CONSTRUCTOR IN C SHARP

In C#, Constructors are the special types of methods of a class which get executed when it's object is created. Constructors are responsible for object initialization and memory allocation of its class. There is always at least one constructor in every class. If you don't write a constructor in class, C# compiler will automatically provide one constructor for that class, called default(parameter less) constructor.

Types of Constructor

Generally, constructors are of three types. But C# doesn't support copy constructor.

1. Default Constructor(Non-Parametrized constructor)
2. Parametrized Constructor
3. Copy Constructor

Default Constructor with example

```
1.  using System;
2. Class example1 {
3.  int roll; float marks;
4.  public example1() //constructor
5. {
6.  roll=0;
7.  marks=0.0;
8.  }
9.  // other members
10.  }
```

The above mentioned constructor example1() doesn't take any argument therefore it is an example of default/non-parameterized constructor. With this type of constructor, the object will be created as follows :

```
1.  example1 obj1 = new example1 ();
```

Parameterized Constructor with example

```
1.  Class example2
```

```
{
int roll;
float marks;
public example2(int a, float b)//constructor
{
 roll = a;
 marks = b;
 }
 //other members
  }
```

The above defined constructor takes two arguments one int and other float to initialize instance members roll and marks for the newly created object and therefore it is called parameterized constructor. With this type of constructor, the object will be created as follows :

```
 example2 obj = new example2( 5 , 33.05F);
```

Constructor Overloading

In C# it possible to overload the constructors, it means we can have more than one constructor with different parameters

```
using System;
public Class OverloadConst
{
public OverloadConst()
{
//Default constructor
}
public OverloadConst(int age)
{
 //Single parameter constructor
 }
 public OverloadConst(int age, string name)
 {
 //two parameter constructor
 }
```

```
16.  }
```

Now, following are the ways through which we can instantiate the object

```
1. OverloadConst obj=new OverloadConst() ; //Default constructor will get called
2. OverloadConst obj=new OverloadConst(33) ; //Single parameter constructor will get called
3. OverloadConst obj=new OverloadConst(35,"asif") ; //Two parameters constructor will get called
```

In above example you have seen that constructor overloading is completely governed by the rule of overloading , i.e, overloading must need two or more method to be different in terms of method signature.

Now, one question here for you , can we have two constructor like this ?

```
1. using System;
2. public Class question {
3.  static question()
4. {
5. // initialize static member only.
6. }
7. public question()
8. {
9. //codes for the first derive class constructor.
10.  }
11.   }
```

The answer to this question is on hold unto the discussion over static constructor.
Many a place you might have been read that a constructor can't be declared as static, this is not true for C# there can be a static constructor in C# .However, it is true for earlier language like C++ .

Static Constructors

Static constructor is a special constructor that gets called before the first object of the class is created. It is used to initialize any static data, or to perform a particular action that needs performed once only.
The time of execution of static constructor is not known but it is definitely before the first object creation - may be at the time of loading assembly.

```
1. //Example1
2. using System;
3. public Class abc
4.  {
5.  static abc()
6. {
7. // initialize static members only.
8.  }
9.  // other methods here.
10.   }
11.  // Example2
12.  using System;
13.  public class MyStaticClass
14.  {
15.  static int count;
16.  static MyStaticClass()
17.  {
18.  count = 0;
19.  Console.WriteLine("Static class is initialized");
20.  }
21.  public static void MyMethod(string name)
22.  {
23.  Console.WriteLine("Static class is initialized " + name);
24.  }
25.  }
26.  MyStaticClass.MyMethod("John") ; //method call
27.  //constructor will be called automatically
28.  Output:
29.  Static class is initialized Hello John
```

Key points about static constructor

1. It can only access the static member(s) of the class.
   Reason : Non static member is specific to the object instance. If static constructor are allowed to work on non static members it will reflect the changes in all the object instance, which is impractical.
2. There should be no parameter(s) in static constructor.
   Reason: Since, It is going to be called by CLR, nobody can pass the parameter to it.
3. Only one static constructor is allowed.
   Reason: Overloading needs the two methods to be different in terms of method/constructor definition which is not possible in static constructor.
4. There should be no access modifier to it.
   Reason: Again the reason is same call to static constructor is made by CLR and not by the object, no need to have access modifier to it

Now, the answer of the question I asked earlier for the following class definition that - can we have two constructors like this?

```
1.  using System;
2. public Class question {
3.  static question()
4.  {
5. //initialize static member only.
6.  }
7. public question()
8. {
9. // codes for the first derive class constructor.
10.   }
11.   }
```

Ans : Yes, this is perfectly valid even if it doesn't match with the rule of overloading concept. The only reason of being it valid is that the time of execution of the two constructors are different, (i) static question() at the time of loading the assembly and (ii) public question() at the time of creating object.

Private Constructor

A private constructor is a special instance constructor used in class that contain static member only. If a class have one or more private constructor and no public constructor then other classes (except nested classes) are not allowed to create instance of this class. Means, we can neither create the object of the class nor it can be inherit by other class.

The public constructor can access the private constructors from within class through constructor chaining.

```
1.  using System;
2. public Class demo {
3. private demo()
4. {
5.  Console.WriteLine("This is no parameter private constructor");
6.  }
7.  public demo(int a):this()
8. {
9. Console.WriteLine("This is one parameter public constructor");
10.   }
11.   // other methods
12.   }
```

The object of the class can be created as :

```
1.  demo obj = new demo(10) ; //it will work fine.
```

But defining object like this won't work.

<="" pre="" style="box-sizing: border-box; overflow: auto; font-family: monospace, monospace; font-size: 1em; vertical-align: top; margin: 10px 0px 0px; font-weight: bold; border-style: none none none solid; border-left-width: 2px; border-left-color: rgb(68, 102, 197); width: 730px; padding: 1px 5px 5px 0px; word-wrap: break-word; background-color: rgb(243, 243, 243);">

```
1.  demo obj = new demo() ; // error of inaccessibility will be occur.
```

Key points about private constructor

1. one use of private construct is when we have only static member.
2. It provide implementation of singleton class pattern
3. Once we provide constructor (private/public/any) the compiler will not add the no parameter public constructor to any class.
4. If we still want to make object of the class having private constructor. We can have a set of public constructor along with the private constructor.

Source : http://www.dotnet-tricks.com/Tutorial/csharp/MSN0140812-Constructor-in-C-Sharp.html