C# Interface

Interface cannot have method body and cannot be instantiated just have signature(s) of method.

It is used *to achieve multiple inheritance* which can't be achieved by class. It is used *to achieve fully abstraction* because it cannot have method body.

Its implementation must be provided by class or struct. The class or struct which implements the interface, must provide the implementation of all the methods declared inside the interface.

C# interface example

Let's see the example of interface in C# which has draw() method. Its implementation is provided by two classes: Rectangle and Circle.

```
1.  using System;
2.
3.  public interface Drawable
4.  {
5.  void draw();
6.  }
7.
8.  public class Rectangle : Drawable
9.  {
10. public void draw()
11. {
12. Console.WriteLine("drawing rectangle...");
13. }
14. }
15.
16. public class Circle : Drawable
17. {
18. public void draw()
19. {
20. Console.WriteLine("drawing circle...");
21. }
22. }
```

```
23. public class TestInterface
24. {
25.     public static void Main()
26.    {
27.     Drawable d;
28.    d = new Rectangle();
29.    d.draw();
30.    d = new Circle();
31.    d.draw();
32.   }
33. }
```

Output:

```
drawing ractangle...
drawing circle...
```

C# Abstract class

In C#, abstract class is a class which is declared abstract. It can have abstract and non-abstract methods. It cannot be instantiated. Its implementation must be provided by derived classes. Here, derived class isFORCED to provide the implementation of all the abstract methods.

Let's see an example of abstract class in C# which has one abstract method draw(). Its implementation is provided by derived classes: Rectangle and Circle. Both classes have different implementation.

```
1. using System;
2. public abstract class Shape
3. {
4. public void display()
5. {
6. Console.WriteLine(“ I m Abstract class simple method ”);
7. }
8. public abstract void draw();
9. }
```

```
10. public class Rectangle : Shape
11. {
12.     public override void draw()
13.     {
14.         Console.WriteLine("drawing rectangle...");
15.     }
16. }
17. public class Circle : Shape
18. {
19.     public override void draw()
20.     {
21.         Console.WriteLine("drawing circle...");
22.     }
23. }
24. public class TestAbstract
25. {
26.     public static void Main()
27.     {
28.         Shape s;
29.         s = new Rectangle();
30.         s.draw();
31.         s = new Circle();
32.         s.draw();
33.     }
34. }
35.
```

Output:

```
drawing ractangle...
drawing circle...
```