Homework 2 : Shapes

This assignment is designed to give you practice writing code and applying lessons and topics for the current module.

This homework deals with the following topics:

- Abstract classes
- Abstract methods
- Simple geometry

The Assignment

In this assignment, you will implement an **abstract** class called *Triangle*, which represents a triangle. (Remember, an abstract class is a class that can contain both abstract methods and concrete methods, and cannot be instantiated. This means you cannot create an instance of an abstract class, but you can, and usually do, extend an abstract class.)

You will also implement two classes that extend the *Triangle* class: *EquilateralTriangle* and *RightTriangle*. These classes represent different kinds of triangles. In this program, the *Triangle* class is the superclass (or parent class), and the *EquilateralTriangle* and *RightTriangle* classes are the subclasses (or extended classes, or child classes).

Triangle Class

Instance Variables

The *Triangle* class has the following instance variables defined for you:

- double sideA - The length of the first side of the triangle
- double sideB - The length of the second side of the triangle
- double sideC - The length of the third side of the triangle

Note, all of these variables are defined with the keyword ***protected***, which means they are accessible from anywhere within the *Triangle* class and from within the *EquilateralTriangle* and *RightTriangle* subclasses.

Constructor

The constructor in the *Triangle* class has been defined and **implemented for you**. You don't need to make any changes to the constructor in this class file.

The constructor has three parameters which represent the three side lengths of the triangle. The code in the constructor first checks to see whether the three given side lengths are valid by calling the static "hasValidSize" method, and then sets the values of the corresponding instance variables in the *Triangle* class. **You'll need to implement the "hasValidSize" method.** Note, a triangle is valid if the sum of any of its two sides is greater than the third side.

Below is the code for the constructor in the *Triangle* class:

```
/**
 * Creates a triangle with the 3 given side lengths.
 * Checks whether the 3 given sides are valid.  If not, throws an
 * IllegalArgumentException.
 * @param sideA first side of triangle
 * @param sideB second side of triangle
 * @param sideC third side of triangle
 */
public Triangle(double sideA, double sideB, double sideC) {
    if (!Triangle.hasValidSize(sideA, sideB, sideC)) {
        throw new IllegalArgumentException("Triangle sides not
        valid.");
    }

    this.sideA = sideA;
    this.sideB = sideB;
    this.sideC = sideC;
}
```

Methods

There are 2 methods defined in the *Triangle* class that **need to be implemented**:

- getPerimeter() - Concrete method that calculates and returns the perimeter of the triangle.

  To calculate the perimeter of a triangle use: sideA + sideB + sideC

- hasValidSize(double sideA, double sideB, double sideC) - Static method that checks the length of the given sides of a triangle to see if they are "valid". This method is called by the constructor in the *Triangle* class.

Each method has been defined for you in the *Triangle* class, but without the code. See the javadoc for each method for instructions on what the method is supposed to do. It should be clear enough. In some cases, we have provided hints and example method calls to help you get started.

For example, below is the code for the "getPerimeter" method in the *Triangle* class. For now, the method just returns 0.0. Read the javadoc, which explains what the method is supposed to do. Then write your code where it says "// TODO". You'll do this for the "hasValidSize" method as well.

```
/**
 * Calculates the perimeter of the triangle.
 * @return the perimeter of the triangle
 */
public double getPerimeter() {
    // TODO Implement method
    return 0.0;
}
```

There is 1 method defined in the *Triangle* class that **needs to be overridden in the *EquilateralTriangle* and *RightTriangle* classes**:

- getArea() - Abstract method that calculates and returns the area of the triangle. (Remember, since this method is abstract, it has to be overridden by the subclasses of *Triangle*.)

  To calculate the area of an equilateral triangle use: $(\sqrt{3} / 4) * sideA^2$
  To calculate the area of a right triangle use: (sideA * sideB) / 2
  You may find Java's `Math.sqrt()` helpful here.

Again, this method has been defined for you in the *Triangle* class (see below), but without the code. Read the javadoc, which explains what the method is supposed to do. Then write your code to **override and implement the method in the *EquilateralTriangle* and *RightTriangle* classes**.

```
/**
```

```
 * Abstract method which calculates the area of the triangle.
 * Should be overridden by subclasses EquilateralTriangle and
 * RightTriangle.
 *
 * @return the area of the triangle
 */
public abstract double getArea();
```

EquilateralTriangle Class

An equilateral triangle is a triangle in which all three sides are equal.

Constructor

The constructor in the *EquilateralTriangle* class has been defined but **needs to be implemented**. It has one parameter which represents one side length of the equilateral triangle. Read the javadoc, which explains what the constructor is supposed to do. Then write your code where it says "// TODO". We have provided hints and example constructor calls to help you get started.

The code in the constructor needs to calculate the lengths of the other two sides of the equilateral triangle and call the constructor in the superclass *Triangle*, with the correct arguments (the length values for all three sides of the triangle). As a reminder, all three sides of an equilateral triangle must be equal.

