C# - INTERFACES

An interface is defined as a syntactical contract that all the classes inheriting the interface should follow. The interface defines the **'what'** part of the syntactical contract and the deriving classes define the **'how'** part of the syntactical contract.

Interfaces define properties, methods, and events, which are the members of the interface. Interfaces contain only the declaration of the members. It is the responsibility of the deriving class to define the members. It often helps in providing a standard structure that the deriving classes would follow.

Abstract classes to some extent serve the same purpose, however, they are mostly used when only few methods are to be declared by the base class and the deriving class implements the functionalities.

Declaring Interfaces

Interfaces are declared using the interface keyword. It is similar to class declaration. Interface statements are public by default. Following is an example of an interface declaration:

```
public interface ITransactions
{
   // interface members
   void showTransaction();
   double getAmount();
}
```

Example

The following example demonstrates implementation of the above interface:

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

namespace InterfaceApplication
{
   public interface ITransactions
   {
      // interface members
      void showTransaction();
      double getAmount();
   }

   public class Transaction : ITransactions
   {
      private string tCode;
      private string date;
      private double amount;
      public Transaction()
      {
         tCode = " ";
         date = " ";
         amount = 0.0;
      }

      public Transaction(string c, string d, double a)
      {
         tCode = c;
         date = d;
         amount = a;
      }
```

```
        public double getAmount()
        {
            return amount;
        }

        public void showTransaction()
        {
            Console.WriteLine("Transaction: {0}", tCode);
            Console.WriteLine("Date: {0}", date);
            Console.WriteLine("Amount: {0}", getAmount());
        }
    }
    class Tester
    {
        static void Main(string[] args)
        {
            Transaction t1 = new Transaction("001", "8/10/2012", 78900.00);
            Transaction t2 = new Transaction("002", "9/10/2012", 451900.00);
            t1.showTransaction();
            t2.showTransaction();
            Console.ReadKey();
        }
    }
}
```

When the above code is compiled and executed, it produces the following result:

```
Transaction: 001
Date: 8/10/2012
Amount: 78900
Transaction: 002
Date: 9/10/2012
Amount: 451900
```