UNIT3

Interfaces: Multiple Inheritance

Introduction

Java does not support multiple inheritance. That is classes in java cannot have more than one superclass. For example

class A **extends class** B **extend**s C

{

…….

…….

}

Is not permitted in java.

Java provides an alternative approach known as interfaces to support the concept of multiple inheritance.

Defining Interfaces

1.The interface is basically a kind of class.

2. Which defines only abstract methods and final fields, means interfaces do not specify any code to implement these methods, Data fields contain only constants.

3. The syntax for defining interface is:

interface Interfacename

{

Variable declarartion;

Method declaration;

}

4.Here, interface is the keyword and Interfacename is any valid java identifier.

5. Here is an example of an interface definition that contains one variable and two method.

```
interface Area
{
    static final float pi=3.14f;
    float compute(float x, float y);
    void show();
}
```

6. Note that the code for method is not included in the interfaces and method declaration simply ends with a semicolon.

7. The class that implements this interface must define the code for the method.

Extending Interfaces

1. Interfaces can be extended from the other interfaces the new interface will inherit all data members and methods of superinterface.
2. This is achieved using the keyword **extends** as shown below.

Syntax: interface one **extends** two

```
{
    Body of two
}
```

3. Ex.
```
interface ItemConstant
{
    int code = 1001;
     String name= "Fan";
}
interface Item extends ItemConstant
{
    void display();
}
```
4. The interface Item can inherit both code and name into it.
5. Note that all variables in an interface are treated as constants although the keywords **static** and **final** are not used.
6. We can also combine several interfaces together into a single interface as shown below.

```
interface ItemConstant
{
    int code = 1001;
     String name= "Fan";
}
interface ItemMethod
{
    void display();
}

interface Item extends ItemConstant, ItemMethod

{
    …….
    …….
}
```

Implementing Interfaces

1. Interfaces are used as superclasses whose properties are inherited by classes this is done as follows.

Syntax:
```
class classname implements interfacename
{
        Body of the class elements
```

