**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.NET\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**-----------------------------------------------------------------**

MVC:-

--------

MODEL VIEW CONTROLER

Database +designing +programming

C#, MVC,ADO.NET, STORED PROCEDURE,WEB SECVICES,PROJECT

C#-developed by :- Hejalberg

in 2000.

.NET FRAMEWORK

--------------------------

.NET :- it was developed dy Microsoft team mainly by ANDERS & HEGALBERG in 2000. It is a framework by Microsoft that provides a environment to write and the whole application on a single plateform .

.NET IDE- VISUAL STUDIO

During the development of .NET,C# was developed.

C -C++ - C#

C# :- C# is general purpose ,high level object oriented programming language.

Many other programming languages are just used to develop a single type of application but C# is used to develop Console based application Web application and Mobile Application.

Besides C# other 30+ Programming languages are provided by Microsoft with .NET framework .According to the developer choice any language can be used in development.

But mostly used programming language with .NET is C# . It is the extension of C++.

Object Oriented Programming Language :-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

object Oriented Programming language mainly provides two feature to the programming .

First is Security and seconds is Code Reusability. In object Oriented programming language every methods are encapsulated/written within classes.

pascal

------

all first word will be capital letter.

What is namespace:-

------------------------

Namespace is the collection of multiple classes. Namespace is the logical collection normally can be think as a folder containing multiple .cs files.

By default, the main namespace of the C# is system.

If you want to access any class declared within a namespace then firstly we have to include the namespace in file.

Syntax:-

----------

using NameSpace\_name;

Classes:-

-------

Classes is the collection data\_member and member\_function.

In simple word classes is the collection of multiple variables and methods.

Any variables/method declared in the class, is directly accessible in any other methods. You do not need to create instance of the class.

Only static member can not access the non-static members of the class.

Syntax:-

-------

class class\_name

{

Data\_member(variable);

Member\_function(methods);

}

Any non static member of class is not directly accessible out of the class.

To access any member of a class, from the outside of class , you have to create instance variable/object of the class.

Object are normally a variable of the class type which is used to access all member (variable ,methods) of a class anywhere.

Syntax to Create object:-

Class\_name object\_name = new class\_name();

Seconds methods:-

Class\_name object\_name; // normal variable of class type

object\_name=new class\_name();// instance variable of class

when a class type variable which is not declared by using new keywords ,stores the reference of another instance variable then is known as **Reference variable** of class.

Suppose there is a class names Program:

Program p=new Program(); // here p is the instance of class Program

Program p1; // here p1 is just a class type variable

p1=p; // now **p1 is the reference variable of class Program** which holds the reference of another instance variable that is p.

**Note : we can create multiple object of a single class.**

**For each object separate memory spaces initialized by compiler.**

**Memory spaces are initialized for only those object created by using new keyword. So new keywords are used to initialize instance variable of class.**

------------------------------------------------------------------------------------------------

Data Type:- datatype is the prefix which specifies the property of any variable ,function . Data type defines total memory spaces consumed by a object (variable/method) and which type of value it can stored.

1. Value type

Simple type

Signed integral value :sbyte short, byte ,int ,long

Unsigned integral value: ushort ,ubyte ,uint, ulong

Unicode character: char

Floating type values: float , double , decimal

Boolean value: bool

Structure

Struct{-------}

Enum: enum e{----------}

Nullable type :null

1. Reference type

Class type

Interface

Array

**Structure of C# First Program:**

**Using System;**

**Namespace myproject1**

**{**

**class MyClass1**

**{**

**Static void Main(Sring[]args)**

**{**

**//statement**

**}**

**}**

**}**

------------------------------------------------------------------------------------------------

using System;

namespace App1

{

class Program

{

static void Main(string[] args)

{

int a, b;

a = 100;b = 200;

Console.WriteLine("Output ="+(a + b));

Console.ReadKey();

}

}

}

decimal v = 4.6m;

Console.WriteLine(v);

char a ='V';

char b ='V';

if (a == b)

Console.WriteLine($"first value of a= {a} second value of b={b}");

string a ="Vinay";

string b ="Maurya";

Console.WriteLine($"{a} {b}");

using System;

namespace App2 // Note: actual namespace depends on the project name.

{

public class Program

{

public static void Main(string[] args)

{

string name;

name=Console.ReadLine()

name=name.Trim('a');

Console.WriteLine(name);

//Console.WriteLine(name);

string[] breakname=name.Split("ch");

Console.WriteLine(breakname.Length);

Console.WriteLine($"{breakname[0]} {breakname[1]} {breakname[2]}");

}

}

}

using System;

namespace App2 // Note: actual namespace depends on the project name.

