COIS 2240 Assignment 2
Question 1:
Problem Description:
In the code provided here, you will find the class GeometricObject and Triangle (as you did in your first assignment). (5 points)

a) Define a *Circle* class that extends *GeometricObject* and contains:
   1. Two double data fields named x and y that specify the centre of the circle with get methods.
   2. A data field radius with a get method.
   3. A no-arg constructor that creates a default circle with (0, 0) for (x, y) and 1 for radius.
   4. A constructor that creates a circle with the specified x, y, and radius.
   5. A method getArea() that returns the area of the circle.
   6. A method getPerimeter() that returns the perimeter of the circle.

b) Define an *EquilateralTriangle* class that extends *Triangle.* The class contains:
   1. One double data field named side to denote the side of the triangle. As we know, an *equilateral triangle* is a special type of triangle where the sides are of the same length.
   2. A constructor that creates an *EquilateralTriangle* with the specified *side*. Pass the value of the *side* to *Triangle's* constructor: *Triangle(double side1, double side2, double side3).* In other words, you have to call *super* and pass *side* for all three arguments: *side1, side2, and side3.*

c) Define a *Rectangle* class that extends *GeometricObject.* The class contains:
   1. Two double data fields named *side1* and *side2* to denote the sides of the rectangle.
   2. A no-arg constructor that creates a default rectangle with 1 for each side.
   3. A constructor that creates a rectangle with the specified *side1* and *side2*.
   4. A method getArea() that returns the area of the rectangle.
   5. A method getPerimeter() that returns the perimeter of the rectangle.

d) Define a *Square* class that extends *Rectangle.* The class contains:
   1. One double data field named *side* to denote the side of the square.
   2. A constructor that creates a *Square* with the specified *side*. Pass the value of the *side* to *Rectangle's* constructor: *Rectangle(double side1, double side2).* In other words, you have to call *super* and pass *side* for both arguments: *side1* and *side2.*
   3. A method getArea() that returns the area of the square.
   4. A method getPerimeter() that returns the perimeter of the square.

Coding:

```
public abstract class GeometricObject {
  private String color = "white";
  private boolean filled;
  private java.util.Date dateCreated;
  /** Construct a default geometric object */
  protected GeometricObject() {
    dateCreated = new java.util.Date();
  }
  /** Construct a geometric object with color and filled
  value */
  protected GeometricObject(String color, boolean filled) {
    dateCreated = new java.util.Date();
    this.color = color;
    this.filled = filled;
  }
  /** Return color */
  public String getColor() {
    return color;
  }
  /** Set a new color */
  public void setColor(String color) {
    this.color = color;
  }
  /** Return filled. Since filled is boolean,
   *  the get method is named isFilled */
  public boolean isFilled() {
    return filled;
  }
  /** Set a new filled */
  public void setFilled(boolean filled) {
    this.filled = filled;
  }
  /** Get dateCreated */
  public java.util.Date getDateCreated() {
    return dateCreated;
  }
  @Override
  public String toString() {
    return "created on " + dateCreated + "\ncolor: " +
  color +
      " and filled: " + filled;
  }


  /** Abstract method getArea */
```

```
  public abstract double getArea();

  /** Abstract method getPerimeter */
  public abstract double getPerimeter();
}

class Triangle extends GeometricObject {
    private double side1 = 1.0, side2 = 1.0, side3 = 1.0;

    /** Constructor */
    public Triangle() {
    }

    /** Constructor */
    public Triangle(double side1, double side2, double
  side3) {
      this.side1 = side1;
      this.side2 = side2;
      this.side3 = side3;
    }

    /** Override method findArea in GeometricObject */
    public double getArea() {
      double s = (side1 + side2 + side3) / 2;
      return Math.sqrt(s * (s - side1) * (s - side2) * (s -
  side3));
    }

    /** Override method findPerimeter in GeometricObject */
    public double getPerimeter() {
      return side1 + side2 + side3;
    }

    /** Override the toString method */
    public String toString() {
      // Implement it to return the three sides
      return "Triangle: side1 = " + side1 + " side2 = " +
  side2 +
        " side3 = " + side3;
    }
  }




class Test {
    public static void main(String[] args) {
```

```
GeometricObject gObjectArray [] = new GeometricObject [4];
//Complete your code here




    }

private static void printAreaAndPerimeter(GeometricObject
  gObject) {

//Complete your code here









    }
  }
```

Submit the following:

1. Implement the classes *Circle, EquilateralTriangle, Rectangle, and Square*.
2. Implement the method *printAreaAndPerimeter* in the *Test* class that prints the area and the perimeter of the passed *GeometricObject*.
3. In the Test class, create an array of *GeometricObject* of size 5. The first element should be assigned to a *Circle object: new Circle(5,5,5).* The second element should be assigned to an *EquilateralTriangle object: new EquilateralTriangle(5).* The third element should be assigned to a *Triangle object: new Triangle(5,5,5).* The fourth element should be assigned to a *Rectangle object: new Rectangle(5,5).* The fifth element should be assigned to a *Square object: new Square(5).*
4. Pass each element in the array to *printAreaAndPerimeter.*
5. Compile, Run, and take a screenshot of the output and submit to Blackboard (you must submit the program regardless whether it complete or incomplete, correct or incorrect)

For question#2 and #3, please use a modelling tool of your choice. You can use draw.io or Microsoft Visio to do the assignment.

Question#2: Draw a sequence diagram for the following scenario: a client wishes to open a new account at a bank branch. To do so, his instance of class *Client* must first be retrieved from the central bank server. For a new client, an instance of Client must be created. An instance of *BankAccount* is then created using the Client object. A deposit must then immediately follow to complete the account creation process. (5 points)

Question#3: Draw an activity diagram for the following scenario: (5 points)

In an online purchasing system, the buyer requests to buy an item. In parallel, the system looks up whether the item exists in the store and verifies if the buyer has an account with the system. If the buyer does not have an account, the system will ask for registration info from the buyer to open an account. If the buyer does not provide registration info, the system exits. If the item does not exist in the store, the system exits. If the item exists, the system will check if the item price is less than or equal to buyer's account balance. If the buyer has enough money in the account to purchase the item, the system completes the purchase order successfully. If the buyer does not have enough money, the system exits.