By using partial class, application development is made faster.

Syntax: -

Partial class class –name

{

Statements;

}

Ex:

```
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace readwriteproperty
{
    public class Program
    {
        //class employee
        //{

        //}
        //class employee
        //{

        //}
        public partial class employee
        {
            //logic here
        }

        public partial class employee
        {
            //logic here
        }
        public partial class employee
        {
            //logic here
        }
        static void Main(string[] args)
        {
            employee e1 = new employee();
        }
    } }
```

Simple way to understand complete OOPS by me

S.no	Type	Class	Sealed Class	Partial class	Abstract class	interface
1	**Instance variable**	✓	✓	✓	✓	x
2	**Static variable**	✓	✓	✓	✓	x
3	**Constants**	✓	✓	✓	✓	x
4	**Instance constructor**	✓	✓	✓	✓	x
5	**Static constructor**	✓	✓	✓	✓	x
6	**Instance methods**	✓	✓	✓	✓	x
7	**Static method**	✓	✓	✓	✓	x
8	**Abstract methods**	X	x	x	✓	✓
9	**Instance properties**	✓	✓	✓	✓	✓
10	**Static properties**	✓	✓	✓	✓	x
11	**Inheritance**	✓	x	✓	✓	✓
12	**Multiple inheritance**	X	x	x	x	✓
13	**Access modifiers**	✓	✓	✓	✓	✓
14	**Object**	✓	✓	✓	x	x

COLLECTIONS

Collection is used to implement data structures in.net.

- It is used to store heterogeneous (dis-similar) data
- Collections are collection of same variable multiple values of different data types.
- In.net to work with collection we have to use the name space
  "Using system .collections"

Types of Collection:-Basically we have five types of collection and below are them.

1. Array list
2. Stack
3. Queue
4. Hash table
5. Sorted list

1) Arraylist:-

- It is used to store heterogeneous values.
- Here the size is not fixed we can declare dynamically
- Itis predefined class under **"using system. Collections"** namespace
- Some predefined methods that are available in array list

1. Add ();

This method is used to add items to arraylist

2. Remove ();

This method is used to remove particular item in array list it removes item by default either start or last values

3. Insert ();

This method is used to insert the item in arraylist at particular index insert (index, value)

4. Remove at ();

This method is used to remove the particular item in arraylist at particular index.

5. Clear ();

This method is used to clear all the values.

6. Reverse();

This method is used to reverse the items in reverse order

7. Sort ();

This method is used to sort the items in array list

Count property:-

- This is used to count the number of items in arraylist

For each syntax;-

The foreach statement repeats a group of embedded statements for each element in an array or an object collection. You do not need to specify the loop bounds minimum or maximum.

Syntax:

For each (datatype variable-name in collections)

The following code loops through all items of an array.

Ex:

```
int j = 0;

int[] myArr = new int[] { 0, 1, 2, 3, 5, 8, 13 };

foreach (int i in myArr )

{

  j = j + i ;

}
```

Difference between For and foreach loop

1. The for loop executes a statement or a block of statements repeatedly until a specified expression or condition becomes to false.
2. There is need to specify the loop bounds (minimum or maximum).

Following is a code example of a simple for loop that starts 0 till <= 5.

```
int j = 0;

for (int i = 1; i <= 5; i++)

{

  j = j + i ;

}
```

```
usingSystem.Collections;

namespace ConsoleApplication1
{
classProgram
    {
staticvoid Main(string[] args)
        {
ArrayListar = newArrayList();
ar.Add(10);
ar.Add(20);
ar.Add(23);
ar.Add(3);
ar.Add(50);
//ar.Remove(3);
//ar.Insert(1,99);
//ar.RemoveAt(2);
//ar.Reverse();
//ar.Sort();
//ar.Clear();
foreach (object o in ar)
            {
Console.WriteLine(o);
            }
Console.ReadLine();


        }
    }
}
```

2) Stack: -

Stack will arrange the data in LIFO (last in first out) format

It has two methods

1. **push**( ) **: -** this is used to add items in the stack
2. **pop**( ) **:-** this is used to remove items from the stack.

```
usingSystem.Collections;

namespace ConsoleApplication1
{
classProgram
    {
staticvoid Main(string[] args)
        {
Stackst = newStack();
st.Push(10);
st.Push(14);
st.Push(13);
st.Push(12);

//st.Pop();
foreach (object o inst)
            {
Console.WriteLine(o);
            }
Console.ReadLine();}}}
```

3) Queue:-

Queue will arrange the data in FIFO (first in first out)

It has two methods

1. **Enqueue:-**it is used to add items to the queue
2. **Dequeue: -** it is used to remove items from the queue

```
usingSystem.Collections;
namespace ConsoleApplication1
{
classProgram
    {
staticvoid Main(string[] args)
        {
Queue q = newQueue();
q.Enqueue(12);
q.Enqueue(15);
q.Enqueue(14);
q.Enqueue(13);
q.Dequeue();

foreach (object o in q)
            {
Console.WriteLine(o);
            }
Console.ReadLine();
        }
    }
}
```

4) Hash table:-

- It will arrange data in the form of key and value format
- It will maintain the data in dictionary entry interface
- It will arrange data based on key and value format.
- It will arrange data in descending order

Ex:-

```
usingSystem.Collections;
namespace ConsoleApplication1
{
classProgram
    {
staticvoid Main(string[] args)
        {
Hashtableht = newHashtable();
ht.Add(1, "Na");
ht.Add(9, "r");
ht.Add(9, "s");
ht.Add(5, "h");
foreach (DictionaryEntry d in ht)
            {
Console.WriteLine("the key is {0} and values is{1}", d.Key, d.Value);
            }
Console.ReadLine();
```

```
        }
    }
}
```