{

public class Program

{

public static void Main(string[] args)

{

string name;

name=Console.ReadLine();

string [] breakline=name.Split(' ');

Console.WriteLine(breakline.Length);

}

}

}

using System;

namespace App2 // Note: actual namespace depends on the project name.

{

public class Program

{

public static void Main(string[] args)

{

string password = "Techpile";

string passbyuser;

passbyuser = Console.ReadLine();

int res = password.CompareTo(passbyuser);

Console.WriteLine(res);

Console.WriteLine(password==passbyuser);

}

}

}

Data type:-

***TASK:-***

using System;

namespace myapp1

{

class Program

{

static void Main(string[] args)

{

// wap to swap two int type value

int num1, num2, temp1;

Console.WriteLine("Enter two int type value");

num1 = int.Parse(Console.ReadLine());

num2 = int.Parse(Console.ReadLine());

temp1 = num1;

num1 = num2;

num2 = temp1;

Console.WriteLine("a value is=" + num1);

Console.WriteLine("b value is =" + num2);

// wap to swap two float type value

float a, b, temp;

Console.WriteLine("Enter two float type value");

a = float.Parse(Console.ReadLine());

b = float.Parse(Console.ReadLine());

temp = a;

a = b;

b = temp;

Console.WriteLine("a value is=" + a);

Console.WriteLine("b value is =" + b);

}

}

}

Some Math Function :-

class Program

{

static void Main(string[] args)

{

// wap to swap two int type value

int num1, num2;

Console.WriteLine("Enter two int type value");

num1 = int.Parse(Console.ReadLine());

num2 = int.Parse(Console.ReadLine());

int c = Math.Min(num1, num2);

Console.WriteLine(“Minumum value is”+c);

}

}

using System;

namespace myapp1

{

class Program

{

static void Main(string[] args)

{

double r;

Console.WriteLine("Enter a radius value to find area of circle :");

r = double.Parse(Console.ReadLine());

double area = (Math.PI) \* Math.Pow(r, 2);

Console.WriteLine($"Area of Circle = {Math.Ceiling(area)}");

}

}

}

------------------------------------------------------------------------------------------------

using System;

namespace myapp1

{

class Program

{

static void Main(string[] args)

{

string name;

Console.WriteLine("Enter Your Company Name :");

name = Console.ReadLine();

if (name.EndsWith('e'))

{

Console.WriteLine("TECHPILE TECHNOLOY PVT. LTD.");

}

}

}

}

-----------------------------------------------------------------------------------------------

1. Only if:- If there is only one condition is given with few statements . if given condition is true then statement will execute, if condition is false then if block will skip and further lines will execute.

Syntax:-

----

If(Condition )

{

// statement.

}

1. If -else statement:-

If there is two block of statement is given with one condition ,one block of if statement and second block for else statement. If given condition is true then if will execute otherwise else will execute without checking another condition.

Syntax:-

if(condition)

{

//statement

}

else

{

// Here you can use any statement.

}

Else if ladder:- If many condition are given as a option and each condition has it’s own statement. If condition will true then statement will execute .

Within given all condition with else-if ladder at one time only one block will execute.

Syntax:-

if(condition)

{

}

else if(condition)

{

}

else if(condition)

{

}

else

{

}

1. Switch statement:- switch statement is used as the optional of else-if ladder statement. If multiple condition have to applied based on the equality operator then instead of else-if ladder,you can use switch statement .That is easy to use and manage.

Switch(value/variable/expression)

{

Case value:

Break;

-

-

-

-

}

Array : Array is the collection of multiple value of similar data type. It can store multiple value in a single variable on different index.

Index of array always start from 0 and max index of array size-1.

