**Properties**

Properties are special kind of class members. We use predefined set and get methods to access and modify them. Property reads and writes are translated to get and set method calls.

class Person{

private string \_name;

public string Name {

get { return \_name; }

set { \_name = value; }

}

}

**Constructor**

## Default Constructor

A constructor without any parameters is called as default constructor means a constructor which does not have any input parameter is known as default constructor.

public class Addition{

public int Mycode = 0;

public Addition(){

Mycode = 9; }

}

## **Parameterized Constructor**

A constructor having one or more parameters is called as parameterized constructor means a constructor which is having single input parameter or multiple input parameters of same data types or different data types are known as parameterized constructor.

public class Addition{

public string itype = "";

public int x = 0;

public int y = 0;

public Addition(string type, int a, int b)

{

itype = type;

x = a;

y = b;

}

}

## **Copy Constructor**

A constructor that contains a parameter of same class type is called as copy constructor.

C# does not provide a copy constructor. A copy constructor enables you to copy the data stored in the member variables of an object of the class into another new object means it helps to copy data stored in one object into another new object of the same instance.

class Car {

public string \_nameofcar = "";

public int \_carno = 0;

public double \_carprice = 0;

public Car(string NameofCar, int CarNumber, double CarPrice) {

\_nameofcar = NameofCar;

\_carno = CarNumber;

\_carprice = CarPrice;

} //Copy Constructor

public Car(Car objCar) {

\_nameofcar = objCar.\_nameofcar;

\_carno = objCar.\_carno;

\_carprice = objCar.\_carprice;

}

}

## **Static Constructor**

Static constructor should be parameter less means it should not contain any input parameter. Program will not execute if static constructor is having any input parameter.

Static constructor can be invoked once for any number instances are created and it is invoked only during the first initialization of instance. It is used to initialize static fields of the class

Static constructor is created using a static keyword as shown below.

Class Addition{

static Addition(){

Console.WriteLine("Static constructor is called");

}

}

## **Private Constructor**

A constructor with "private" access modifier in a class is called as private constructor.

A class with private constructor cannot be inherited.

We cannot create an object of the class which is having a private constructor. Program will not allow us to create an object of a class having private constructor.

public class Employee {

public static int emplcounter = 9;

private clsEmployee() {

}

}

**Virtual Keyword**  
  
The Virtual keyword is used for generating a virtual path for its derived classes on implementing method overriding. The Virtual keyword is used within a set with an override keyword. It is used as:

class A

{

public virtual void show()

{

Console.WriteLine("Hello: Base Class!");

Console.ReadLine();

}

}

**Override Keyword**  
  
The Override keyword is used in the derived class of the base class in order to override the base class method. The Override keyword is used with the virtual keyword, as in:

class A

{

public virtual void show()

{

Console.WriteLine("Hello: Base Class!");

Console.ReadLine();

}

}

// Derived Class

class B : A

{

public override void show()

{

Console.WriteLine("Hello: Derived Class!");

Console.ReadLine();

}

}

**New Keyword**  
  
The New keyword is also used for polymorphism but in the case of method overriding.In simple words we can say that we are changing what the base class does for the derived class.

class A

{

public void show()

{

Console.WriteLine("Hello: Base Class!");

Console.ReadLine();

}

}

class B : A

{

public new void show()

{

Console.WriteLine("Hello: Derived Class!");

Console.ReadLine();

}

}