5) Sortedlist:-

- It will arrange data in key and value pair format.
- It will maintain data in dictionary-entry interface
- It will arrange data based on key
- It will arrange data in ascending order

```
usingSystem.Collections;
namespace ConsoleApplication1
{
classProgram
    {
staticvoid Main(string[] args)
        {
SortedListsl = newSortedList();
sl.Add(1, "c");
sl.Add(2, "b");
foreach (DictionaryEntry d insl)
            {
Console.WriteLine("the key is {0} and values is{1}", d.Key, d.Value);
            }
Console.ReadLine();
        }
    }
}
```

Exception Handling

- Exception means it is a runtime error
- Here we are having three blocks i.e try ,catch and finally
- If you enter a value it will go to try block
- Exception catches in catch block and finally block will give output

Syntax:-

Try

{

//code expected to generate a runtime ever

}

Catch (object exceptiontype)

{

// what to do in exception case

}

Finally

{

//code to close the pointer object

}

Ex:-

```
classProgram
    {
staticvoid Main(string[] args)
        {
try
            {
int x, y;
Console.WriteLine("enter two numbers");
                x = int.Parse(Console.ReadLine());
                y = int.Parse(Console.ReadLine());
int z = x / y;
Console.WriteLine("the division is " + z);

            }

catch (DivideByZeroException e1)
            {
// Console.WriteLine(e1.Message);
Console.WriteLine("divide by zero is not possible");
            }
catch (FormatException e2)
            {
Console.WriteLine("plz enter only integer values");
//Console.WriteLine(e2.Message);
            }

catch (Exception e)
            {
Console.WriteLine(e.
                    Message);
            }
finally
            {
Console.WriteLine("i am from finally");
            }
Console.ReadKey();
        }
    }
```

GENERICS

Generics are used to represent type safe data structures.

- These are introduced from .net 2.0version
- These are also called as general data types
- By using generics, we can confirm data types at runtime
- Generics avoids types casting like boxing of Unboxing
- These are implemented by using “generic “notation
- It consists of place holder and type parameter
- Placeholder is represented by angular braces (“<>”)
- Type parameter is a parameter or available name used to pass the required data type name to the generic notation.
- Generics are defined by using “<>’
- We can apply generics for variables, methods. Class, constructors, properties etc

Syntax:-

<tp> (tp variable)

Ex: -

```
classProgram
    {
publicstaticvoid display<tp>(tp t)
        {
Console.WriteLine(t);
        }
staticvoid Main(string[] args)
        {
display<int>(10);
Console.ReadLine();
        }
    }
```

Ex:2

```
classcollege<tp>
    {
publicvoid display(tp t)
        {
Console.WriteLine(t);
        }
    }
classggg
    {
staticvoid Main(string[] args)
        {
college<int> e1 = newcollege<int>();
e1.display(101);
Console.ReadLine();
        }
    }
```

If we declare a class as generic class , we have to confirm the data type at the time ofcreating the object

Note: - by using generics, we can avoid function over-loading

Draw backs generics:-

- It does not support arithmetic operations
- We can't pass more than one data type at a time

DELEGATES

Delegates are used to store multiple methods.
It is a type safe function pointer which is used to store information about method

Steps to work with delegates

Step1: complete class level code like variables methods. Etc
Public void show ()
{
}
Step 2: Create delegates
Syntax: <Access modifier >delegate return type delegate name ();
Step3: Create object to class
Step4: Create an object for delegate and store the method values in delegate
Step 5: Call the delegate
Syntax: Delegate object name ();

Types of delegates:-

- Single Cast
- Multicast

Note:-

1. We can add and store multiple methods for a delegate class
2. When we call the delegate, it will call all the methods
3. We can add multiple methods to delegates, by using "+ and ="
4. To remove methods from delegates using "- and ="

Ex 1:

```
Class test
    {
Public void p1()
        {
Console.WriteLine("this is p1 method");
        }
    }
Public delegate void dname();
Class Program
    {
Static void Main(string[] args)
        {
test t1 = newtest();
dname d1 = newdname(t1.p1);
d1();
Console.ReadKey();

        }
    }
```

Ex2:

```
classtest
    {
publicvoid p1()
        {
Console.Write("AP");
        }
publicvoid p2()
        {
Console.WriteLine("EC");
        }
    }
publicdelegatevoiddname();
classProgram
    {
staticvoid Main(string[] args)
        {
dname d1, d2, d3, d4, d5, d6;
test t1 = newtest();
            d1 = newdname(t1.p1);
            d2 = newdname(t1.p2);
            d3 = d1 + d2;
Console.WriteLine("From d3");
d3();
            d4 = d3 + d3;
Console.WriteLine("From d4");
d4();
            d5 = d4 - d1;
Console.WriteLine("From d5");
d5();
            d6 = d5 - d3 + d2;
Console.WriteLine("From d6");
d6();
Console.ReadKey();
        }
    }
```

MultiThreading

Creating our own execution process is called multi-threading
To work with threading we have to import the name space **"system. Threading "**

Ex:-

```
usingSystem.Threading;
namespacemultithreaddemo
{

publicclasstest
    {
publicstaticvoid p1()
        {
for (int i = 0; i <= 10; i++)
            {
Console.WriteLine(i + " ");
Thread.Sleep(500);
            }
        }
publicstaticvoid p2()
        {
for (int k = 11; k <= 20; k++)
            {
Console.WriteLine(k + " ");
Thread.Sleep(500);
            }
        }
classProgram
        {
staticvoid Main(string[] args)
            {
ThreadStart ts1 = newThreadStart(test.p1);
ThreadStart ts2 = newThreadStart(test.p2);
Thread t1 = newThread(ts1);
Thread t2 = newThread(ts2);
t1.Start();
t2.Start();
Console.ReadKey();

            }
        }
    }
}
```