Syntax to declare Array:-

Data\_type[] var\_name=new dat\_type[size];

Ex:- Let’s create a int type array with size 10;

Int[] students =new int[10];

using System;

namespace ConsoleApp3

{

class Program

{

static void Main(string[] args)

{

int[] arr = new int[10];

Console.WriteLine("Hello World!");

Console.WriteLine(arr[2]);

}

}

}

using System;

namespace ConsoleApp3

{

class Program

{

// wap to input 10 cities name and print only odd index city

static void Main(string[] args)

{

string[] city = new string[10];

Console.WriteLine("Enter 10 cities : ");

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

city[i] = Console.ReadLine();

for (var i = 1; i <10; i = i + 2)

Console.WriteLine(i+" index city : " + city[i]);

}

}

}

using System;

namespace ConsoleApp3

{

class Program

{

// wap to input 10 cities name and print only odd index city

static void Main(string[] args)

{

string[] city = new string[10];

Console.WriteLine("Enter 10 cities : ");

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

city[i] = Console.ReadLine();

for (var i = 1; i <10; i = i + 2)

Console.WriteLine(i+" index city : " + city[i]);

}

}

}

using System;

namespace ConsoleApp3

{

class Program

{

static void Main(string[] args)

{

string[] city1 = { "vinay", "maurya", "kishan", "kumar", "shani", "kumar" };

string[] city2 = new string[city1.Length];

city1.CopyTo(city2, 0);

city1[0] = "prasoon";

Console.WriteLine($"{city1[0]} {city2[0]}");

Array.Sort(city1);

for (int i = 0; i < city1.Length; i++)

Console.WriteLine(city1[i]);

}

}

}

One Dimensional Array:-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

one Dimensional array also called 1D array is used to store multiple value in a series on different index of array.

int[] arr\_name=new int[size];

Length: Length is the Property of array which is used to get total number of element in array.

Many function are used to the array type can be called by using Array class.

for the Example:-

---------------

Array.Sort(array);

Array.IndexOf()

Array.Reverse()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Two Dimensional Array:-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Also called 2D array . It is used to store values in form of rows and column., like a table,a matrix. In 2D array number of column in each rows are same. it can not be different.

Syntax to declare a 2D array:-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Data\_type[,] array\_name=new data\_type[rows\_size,column\_name];

Example:-

--------

Let's Create 2D Array with 2 rows and 3 Column to store int type value.

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

Direct Initialization :-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int[,] arr=new int[,] {{2,3,4},{4,5,7}}

Console.WriteLine(arr.Length);

int[,] arr={{2,3,4},{4,5,7}}

Console.WriteLine(arr.Length);

using System;

namespace Program

{

class Program

{

static void Main(string[] args)

{

int[] arr = new int[10];

Console.WriteLine("Enter 10 Element of array");

int i;

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

arr[i] = int.Parse(Console.ReadLine());

foreach(int val in arr)

{

Console.WriteLine($"Power 2 of {val} is {Math.Pow(val,2)}");

}

}

}

}

------------------------------------------------------------------------------------------------

2 3 4

5 6 7

6 7 8

Output:-

---

4

6

6

using System;

namespace Program

{

class Program

{

static void Main(string[] args)

{

int[,] num = new int[3, 3];

int i, j;

Console.WriteLine("Enter 9 Element :");

for (i = 0; i < 3; i++)

{

for (j = 0; j<3; j++)

{

num[i, j] = int.Parse(Console.ReadLine());

}

}

Console.WriteLine("Matrix is ");

// let's print 2D array...........

for (i = 0; i < 3; i++)

{

for (j = 0; j < 3; j++)

{

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

}

Console.WriteLine();

}

// digonal matrix

Console.WriteLine("Diagonal array is :");

for (i = 0; i < 3; i++)

{

for (j = 0; j < 3; j++)

{

if(i+j==2)

{

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

}

}

Console.WriteLine();

}

}

}

}

==============================================

using System;

namespace Program

{

class Program

{

static void Main(string[] args)

{

// wap to input 3\*4 matrix and print last element of each element.

int[,] num = new int[3, 4];

int i, j;

Console.WriteLine("Enter 12 Element :");

for (i = 0; i < 3; i++)

{

for (j = 0; j < 4; j++)

{

num[i, j] = int.Parse(Console.ReadLine());

}

}

Console.WriteLine("Matrix is ");

// let's print 2D array...........

for (i = 0; i < 3; i++)

{

for (j = 0; j < 4; j++)

{

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

}

Console.WriteLine();

}

// digonal matrix

Console.WriteLine("Diagonal array is last Element :");

for (i = 0; i < 3; i++)

Console.WriteLine(num[i, 3] + " ");

}

}

}

Jagged Array:-

---------------

Jagged array is same as 2D array but the only difference is number of columns in each row in two dimensional array is same but in Jagged array each row can have difference number of column.

Normally it is good to say that jagged array is the collection of multiple one dimensional array.

So jagged array is the collection of multiple rows where each rows is the collection of a array. And each array can have difference number of elements.

