Sir Syed University of Engineering & Technology, Karachi

Introduction to Classes, Objects & Methods Part2

Course Code : CS-127

Course Title : Object Oriented Programming

Semester : 2nd

Constructors

- In C#, constructor is a special method which is invoked automatically at the time of object creation. It is useful to initialize and set default values for the data members of the new object. The constructor in C# has the same name as class.

There can be two types of constructors in C#.

- ✓ Default constructor
- ✓ Parameterized constructor

Create a Default Constructor: Example

A constructor with no parameters is called a default constructor.

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication43
{
   class car
   {
      public string name;
      public car()
      {
         Console.WriteLine(" Default Constructor invoked... ");
         name = "Civic 2020";
      }
      public void displaycarinfo()
      {
         Console.WriteLine(" Car Name: "+name);
      }
   }
```

```
class testcar
  {
     static void Main(string[] args)
     {
        car honda = new car();
        honda.displaycarinfo();
        Console.ReadKey();

     }
  }
}
```

Constructors

- Note that the constructor name must match the class name, and it cannot have a return type (like void or int).
- Also note that the constructor is called when the object is created.
- A constructor with no parameters is called a default constructor.
- All classes have constructors by default:

"if you do not create a class constructor yourself, C# creates one for you. However, then you are not able to set initial values for fields."

C# Parameterized Constructor

- A constructor which has parameters is called parameterized constructor.

- It is used to provide different values to distinct objects.

Create a Parameterized Constructor: Example

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

namespace ConsoleApplication43
{
   class car
   {
      public string name;
      public string color;
      public int year;
      public car(string modelname,string modelcolor, int modelyear)
      {
         name = modelname;
         color = modelcolor;
         year = modelyear;

      }
      public void displaycarinfo()
      {
         Console.WriteLine("Name :"+name+" Color : "+color+" Year : "+year);
      }

   }
```

```
class testcar
  {
     static void Main(string[] args)
     {

        car honda = new car("civic","White",2020);
        honda.displaycarinfo();

        Console.ReadKey();

     }
  }
}
```

Static keyword in C#

Static is a keyword in C# which is applicable for the following:

- ✓ Variables
- ✓ Methods
- ✓ Constructor
- ✓ Classes

Static Variable

- A static variable is declared with the help of static keyword.
- When a variable is declared as static, then a single copy of the variable is created and shared among all objects at the class level.
- Static variables are accessed with the name of the class, they do not require any object for access.

Static Variable : Example

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

namespace ConsoleApplication45
{
   class Vehicle
   {
      public static string name="Civic";
   }

   class TestVehicle
   {
      static void Main(string[] args)
      {
         Console.WriteLine(Vehicle.name);
         Console.Read();

      }
   }}
```

Static Method

- A static method is declared with the help of static keyword.
- Static methods are accessed with the name of the class.
- A static method can access static and non-static fields.

Static Method: Example

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

namespace ConsoleApplication45
{
   class Numbers
   {
      public int findminimum1(int num1, int num2)
      {
         int min = num2;
         if (num1 < num2) { min = num1; }
         return min;
      }
      public static int findminimum2(int num1, int num2)
      {
         int min = num2;
         if (num1 < num2) { min = num1; }
         return min;
      } }
```

```
class testnumber
   {
      static void Main(string[] args)
      {
         int min;
         min = Numbers.findminimum2(4, 6);
         Console.WriteLine(" Minimum Value is : "+min);
         Console.Read();
      }
   }
}
```

Advantage of C# static keyword

- Memory efficient: Now we don't need to create instance for accessing the static members, so it saves memory.

- Moreover, it belongs to the type, so it will not get memory each time when instance is created.

Static Constructor

- In c#, Static Constructor is useful to perform a particular action only once throughout the application.

- If we declare a constructor as static, then it will be invoked only once irrespective of the number of class instances and it will be called automatically before the first instance is created.

- Generally, in c# the static constructor will not accept any access modifiers and parameters. In simple words, we can say it’s parameterless.

Properties of Static Constructor

The following are the properties of static constructor in c# programming language.

- Static constructor in c# won't accept any parameters and access modifiers.
- The static constructor will invoke automatically, whenever we create the first instance of a class.
- The static constructor will be invoked by CLR so we don't have a control on static constructor execution order in c#.
- In c#, only one static constructor is allowed to create.

C# Static Constructor Example

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

namespace ConsoleApplication45
{
   class user
   {
      public user()       //Default Constructor
      {
         Console.WriteLine(" A Default Constructor... ");
      }
      static user()      //Static Constructor
      {
         Console.WriteLine(" A Static Constructor... ");
      }
   }
```

```
class testuser
  {
     static void Main(string[] args)
     {
        user u = new user();
        u.test();
        Console.Read();

     }
  }
}
```

Note:

When you access a static method or variable, you don't need to create an instance of the class, but the memory for the static member is allocated when the program starts running. The memory for a static member is allocated in a special area of memory called the static memory area or static data area, which is allocated by the operating system when the program starts running.

When you run the program, the static memory area is loaded into memory, and the static method or variable is available for use. Because the memory for the static member is allocated when the program starts running, you don't need to use the **new** keyword to create an instance of the class to access the static member.