**Object-Oriented Programming**

Lab session #1

**Question 1**: Rectangle Visualization (25 points)

Write a class Rectangle which has

* Two attributes *width* and *height* with appropriate getter methods
* A constructor that requires values for width and height of the rectangle. If either of the inputs is negative, print an error message and set the corresponding attribute to 1.
* A *visualize* method to display the rectangle using \* symbol

Write a class TestRectangle with a *main* method to instantiate 5 different rectangles and visualize them.

**Rectangle.java**

public class Rectangle {

    public double width;

    public double height;

    public Rectangle (double width, double height) {

        this.width = width;

        this.height = height;

        if (width < 0 || height < 0) {

            System.out.print("Width and height must be positive\n");

            if (width < 0) {

                this.width = 1;

            }

            if (height < 0) {

                this.height = 1;

            }

        }

    }

    public void display() {

        for (int i=0; i<height; i++) {

            for (int j=0; j<width; j++) {

                System.out.print("\*");

            }

            System.out.println();

        }

    }

}

**TestRectangle.java**

public class TestRectangle {

    public static void main(String[] args) {

        Rectangle r1 = new Rectangle(5, 3);

        r1.display();

        Rectangle r2 = new Rectangle(2, 4);

        r2.display();

        Rectangle r3 = new Rectangle(4, 0);

        r3.display();

        Rectangle r4 = new Rectangle(-4, 5);

        r4.display();

        Rectangle r5 = new Rectangle(-5, -5);

        r5.display();

    }

}

**Output**

A screenshot of a computer

Description automatically generated with medium confidence

**Question 2**: Triangle Verification (25 points)

Write a Triangle class which has 3 attributes, the length of 3 sides. The class has appropriate constructor and get methods. This class has a method String verify() to check and return type of the Triangle. The types can be Not Triangle, Equilateral, Isosceles or Scalene. Create another class which has a main() method to receive the length of 3 sides, verify it and display the result.

**Triangle.java**

public class Triangle {

    public int size1;

    public int size2;

    public int size3;

    public Triangle (int size1, int size2,int size3) {

        this.size1 = size1;

        this.size2 = size2;

        this.size3 = size3;

    }

    public String verify() {

        if (size1+size2>size3 && size1+size3>size2 && size2+size3>size1) {

            if (size1==size2 && size2==size3) {

                return "Equilateral\n";

            }

            else if (size1==size2 || size2==size3 || size1==size3) {

                return "Isosceles\n";

            }

            else {

                return "Scalene\n";

            }

        }

        else {

            return "Not Triangle\n";

        }

    }

}

**TestTriangle.java**

public class TestTriangle {

    public static void main(String[] args) {

        Triangle t1 = new Triangle(3, 4, 5);

        System.out.print(t1.verify());

        Triangle t2 = new Triangle(3, 3, 3);

        System.out.print(t2.verify());

        Triangle t3 = new Triangle(3, 3, 4);

        System.out.print(t3.verify());

        Triangle t4 = new Triangle(3, 4, 6);

        System.out.print(t4.verify());

    }

}

**Output**

A screenshot of a computer

Description automatically generated with medium confidence

**Question 3**: Distance (25 points)

Write a Point class that has private attributes for coordinates x and y. The class has constructor to get values for x and y of the point. In the class WITHOUT having getter methods for x and y, write a method

*double distance(Point target)*

to compute the distance from the current point and the given target point.  
Note: the distance *d* between two points A and B can be computed with the following formula

Write a class with a main method to test the class Point and the distance method

**Point.java**

public class Point {

    private double x;

    private double y;

    public Point() {

    }

    public Point(double x, double y) {

        this.x = x;

        this.y = y;

    }

    public double distance(Point target)   {

        return Math.sqrt((this.x - target.x)\*(this.x -target.x) + (this.y - target.y)\*(this.y - target.y));

    }

}

**Test.java**

public class Test {

    public static void main(String[] args) {

        Point p1 = new Point(2, 2);

        Point p2 = new Point(4, 4);

        System.out.println(p2.distance(p1));

    }

}

**Output**

A screenshot of a computer

Description automatically generated with medium confidence

**Question 4**: E-commerce Order (25 points)

You are required to program Order and Item class satisfied the requirements as follows:

1. Each order has an ID

2. Each order has a list of Items

3. Each item has an ID, a name and a price

4. Each class has appropriate constructors, get and set methods

5. Class Order has a method double calculateAverageCost() to calculate the average of the cost

of all items in an order

6. Create a class which has a main() method to receive the inputs of items of an order from

keyboard and display the averageCost

**Item.java**

public class Item {

    public String ID;

    public String name;

    public double price;

    public Item(String ID, String name, double price) {

        this.ID = ID;

        this.name = name;

        this.price = price;

    }

    public String getID() {

        return ID;

    }

    public String getName() {

        return name;

    }

    public double getPrice() {

        return price;

    }

    public void setID(String ID) {

        this.ID = ID;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setPrice(double price) {

        this.price = price;

    }

}

**Order.java**

public class Order {

    public String ID;

    public Item[] items;

    public Order(String ID, Item[] items) {

        this.ID = ID;

        this.items = items;

    }

    public String getID() {

        return ID;

    }

    public Item[] getItems() {

        return items;

    }

    public void setID(String ID) {

        this.ID = ID;

    }

    public void setItems(Item[] items) {

        this.items = items;

    }

    public double calculateAverageCost() {

        double total = 0;

        for (int i=0;i<items.length;i++) {

            total += items[i].getPrice();

        }

        return total/items.length;

    }

}

**Test.java**

import java.util.Scanner;

public class Test {

    public int n=0;

    public Item[] items = new Item[n];

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        Test t = new Test();

        System.out.print("Enter number of items: ");

        t.n = input.nextInt();

        t.items = new Item[t.n];

        for (int i=0;i<t.n;i++) {

            System.out.print("Enter item ID: ");

            String ID = input.next();

            System.out.print("Enter item name: ");

            String name = input.next();

            System.out.print("Enter item price: ");

            double price = input.nextDouble();

            t.items[i] = new Item(ID, name, price);

        }

        Order o = new Order("1", t.items);

        System.out.println("Average cost: " + o.calculateAverageCost());

    }

}

**Output**

A screenshot of a computer

Description automatically generated with medium confidence