C# Interfaces

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.

C# Interfaces

- 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 a few methods are to be declared by the base class and the deriving class implements the functionalities.

C# Interfaces

- Interface members are implicitly public abstract(virtual).
- Interface members must not be static.
- Classes and structs may implement multiple interfaces.
- Interfaces can extend other interfaces.

C# Interfaces

- Implemented by Classes and Structs
- A class can inherit from a single base class, but implement multiple interfaces.
- A struct cannot inherit from any type, but can implement multiple interfaces.
- Implemented interface methods must not be declared as override.

C# Interfaces

- Declaring Interfaces
  - Interfaces are declared using the interface keyword.
  - It is similar to class declaration.
  - Interface statements are public by default.

C# Interfaces

- An interface defines **<u>but does not implement</u>** a group of methods, properties, events and indexers

```
public interface IDisplayable
{
    void Display();
}
```

C# Interfaces

- Any class may then inherit the interface, which is then a contract that the class implements the methods etc. defined in the interface

public class CurrentAccount : Account, IDisplayable

C# Interfaces

- Interfaces are used to support the concept of multiple inheritance in C#
- The inheriting class must provide an implementation for the interface members

```
#region IDisplayable Members
public void Display()
{
    Console.WriteLine("{0}   {1,}  {2,}", Id, Balance, CreditLimit.);
}
#endregion
```

C# Interfaces – Implicit vs Explicit

- Interface members may be implemented either implicitly or explicitly in a class that inherits the interface
- A method is implemented implicitly as follows

```
public  void Display()
{
    Console.WriteLine("{0}   {1,}  {2,}", Id, Balance, CreditLimit.);
}
```

C# Interfaces – Implicit vs Explicit

- A method is implemented explicitly as follows

```
 public  void IDisplayable.Display()
{
     Console.WriteLine("{0}   {1,}  {2,}", Id, Balance, CreditLimit.);
}
```

C# Interfaces – Implicit vs Explicit

- If a method is implemented implicitly then it may be accessed through the class or the interface

```
CurrentAccount myAcc = new CurrentAccount ();
myAcc.Display();
IDisplayable myInterface = myAcc;
myInterface.Display();
```

C# Interfaces – Implicit vs Explicit

- If a method is implemented explicitly then it may only be accessed through the interface

```
CurrentAccount myAcc = new CurrentAccount ();
IDisplayable myInterface = myAcc;
myInterface.Display();
```

C# Interfaces – Implicit vs Explicit

- In general use the implicit implementation