WINDOWS FORM

Steps to Create windows Forms Project

Step1: Launch Visual studio

Step 2: Create new project

Step 3: Select below from the dropdown

Language: C#

Platform: Windows

Project type: Desktop

Step 4: Select Windows Forms App (.Net framework)

Step 5: Click on next

Step 6: Select below configurations

Project name: Specify the project name based on our requirement

Location: Specify the path of solution where we want to save the application

Step 7: Click on create

Points to remember in Windows Forms

1. Here we can vary the form size at design time according to our requirement.

2. Once the form size is fixed it can't be changed when using in other application

3. Design the form and build the solution

4. User control is generated in our local system (DLL)

5. Drag and drop the controls from toolbox and start running the application by pressing F5.

Different Controls in windows forms

Label: -Label control will acts as a caption for other controls.

Textbox: - It is a data entry control which is used to accept data from the user.

It has 3 types

1. Single Line.

2. Multi Line.

3. Password

Single Line: - If text mode is Single lines then the text box control will acts as Single line text box.

Password: - If text mode is password then textbox will act as a password textbox

Properties

UseSystemPasswordChar: we can make it true or false

Multi Line: - If text mode is multiline then textbox will acts as multiline text box

Property: Multiline: true or false

Maxlenght: - This is used to set the max number of character that what we are

Entering the textbox

Common properties

Name property: it is used to identify the controls like textbox, label, checkbox etc...

Whenever we can to get the value from specific control we will use Name property.

Text: For label if we want to give proper name or caption to any specific control then We will use Text property.

Message Box: If we want display some message then we will use message box in windows forms.

Check Box:-

This is used to select more than one item among group of items.

Radio Button:-

It is used to select one item from group of items.

Properties:-

Text: - This is used to display the text for the check box /radio button control.

Checked [True or False]: This Property will become true. Whenever user check the check box / radio button at run time.

Combo box:

This is also called as drop down list

It is used to select one item from the given list

How to add items to combo box?

Right click on properties and the select Items --> Collection and enter the values

We can add values to combo box in two ways

Step 1:

Click on Combo box --> Edit Items --> press enter after entering value --> click on Ok

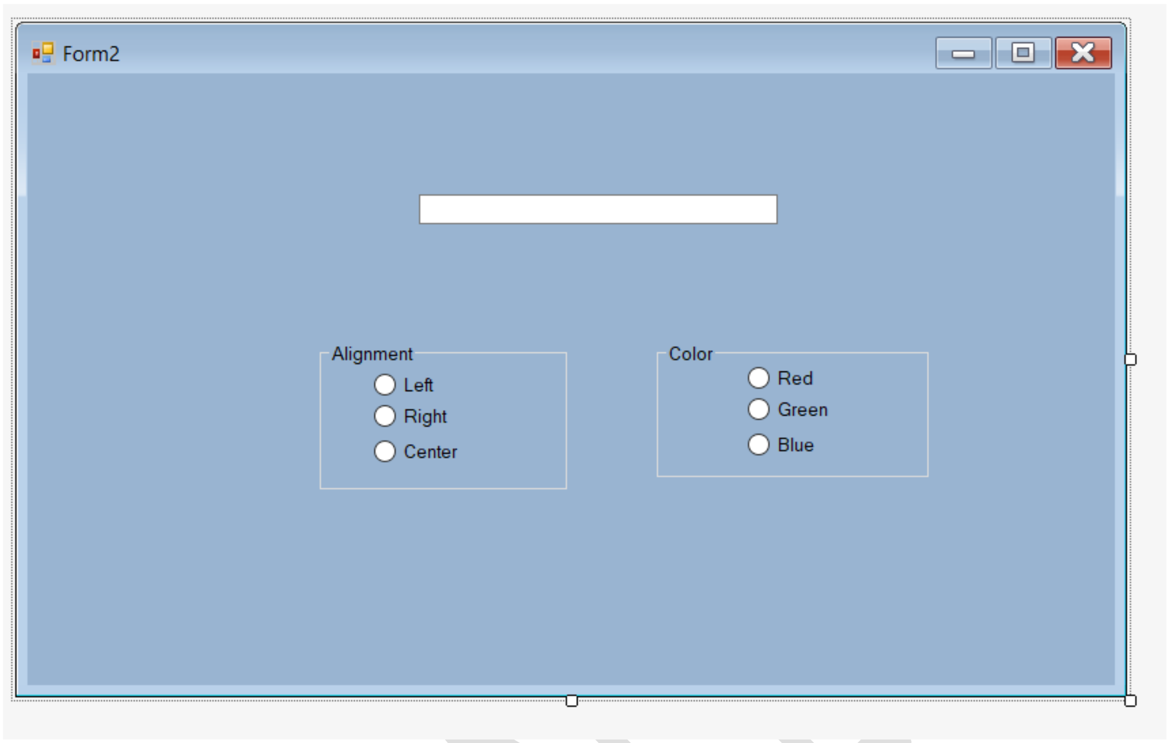
Step 2: Right click on Combo box -->properties--> Items --> Click on collection -->Press enter after entering value --> click on Ok

How to set Default value to combo box?

