Practical 2

Aim: Demonstrate Inheritance, Abstract class and Interfaces in C#. (java file conversion)

1: Inheritance

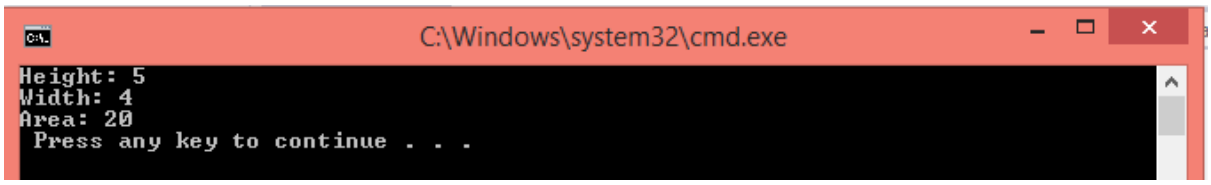
Note: If you are using the inheritance in C# and are want to override (hide) the base class method but forget to use the 'new' keyword then you will get the warning: "Use the new keyword if hiding was intended(CS0108)".So all you need to do is to put the keyword 'new' before the overriding method.

PROGRAM

```
using System;
namespace InheritanceApplication
{
  class Shape
  {
    protected Shape(int w,int h)
        {
                width=w;
                height=h;
        }
    protected int width;
    protected int height;
    public void Display()
   {
      Console.WriteLine("Height: {0}", height);
      Console.WriteLine("Width: {0}", width);
    }
  }
   class Rectangle: Shape // Derived class
  {
        public Rectangle(int w,int h):base(w,h)
        {}
    public int getArea()     {
      return (width * height);
    }
    new public void Display()     {
      base.Display();
      Console.WriteLine("Area: {0}", getArea());
    }
  }
  class RectangleTester
  {
```

```
    static void Main(string[] args)
    {
        Rectangle Rect = new Rectangle(4,5);
        Rect.Display();
        Console.ReadKey();
    }
  }
}
```

OUTPUT



2: Abstract class

Note: Classes can be declared as abstract by putting the keyword abstract before the class definition. An abstract class cannot be instantiated. The purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share.

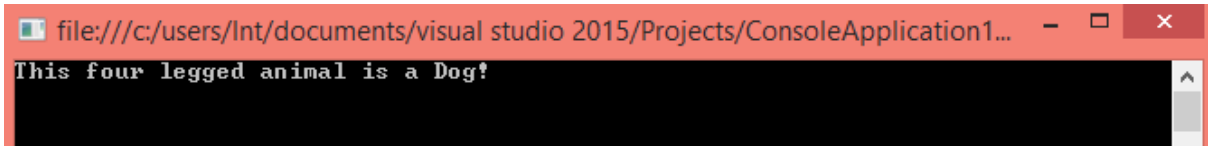
PROGRAM

```
using System;
namespace AbstractClasses
{
   class Program
   {
      static void Main(string[] args)
      {
         Dog dog = new Dog();
         Console.WriteLine(dog.Describe());
         Console.ReadKey();
      }
   }

   abstract class FourLeggedAnimal
   {
      public virtual string Describe()
      {
         return "Not much is known about this four legged animal!";
      }
   }
   class Dog : FourLeggedAnimal
   {
```

```
        public override string Describe()
        {
      return "This four legged animal is a Dog!";
        }
   }
}
```

#OUTPUT



3: <u>Interfaces</u>

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

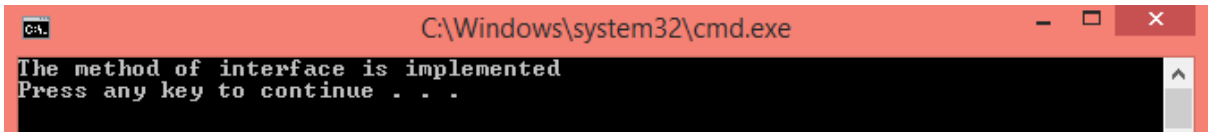
<u>PROGRAM</u>

```
using System;
namespace sampleInterface
{
  public interface Iabc
  {
    // interface members
    void Display();
  }

  public class A:Iabc
        {
                public  void Display()
                {
                        Console.WriteLine("The method of interface is implemented");
                }
        }
  class EntryClass
  {
        public static void Main()
        {
                A a=new A();
                a.Display();
        }
  }
```

}

#OUTPUT


C:\Windows\system32\cmd.exe
The method of interface is implemented
Press any key to continue . . .