}

2. A class can extend another class while implementing interfaces.

class classname **extends** superclass **implements**

interface1,interface2,……

{

Body of classname

}

3. The implementation of interfaces can take various forms as illustrated in Fig. below

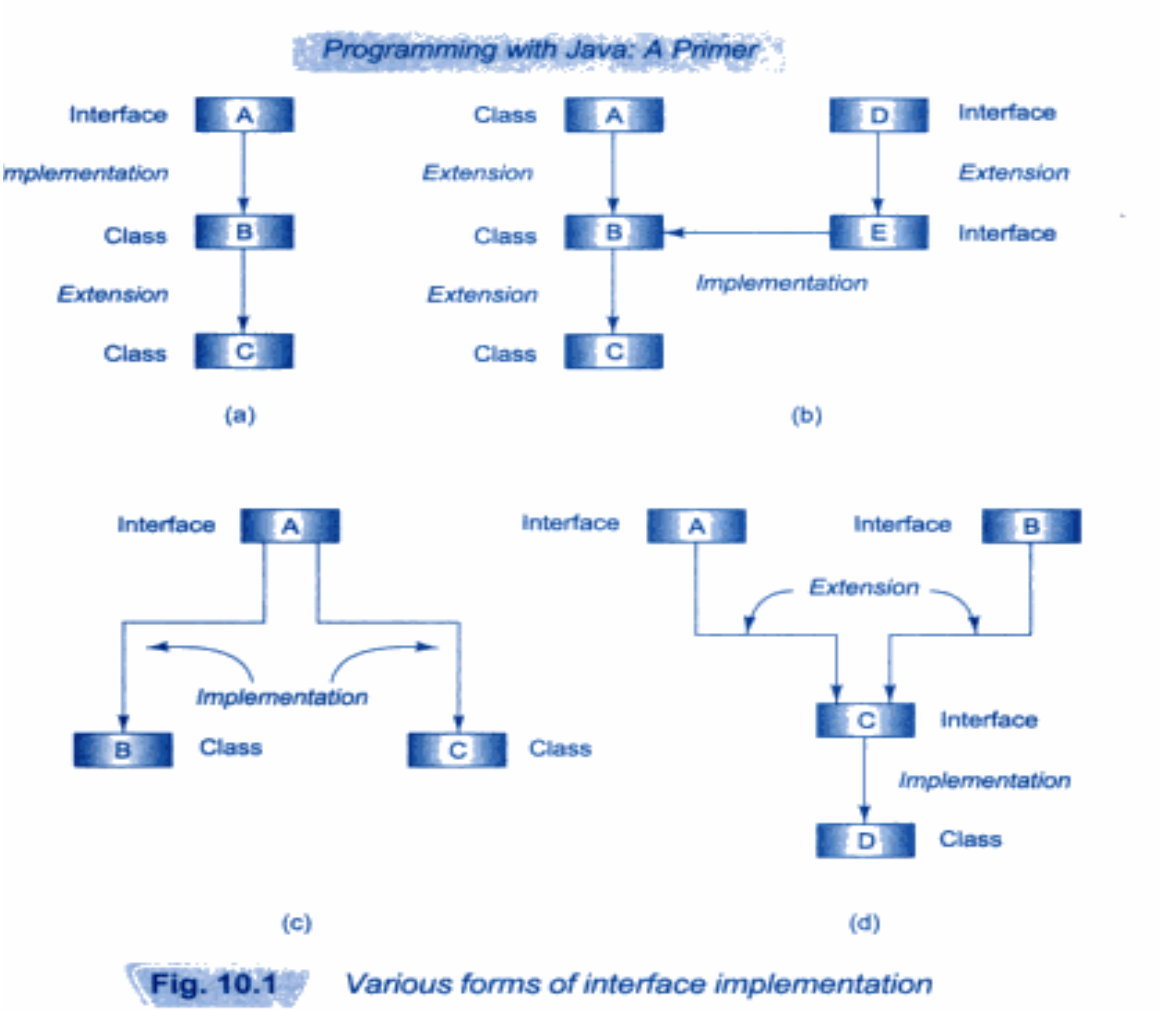

Fig. 10.1 *Various forms of interface implementation*

4. Implementation of class types is illustrated in below program. First create an interface **Area** and implement the same in two different classes, **Rectangle** and **Circle.**
5. Here we create a object of Type **Area** interface and assign the reference of the rectangle object rec to area.
6. We use area object to call compute method of rectangle and circle class by defining interface object as shown below.

```
Area area;
```

/* Implementing Interfaces Example1

```
interface Area
{
    float pi=3.14f;
    float compute(float x,float y);
}
class rectangle implements Area
{
    public float compute(float x,float y)
    {
        float a=x*y;
        return(a);
    }
}
class circle implements Area
{
    public float compute(float x,float y)
```

```
        {
                float a=pi* x*x;
                return(a);
        }
}
class m
{
        public static void main(String args[])
        {
                rectangle rect=new rectangle();
                circle cir=new circle();
                Area area;    //Interface object
                area = rect;     //area refers to rect object
                System.out.println("Area of Rectangle = " + area.compute(10,20));
                area = cir;       //area refers to cir object
                System.out.println("Area of Circle = " + area.compute(10,0));
        }
}
```

/* Implementing Interface Example 2

```
interface A
{
        int x=10;
        int y=20;
}
```

```
interface B extends A
{
        void display();
        void mul();
}
Class C implements B
{
        public void display()
        {
                System.out.println("X="+x);
                System.out.println("Y="+y);
        }
        public void mul()
        {
                int z=x*y;
                System.out.println("Result="+z);
        }
}
class m
{
        public static void main(String args[])
        {
                C c1=new C();
                c1.display();
                c1.mul();
```

```
        }
}
```

Accessing Interface Variables

1. Interfaces can be used to declare a set of constants that can be used in different classes
2. These constants values will be available to any class that implements the interface.

```
/* Implementing multiple inheritance using interfaces
interface sport
{
        int smark=70;
        void putswt();
}
class student
{
        int rollno;
        void getnum(int n)
        {
                rollno=n;
        }
        void putnum()
        {
                System.out.println("roll no"+rollno);
```

```
        }
}
class test extends student
{
        int part1,part2;
        void getmarks(int m1,int m2)
        {
                part1=m1;
                part2=m2;
        }
        void putmarks()
        {
                System.out.println("Part1 = " + part1);
                System.out.println("Part2 = " + part2);
        }
}

class Result extends test implements sport
{
        int tot;
        public void putswt()
        {
                System.out.println("Sports M= "+smark);
        }
```

```
        void display()
        {
                tot=part1+part2+smark;
                putnum();
                putmarks();
                putswt();
                System.out.println("TOTAL= "+tot);
        }
}
class m
{
        public static void main(String arg[])
        {
                Result s1=new Result();
                s1.getnum(123);
                s1.getmarks(60,65);
                s1.display();
        }
}
```