Right click on Combo box --> properties--> text --> Enter Default value

Ex: Select or some country name like India USA...etc

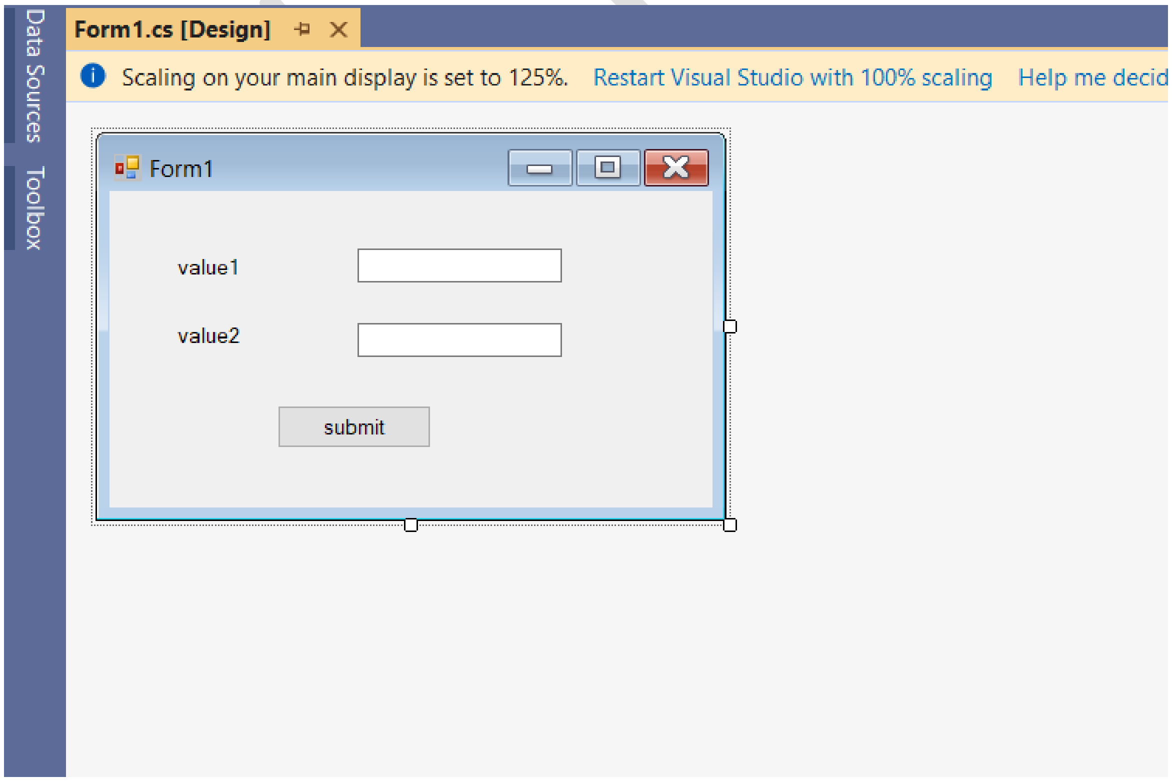
Ex: -Windows form program for alignment and color changes



Below is the Form.cs code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using WinodwsDemo;
using System.Data.SqlClient;

namespace WindowsFormsApp10
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

        }
        }
        private void radioButton1_CheckedChanged(object sender, EventArgs e)
       {
            textBox1.TextAlign = HorizontalAlignment.Left;
        }
private void radioButton2_CheckedChanged(object sender, EventArgs e)

       {
            textBox1.TextAlign = HorizontalAlignment.right;
        }

private void radioButton3_CheckedChanged(object sender, EventArgs e)

       {
            textBox1.TextAlign = HorizontalAlignment.Center;
        }

private void radioButton4_CheckedChanged(object sender, EventArgs e)

       {
            textBox1.TextAlign = HorizontalAlignment.Red;
        }

private void radioButton5_CheckedChanged(object sender, EventArgs e)

       {
            textBox1.TextAlign = HorizontalAlignment.Green;
        }

    }
}
```

Calculate sum using Windows forms

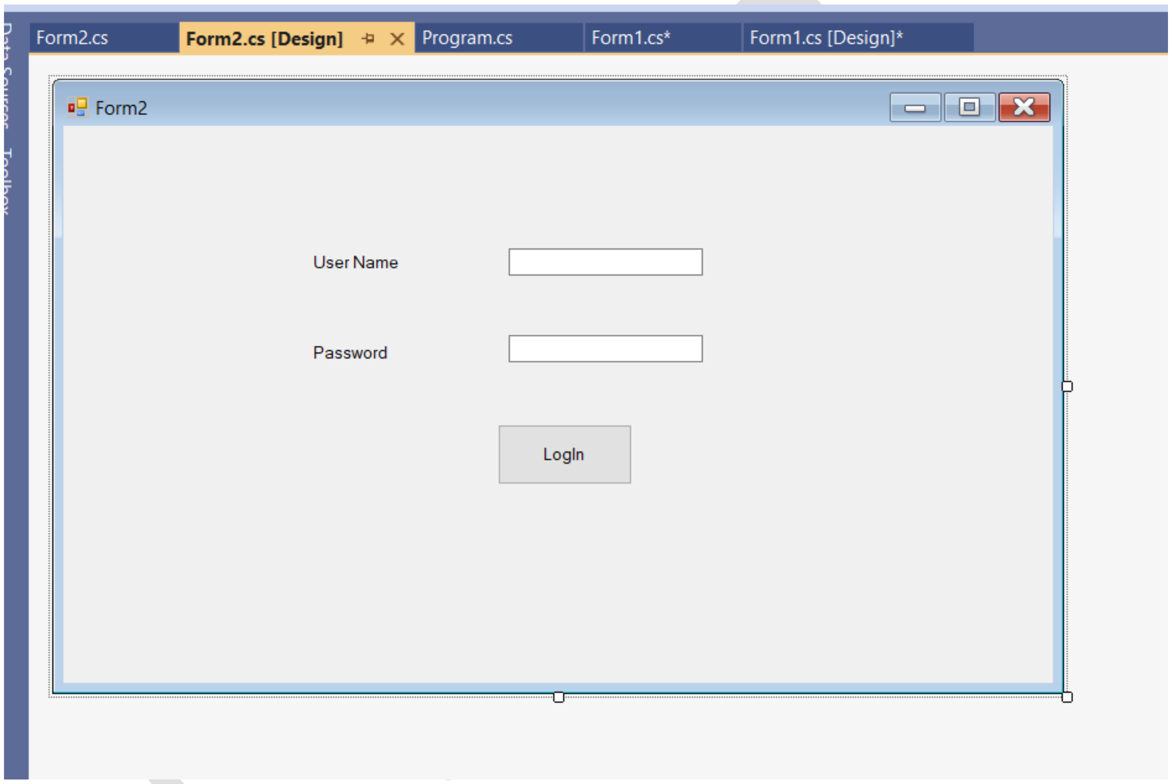

Form1.cs [Design]

Scaling on your main display is set to 125%. Restart Visual Studio with 100% scaling Help me decid

Double click on button and write the following code

