EXPERIMENT NO. 5
PART A

A.1 AIM: -

a) *WAP to display area of square and rectangle. Make use of interface to define templates of methods to be there in the derived classes*
b) *WAP to calculate volume of sphere using multilevel inheritance demonstrating method overriding. The base class method will accept the radius from the user. A class will be derived from the above mentioned class that will have a method to find and display the area of a circle and another class derived from this will have methods to calculate and display the volume of the sphere.*
c) *WAP to display volume of sphere and hemispehere. Make use of abstract class*

A.2 Prerequisite

C Programming

A.3 Outcome

After successful completion of this experiment students will be able to

1. Code a program using java constructs
2. Given an algorithm student will be able to formulate program that correctly implements the algorithm

A.4 Theory

a) An abstract class is a class that is declared `abstract`—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

```
abstract void moveTo(double deltaX, double deltaY);
```

If a class includes abstract methods, then the class itself must be declared `abstract`, as in:

```
public abstract class GraphicObject {
  // declare fields
  // declare nonabstract methods
  abstract void draw();
}
```

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared `abstract`.

Program

```
abstract class Shape{
abstract void draw();
}

class Rectangle extends Shape{
```

```
void draw(){System.out.println("drawing rectangle");}
}

class Circle extends Shape{
void draw(){System.out.println("drawing circle");}
}

class Test{
public static void main(String args[]){
Shape s=new Circle();
//In real scenario, Object is provided through factory method
s.draw();
}
}
```

Output: drawing circle

b) An interface declaration consists of modifiers, the keyword `interface`, the interface name, a comma-separated list of parent interfaces (if any), and the interface body. For example:

```
public interface GroupedInterface extends Interface1, Interface2, Interface3 {

    // constant declarations

    // base of natural logarithms
    double E = 2.718282;

    // method signatures
    void doSomething (int i, double x);
    int doSomethingElse(String s);
}
```

The `public` access specifier indicates that the interface can be used by any class in any package. If you do not specify that the interface is public, then your interface is accessible only to classes defined in the same package as the interface.

An interface can extend other interfaces, just as a class subclass or extend another class. However, whereas a class can extend only one other class, an interface can extend any number of interfaces. The interface declaration includes a comma-separated list of all the interfaces that it extends.

Program

```
interfaceIntExample{
public void sayHello();
 }
}
public class JavaInterfaceExample implements IntExample{
public void sayHello(){
System.out.println("Hello Visitor !");
```

```
  }
  public static void main(String args[]){
    //create object of the class
    JavaInterfaceExamplejavaInterfaceExample = new JavaInterfaceExample();
    //invoke sayHello(), declared in IntExample interface.
    javaInterfaceExample.sayHello();
  }}
```

output

Hello Visitor!

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll No.	Name:
Program:	Division:
Semester:	Batch :
Date of Experiment:	Date of Submission:
Grade :	

B.1 Software Code written by student:

A)

```
package exp_5.ques1;

interface Shape {
	public int area();
	public int perimeter();
}

----------X----------X----------

package exp_5.ques1;

class Rectangle implements Shape {
	private int len, bre;

	Rectangle(int a, int b) {
		this.len = a;
		this.bre = b;
```

```
	}

	public int area() {
		return this.len * this.bre;
	}

	public int perimeter() {
		return (2 * this.len) + (2 * this.bre);
	}
}
```

----------X----------X----------

```
package exp_5.ques1;

class Triangle implements Shape {
	private int side1, side2, side3;

	Triangle(int a, int b, int c) {
		this.side1 = a;
		this.side2 = b;
		this.side3 = c;
	}

	public int area() {
		int s = side1 + side2 + side3;
		return (int) (Math.sqrt(s * (s - side1) * (s - side2) * (s - side3)));
	}

	public int perimeter() {
		return this.side1 + this.side2 + this.side3;
	}
}
```

----------X----------X----------

```
package exp_5.ques1;

import java.util.*;

class Main {
	public static void main(String[] args) {
		Scanner in = new Scanner(System.in);
		Shape anyShape;
		anyShape = new Rectangle(in.nextInt(), in.nextInt());
```

```
		debug(anyShape.area());
		anyShape = new Triangle(in.nextInt(), in.nextInt(), in.nextInt());
		debug(anyShape.area());
	}

	static void debug(String msg) {
		System.out.println(msg);
	}

	static void debug(int msg) {
		System.out.println(msg);
	}
}
```

----------X----------X----------

----------X----------X----------

B)

```
package exp_5.ques2;

import java.util.*;

class Input {
	int radii(){
		Scanner in = new Scanner(System.in);
		return in.nextInt();
	}
}
```

----------X----------X----------

```
package exp_5.ques2;

class CalcCircle extends Input {
	int area() {
		int radius = radii();
		return (int) (3.14 * radius * radius);
	}

	void display() {
		System.out.println(area());
	}
}
```