Syntax:-

Int[][] arr=new int[row\_size][col\_size];

Example:-

using System;

namespace jagged

{

class Program

{

static void Main(string[] args)

{

int[][] marks = new int[3][];

int i, j;

marks[0] = new int[] { 2, 4, 5, 6, 7, };

marks[1] = new int[] { 22, 43, 5, 6, 7, };

marks[2] = new int[] { 2, 4, 7, };

Console.WriteLine("Matrix is:");

for (i = 0; i <= 2; i++)

{

for (j = 0; j < marks[i].Length; j++)

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

Console.WriteLine();

}

}

}

}

using System;

namespace jagged

{

class Program

{

// jagged Array concept.

static void Main(string[] args)

{

int[][] marks = new int[3][];

int i, j;

marks[0] = new int[] { 2, 4, 5, 6, 7, };

marks[1] = new int[] { 22, 43, 5, 6, 7, };

marks[2] = new int[] { 2, 4, 7, };

Console.WriteLine("Matrix is:");

foreach (int[] arr in marks)

{

foreach (int val in arr)

Console.Write(val + " ");

Console.WriteLine();

}

}

}

}

*Addition of Jagged Array value:-*

using System;

namespace jagged

{

// wap to create a jagged array and print sum of all element of each row .

class Program

{

// jagged Array concept.

static void Main(string[] args)

{

int[][] marks = new int[3][];

int i, j,sum;

marks[0] = new int[] { 2, 4, 5, 6, 7, };

marks[1] = new int[] { 22, 43, 5, 6, 7, };

marks[2] = new int[] { 2, 4, 7, };

Console.WriteLine("Matrix is:");

foreach (int[] arr in marks)

{

foreach (int val in arr)

Console.Write(val + " ");

Console.WriteLine();

}

Console.WriteLine("Sum of all Element :");

foreach (int[] arr in marks)

{

sum = 0;

foreach (int val in arr)

sum = sum + val;

Console.Write(sum + " ");

Console.WriteLine();

}

}

}

}

New Way to Declare Jagged array :-

using System;

namespace jaggad1

{

class Program

{

static void Main(string[] args)

{

int[][] marks = new int[][] {

new int[]{ 2,3,4,5},

new int []{ 3,4,5,6,7}

};

Console.WriteLine(marks[0].Length);

}

}

}

--------------------

using System;

namespace jaggad1

{

class Program

{

// wap to make a jagged array with 3 rows and take user input of no. of cols and print sum of each rows.

static void Main(string[] args)

{

int[][] mat = new int[3][];

int i, j, size;

for(i=0;i<=2;i++)

{

Console.WriteLine($"Enter size of {i+1} row");

size = int.Parse(Console.ReadLine());

mat[i] = new int[size];

Console.WriteLine($"Enter {size} elements of {i+1} row");

for (j = 0; j < size; j++)

mat[i][j] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Here all elements of Jagged array");

foreach(int [] arr in mat)

{

foreach (int val in arr)

Console.Write(val);

Console.WriteLine();

}

}

}

}

-------------------------------------------------------------------

Problem with array :-

1. Fixed size
2. You can not insert any item in the middle of array
3. You can not any item from the middle of array.

***Collection :-***

**Dynamic** Array collection is also the group of multiple element which can holds variable size values and provides many function to insert , delete ,add, sort and many more.

It was introduced in C# 1.0.

Many Collection type are present like stack, queue, ArrayList,, linkedList , HashTable etc.

The all type of collection of class.

All classes of collection of present in a namespace.

System.Collection;

using System;

using System.Collections;

namespace collection

{

class Program

{

static void Main(string[] args)

{

ArrayList arrayList = new ArrayList();

arrayList.Add("Vinay");

arrayList.Add("Prasoon");

arrayList.Add("Maurya");

arrayList.Add(110);

arrayList.Add("@");

arrayList.Add("gmail");

arrayList.Add(".com");

//arrayList.RemoveAt(6);

Console.WriteLine(arrayList.Count);

foreach (object val in arrayList)

Console.Write(val);

}

}

}

------------------------------------------------------------------------------------------------

ArrayList:-

ArrayList is a class of collection (Non -Generic Collection).it works same as Array that means it can store multiple values and each values can be accessed by using index.

But the major difference between array and arrayList is Array is fixed size collection but ArrayList is variable size collection.

It automatic sets the capacity of collection to the 4. And increases twice if capacity fails.

You can set capacity of ArrayList explicitly.

