

Seneca College

Sep 18, 2019

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

JAC444**Demo & Final Code Due date****Sep 27, 2019**

Workshop 2

Notes:

- i. Each task should be presented during the lab, demo worth 70% of the workshop marks and code uploading worth the other 30%.
- ii. Make sure you have all security and check measures in place, like wrong data types etc., no need to implement Exception as we haven't covered yet. There are other ways to handle bad input data.
- iii. Given output structure is just for student to have a glimpse what the output can look, student are free to make the output better in any way.
- iv. The final should be submitted by the midnight to avoid late penalties which are 10% each day late.

Other inputs can be given during demo, so make sure you test your program properly.

Description:

The following workshop lets you practice basic java coding techniques, creating classes, methods, using arrays, Inheritance, Polymorphism.

Task 1: Design a class named **MyPoint** to represent a point with **x** and **y** coordinates. The class contains:

- The data fields **x** and **y** that represent the coordinates with getter methods.
- A no-arg constructor that creates a point **(0, 0)**.
- A constructor that constructs a point with specified coordinates.
- A method named **distance** that returns the distance from this point to a specified point of the **MyPoint** type.
- A method named **distance** that returns the distance from this point to another point with specified **x** and **y** coordinates.

Define another class named **Triangle2D** class that contains:

- Three points named **p1**, **p2**, and **p3** of the type **MyPoint** with getter and setter methods.
- A no-arg constructor that creates a default triangle with the points **(0, 0)**, **(1, 1)**, and **(2, 5)**.
- A constructor that creates a triangle with the specified points.
- A method **getArea()** that returns the area of the triangle.

- A method **getPerimeter()** that returns the perimeter of the triangle.
- A method **contains(MyPoint p)** that returns **true** if the specified point **p** is inside this triangle (see Figure A).
- A method **contains(Triangle2D t)** that returns **true** if the specified triangle is inside this triangle (see Figure B).
- A method **overlaps(Triangle2D t)** that returns **true** if the specified triangle overlaps with this triangle (see Figure C).

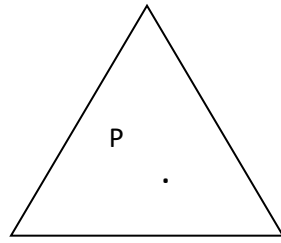


Figure (A)

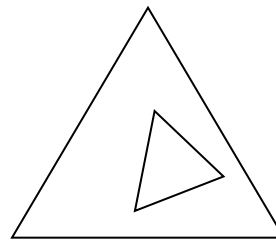


Figure (B)

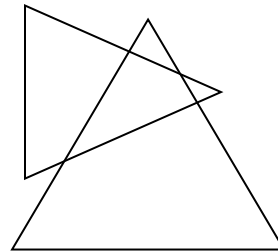
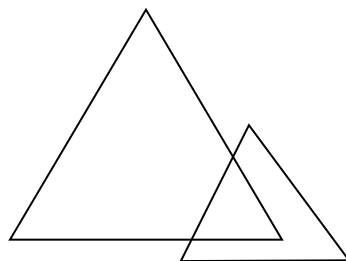


Figure (C)

Write a test program that creates a **Triangle2D** objects **t1** using the constructor **new Triangle2D(new MyPoint(2.5, 2), new MyPoint(4.2, 3), new MyPoint(5, 3.5))**, displays its area and perimeter, and displays the result of **t1.contains(3, 3)**, **r1.contains(new Triangle2D(new MyPoint(2.9, 2), new MyPoint(4, 1), MyPoint(1, 3.4)))**, and **t1.overlaps(new Triangle2D(new MyPoint(2, 5.5), new MyPoint(4, -3), MyPoint(2, 6.5)))**.

Create another **Triangle2D** object **t2** using the constructor **new Triangle2D(new MyPoint(0, 0), new MyPoint(0, 2), new MyPoint(2, 0))**, displays its area and perimeter, and display the results of **t2.contains(1, 1)**, **r2.contains(new Triangle2D(new MyPoint(4, 5), new MyPoint(10.5, 3.2), new MyPoint(-0.5, -10.5)))**, and **t2.overlaps(new Triangle2D(new MyPoint(1, 1.7), new MyPoints(-1, 1.7), new MyPoints(0, -3)))**.

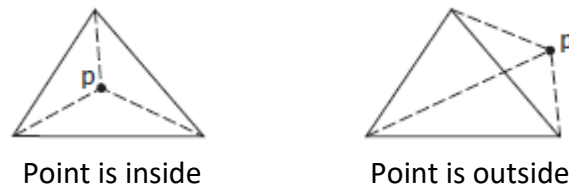
Note: Students can also design the program to take these points as input from the user as well using Scanner object.

Hint: Formula to compute the area of the Triangle:

$$s = (\text{side1} + \text{side2} + \text{side3})/2;$$

$$\text{area} = \sqrt{s(s - \text{side1})(s - \text{side2})(s - \text{side3})}$$

To detect whether a point is inside a triangle, draw three dashed lines, as shown in Figure D.



If the point is inside a triangle, each dashed line should intersect a side only once. If a dashed line intersects a side twice, then the point must be outside the triangle. For the algorithm of finding the intersecting point of two lines,

The intersecting point of two lines can be found by solving the following linear equation:

$$(y_1 - y_2)x - (x_1 - x_2)y = (y_1 - y_2)x_1 - (x_1 - x_2)y_1$$

$$(y_3 - y_4)x - (x_3 - x_4)y = (y_3 - y_4)x_3 - (x_3 - x_4)y_3$$

The linear equations can be solved using the Cramer's rule:

Cramer's rule for solving 2 x 2 system of linear equation

$$\begin{matrix} ax + by = e \\ cx + dy = f \end{matrix} \quad x = \frac{ed - bf}{ad - bc} \quad y = \frac{af - ec}{ad - bc}$$

If $ad - bc = 0$, the equations has no solution. (mean they are parallel lines).

You can also explore java.awt.geom.Line2D class for the calculation.

Marking Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Document all the classes properly
- Not have any debug/ useless code and/ or files in the assignment
- Tester class is must which will be having only main method to control the flow.

Deliverables and Important Notes:

- You are supposed to show up AND hand in your solution in person (run the solution and/or answer related Qs) in lab time.
- In case you don't show up OR hand in/run the required task in the lab, you could submit your final solution (described below) on the due date but note that there would be a 70% penalty! Late submissions would result in additional 10% penalties for each day or part of it.
- In this case, you should zip *only the Java files* to a file named after your Last Name followed by the first 3 digits of your student ID. For example, if your last name is **Savage** and your ID is **354874345** then the file should be named **Savage354.zip**. Finally upload your zip file to the blackboard
- Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the assignments, but the final solution may not be copied from any source.