Sir Syed University of Engineering & Technology, Karachi

Abstract class and interface

Course Code : CS-127

Course Title : Object Oriented Programming

Semester : 2nd

Batch - 2020

Department of Computer Science / Information Technology

Interface

- A contract between the two or more humans binds them to act as per the contract. In the same way, an interface includes the declarations of related functionalities. The entities that implement the interface must provide the implementation of declared functionalities.

- Interface can be defined using the interface keyword.

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

Interface

- It is used *to achieve multiple inheritance* which can't be achieved by class.
- 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.
- Interfaces can’t have private members.
- By default all the members of Interface are public and abstract.
- Interface cannot contain fields because they represent a particular implementation of data.

C# Interface Example

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

namespace ConsoleApplication63
{
   interface Icolor
   {
      void coloring();//Interface cannot have method body
      void info();//just have signature(s) of method
   }
   class red : Icolor
   {
      public void info()
      {
         Console.WriteLine(" i m the red class... ");
      }
      public void coloring()
      {
         Console.ForegroundColor = ConsoleColor.White;//ForegroundColor
property in the console class
         Console.BackgroundColor = ConsoleColor.Red;
      }
   }
   class blue : Icolor
   {
      public void info()
      {
         Console.WriteLine(" i m the blue class... ");
      }
      public void coloring()
      {
         Console.ForegroundColor = ConsoleColor.White;//ForegroundColor
property in the console class
         Console.BackgroundColor = ConsoleColor.Blue;
      }

   }
class green : Icolor
   {
      public void info()
      {
         Console.WriteLine(" i m the green class... ");
      }
      public void coloring()
      {
         Console.ForegroundColor =
ConsoleColor.White;//ForegroundColor property in the console
class
         Console.BackgroundColor = ConsoleColor.Green;
      }
   }
   class Program
   {
      static void Main(string[] args)
      {
         Icolor clr = new red();//we created an instance for
interface "Icolor" using red class;
         clr.coloring();
         clr.info();
         Icolor clb = new blue();
         clb.coloring();
         clb.info();
         Icolor clg = new green();
         clg.coloring();
         clg.info();
         Console.Read();
      }
   }
}
```

Multiple Inheritance with Interface

- As discussed, c# will not support multiple inheritance of classes but that can be achieved by using the interface.
- Following is the example of implementing a multiple inheritance using interfaces in c# programming language.

Multiple Inheritance with Interface Example

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

•       namespace ConsoleApplication63
•       {
•          interface Iname
•          {
•             void displayname(string name);
•          }
•          interface Ilocation
•          {
•             void displaylocation(string location);
•          }
•          interface Iage
•          {
•             void displayage(string age);
•          }
•          class User : Iname, Ilocation, Iage
•          {
•            public void displayname(string name)
•             {
•                Console.WriteLine(" Name :  "+name);
•             }
•            public void displaylocation(string location)
•             {
•                Console.WriteLine(" Location :  " + location);
•             }
•            public void displayage(string age)
•             {
•                Console.WriteLine(" Age :  " + age);
•             }
•          }
```

```
•          class Program
•          {
•             static void Main(string[] args)
•             {

•                User u = new User();
•                u.displayname(" Ali Ahmed ");
•                u.displaylocation(" Karachi ");
•                u.displayage("65");
•                Console.Read();
•             }
•          }
•       }
```

Abstract Classes and Methods

- Data abstraction is the process of hiding certain details and showing only essential information to the user.

- Abstraction can be achieved with either abstract classes or interfaces.

The abstract keyword is used for classes and methods:

✓ **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

✓ **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from).

Abstract class

- 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 is FORCED to provide the implementation of all the abstract methods.