1st method : ArrayList arraylist=new ArrayList(30);

2nd method : arraylist.capacity(40);

Second Difference between Aarray and ArrayList is, in ArrayList you can add, delete , values anywhere in the ArrayList by using some pre-defined function Add(), Insert(), InsertAt() , Remove(), RemoveAt().

Count Property is used to get number of total elements in Arraylist.

Console.WriteLine(arraylist.Count);

using System;

using System.Collections;

namespace collection

{

class Program

{

static void Main(string[] args)

{

ArrayList arrayList = new ArrayList();

arrayList.Add("Vinay");

arrayList.Add("Prasoon");

arrayList.Add("Maurya");

arrayList.Add(110);

arrayList.Add("@");

arrayList.Add("gmail");

arrayList.Add(".com");

Console.WriteLine(arrayList.Count);

foreach (object val in arrayList)

Console.Write(val);

}

}

}

using System;

***List Example type Array:-***

using System.Collections.Generic;

namespace List1

{

class Program

{

static void Main(string[] args)

{

List<int> list = new List<int>();

list.Add(20);

list.Add(40);

list.Add(50);

list.Add(5);

list.Add(01);

foreach (int val in list)

Console.WriteLine(val);

}

}

}

------------------------------------------------------------------------------------------------

using System;

namespace TAsk1

{

class Program

{

//wap to Create a parametrized function to get number of spaces in the string

static void Main(string[] args)

{

Program p1 = new Program();

string name = "vinay";

Console.WriteLine(p1.IsContains(name,'i'));

}

bool IsContains(string str,char chr)

{

int i, count = 0;

for(i=0;i<str.Length;i++)

{

if(str[i]==chr)

{

count++;

}

}

if(count==0)

{

return false;

}

else

return true;

}

}

}

-------------------------------------------------------------------------------------------

using System;

namespace index

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

Program p1 = new Program();

Console.WriteLine("Enter any String to find Given Value");

string name = Console.ReadLine();

int index = p1.LastIndexOf(name,'i');

if(index==-1)

Console.WriteLine($" Character is not present in any index ");

else

Console.WriteLine($" Character is present in {index} index ");

}

int LastIndexOf(string name, char val)

{

int i,index = -1;

for (i=0;i<name.Length;i++)

{

if(name[i]==val)

{

index = i;

}

}

return index;

}

}

}

***HashTable:-***

Hashtable is the Non-generic collection of elements .Non- generic collection means is has property of auto resizing. That means with the increment of number of elements size of non -generic collection increments.

There is no Concept of fixed sizing like array.

ArrayList, Hashtable all are the non -generic collection .But the difference between ArrayList andHashTable is ArrayList store values on the different indexes while HashTable store values in the pair of **key and value.**

Many different pre-defined function are present to perform different operations on the HashTable. Like insert, remove, delete elements key wise.

*Syntax to declare A HashTable Obeject.*

HashTable obj\_name=new HashTable();

Syntax to add elements in HashTable :

Obj\_name.Add(object key,object value);

***List :-***

List is a generic collection of c#. it has a namespace: using System.Collection.Generic;

List is a collection of multiple elements just like Array,Arraylist and many more.

List is a generic collection So it has some properties:-

1.List has auto resizing Property. That means capacity of list increment with number of elements. You do not need to initialize size at the time of declaration of list.

2. List is the collection of specific type elements so it can be think as strongly typed collection.

3. Many methods are given to perform different operations on list object like insert ,delete, remove manipulate, sort and many more.

4. So any elements can be added or removed from the middle of list that is not possible Array.

***Syntax to declare a List Object :-***

List<T> obj\_name=new List<T>();

Where T is the specific type like , string ,int float, bool…..

List also stores elements on the indexes just like Array.

*Syntax to add elements in list :-*

*­*obj\_name.Add(“element”);

-------------------------------------------------------------------

***Dictionary :-***

Dictionary is also a class of Generic type just like list.

Only difference between list and Dictionary is list stores values index wise where Dictionary store values in the pair of key and value.

