Java Programming

Unit – 3

Interfaces, Packages and Exceptions

Dr. Y. J. Nagendra Kumar
Professor of IT
Dean - Technology and Innovation Cell - GRIET

Interfaces

Interfaces

- Java does not support multiple inheritance. That is, classes in java cannot have more than one superclass.
- Java provides an alternate approach known as interfaces to support the concept of multiple inheritance.
- Interfaces are syntactically similar to classes, but they define only abstract methods and final fields.
- This means that interfaces do not specify any code to implement these methods and data fields contain only constants.
- Once it is defined, any number of classes can implement an interface. Also, one class can implement any number of interfaces.

Defining an Interface

- An interface is defined much like a class. This is the general form of an interface:

access interface *name*
{
type final-varname1 = value;
type final-varname2 = value;
// ...
type final-varnameN = value;
return-type method-name1(*parameter-list*);
return-type method-name2(*parameter-list*);
// ...
return-type method-nameN(*parameter-list*);
}
Here, *access* is either **public** or not used

Interfaces

- Methods are, essentially, abstract methods
- Each class that includes an interface must implement all of the methods.
- Variables can be declared inside of interface declarations. They are implicitly **final** and **static**, meaning they cannot be changed by the implementing class.
- All methods and variables are implicitly **public** if the interface, itself, is declared as **public**.

Ex:

```
interface shape
{
void area(int param);
}
```

Implementing Interfaces

- Once an **interface** has been defined, one or more classes can implement that interface.
- To implement an interface, include the **implements** clause in a class definition, and then create the methods defined by the interface.

Syntax:

access class *classname* [extends *superclass*]
[implements *interface* [,*interface...*]]
{
    // class-body
}

Interfaces contd.,

- Here, *access* is either **public** or not used. If a class implements more than one interface, the interfaces are separated with a comma.
- The methods that implement an interface must be declared **public**.
- Also, the type signature of the implementing method must match exactly the type signature specified in the **interface** definition.

Interfaces

```
class circle implements shape
{
        // Implement shape's interface
        public void area(int p)
        {
        System.out.println("Area of Circle "+3.14*p*p);
        }
}
```

Interfaces Second Example

```
class student
{
        int rollno;

        void getno(int a)
        {
                rollno=a;
        }
        void putno()
        {
System.out.println(" Roll Number : "+rollno);
        }
}
```

```
class test extends student
{
    double m1,m2;

    void getmarks(double a,double b)
    {
        m1=a; m2=b;
    }
    void putmarks()
    {
    System.out.println("M1 : "+m1+" M2 : "+m2);
    }
}
```

Interfaces Second Example contd.,

```
interface sports
{
        final static double spwt=10;
        void putspwt();
}
class result extends test implements sports
{
        double total;
        public void putspwt()
        {
        System.out.println("Sports Marks Weightage
        : "+spwt);
        }
        void show()
        {       total=m1+m2+spwt;
                putno();
                putmarks();
                putspwt();
        System.out.println("Total Marks :"+total);
        }
}
class interface2
{       public static void main(String ar[])
        {       result r=new result(); r.getno(786);
                r.getmarks(78.5,65.25);  r.show();
        }
}
```

End of Unit III Part 1