Example of abstract class

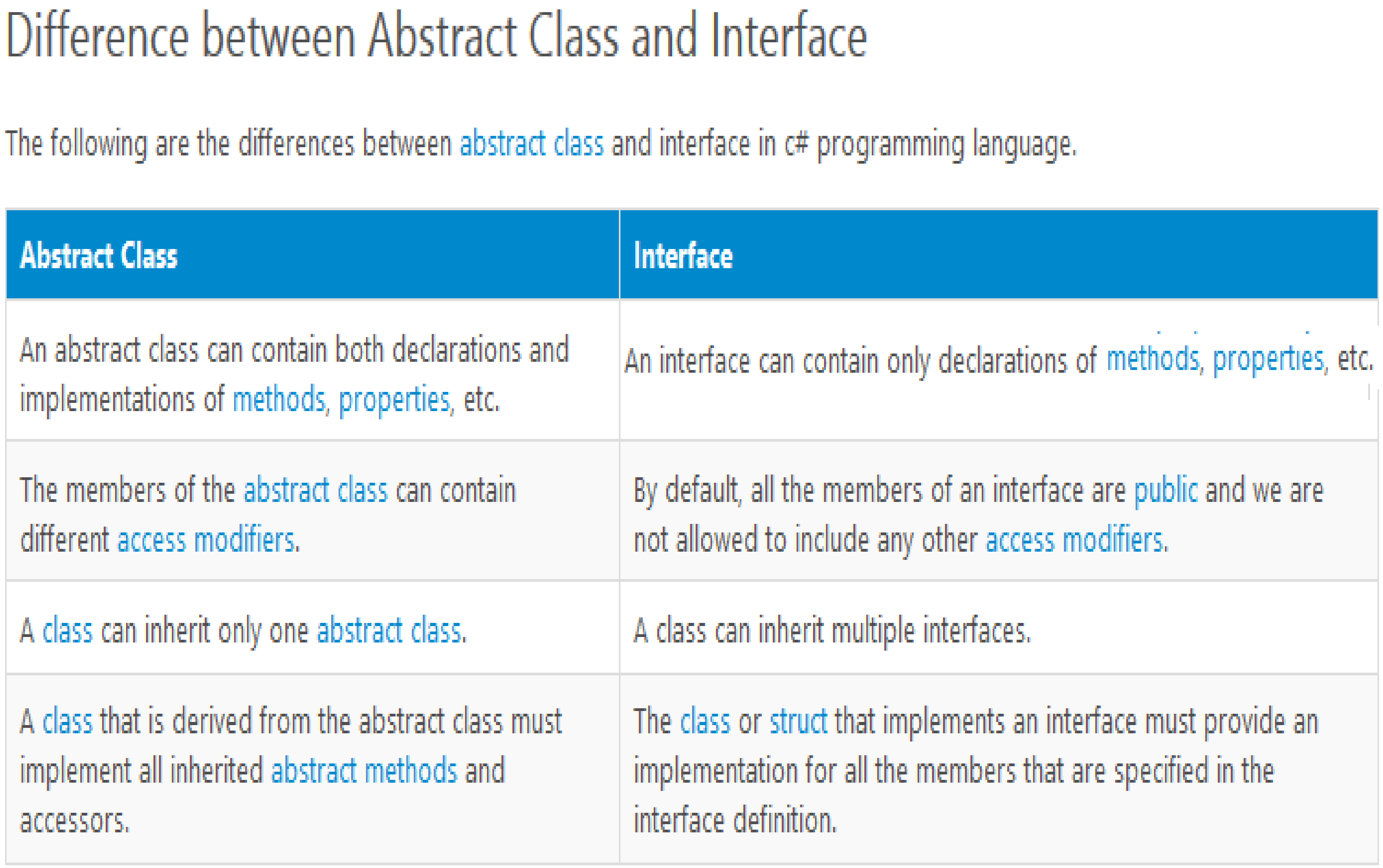
```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication63
{
   abstract class colors
   {
      public abstract void color();
      public void displayinfo()
      {
         Console.WriteLine(" i m the simple method of abstract class ");
      }
   }
   class red : colors
   {
      public override void color()
      {
         Console.ForegroundColor =
ConsoleColor.White;//ForegroundColor property in the console class
         Console.BackgroundColor = ConsoleColor.Red;
         Console.WriteLine(" i m the red class.... ");
      }
   }
   class blue : colors
   {
      public override void color()
      {
         Console.ForegroundColor =
ConsoleColor.White;//ForegroundColor property in the console class
         Console.BackgroundColor = ConsoleColor.Blue;
         Console.WriteLine(" i m the blue class.... ");
      }

   }
class green : colors
   {
      public override void color()
      {
         Console.ForegroundColor =
ConsoleColor.White;//ForegroundColor property in the console class
         Console.BackgroundColor = ConsoleColor.Green;
         Console.WriteLine(" i m the green class.... ");
      }

   }
   class Program
   {
      static void Main(string[] args)
      {
         colors clr = new red();
         clr.color();
         clr.displayinfo();
         colors clb = new blue();
         clb.color();
         clb.displayinfo();
         colors clg = new green();
         clg.color();
         clg.displayinfo();

         Console.Read();
      }
   }
}
```



Difference between Abstract Class and Interface

The following are the differences between abstract class and interface in c# programming language.

Abstract Class	Interface
An abstract class can contain both declarations and implementations of methods, properties, etc.	An interface can contain only declarations of methods, properties, etc.
The members of the abstract class can contain different access modifiers.	By default, all the members of an interface are public and we are not allowed to include any other access modifiers.
A class can inherit only one abstract class.	A class can inherit multiple interfaces.
A class that is derived from the abstract class must implement all inherited abstract methods and accessors.	The class or struct that implements an interface must provide an implementation for all the members that are specified in the interface definition.