INTERFACES AND ABSTRACT CLASSES

ABSTRACT CLASSES

- An abstract class can be inherited by other classes, but not used to create an object.
- An abstract method is also created using the abstract keyword. Abstract methods have no body. They cannot have private access.
- An abstract class may not contain abstract methods, but any class that does contain abstract methods must be declared as abstract.

ABSTRACT CLASSES

Example

```
abstract class Shape
{
    protected string Name;
    public virtual double PrintName()
    {

Console.WriteLine(Name);
    }
    public abstract double GetArea();
}
```

INTERFACES

An interface is a special type of coding element that provides many of the advantages of multiple inheritance. An interface defines a set of public methods that can be implemented by a class.

A class that implements an interface must provide an implementation for each method defined by the interface.

INTERFACES VS ABSTRACT CLASSES

Advantages of an abstract class

- Can use instance as well as static variables and constants
- Can define regular methods that contain code (concrete), and abstract methods
- Can define fields

INTERFACES VS ABSTRACT CLASSES

Advantages of an Interface

- A class can directly implement multiple interfaces
- Any object created from a class that implements an interface can be used wherever the interface is accepted

INTERFACES

Example

```
interface IShape
{
    void GetArea();
}
```

HOW TO WORK WITH INTERFACES

- An interface can inherit one or more interfaces
- An interface can't inherit a class
- A class that implements an interface must implement all the methods declared by the interface as well as all the methods declared by any inherited interfaces unless the class is defined as abstract.