Assignment 1

INFO 2313

Due: 17th Mar 2024 (11.59 PM)

Max Marks – 40

Objectives:

- Use Objects, Classes, Inheritance, Abstract Classes, Interfaces and ArrayLists.
- Explain how programming instructions are executed when several classes are involved

Instructions:

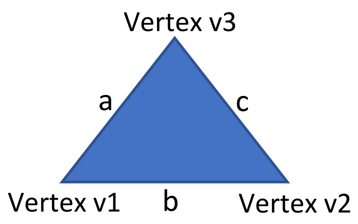
- Solve the programs as per instructions given in each question. Use the basic template with comments is being shown by the instructor in each problem definition.

Submission:

- Only one PDF submission is allowed. This submission should have the Codes pasted in text format along with the output screenshots.

No Extensions Allowed. Hard Deadline. 15-24 Hours Late – 10% deduction. >24 Hours Late, 0 Marks.

Problem 1: Write a program that finds out the area of a triangle using Heron's formula. The code template below uses the principles of inheritance to find out the area in a hierarchical manner. First read the mathematical formula for Heron's Computation of area of the triangle, then code it. **[15 Marks]**



Heron's Formula

$$S = \frac{A + B + C}{2}$$

$$\text{Area} = \sqrt{S(S - A)(S - B)(S - C)}$$

```
The triangle has 3 points v1,v2 and v3. Each point has 2 coordinates X,Y.
The Point class defines the coordinates (x,y) of a point. LineSegment defines
```

the length of the 2 points. Triangle area computation uses 3 LineSegments to find the area of the Triangle using Heron's Formula.

Formula for length of a line segment using 2 Points v1 and v2.

$$\sqrt{(x_{v1} - x_{v2})^2 + (y_{v1} - y_{v2})^2}$$

```
public class Point {

        private double x;
        private double y;


//code the default and constructor using Fields for Point class. [2 Marks]


//Code the setters and getters for the Point class. [2 Marks]


}
```

```
public class LineSegment extends Point {

        Point v1;
        Point v2;

        public double length(Point v1, Point v2){

/* Here, code the length between 2 points v1 & v2 using the formula given
above [3 Marks]*/

                return  ;

        }

}
```

```
public class Triangle {

        public static void main(String[] args) {
/*Initialize the points v1, v2 and v3 using coordinates (0,0), (3,0) and
(0,4) respectively. Use setters and getters. Do not change the constructors
defined below [2 Marks] */

                Point v1 = new Point(0,0);
                Point v2 = new ;
                Point v3 = new ;

//code the values of side lengths a, b and c of the triangle [3 Marks]

                double a =;
                double b =;
                double c =;

//code the s (half perimeter) of the triangle [1 Mark]

                double s =;

//code the area of the triangle using Heron's Formula [2 Marks]

                double area =;
                System.out.println(area);
//Area of the triangle is 6 with the defined points (0,0), (3,0) and (0,4).

        }

}
```

Problem 2: In this exercise you are going to create an Interface called Lockable. The lockable interface should be used by classes who wish to have objects that may enter or leave a locked state. We will say that if an object is in a locked state, then certain methods of the object cannot be executed. For example, if a class called Door has an open method, then if a door object enters the locked state, calling door.open() would not succeed. We would have to unlock the door before we could successfully call the open method.

Create the Lockable interface that includes the following methods: setKey, lock, unlock, and isLocked. The setKey, lock, and unlock methods take an integer parameter that represents the key. The setKey method establishes the key. The lock and unlock methods lock and unlock the object, but only if the key passed in is correct. The isLocked method returns a boolean that indicates whether or not the object is locked. A Lockable object represents an object whose regular methods can be frozen: if the object is locked, the methods cannot be invoked; if it is unlocked, they can be invoked. **[5 Marks]**

Below is a class called Door. A Door can be open or closed and has methods to open the door and close the door.

Make Door implement the Lockable interface and provide implementations for the abstract methods of Lockable. Initialize the door to be unlocked and have a key value of 0. Use the locked state to prevent the door from being opened. **[5 Marks]**

```
public class Door{

        private boolean isOpen;


        public Door() {
                isOpen = false;
        }


        public void open() {

                        isOpen = true;
        }
```

```
	public void close() {
		isOpen = false;
	}

	public boolean getState() {
		return isOpen;
	}

	public String toString() {
		if(isOpen)
			return "The door is open";
		else return "The door is closed";
	}
```

Here is the Account class, Make Account implement the Lockable interface. When an Account becomes locked, you cannot make deposits or withdrawals. **[5 Marks]**

```
public class Account{

	private static int baseNum = 100000000;

	private int number;

	private double balance;


	public Account() {

		this.number = baseNum++;

		this.balance = 0;

	}


	public Account(double balance) {

		this.number = baseNum++;

		this.balance = balance;

	}


	public void deposit(double amount) {

			if (amount > 0)

				balance += amount;

	}


	public void withdraw(double amount) {
```

```
                if (amount > 0 && amount <= balance)
                      balance -= amount;
        }

        public int getNumber() {
              return number;
        }

        public double getBalance() {
              return balance; }
```

Problem 3: Create an abstract class called GeometricFigure. Each figure includes a height, a width, a figure type, and an area. Include an abstract method to determine the area of the figure. **[2 Marks]**

Add an interface called SidedObject that contains a method called displaySides(); this method displays the number of sides the object possesses. **[2 Marks]**

Create two concrete subclasses called Square and Triangle that extends GeometricFigure abstract class and implements the SidedObject Interface. **[3 Marks]**

Create an FigureTest Class that demonstrates creating objects of both subclasses, and store the Objects in an array and display their output for different field variables. **[3 Marks]**