Private Constructors

Private constructors, the constructors with the "private" access modifier, are a bit special case. It is because we can neither create the object of the class, nor can we inherit the class with only private constructors. But yes, we can have the set of public constructors along with the private constructors in the class and the publicconstructors can access the private constructors from within the class through constructor chaining.

Say for example, my class is something like this :

Hide   Copy Code

public class myClass

{

private MyClass()

{

Console.WriteLine("This is no parameter Constructor");

}

public MyClass(int var):this()

{

Console.WriteLine("This is one parameter Constructor");

}

*// Other class methods goes here*

}

Then we can create the object of this class by the statement:

Hide   Copy Code

MyClass obj = new MyClass(10);

The above statement will work fine, but the statement

Hide   Copy Code

MyClass obj = new MyClass();

will raise an error : *'Constructors.MyClass.MyClass()' is inaccessible due to its protection level*

It is possible to have the class with only the private constructors. But yes as I said, such class can neither be instantiated nor be inherited. If we try to inherit the class with only private constructors then we will get the same error as above. Also recall, once you provide constructor from your side the compiler will not add the no-parameter public constructor to your class.

Well, one of the usage scenarios of such class could be – when you have only static members in the class and you don't need to instantiate it.

Phew… lost… Anything left in constructors? Yes, Static Constructors. Ha!! Now, what are they? Let us see..

Static Constructors

This is a new concept introduced in C#. By new here, I mean that it was not available for the C++ developers. This is a special constructor and gets called before the first object is created of the class. The time of execution cannot be determined, but it is definitely before the first object creation - could be at the time of loading the assembly.

The syntax of writing the static constructors is also damn simple. Here it is:

Hide   Copy Code

public class myClass

{

static myClass()

{

*// Initialization code goes here.*

*// Can only access static members here.*

}

*// Other class methods goes here*

}

Notes for Static Constructors:

1. There can be only one static constructor in the class.
2. The static constructor should be without parameters.
3. It can only access the static members of the class.
4. There should be no access modifier in static constructor definition.

Ok fine, all the above points are fine, but why is it like that? Let us go step by step here.

Firstly, the call to the static method is made by the CLR and not by the object, so we do not need to have the access modifier to it.

Secondly, it is going to be called by CLR, who can pass the parameters to it, if required. So we cannot have parameterized static constructor.

Thirdly, non-static members in the class are specific to the object instance. So static constructor, if allowed to work on non-static members, will reflect the changes in all the object instances, which is impractical. So static constructor can access only static members of the class.

Fourthly, overloading needs the two methods to be different in terms of methods definition, which you cannot do with Static Constructors, so you can have at the most one static constructor in the class.

Now, one question raises here, can we have two constructors as:

Hide   Copy Code

public class myClass

{

static myClass()

{

*// Initialization code goes here.*

*// Can only access static members here.*

}

public myClass()

{

*// Code for the First myDerivedClass Constructor.*

}

*// Other class methods goes here*

}

This is perfectly valid, though doesn't seem to be in accordance with overloading concepts. But why? Because the time of execution of the two methods are different. One is at the time of loading the assembly and one is at the time of object creation.

Constructors FAQs

1. Is the constructor mandatory for a class?

Yes, it is mandatory to have the constructor in the class and that too should be accessible for the object i.e., it should have a proper access modifier. Say, for example, we have only private constructor(s) in the class and if we are interested in instantiating the class, i.e., want to create an object of the class, then having only private constructor will not be sufficient and in fact it will raise an error. So, proper access modifies should be provided to the constructors.

1. What if I do not write the constructor?

In such case, the compiler will try to supply the no parameter constructor for your class, behind the scene. Compiler will attempt this only if you do not write the constructor for the class. If you provide any constructor (with or without parameters), then compiler will not make any such attempt.

1. What if I have the constructor public myDerivedClass(), but not the public myBaseClass()?

It will raise an error. If either the no parameter constructor is absent or it is in-accessible (say it isprivate), it will raise an error. You will have to take the precaution here.

1. Can we access static members from the non-static (normal) constructors?

Yes, we can. There is no such restriction on non-static constructors. But there is one on static constructors that it can access only static members.