**What is  constructor?**  
A special method of the class that will be automatically invoked when an instance of the class is created is called a constructor. The main use of constructors is to initialize private fields of the class while creating an instance for the class. When you have not created a constructor in the class, the compiler will automatically create a default constructor in the class. The default constructor initializes all numeric fields in the class to zero and all string and object fields to null.   
  
**Some of the key points regarding the Constructor are:**

* A class can have any number of constructors.
* A constructor doesn't have any return type, not even void.
* A static constructor can not be a parametrized constructor.
* Within a class you can create only one static constructor.

**Constructors can be divided into 5 types:**

1. Default Constructor
2. Parametrized Constructor
3. Copy Constructor
4. Static Constructor
5. Private Constructor

Now let us see  each constructor type with example as below

**Default Constructor**  
A constructor without any parameters is called a default constructor; in other words this type of constructor does not take parameters. The drawback of a default constructor is that every instance of the class will be initialized to the same values and it is not possible to initialize each instance of the class to different values. The default constructor initializes:

1. All numeric fields in the class to zero.
2. All string and object fields to null.

**Example**

using System;  
namespace DefaultConstractor  
 {  
    class addition  
    {  
        int a, b;

        public addition()   //default contructor

        {

            a = 100;

            b = 175;

        }

        public static void Main()

        {

            addition obj = new addition(); //an object is created , constructor is called

            Console.WriteLine(obj.a);

            Console.WriteLine(obj.b);

            Console.Read();

        }

      }

    }

Now run the application, the output will be as in the following:



**Parameterized Constructor**

A constructor with at least one parameter is called a parametrized constructor. The advantage of a parametrizedconstructor is that you can initialize each instance of the class to different values.

using System;

namespace Constructor

{  
    class paraconstrctor  
    {  
      public  int a, b;  
      public paraconstrctor(int x, int y)  // decalaring Paremetrized Constructor with ing x,y parameter

        {

            a = x;

            b = y;

        }

   }

    class MainClass

    {

        static void Main()

        {

            paraconstrctor v = new paraconstrctor(100, 175);   // Creating object of Parameterized Constructor and ing values

            Console.WriteLine("-----------parameterized constructor example by vithal wadje---------------");

            Console.WriteLine("\t");

            Console.WriteLine("value of a=" + v.a );

            Console.WriteLine("value of b=" + v.b);

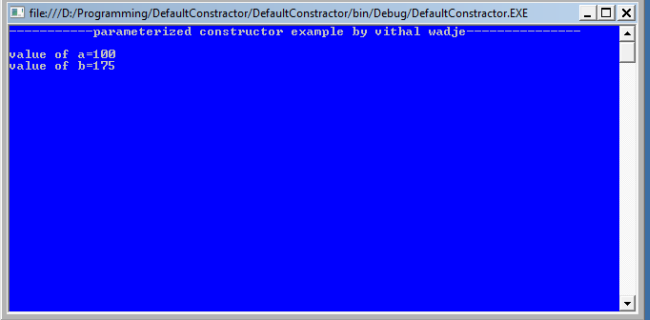
            Console.Read();

        }

    }

}

Now run the application, the output will be as in the following:



**Copy Constructor**  
The constructor which creates an object by copying variables from another object is called a copy constructor. The purpose of a copy constructor is to initialize a new instance to the values of an existing instance.  
  
**Syntax**  
public employee(employee emp)  
{  
name=emp.name;  
age=emp.age;  
}  
  
The copy constructor is invoked by instantiating an object of type employee and ing it the object to be copied.  
  
**Example**

employee emp1=new  employee (emp2);

Now, emp1 is a copy of emp2.   
So let us see its practical implementation.

using System;

namespace copyConstractor

{

    class employee

    {

        private string name;

        private int age;

        public employee(employee emp)   // declaring Copy constructor.

        {

            name = emp.name;

            age = emp.age;

        }

        public employee(string name, int age)  // Instance constructor.