----------X----------X----------

```
package exp_5.ques2;

class CalcSphere extends CalcCircle{
	int volume() {
		int radius = radii();
		return (int) (4 / 3 * 3.14 * radius * radius * radius);
	}

	void display() {
		System.out.println(volume());
	}
}
```

----------X----------X----------
----------X----------X----------

C)

```
package exp_5.ques3;

abstract class Volume {
	abstract int calcVol(int radius);

	abstract void display();
}
```

----------X----------X----------

```
package exp_5.ques3;

import java.util.*;

class Sphere extends Volume {
	int calcVol(int radius) {
		return (int) (4 / 3 * 3.14 * radius * radius * radius);
	}

	void display() {
		System.out.println("Enter the radius of Sphere: ");
		Scanner in = new Scanner(System.in);
		System.out.println(calcVol(in.nextInt()));
```

```
	}
}
```

----------X----------X----------

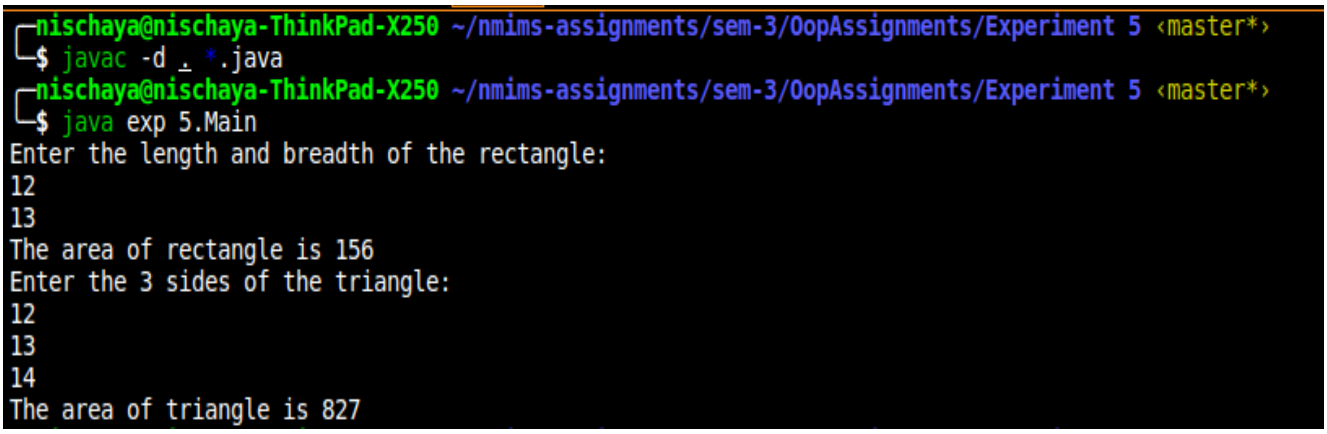
```
package exp_5.ques3;

import java.util.*;

class Hemisphere extends Volume {
	int calcVol(int radius) {
		return (int) ((4 / 3 * 3.14 * radius * radius * radius) / 2);
	}

	void display() {
		System.out.println("Enter the radius of Hemisphere: ");
		Scanner in = new Scanner(System.in);
		System.out.println(calcVol(in.nextInt()));
	}
}
```

----------X----------X----------

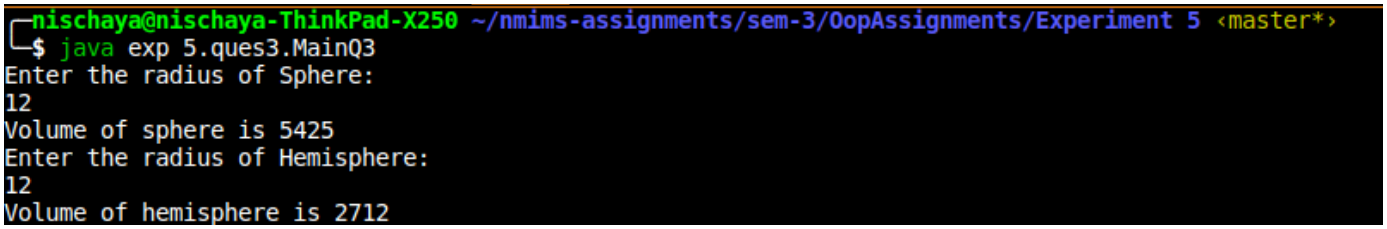
```
package exp_5.ques3;

class MainQ3 {
	public static void main(String[] args) {
		Volume vol;
		vol = new Sphere();
		vol.display();
		vol = new Hemisphere();
		vol.display();
	}
}
```

B.2 Input and Output:

A)



C)



B.3 Conclusion:

I got to learn about Interfaces and Abstract classes in Java.