Experiment No. 3

Title: Implementing the concept of Interface.
Aim: To study 1. Defining interface 2. Implementing interface. 3. Implementing multiple interfaces. 4. Extending interface

Theory:

Introduction

Interfaces are used to encode similarities which classes of various types share, but do not necessarily constitute a class relationship.

An interface in Java is a blueprint of a **class**. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritances in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

For example:

```
interface Bounceable
{
        public abstract void setBounce();

/*Interface methods are by default public and abstract and themethods in an interface ends
with a semicolon not with curly brace.*/
}
```

Defining an interface

Interfaces are defined with the following syntax:

[*visibility*] interface ***InterfaceName*** [extends *other interfaces*] {

constant declarations

member type declarations

abstract method declarations}

The body of the interface contains abstract methods, but since all methods in an interface are, by definition, abstract, the abstract keyword is not required. Since the interface specifies a set of exposed behaviors, all methods are implicitly public.

Implementing an interface

The syntax for implementing an interface uses this formula:

... implements *InterfaceName*[, *another interface*, *another*, ...] ...

If a class implements an interface and is not abstract, and does not implement all its methods, it must be marked as abstract. If a class is abstract, one of its subclasses is expected to implement its unimplemented methods.

Classes can implement multiple interfaces

public class Frog **implements** Predator, Prey { ... }

Sample Program:

```
interface myinter
{
        void meth1();
        void meth2();
}
class myclass implements myinter
{
        public void meth1()
        {
                System.out.println("This is meth1");
        }

        public void meth2()
        {
                System.out.println("This is meth2");
        }
}
class DemoInter
{
        public static void main(String args[])
        {
                myclass  m  =new  myclass();
```

```
                m.meth1();
                m.meth2();

        }
    }
```

Program for extending Interface

```
interface myinter1
{
        void meth1();
}

interface myinter2 extends myinter1
{
        void meth2();
}

class myclass implements myinter2
{
        public void meth1()
        {
                System.out.println("This is meth1");
        }

        public void meth2()
        {
                System.out.println("This is meth2");
        }
  }
class DemoInterextends
 {
        public static void main(String args[])
        {
                myclass m1 =new myclass();
                m1.meth1();
                m1.meth2();
        }
  }
```

Statement: Create Vehicle Interface with name, maxPassanger, and maxSpeed variables. Create LandVehicle and SeaVehicle Inteface from Vehicle interface. LandVehicle has numWheels variable and drive method. SeaVehicle has displacement variable and launch method. Create Car class from LandVehicle, HoverCraft from LandVehicle and SeaVehicle interface. Also create Ship from SeaVehicle. Provide additional methods in HoverCraft as enterLand and enterSea. Similarly provide other methods for class Car and Ship. Demonstrate all classes in a application.

Program:

```
import java.util.*;

interface Vehicle
{

	int max_passenger=10;
	int max_filled=7;
}

interface Landvehicle extends Vehicle
{
	int num_wheel=4;
	public void drive();
}

interface Seavehicle extends Vehicle
{
	int displacement=20;
	public void launch();
}

abstract class Car implements Landvehicle
{
	abstract void display1();
}

class Howercraft implements Landvehicle, Seavehicle
{

	public void drive()
	{
		System.out.println("Number of wheel "+num_wheel+" Number of passenger "+max_passenger+" Number of filled "+max_filled+"\n");
	}

	public void launch()
```

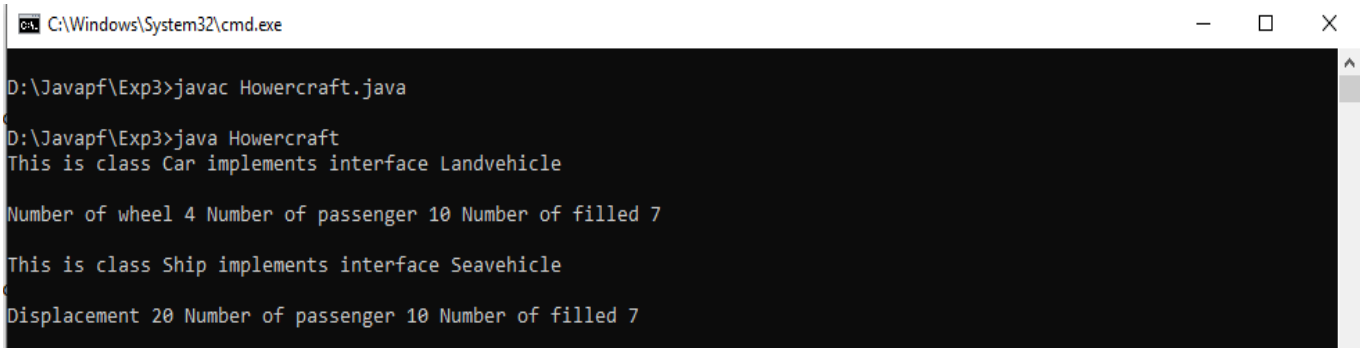
```
        {
                System.out.println("Displacement "+displacement+" Number of passenger "+max_passenger+" Number of filled "+max_filled);

        }
        public void display1()
        {
                System.out.println("This is class Car implements interface Landvehicle \n");
        }
        public void display2()
        {
                System.out.println("This is class Ship implements interface Seavehicle \n");
        }
        public static void main(String[] args)

        {
                Howercraft h=new Howercraft();
                h.display1();
                h.drive();
                h.display2();
                h.launch();
        }
}

abstract class Ship implements Seavehicle
{
        abstract  void display2();
}
```

Output:-



Conclusion: Thus we have studied how to create and implement interface and how to overcome problem of multiple inheritance using interface.