CHAPTER - 5

INTERFACES

5. Interfaces

Interface is a collection of method declaration and constants that one or more classes of objects will use. Interface definition is same as class except that it consists of the methods that are declared have no method body. They end with a semicolon after the parameter list. Syntax for an interface is as follows.

```
Syntax: <access specifier> interface <it-name>
        {
                type varname1=value;
                type varname2=value;
                .
                .
                returntype method-name1(parameter-list);
                returntype method-name2(parameter-list);
                .
                .
        }
```

Here,

- access specifier is always public only public access specifier indicates that the interface can be used by any class. Otherwise, the interface will accessible to class that are defined in the same package as in the interface.
- interface keyword is used to declare the class as an interface.
- it-name is the name of the interface and it is a valid identifier.
- Variables declared inside the interface are implicitly final and static. They cannot be changed in the implementing class.
- All the methods in an interface are implicitly abstract methods. Method implementation is given in later stages.

- The main difference between a class and interface is class contains methods with method body and the interface contains only abstract methods.

Example:

```
public interface shape
{
        int radius=2;
        public void area(int a);
}
```

5.1. Implementing Interfaces:

Once an interface has been defined, one or more classes can implement that interface. "implements" keyword used for implementing the classes. The syntax of implements is as follows.

Syntax: class class-name implements interface1, interface2, .. interfacen

```
{
        ........
        ........          // interface body
        ........
}
```

If a class implements more than one interface, the interfaces are separated with a comma operator. The methods that implement an interface must be declared public. Also type signature of the implementing method must match exactly the type signature specified in the interface definition.

Example:

```
interface it1
{
        int x=10,y=20;
        public void add(int a,int b);
        public void sub(int a,int b);
}
class it2 implements it1
{
        public void add(int s,int w)
        {
        System.out.println("Addition="+(s+w));
        }
        public void sub(int s,int w)
        {
        System.out.println("Subtraction="+(s-w));
        }
```

```
    public static void main(String[] args)
    {
    it2 obj=new it2();
    obj.add(3,4);
    obj.sub(5,2);
    System.out.println(obj.x+obj.y);
    obj.x=70;          // error since x is final variable in interface
    }
}
```

Note: 1. Interface methods are similar to the abstract classes so, that it cannot be instantiated.

2. Interface methods can also be accessed by the interface reference variable refer to the object of subclass. The method will be resolved at run time. This process is similar to the "super class reference to access a subclass object".

Example:

```
interface it1
{
    int x=10,y=20;
    public void add(int a,int b);
    public void sub(int a,int b);
}
class it2 implements it1
{
    public void add(int s,int w)
    {
    System.out.println("Addition="+(s+w));
    }
    public void sub(int s,int w)
    {
    System.out.println("Subtraction="+(s-w));
    }
    public static void main(String[] args)
    {
    it2 obj=new it2();
    it1 ref;
    ref=obj;
    System.out.println(ref.x+ref.y);
    }
```

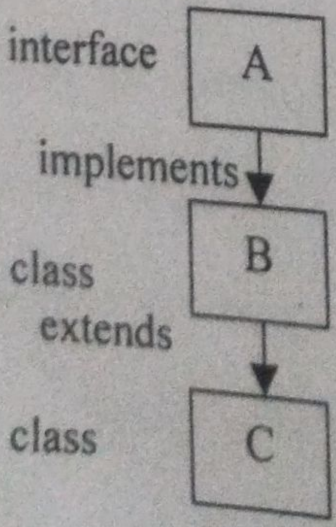
3. If a class includes an interface but does not fully implement the methods defined by that interface, then the class becomes abstract class and must be declared as abstract in the first line of its class definition.

Example:

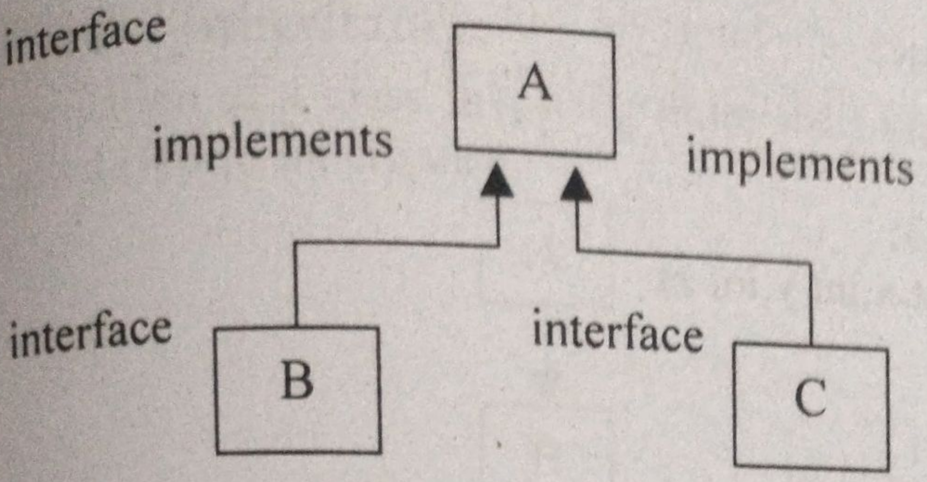
```
interface it1
{
        int x=10,y=20;
        public void add(int a,int b);
        public void sub(int a,int b);
}
abstract class it2 implements it1
{
        public void add(int s,int w)
        {
        System.out.println("Addition="+(s+w));
        }
}
class it3 extends it2
{
        public void sub(int s,int w)
        {
        System.out.println("Subtraction="+(s-w));
        }
        public static void main(String[] args)
        {
        it3 obj=new it3();
        obj.add(5,6);
        }
}
```