Namespace of Dictionary : using.System.Collections.Generic;

*Syntax to declare a Dictionary :-*

Dictionary <Tkey,Tvalue> Obj\_name=new DictionaryTkey,Tvalue>();

Example:-

Dictionary <string,string> dict=new Dictionary<string,string>();

Syntax: to Add elements in dictionary :-

Obj\_name.Add(“name”,”Ram Singh”);

using System;

using System.Collections.Generic;

namespace dict

{

class Program

{

static void Main(string[] args)

{

Dictionary<string, string> dic = new Dictionary<string, string>();

dic.Add("Name","Techpile");

dic.Add("DOB","15 OCT");

foreach(KeyValuePair<string,string> pair in dic)

{

Console.WriteLine(pair.Key + " " + pair.Value);

}

}

}

}

-----------

using System;

using System.Collections;

namespace dict

{

class Program

{

static void Main(string[] args)

{

Hashtable dic = new Hashtable();

dic.Add("Name","Techpile");

dic.Add("DOB","15 OCT");

foreach(DictionaryEntry pair in dic)

{

Console.WriteLine(pair.Key + " " + pair.Value);

}

}

}

}

Constructor:-

-------------

Constructor is method declared within the class that have same name as the class.

Constructor should not be private.

Constructor call automatic with creation of object.when we create object of a class by using

new keyword then statements of constructor executes automatic.

So their is no need to call constructor explicitly.

Constructor does not have any return type not void.

A class may have more that one overloaded constructor.

An implicit Constructor is always with each class which performs initialization of class variables on behalf of programmer at the time of object creation.

So default value of class variable defined by implicit constructor is :

int type variable 0

string type variable null

float type variable 0

-------------------------------------------------------------------------------------------------------------------

Type of Constructor:-

--------------------------

1. Default constructor

explicit Constructor/defined by programmer

implicit constructor/defined by compiler

2.Paramerterized Constructor

3.Copy Constructor

4.static constructor

Default Constructor:-

Default Constructor is also known as parameter-less constructor.

A method with same name of class that has no parameter is default Constructor.

The default Constructor which coding is intentionally changed by programmer is known as explicit default Constructor.

Default Constructor does not need to be called explicitly. Is will automatic run with the creation of object.

Mostly Constructor are used to initialize class variables which is being used by another function of class.

Class Test

{

Public Test()// default explicit constructor

{

// statement

}

}

Parameterized Constructor:-

A method with same name of class that has some arguments passed with it is known as **parameterized Constructor.**

Mostly parameterized Constructor are used to receive values from the outside of class and initialize those values into class variables.

If you want to initialize class variables with the value given by caller then you can declare a parameterized constructor to do the same.

Class test

{

int a ,b;

test(int x, int y)

{

a=x ;

b=y;

}

}

Copy Constructor:-

Copy constructor is a method of class that same name as the class and has parameter of **same class type variable**.

Copy constructor is used to copy data of one object to another object.

class test{

test(test t)// Copy Constructor

{

}

}

using System;

namespace ConsoleApp5

{

class Program

{

static void Main(string[] args)

{

string name;

int age;

Console.WriteLine("please ! Eneter your Name and Age:" );

name = Console.ReadLine();

age = int.Parse(Console.ReadLine());

student s = new student(name,age);

s.showDetails();

student s2 = new student(s);

s2.showDetails();

}

}

class student

{

string name;

int age;

public student(string name, int age)// paramerterized constructor

{

this.name = name;

this.age = age;

}

public student(student s)// copy constructor

{

name = s.name;

age = s.age;

}

public void showDetails()

{

Console.WriteLine($"Student name is {name} and age is : {age} year old");

}

public string data // getter oroperty

{

get

{

return $"Student name is {name} and age is : {age} year old";

}

}

}

}

Math.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp5

{

class math

{

int a, b;

public math()

{

a = 50;

b = 40;

}

public math(int a, int b)

{

this.a = a;

this.b = b;

}

public math(math maith)

{

a = maith.a;

b = maith.b;

}

public void add()

{

Console.WriteLine($" Addition is = {a+b}");

}

}

}

Polymorphism :-

--------------------

Polymorphism is OOP concept where same name of function performs differently.

So if you use same name of function in different forms this is polymorphism.

There are two type of polymorphism :-

1.compile time polymorphism

a. method overloading

2.Run time Polymorphism

a. Method overriding

b. Method hiding/Sampling

when Polymorphism performs within same class then is called Method overloading and when Polymorphism performs within parent- child class then is called method overriding.

**Method Overloading/Compile time Polymorphism:-**

When a class have more than one function with same name but different parameter then it is called method overloading.