```
private void button1_Click_1(object sender, EventArgs e)
        {
            Class1 obj = new Class1();
            int value1, value2, sum;
            value1 =Convert.ToInt32(textBox1.Text);
            value2=Convert.ToInt32(textBox2.Text);
            sum = obj.Add(value1, value2);
            MessageBox.Show("sum is "+ sum);
        }
```

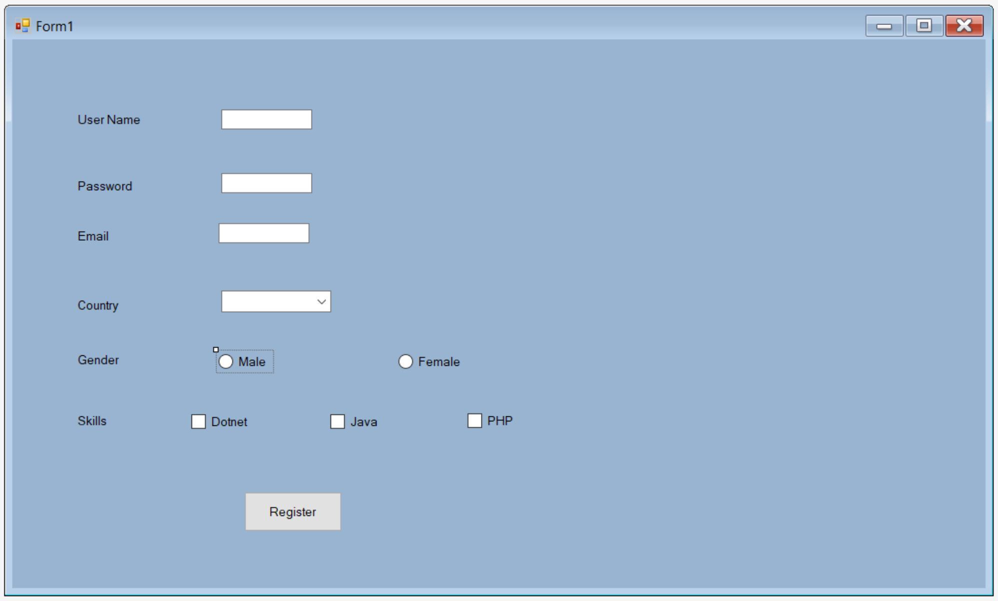
Ado. Net program to validate username and password from databse



Double click on button and write the following code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp10
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string username = textBox1.Text;
            string password = string.Empty;
            //step 2 create connection
       SqlConnection con = new SqlConnection("data
source=.;database=student;integrated security=yes");
            //pass the query
            con.Open();
            //craete a table in database with columns as username and password
            string query = "select password from stu where
username='"+username +"'";
            SqlCommand cmd = new SqlCommand(query, con);
            password = cmd.ExecuteScalar().ToString(); //value coming from sql
server
            if (password==textBox2.Text)
            {
                MessageBox.Show("Valid credentials ");
            }
            else
            {
                MessageBox.Show("inValid credentials ");
            }
            con.Close();

        }
    }
}
```

Ado. Net program to save values in Database

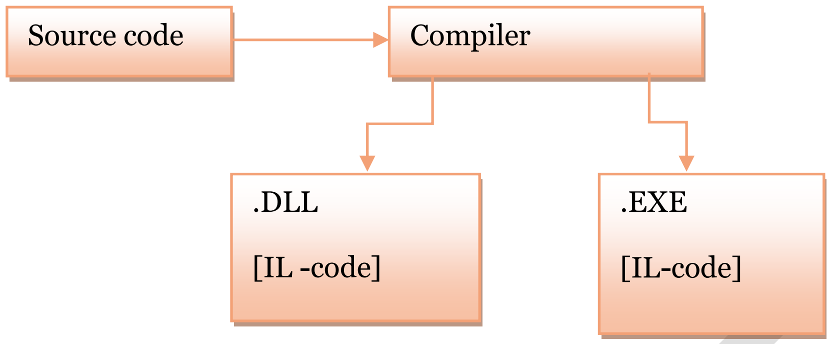


Double click on button and write the following code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using WinodwsDemo;
using System.Data.SqlClient;

namespace WindowsFormsApp10
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