        {

            this.name = name;

            this.age = age;

        }

        public string Details     // Get deatils of employee

        {

            get

            {

                return  " The age of " + name +" is "+ age.ToString();

            }

        }

    }

    class empdetail

    {

        static void Main()

        {

            employee emp1 = new employee("Vithal", 23);  // Create a new employee object.

            employee emp2 = new employee(emp1);         **// here is emp1 details is copied to emp2.**

            Console.WriteLine(emp2.Details);

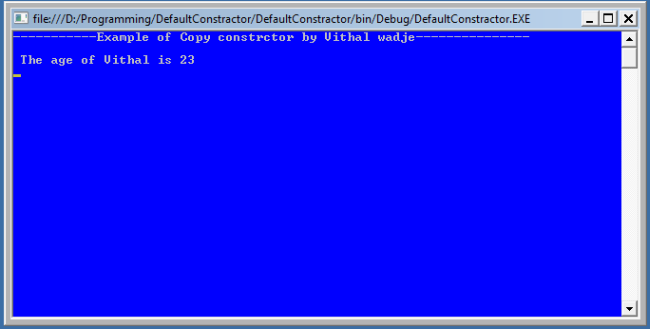
            Console.ReadLine();

        }

    }

}

Now run the program, the output will be as follows:

  
  
**Static Constructor**  
When a constructor is created as static, it will be invoked only once for all of instances of the class and it is invoked during the creation of the first instance of the class or the first reference to a static member in the class. A static constructor is used to initialize static fields of the class and to write the code that needs to be executed only once.  
  
**Some key points of a static constructor is:**

1. A static constructor does not take access modifiers or have parameters.
2. A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.
3. A static constructor cannot be called directly.
4. The user has no control on when the static constructor is executed in the program.
5. A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.

**Syntax**  
class employee  
 {// Static constructor  
  static employee(){}  
 }

Now let us see it with practically

using System;

namespace staticConstractor

{

public class employee

{

    static employee() // Static constructor declaration{Console.WriteLine("The static constructor ");

}

public static void Salary()

 {

    Console.WriteLine();

    Console.WriteLine("The Salary method");

 }

}

class details

{

    static void Main()

    {

        Console.WriteLine("----------Static constrctor example by vithal wadje------------------");

        Console.WriteLine();

        employee.Salary();

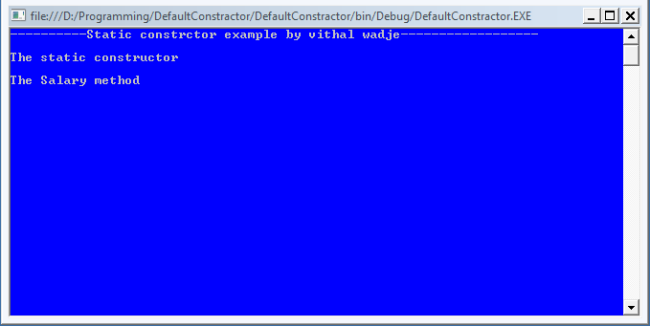
        Console.ReadLine();

    }

  }

}

Now run the program the output will be look as in the following:



**Private Constructor**

When a constructor is created with a private specifier, it is not possible for other classes to derive from this class,

neither is it possible to create an instance of this class. They are usually used in classes that contain static members

only. Some key points of a private constructor are:

1. One use of a private constructor is when we have only static members.
2. It provides an implementation of a singleton class pattern
3. Once we provide a constructor that is either private or public or any, the compiler will not add the parameter-less public constructor to the class.

Now let us see it practically.

using System;

namespace defaultConstractor

{

    public class Counter

    {

        private Counter()   //private constrctor declaration

        {

        }

        public static int currentview;

        public static int visitedCount()

        {

            return ++ currentview;

        }

    }

    class viewCountedetails

    {

        static void Main()

        {

            // Counter aCounter = new Counter();   // Error

            Console.WriteLine("-------Private constructor example by vithal wadje----------");

            Console.WriteLine();

            Counter.currentview = 500;

            Counter.visitedCount();

            Console.WriteLine("Now the view count is: {0}", Counter.currentview);

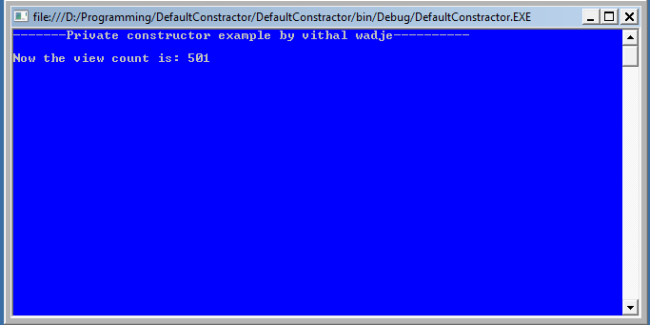
            Console.ReadLine();

        }

    }

}

Now run the application; the output is:



If you uncomment the preceding statement that is commented in the above program then it will generate an error

because the constructor is inaccessible (private).