Parameter of function may be different 3 Ways:-

1.By different number of parameter

2.By different data type of parameter

3. By different sequence of parameter

using System;

namespace MethodOverloading

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

calculator cal = new calculator();

cal.add(45,60);

}

}

class calculator {

public void add(int a, int b)

{

Console.WriteLine($"Addition ={a+b}");

}

public void add(int a, int b, int c)

{

Console.WriteLine($"Addition ={a + b+c}");

}

public void add(float a, float b)

{

Console.WriteLine($"Addition ={a + b}");

}

public void add(float c,float b, float a)

{

Console.WriteLine($"Addition ={a + b+c}");

}

}

}

**Method Overriding:-**

Overriding may be done within - parent child class. When a parent class and child class has same function with same name , same parameter then this is Overriding.

using System;

namespace MethodOverloading

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

Math m = new Math();

m.add(25,20);

}

}

class calculator {

public void add(int a, int b)

{

Console.WriteLine($"Addition ={a+b}");

}

public void add(int a, int b, int c)

{

Console.WriteLine($"Addition ={a +b+ c}");

}

public void add(float a, float b)

{

Console.WriteLine($"Addition ={a + b}");

}

public void add(float c ,float b, float a)

{

Console.WriteLine($"Addition ={a + b+c}");

}

}

class Math : calculator

{

public new void add(int a ,int b)// method overriding

{

Console.WriteLine($" Reault is = {a+b} ");

}

}

}

Method Hiding/Sampling:- Method hiding sampling is the example of method overriding. When parent class and child class have same name function with same parameter then re-implementation of function in child class is known as method hiding.

**Difference between Method overriding -and Hiding:**

* Method overriding is done on the virtual function of parent .
* Method hiding can be done by on any function of parent.
* Method overriding is done by using overriding keyword in class functions.
* Method hiding is done by using new keyword in child class functions.
* When child class have same name functions as parent class with same signature (parameter ,return type) then this is called method overriding.
* When class have same name function with same parameters as parent class then this is known as method hiding.

**Inheritance :-**

Inheritance is methodology in OOP to use members of pre-existing class in a new class . So new class can use the member of parent class at it is the owner of members.

In Inheritance the class which properties are inherited is known as Parent class/base class/super class and the class which is inheriting the property is known as child cla ss/ derived class/ sub class.

In C# inheritance is performed by using (:) operator.

All protected member of a class is not accessible outside except of child class.

Important points about Inheritance.

* By creating object of parent class, only members of parent class is accessible. In any way it can not access member of it’s child.
* By default each classes of C# is child of object class. Object is declared within System namespace.
* ToString () Equals() , GetHashCode() , Gettype () these are the Functions of Object Class that is accessible with almost all classes of C#. You don’t need to inherit it by using(:) .it is Implicitly inherited.
* By using object of class You can Access both ,member of it’s parent class and members declared within it.
* By default , Default Constructor of parent class executes when we create object of child class .

Enum :-

Enum is a fixed value collection which can be used as a type anywhere. Enum type Variable contains a fixed value list that is only acceptable for those variables.

When you want to initialize a enum type variable then their will be fixed list that is acceptable for this variable.

Syntax:-

------------

<Access-modifier>enum <variable\_name>

{

Value\_list\_seprated\_by\_comma

}

Public enum week

{

Sunday, Monday,Tuesday,wednesday

}

Now you can use week as the type anywhere. And week type members only accepts values as defined in the list of week.

Difference between Fields of class and Property of class is nothing but only that Property can have only getter of setter but fields always have both.

Conditions can be applied with getter or setter of property but fields can not contains any extra code.

*Use of TempData:-*

***Home.Controller:-***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using webProject\_7.Models;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

namespace webProject\_7.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

public ActionResult Register()

{

return View();

}

[HttpPost]

public ActionResult Register(string name,string userid,string pass,string cpass,string mobile,HttpPostedFileBase pic)

{

if (cpass == pass)

{

string path = System.IO.Path.Combine(Server.MapPath("/content/profile/"),pic.FileName);

pic.SaveAs(path);

Register reg = new Register() { name = name, userid = userid, password = pass, mobile = mobile, picture =pic.FileName};

TempData["registerdata"] = reg;

string str =ConfigurationManager.ConnectionStrings["mystr"].ToString();

SqlConnection connection = new SqlConnection(str);

connection.Open();

SqlCommand command = new SqlCommand("sp\_employee",connection);

command.CommandType = CommandType.StoredProcedure;

command.Parameters.AddWithValue("@name",name);

command.Parameters.AddWithValue("@userid",userid);

command.Parameters.AddWithValue("@password",pass);

command.Parameters.AddWithValue("@mobile",mobile);

command.Parameters.AddWithValue("@picture",pic.FileName);

int res = command.ExecuteNonQuery();

if (res>0)