5.2 Various forms of Interface Implementations:

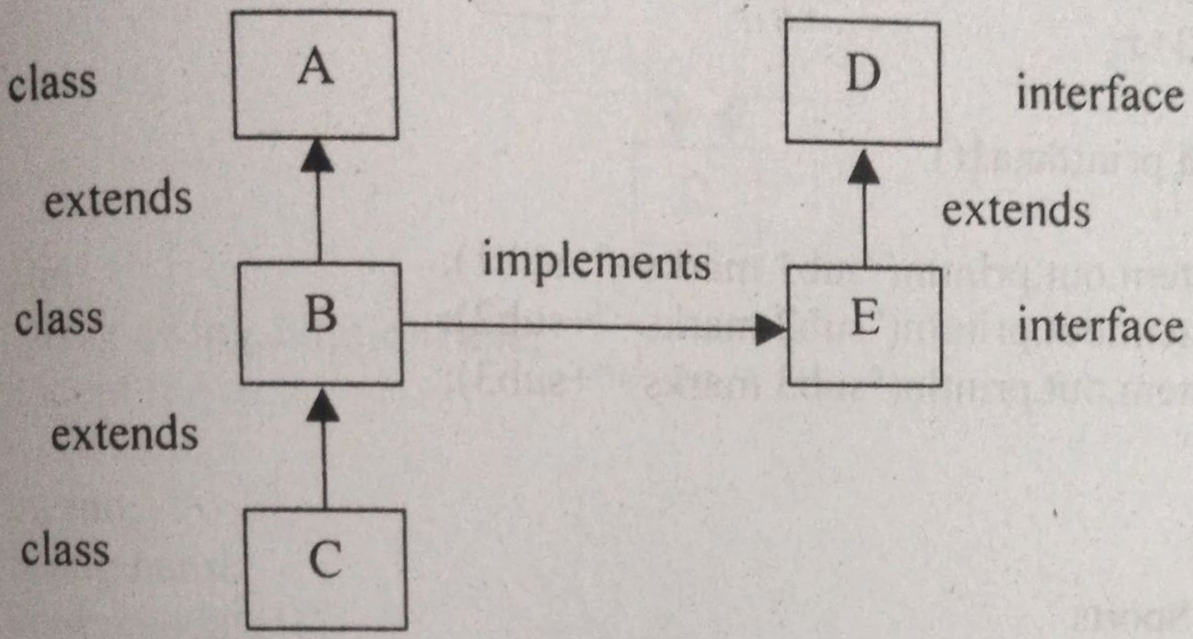
1.



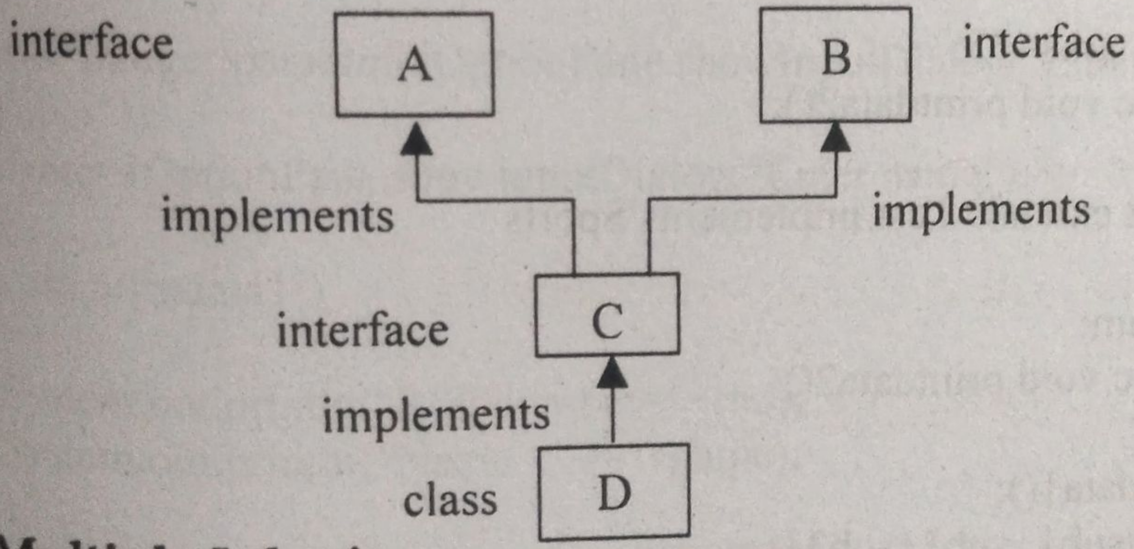
2.