        }
        private void button1_Click(object sender, EventArgs e)
        {
            string username = textBox1.Text;
            string password = textBox2.Text;
            string email = textBox3.Text;
            string country = string.Empty;
            string gender = string.Empty;
            string skills = string.Empty;
            if (comboBox1.SelectedItem != null)
            {
                country=comboBox1.SelectedItem.ToString();
            }
            if (radioButton1.Checked == true)
            {
                gender = radioButton1.Text;
            }
            else if (radioButton2.Checked == true)
            {
                gender = radioButton2.Text;
            }
            if (checkBox1.Checked == true)
            {
                skills = skills+checkBox1.Text + " ,";
            }
            if (checkBox2.Checked == true)
            {
                skills = skills+checkBox2.Text + " ,";
            }

            if (checkBox3.Checked == true)
            {
                skills = skills+checkBox3.Text + " ,";
            }
            //step2
            //create connection
      SqlConnection con = new SqlConnection("data
source=.;database=student;integrated security=yes");
            //step 3
            //open connection
            con.Open();
            //step 4
            //pass the query
 string query = "insert into Stu values( '" + username + "' ,'" + password +
"','" + email +
                "','" + gender + "','" + country + "','" + skills + "')";
            SqlCommand cmd = new SqlCommand(query, con);
            //step 5
            //execute query
            cmd.ExecuteNonQuery();
            //step 6
            //close connection
            con.Close();
            Form2 obj = new Form2();
            obj.Show();
        }


        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {

        }
    }
}
```

ASSEMBLY

- ❖ Assembly is a logical unit of “Types” and “Resources “



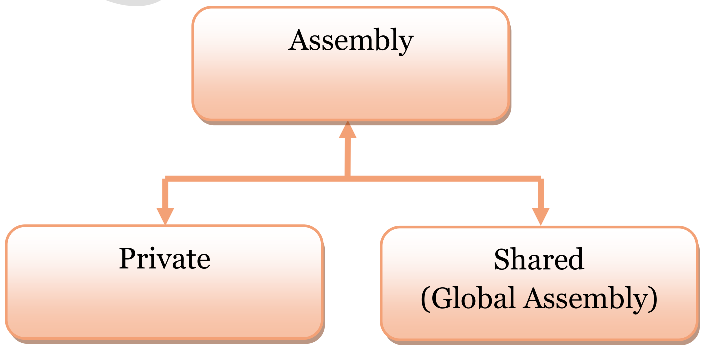
Differences between .DLL and EXE?

EXE (application)	DLL (library)
1.self-executable	1.dependent on an exe application
2.it will have entry point Main( )	2.it will not have any entry point
3.it will not have standardization of COM (as reusability ,better maintenance)	3.Follows the standard of com (as reusability ,better maintenance language inter-operability ,security)

DLL assembly:-

- ➢ Assembly is a small unit of deployment that provides reusability
- ➢ Versioning, maintained as well as security

Types of Assemblies:-



Private assembly:-

- It is used in single application where registration of assembly is not required

Shared assembly:-

- It assembly has to be used in more and more application the assembly needs to be registered under GAC registry .GAC(Global assembly cache)

Creating a private assembly:-

Steps to work with Class library

Step 1:

Create class library project

Steps to create class library project

1. Click on new project

2. Select Class Library (.Net Framework)

3. Change project settings like project name, location

4. Click on create

5. A class library file will be created without main method..

6. Complete class level code

7. Build the solution and close the project

Note: We need to remember path of the project, project name and class name

Program structure:-