{

Response.Write("<script>alert('Record Added')</script>");

}

else

{

Response.Write("<script>alert('Record Not Added Please ! try Again.')</script>");

}

connection.Close();

return RedirectToAction("SignIn");

}

else

{

Response.Write("<script>alert('Password and Confirm Password Should be Same ..')</script>");

return View();

}

}

public ActionResult SignIn()

{

if (TempData["registerdata"] != null)

{

Register s = TempData["registerdata"] as Register;

TempData.Keep();

}

return View();

}

[HttpPost]

public ActionResult SignIn(string userid,string pass)

{

Register s = TempData["registerdata"] as Register;

if (s.userid == userid && s.password == pass)

{

Response.Write("<script>alert('Welcome to Myweb')</script>");

return RedirectToAction("Welcome");

}

else

{

Response.Write("<script>alert('Userid password is Incorrect.')</script>");

return View();

}

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

public ActionResult Welcome()

{

Register s = TempData["registerdata"] as Register;

return View();

}

}

}

*Register.cshtml:-*

@{

ViewBag.Title = "Register";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<style>

.register{

min-height:300px;

box-shadow:0px 0px 10px 10px orange inset;

padding:50px;

margin:20px;

border-radius:20px;

}

.form-control {

margin-bottom:10px;

}

.btn:hover{

background:green;

}

</style>

<div class="row">

<div class="col-sm-4"></div>

<div class="col-sm-4 register">

<h1 class="text-center text-warning" style="font-family:Century;">Register Here </h1>

<form action="" method="post" enctype="multipart/form-data">

Name :<input type="text"name="name" required class="form-control" placeholder="Enter Your Name.."/>

UserId :<input type="text" name="userid" required class="form-control" placeholder="Enter Your UserId.."/>

Password :<input type="password"name ="pass" required class="form-control" placeholder="Enter Your password.."/>

Confirm Password :<input type="password" name="cpass"required class="form-control" placeholder="Enter Your confirm password.."/>

Mobile No :<input type="number" name="mobile" class="form-control"required placeholder="Enter Your number.."/>

Picture :<input type="file" name="pic" required class="form-control"/>

<input type="submit" class="btn btn-warning"/>

</form>

</div>

<div class="col-sm-4"></div>

</div>

*Signin.cshtml:-*

@{

ViewBag.Title = "SignIn";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<style>

.register {

min-height: 300px;

box-shadow: 0px 0px 10px 10px orange inset;

padding: 50px;

margin: 20px;

border-radius: 20px;

}

.form-control {

margin-bottom: 20px;

}

.btn:hover {

background: green;

}

</style>

@{var s = TempData["registerdata"] as webProject\_7.Models.Register;}

<div class="row">

<div class="col-sm-4"></div>

<div class="col-sm-4 register">

<h3 style="font-family:Century; font-weight:bolder;">Welcome @s.name</h3>

<h2 class="text-center text-warning" style="font-family:Century;">֍ Signin Here ֍</h2>

<form action="" method="post">

UserId :<input type="text" name="userid" class="form-control" placeholder="Enter Your UserId.." />

Password :<input type="password" name="pass" class="form-control" placeholder="Enter Your password.." />a

<input type="submit" class="btn btn-warning" value="login"/>

</form>

</div>

<div class="col-sm-4"></div>

</div>

*Welcome.cshtml:-*

@{

ViewBag.Title = "Welcome";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@{var s = TempData["registerdata"] as webProject\_7.Models.Register;}

<div class="row">

<div class="col-sm-4"></div>

<div class="col-sm-4">

<img src="~/Content/profile/@s.picture" height="200px"/><br /><br/>

<p><b>Name :</b>@s.name</p>

<p><b> UserId :</b>@s.userid</p>

<p><b>Password :</b>@s.password</p>

<p><b>Mobile No :</b>@s.mobile</p>

</div>

<div class="col-sm-4"></div>

</div>

**Standard View:-**standard view is a separate page of .cshtml. which needs no data or very small data form controller.

ViewBag or ViewData is used to transfer data from controller to View.

**Stricty Typed View:-**

Stricty typed view is a page with .cshtml extension which large amount of data from Controller.

Model of ViewModal is used to

SqlDataAdapter

ExecuteScaler

ExecuteNonQuery

SqlCommand

ExecuteNonReader

Filtration

RouteConfig

MVC Cycling