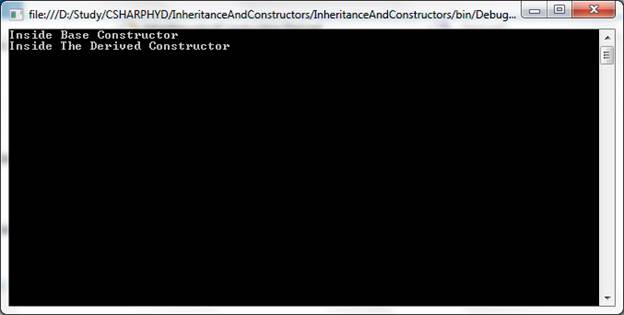
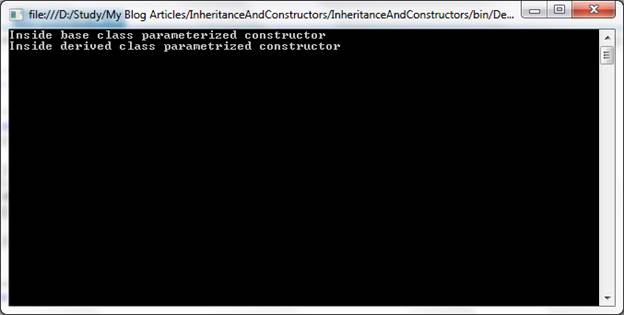
**Inheritance And Constructors**

If you provide the constructor in the derived class, you must provide an appropriate constructor in the base class. If you have provided a constructor in the derived class and no constructor in the base class, the compiler will flag an error.  
  
Now, suppose I have a constructor in the derived/subclass class and in the base class as well, remember that the base class constructor will be called first and then derived class constructor. If I create the instance of the derived class, let us see this behavior with an example.  
  
Suppose, I have two classes, given below:

1. **class** Base
2. {
3. **public** Base() {
4. Console.WriteLine("Inside Base Constructor");
5. }
6. ~Base() {
7. Console.WriteLine("Inside Base Destructor");
8. }
9. }
10. **class** Derived: Base {
11. **public** Derived() {
12. Console.WriteLine("Inside The Derived Constructor");
13. }
14. ~Derived() {
15. Console.WriteLine("Inside The Derived Destructor");
16. }
17. }

The derived class is getting inherited from the base class. Both the classes have a default constructor, where I am printing appropriate messages.  
  
Now, if I create an instance of a derived class as  
Derived obj=new Derived();  
  
The output of my program will be:  
  
  
  
The important point to note from this output is that the base class constructor is being called first and then derived class constructor. Many interviewers will give code snap like this and will ask for the output. Thus, remember it. Another point to note is that the default constructor (constructor without any parameters is called a default constructor) of base class is being called automatically from the derived class constructor.  
  
Now, I want to manually call the constructor of the base class from the derived class constructor. We need to do this, using base keyword.  
  
Let us see the example.  
  
Suppose, I have two classes, given below:

1. **class** Class1
2. {
3. **protected** **int** a, b;
4. **public** Class1() {
5. a = 0;
6. b = 0;
7. Console.WriteLine("Inside base class default constructor");
8. }
9. **public** Class1(**int** a, **int** b) {
10. **this**.a = a;
11. **this**.b = b;
12. Console.WriteLine("Inside base class parameterized constructor");
13. }
14. }
15. **class** Class2: Class1 {
16. **int** c;
17. **public** Class2(**int** a, **int** b, **int** c): **base**(a, b) {
18. **this**.c = c;
19. Console.WriteLine("Inside derived class parametrized constructor");
20. }
21. }

Observe the line public Class2(int a, int b, int c) : base(a, b)  
  
Now, here I am calling the base class constructor, using base(a, b)  
  
Now, when I create the instance of the derived class as  
  
Class2 obj2 = new Class2(2, 3, 4);  
  
Thus, the base class constructor, that takes two parameters is called first and then the derived class constructor. Output of this is:  
  
  
  
Note that only the default constructors of the base class will be called automatically, when we create an instance of the derived class, as we have seen in the first example. If we want to call parameterized constructor of the base class, you need to use a base keyword. One thing to note is, in inheritance from the derived class constructor, the base class constructor will be called either automatically or manually (using base keyword)  
  
If we create a constructor in the derived class, no constructor in the base class compiler will flag an error. Thus, when you create a constructor in the derived class, you must create a constructor in the base class as well.