```
Name space class library

{

Public class class name

{

}

}
```

Step 2: How to use class library now?

1. Create one console application

2. Add the reference of class library project

3. Right click on references folder--> Add reference

4. Click on browse -->open previously saved class library project path

5. Bin --> debug -->Select the DLL file

6. Click on Add --> Click on Ok

7. Now we can see the class library dll in references folder.

8. Add the namespace

9. Create object to class library class and access class library functions.

10. Complete the console application level code

11. Run the project and verify the output.

REMOTING

Remoting is a communication between .net assemblies present in one system with.net assemblies present in the other system connected within the network

Steps to perform and understand Remoting

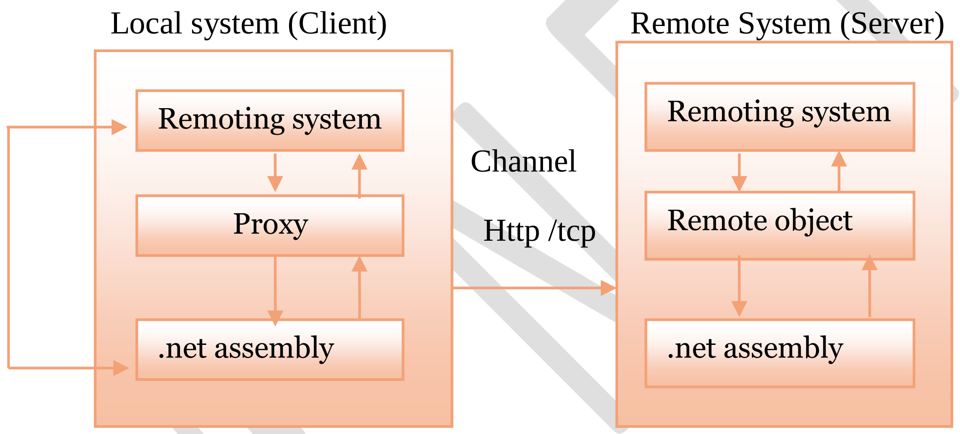
1. Architecture

2. Channels (http/tcp) we are used to communicate

3. Request

4. Response

1. Architecture:-



- Whenever client wants to send a request to server initially .net assembly will send the request to remoting system which is present in client system or local system
- Then remoting system will send message to proxy which tells about whether server has accepted the request or not and proxy will send that message to .net assembly
- If the request is accepted it will be send to .net assembly present in sever system
- Then .net assembly will create an response object which is called remote object
- Remote object will send the request to Remoting system either by value or by reference
- And finally Remoting system will give the response

Note: - Request and response are provided through a channel from local system to server system.

The channel may be Http or Tcp

Differences between TCP and HTTP

Tcp	Http
1.It is used for **intranet** based application	1.It is used for intranet and internet based application
2.It is mandatory to define a port explicitly	2.It can be used the default port of web server and can also, define a port explicitly
3.If the network is enabled with fire walls or proxy server then the data con not be passed through the network	3.Data can be passed through fire walls
4.In –build security is not supported and hence user has to define the security explicitly	4.It can use default security of the web server
5. Performance will be very fast.	5. Performance will be very slow.

Request:-

A client can send request in 2 ways

CAO:- (client activated object)

It is used for sending a request to the server if only remote class method is invoked within the client application a request will not be provided to the Remoting system for further processing

SAO:- (server activated object)

It is used for sending a request to the server if any object is defined for the remote class then the request will be given to the remoting system for further processing, it has 2 types,

1. Single call method:-

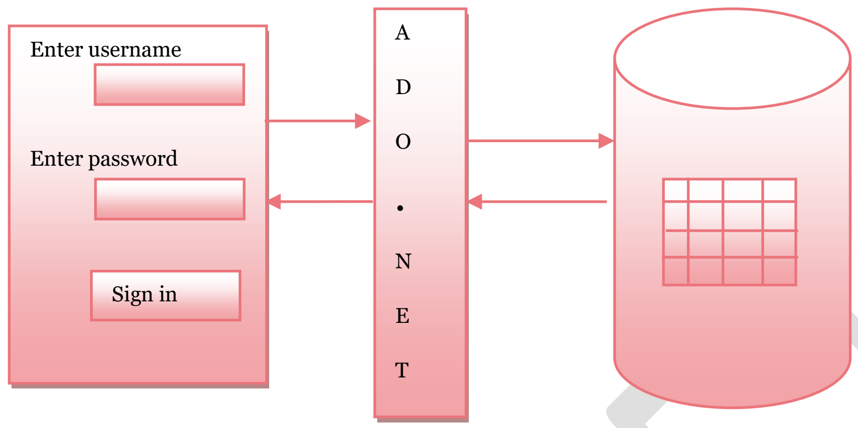
It is used for every client request and object is created every time in the server once the response is provided form the server then the object created in the server will be sent to Garbage collector implicitly.

2. Single ton method:-

When client gives the first request one object will be created in the server and the same object will be used for further client request for providing response.

ADO.Net

Active database object



Front End Application **Back End Application**

Front end application (F E):-

The application where user will interact is called frontend application.

FE can be developed by using FE tools like .net, java....etc

Back End Application (B E):-

The application where exactly user data will be maintained is called as BE application.

BE application can be developed by using BE tools like MSSQL, ORACLE, and SQL.

ADO.Net is a database object model which is used to create communication between FE applications and BE application.

SQL Server 2018

Server Type:- Database Engine

Server Name: -computer name or. (Dot) or localhost

Authentication: - Windows authentication / SQL server authentication

Username

Password

Server type :- Database engine .because we are working with SQL server as database engine

Server Name: - We have to give computer name where SQL server software was installed.

Authentication: - It is a process of checking of user credentials like username and password.

We can connect to SQL server database in two ways.

1) Windows authentication
2) SQL server authentication.

When we connect to SQL server database using **windows authentication**, It is not required to give username and password.

Where we connect to SQL server database using **SQL server authentication** then we have to give username and password.

Syntax for creating Data Base Object:-

Go to SQL server →right click on DB then you will find a option new DB→give your DB name→ Add→OK→and then you will find your newly created DB Name→ and go to that DB→ and Explore that one→and you will find table option there→now right click on table→and select New Table→add the columns and write the corresponding datatype of that columns→Save Table with some name→Now you will find the newly added table in the table list.

To connect with SQl server we have to use following name space

Using System.Data;

Using System.Data.SQLClient;

ADo.net supports two types of architecture

1. Connection oriented architecture
2. Disconnection oriented architecture.

Connection oriented Architecture

Steps to work with Connection Oriented Procedure:-

Step 1:- Declare Namespace using System.Data;

Using System.Data.SQLClient;

Step2:- Create a Connection

SQLClient Namespace provides predefined class called SQLConnection class. By using SQLConnection class we can create the connection to the Database. We can connect to SQL DB into 2 ways by using

- ➔ Windows Authentication
- ➔ SQL Server Authentication

When we connect to SQL server using windows Authentication there is no need of username & password and the connection string.

SqlConnection con=new SqlConnection (“Data Source = **.**; Database=Database name, Integrated Security=”Yes “);

When we connect SQLserver database using SQLserver authentication

SqlConnection con = new SqlConnection (“data source= . , database= database name, username=”sa”, password=”abc”);

Step 3: -

SqlConnection class provides predefined methods like open(), close()

Open() :-It is used to open the connection

Close () : - It is used to close the connection

Syntax for calling method:-

Con. Open ();

Step 5:-

SQLClient namespace provides a predefined class called Sqlcommand class. It is used to pass the query

Syntax:-

Sqlcommand cmd = new Sqlcommand ("pass the query", con);

Step 6 :

Execute query.

Sql command class provides three predefined methods

1) ExecuteScalar ()
2) ExecuteReader ()
3) ExecuteNonquery()

Syntax for calling method

Cmd. ExecuteNonquery ();

Step 7:-

close the connection.

SqlConnection class provides a predefined method close.

It is used to close the connection.

Syntax:

Con.close ();

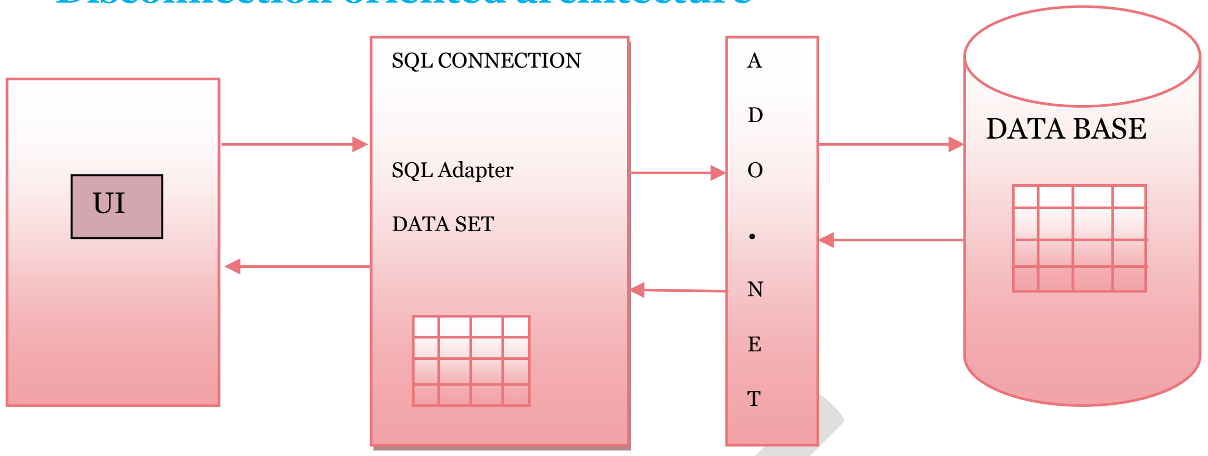
EXAMPLE:-

