Need of Interfaces

Few main reasons for using the interfaces are as follows:

- The main feature of Object Oriented Programming is the ability to reuse the code. One way to achieve this is by using the interface although it is limited to reusing the classes within a program.
- Java does not support multiple inheritance. Since multiple inheritance is an important concept in OOP paradigm, java provides an alternate way of implementing this concept using interface.
- Interface also provide the another feature of Object Oriented Programming i.e. abstraction.

Introduction

- It is a collection of abstract methods.
- A class implements an interface, by inheriting the abstract methods of the interface.
- All the methods of the interface need to be defined in the class that implements interface (Except abstract class).
- Writing an interface is similar to writing a class, but
  - A class describes the attributes and behaviours of an object.
  - An interface contains behaviours that a class implements.

Syntax

```
interface <interface_name>
{
	variable declaration;
	method declaration;
}
```

- All variables are declared as constants.
- Methods declaration will contains the list of methods without any body.

Example

<u>Test.java</u>

```
interface Area
{
	final static float pi=3.14F;
	float calculate (float x, float y);
}
class Rectangle implements Area
{
	public float calculate(float x, float y)
	{
		return(x*y);
	}
}
class Circle implements Area
{
        public float calculate(float x, float y)
        {
                return(pi*x*x);
        }
}
class Test
{
        public static void main(String args[])
        {
                Rectangle obj1=new Rectangle();
                Circle obj2=new Circle();
                System.out.println("Area of Rectangle is "+obj1.calculate(10,20));
                System.out.println("Area of Circle is "+obj2.calculate(10,0));
        }
}
```

Output:

```
Area of Rectangle is 200.0
Area of Circle is 314.0
```

Extending interfaces

Like classes, interfaces can also be extended. That is, an interface can be sub interfaced from other interfaces. The new sub interface will inherit all the members of the super interface in the manner similar to subclasses. This is achieved using the keyword "*extends*" as shown below:

```
interface name1
{
        body of name1
}
interface name2 extends name1
{
        body of name2
}
```

Implementing Interfaces

A class can extend another class while implementing interfaces. When a class implements more than one interface, they are separated by a comma. This is achieved using the keyword "*implements*" as shown below:

```
interface name1
{
        body of name1
}
interface name2
{
        body of name2
}
```

```
class cname1
{
        body of cname1
}
class cname2 implements name1, name2
{
        body of cname2
}
```

Difference between Abstract Class & Interface

Abstract Class	Interface
It can have abstract and non-abstract methods.	It can have only abstract methods.
It doesn’t support multiple inheritance.	It supports multiple inheritance.
It can have final, non-final, static and non-static variables.	It can have only static and final variables.
It can have static methods, main methods or constructors.	It cannot have static methods, main methods or constructors.
It can provide the implementation of interface.	It cannot provide the implementation of abstract class.
“abstract” keyword is used to declare abstract class.	“interface” keyword is used to declare interface.

Questions asked in semester paper

No Questions