3.

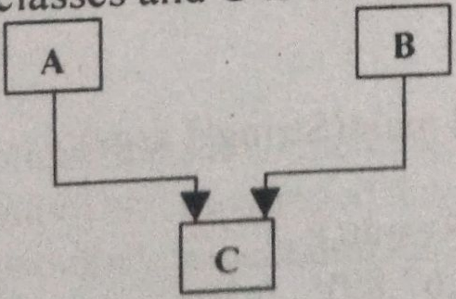


4.



5.3 Multiple Inheritance:

Multiple Inheritances does not support by Java directly. Multiple Inheritances enables to derive a class from multiple parent classes. Let A and B are parent classes and C is the derived class



Java provides Interface approach to support the concept of Multiple Inheritance. A class can also implements multiple interfaces.

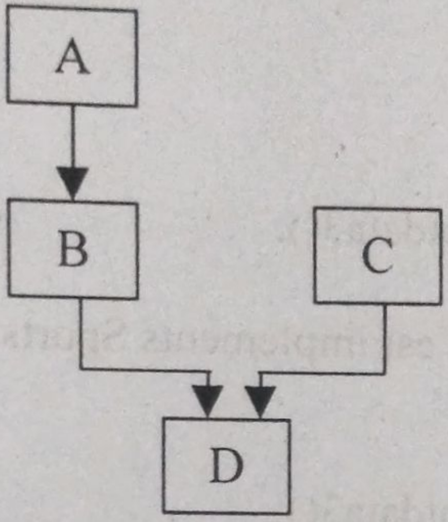
Example:

```
class Test
{
        int sub1,sub2,sub3;
        void readdata1(int x,int y,int z)
        {
        sub1=x;
        sub2=y;
        sub3=z;
        }
        void printdata1()
        {
        System.out.println("sub1 marks="+sub1);
        System.out.println("sub2 marks="+sub2);
        System.out.println("sub3 marks="+sub3);

        }
}
interface Sports
{
        int smarks=55;
        public void printdata2();
}
class Result extends Test implements Sports
{
        int sum;
        public void printdata2()
        {
        printdata1();
        sum=sub1+sub2+sub3+smarks;
        System.out.println("Total Marks="+sum);
        }
}
class mul
{
        public static void main(String[] args)
        {
        Result obj=new Result();
        obj.readdata1(35,67,89);
        obj.printdata2();
        }
}
```

5.4 Hybrid Inheritance:

Derivation of a class involving more than one form of Inheritance is called Hybrid Inheritance.



Example:

```
import javax.swing.JOptionPane;
class Student
{
        int rno;
        String name;
        void readdata1()
        {
        rno=Integer.parseInt(JOptionPane.showInputDialog("Enter
roll number"));
        name=JOptionPane.showInputDialog("Enter name");
        }
        void printdata1()
        {
        System.out.println("Roll number="+rno);
        System.out.println("Name       ="+name);
        }
}
class Test extends Student
{
        int m1,m2,m3;
        void readdata2()
        {
        m1=Integer.parseInt(JOptionPane.showInputDialog("Enter sub1 marks"));
        m2=Integer.parseInt(JOptionPane.showInputDialog("Enter sub2 marks"));
        m3=Integer.parseInt(JOptionPane.showInputDialog("Enter sub3 marks"));
        }
        void printdata2()
        {
```

```
        System.out.println("Sub1="+m1+"
         Sub2="+m2+"   Sub3="+m3);
        }
}
interface Sports
{
        int smarks=55;
        public void printdata3();
}
class Result extends Test implements Sports
{
        int sum;
        public void printdata3()
        {
        System.out.println("Sports marks="+smarks);
        }
        public void readdata3()
        {
        readdata1();
        readdata2();
        }.
        public void printdata4()
        {
        sum=m1+m2+m3+smarks;
        printdata1();
        printdata2();
        printdata3();
        System.out.println("Total Marks="+sum);
        }
}
class hy
{
        public static void main(String[] args)
        {
        Result obj=new Result();
        obj.readdata3();
        obj.printdata4();
        }
}
```