Laboratory Assignment № 3

Dr. Duncan, CSC 1351, Louisiana State University 09/10/2015

Implementing a Subclass and Abstract Methods

Definition 1. ***Inheritance*** *is the process by which one class (type), the derived class, is defined by extending the behavior of an existing class (type), the base class. In Java the existing class is called the superclass and the new class is called a subclass. The Object class in Java is the cosmic class; that is, every class inherits the Object class. In today's lab session, you will define three classes: Polygon, a superclass that describes a polygon, Triangle, a subclass of the Polygon class, and TriangleDemo, a program to test your implementation of the superclass and subclass.*

The Polygon Class

Definition 2. *A* ***polygon*** *can be defined as a geometric object consisting of a number of points (called vertices) and an equal number of line segments (called sides), namely a cyclically ordered set of points in a plane, with no three successive points collinear, together with the line segments joining consecutive pairs of the points.*

The *Polygon* class will consist of an array of type *Point2D.Double*, named *vertices*, as its instance variable. This class will be an *abstract* class. View the online Java API documentation to find the package in which *Point2D.Double* is defined. When declaring the instance variable *vertices*, use the *protected* access specifier. This class will consist of the following methods:

1. *public Polygon()* : a default constructor that creates a degenerate polygon consisting of one point, (0,0).

2. *public Polygon(Point2D.Double[])* : a parameterized constructor that takes an array of points as an explicit parameter and creates a polygon.

3. *protected static double distance(Point2D.Double , Point2D.Double)* : an auxilliary method used to find the Euclidean distance between two points. Given points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, the Euclidean distance between them is given by the formula:

$$\overline{p_1 p_2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

4. *public abstract double area();* : an abstract method which when implemented computes the area of the polygon.

5. *public double perimeter()* : an instance method that computes the perimeter of the polygon.
6. *public Point2D.Double[] getVertices()* : returns an array of points that are vertices of the polygon.
7. *public void setVertices (Point2D.Double[])* : modifies the vertices of a polygon using the points in the array, the explicit parameter of the method.

The Triangle Class

Definition 3. *A triangle is a three-sided polygon. It consists of three vertices.*

Define a class *Triangle*, a subclass of the Polygon class. The Triangle class will consist of the following methods:

1. *public Triangle(Point2D.Double[])* : a parameterized constructor that takes, as an explicit parameter, an array of three points.
2. Implement the abstract *area()* method defined in the superclass, Polygon.

$$s = \frac{\overline{p_1p_2} + \overline{p_2p_3} + \overline{p_1p_3}}{2} \tag{2}$$

$$a = \sqrt{s\left(s - \overline{p_1p_2}\right)\left(s - \overline{p_2p_3}\right)\left(s - \overline{p_1p_3}\right)} \tag{3}$$

   where s, is half the perimeter of the triangle, $\overline{p_ip_j}$, is the Euclidean distance between points p_i and p_j, and a is the area of the triangle.
3. Override the *public String toString()* method so that it returns a string representation of the triangle in the form $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$, where (x_1, y_1), (x_2, y_2), and (x_3, y_3) are the vertices of the triangle.

The *TriangleDemo* Class

Define a class, *TriangleDemo*, consisting of only one method, the main method, that does the following:

1. Creates a triangle whose vertices are $(9, 9)$, $(11, 14)$ and $(17, 13)$.
2. Creates a second triangle whose vertices are $(5, 6)$, $(5, 46)$ and $(14, 6)$.
3. Prints the coordinates of the first triangle and its area and perimeter to the nearest ten thousandths.
4. Prints the coordinates of the second triangle and its area and perimeter to the nearest ten thousandths.

Since we have not studied exceptions and exception-handling, we will assume that points used to create each triangle are non-collinear.

Additional Requirements

Do not define any other class or implement any additional method other than those described in this handout. Compile and run your program to make sure that it works. Write header comments for each class using the following Javadoc documentation template:

```
/**
 * Explain the purpose of this class; what it does <br>
 * CSC 1351 Lab # 3
 * @author YOUR NAME
 * @since DATE THE CLASS WAS WRITTEN
 */
```

For the *Triangle* class, after the @since line, add the following Javadoc:

```
 * @see Polygon
```

For the *TriangleDemo* class, after the @since line, add the following Javadoc:

```
 * @see Triangle
```

Remove all Netbeans-generated Javadoc documentation. Add Javadoc documentation for each instance variable and method in both *Polygon* and *Triangle* classes. Run the Javadoc utility to make sure that it generates documentation for the *Polygon* and *Triangle* classes. Locate your source files, *Polygon.java*, *Triangle.java* and *TriangleDemo.java* and enclose them in a zip file, *YOURPAWSID_lab03.zip*, and submit your lab assignment for grading using the digital dropbox set up for this purpose. Here is a sample program interaction, where ... denotes a numeric value to the nearest ten thousandths:

Listing 1: Sample Run.

```
1 The first triangle: {(9.0,9.0),(11.0,14.0),(13.0,17.0)}
2 perimeter: ...
3 area: ...
4
5 The second triangle: {(5.0,6.0),(5.0,46.0),(14.0,6.0)}
6 perimeter: ...
7 area: ...
```