```
using System.Data.SqlClient;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp42
{
    internal class Program
    {
        public static void Main()
        {
            //step 1 Declare namespace
            int eno;
            string ename;
            Console.WriteLine("Enter employee number");
            eno = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter employee name");
            ename = Console.ReadLine();
            //step2
            //create connection
            SqlConnection con = new SqlConnection("data 
source=.;database=employee;integrated security=yes");
            //step 3
            //open connection
            con.Open();
            //step 4
            //pass the query
            string query = "insert into emp values( '" + eno + "' ,'" + ename 
+ "')";
            SqlCommand cmd = new SqlCommand(query, con);
            //step 5
            //execute query
            cmd.ExecuteNonQuery();
            //step 6
            //close connection
            con.Close();
            Console.WriteLine("records inserted sucessfully");
            Console.ReadLine();
        }
    }

}
```

Drawbacks of connection oriented architecture:

1. In connection oriented architecture we have to open the connection and close the connection
   if user open the connection and fails to close the connection ,the database will be maintained so the network traffic will be increased and the performance will be decreased
2. SQLDataReader class : it reads the data in forward only –direction that means top to bottom we can't read the data in bi-directional
3. Connection architecture will directly communicate with original database .we can't store the database in temporary memory

Disconnection oriented architecture



Steps to work with Disconnection Oriented Procedure:-

Step1:-

Declare the namespace

using System.Data;

using System.Data.SQLClient;

Step2:-

Create Connection

SQLConnection con=new SQLConnection();

Step3:-

Pass the Query.

Using Data Adapter Class

When we pass the Query.using SQL DataAdapter Class.

Data Adapter Class:-

It will directly open the connection to the database and it will execute the Query and result will be stored in dataset and Finally SQL Data Adapter class will be closes. That equals to opening and closing connections to the database and executes the query by SQL Data Adapter automatically.

Syntax:-

SQL Data Adapter Da=new SQL Data Adapter("pass the Query,con);

Step4:-

Create a object for Dataset

Dataset is a Temporary Database is used to store the data and retrieve the data from the database.

By Data Adapter and make it available for Dotnet applications

Dataset will read the data bi-directionally that means top – bottom and bottom –top

Syntax:-

Dataset ds = new Dataset ();

Step 5 :-

Fill the Data Adapter data in temporary database called dataset

SQL Data Adapter class consists of predefined method called fill().

This method is used to fill response in dataset

da.fill(ds, “tablename”);

Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//step 1
//Download System.Data.SqlClient from nuget packages
using System.Data;
using System.Data.SqlClient;

namespace ConsoleApplication4
{
    public class Program
    {
        static void Main(string[] args)
        {
            //step 2 create connection
            SqlConnection con = new SqlConnection("data 
source=.;database=employee;integrated security=yes");
            //pass the query
            SqlCommand cmdQuery = new SqlCommand("Select * from emp", con);
            SqlDataAdapter sda = new SqlDataAdapter(cmdQuery);
            DataSet dsData = new DataSet();
            sda.Fill(dsData);
            Console.WriteLine("Employee details Are");
            Console.WriteLine(dsData.Tables[0].Rows[0][0]);
            Console.WriteLine(dsData.Tables[0].Rows[0][1]);
            Console.ReadLine();
        }

    }
}
```

****END****