Name of the Course : Java 9 Core principles

Level : Difficult

Tool Stack : Marker Interfaces, abstract class , abstract method, array of Object , **instanceof** operator, Encapsulation and access specifiers

Problem Statement :

Mr. Roshan who is software developer want to develope app Area-Volume Caculator for his math teacher to solve problem easily. They need to automate the process of finding Area and Volume of different shapes.

Description : Create an abstract public class Shape with the below public abstract methods :

public double area()

public double volume()

Create a public class Rectangle with private attributes :

double length

double width

Write the public getters and setters for these attributes.

Rectangle class should inherit the Shape class

Area of rectangle is length \* width

Create a public class Triangle with private attributes :

double base

double height

Write the public getters and setters for these attributes.

Triangle class should inherit the Shape class

Area of triangle is 1/2 \* base \* height

Create a public class Cube with private attributes :

double length

double width

double height

Write the public getters and setters for these attributes.

Cube class should inherit the Shape class

Area of cube is 2 \* length \* width + 2 \* length \* height + 2\* width \* height

Volume of cube is length \* width \* height

Create a public class Sphere with private attributes :

double radius

Write the public getters and setters for these attributes.

Sphere class should inherit the Shape class

Area of Sphere is 4 \* PI \* radius2

Volume of Sphere is ( 4 \* PI \* radius3 ) / 3

Note : You should use Math.PI for the value of PI

For Rectangle and Triangle class the method volume should return -1.

Create an interface Spatial which is a marker interface. Classes that has proper implementation for volume should implement this interface.

Create a public Main class which has the main method.

Create an array of Shape of size 5.

Get the Shape type and the corresponding attributes and store those objects in the array.

Print the area and volume of the objects created.

Volume of the array object should be printed only if it is of Spatial Type.

**Code:**

**package** main.java.yaksha;

**abstract** **public** **class** Shape {

**public** **abstract** **double** area() ;

**public** **abstract** **double** volume();

}

**package** main.java.yaksha;

**public** **interface** Spatial {

}

**package** main.java.yaksha;

**public** **class** Rectangle **extends** Shape{

**private** **double** length;

**private** **double** width;

**public** **double** getLength() {

**return** length;

}

**public** **void** setLength(**double** length) {

**this**.length = length;

}

**public** **double** getWidth() {

**return** width;

}

**public** **void** setWidth(**double** width) {

**this**.width = width;

}

@Override

**public** **double** area() {

**return** length\*width;

}

@Override

**public** **double** volume() {

**return** -1;

}

}

**package** main.java.yaksha;

**public** **class** Triangle **extends** Shape {

**private** **double** base;

**private** **double** height;

**public** **double** getBase() {

**return** base;

}

**public** **void** setBase(**double** base) {

**this**.base = base;

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

}

@Override

**public** **double** area() {

**return** 0.5 \* base \* height;

}

@Override

**public** **double** volume() {

**return** -1;

}

}

**package** main.java.yaksha;

**public** **class** Cube **extends** Shape **implements** Spatial {

**private** **double** length;

**private** **double** width;

**private** **double** height;

**public** **double** getLength() {

**return** length;

}

**public** **void** setLength(**double** length) {

**this**.length = length;

}

**public** **double** getWidth() {

**return** width;

}

**public** **void** setWidth(**double** width) {

**this**.width = width;

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

}

@Override

**public** **double** area() {

**return** 2 \* length \* width + 2 \* length \* height + 2\* width \* height;

}

@Override

**public** **double** volume() {

**return** length \* width \* height;

}

}

**package** main.java.yaksha;

**public** **class** Sphere **extends** Shape **implements** Spatial {

**private** **double** radius;

**public** **double** getRadius() {

**return** radius;

}

**public** **void** setRadius(**double** radius) {

**this**.radius = radius;

}

@Override

**public** **double** area() {

**return** 4\*Math.***PI***\*radius;

}

@Override

**public** **double** volume() {

**return** ( 4 \* Math.***PI*** \* radius ) / 3;

}

}

**package** main.java.yaksha;

**import** java.lang.reflect.Array;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Shape s[] = **new** Shape[5] ;

Scanner sc=**new** Scanner(System.***in***);

**for**(**int** i=0;i<5;i++) {

System.***out***.println("Enter the type of shape [Rectangle/Triangle/Cube/Sphere]");

String shape=sc.next();

**if**(shape.equals("Rectangle")) {

Rectangle r=**new** Rectangle();

**double** length=sc.nextDouble();

**double** width=sc.nextDouble();

r.setLength(length);

r.setWidth(width);

s[i]=r;

}**else** **if**(shape.equals("Triangle")) {

Triangle t=**new** Triangle();

**double** base=sc.nextDouble();

**double** height=sc.nextDouble();

t.setBase(base);

t.setHeight(height);

s[i]=t;

}**else** **if**(shape.equals("Cube")) {

Cube c=**new** Cube();

**double** length=sc.nextDouble();

**double** width=sc.nextDouble();

**double** height=sc.nextDouble();

c.setLength(length);

c.setWidth(width);

c.setHeight(height);

s[i]=c;

}**else** **if**(shape.equals("Sphere")) {

Sphere sp=**new** Sphere();

**double** radius=sc.nextDouble();

sp.setRadius(radius);

s[i]=sp;

}

}

**for**(**int** i=0;i<5;i++) {

**double** ar=s[i].area();

System.***out***.println("Area :"+ar);

**if**(s[i] **instanceof** Spatial) {

**double** vol=s[i].volume();

System.***out***.println("Volume :"+vol);

}

}

}

}

Junit Testing

**class** MainTest {

@Test

**void** testArea() **throws** IOException {

Shape s;

Rectangle s1=**new** Rectangle();

s1.setLength(20);

s1.setWidth(30);

*assertEquals*(600.0, s1.area());

Triangle s2=**new** Triangle();

s2.setBase(20);

s2.setHeight(30);

*assertEquals*(300.0, s2.area());

}

@Test

**void** testVolume() **throws** IOException {

Shape s;

Cube s1=**new** Cube();

s1.setLength(20);

s1.setWidth(30);

s1.setHeight(40);

*assertEquals*(24000.0, s1.volume());

}

}

Test Data1

Enter the type of shape [Rectangle/Triangle/Cube/Sphere]

Rectangle

20

30

Enter the type of shape [Rectangle/Triangle/Cube/Sphere]

Triangle

20

30

Enter the type of shape [Rectangle/Triangle/Cube/Sphere]

Cube

20

30

40

Enter the type of shape [Rectangle/Triangle/Cube/Sphere]

Sphere

20

Enter the type of shape [Rectangle/Triangle/Cube/Sphere]

Triangle

50

100

Area :600.0

Area :300.0

Area :5200.0

Volume :24000.0

Area :251.32741228718345

Volume :83.77580409572782

Area :2500.0

sLearning outcome: Participant could able to learn how to use Marker Interfaces, abstract class , abstract method, array of Object , **instanceof** operator, Encapsulation and access specifiers.