Below is the provided code for the constructor in the *EquilateralTriangle* class:

```
/**
 * Creates an equilateral triangle with 3 sides of the given side
 * length.
 * (All 3 sides of an equilateral triangle have the same length, so
we
 * only need one side length to create the triangle.)
 * Calls constructor in parent Triangle class by calling:
 * super(sideA, sideB, sideC)
 *
 * @param sideLength for all 3 sides
 */
public EquilateralTriangle(double sideLength) {
    // TODO Implement constructor
}
```

Methods

As noted, the abstract "getArea" method defined in the *Triangle* class **needs to be overridden in the *EquilateralTriangle* class**.

To calculate the area of an equilateral triangle use: $(\sqrt{3} / 4) * sideA^2$
Again, you may find Java's `Math.sqrt()` helpful here.

RightTriangle Class

A right triangle is a triangle in which one angle is a right angle.

Constructor

The constructor in the *RightTriangle* class has been defined but **needs to be implemented**. It has two parameters which represent the lengths of the two shorter sides of the right triangle. Read the javadoc, which explains what the constructor is supposed to do. Then write your code where it says "// TODO". We have provided hints and example constructor calls to help you get started.

The code in the constructor needs to calculate the length of the third side, which is called the *hypotenuse*. To do this, **implement the static "getHypotenuse" method** and calculate the *hypotenuse* using: $\sqrt{((sideA * sideA) + (sideB * sideB))}$
Then call the constructor in the superclass *Triangle*, with the correct arguments (the length values for all three sides of the triangle).

Below is the provided code for the constructor in the *RightTriangle* class:

```
/**
 * Creates a right triangle with given 2 sides and calculates the
 * hypotenuse (3rd and longest side).
 * Calls constructor in parent Triangle class by calling:
 * super(sideA, sideB, sideC)
 *
 * @param sideA is the first right-angled side
 * @param sideB is the second right-angled side
 */
public RightTriangle(double sideA, double sideB) {
    // TODO Implement constructor
}
```

Methods

As noted, the abstract "getArea" method defined in the *Triangle* class **needs to be overridden in the *RightTriangle* class**.

To calculate the area of a right triangle use: (sideA * sideB) / 2

You'll also need to implement the static "getHypotenuse" method which calculates the hypotenuse (longest side) for the right triangle, based on the two given shorter sides.
To calculate the hypotenuse of a right triangle use: √((sideA * sideA) + (sideB * sideB))
The "getHypotenuse" method is called by the constructor in the *RightTriangle* class.

Run Method & Main Method

The static "run" method has been defined and **implemented for you** (see example below). It's designed purely to call the other methods in the program, and to help you run the program. Nothing needs to be completed in the "run" method.

The static "main" method has also been defined and **implemented for you** (see example below). The "main" method is required as the entry point of the program, but it simply runs the program by calling the static "run" method in the *Triangle* class. Nothing needs to be completed in the "main" method.

```
/**
 * Runs and controls the program, creating different kinds of
 * triangles and printing useful information about them for
debugging.
 */
public static void run() {

    System.out.println("---------------------------------------");

    // Results should be:
    //  1.0, 1.0, 1.0
    //  0.4330 ...
    //  3.0
    EquilateralTriangle et = new EquilateralTriangle(1);

    System.out.println("Equilateral Triangle's sides are: " +
et.sideA + ", "+ et.sideB + ", " + et.sideC);
    System.out.println("Equilateral Triangle's area is: " +
et.getArea());
```

```
    System.out.println("Equilateral Triangle's perimeter is: " +
et.getPerimeter());

    System.out.println("---------------------------------------");
}

/**
 * Main method to run the program.
 * @param args
 */
public static void main(String args[]) {
    //call static run method in Triangle class
    Triangle.run();
}
```

Tips for this Assignment

In this assignment, some tips are given as follows:

- Geometry Review:
  - For a triangle to have a valid size the sum of two side lengths of the triangle must always be greater than the third side. This must be true for all three combinations of added side lengths.
  - The perimeter of a triangle is the sum of all the sides of the triangle.
  - Pythagorean Theorem: In a right triangle, $c^2 = a^2 + b^2$ or $c = \sqrt{a^2 + b^2}$. Where *c* is the hypotenuse and *a* and *b* are the other two sides of the triangle.
  - Right Triangle: a triangle that has one 90° angle. The side opposite the 90° angle is the hypotenuse.
  - Equilateral Triangle: a triangle in which all three sides are equal.
  - Area of a Triangle: to find the area of a triangle, multiply the base by the height and then divide by 2.
    - For a Right triangle with sides *a*, *b* and hypotenuse *c*: Area = (a * b) / 2
    - For an Equilateral triangle with sides *a*: Area = $(\frac{\sqrt{3}}{4}) * a^2$
- Java Math Class:
  - As mentioned above, you may find `Math.sqrt()` helpful for this assignment. For more information on the functionality available in the Java Math Class see: https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Submission

You have been provided with *Triangle.java*, *EquilateralTriangle.java*, and *RightTriangle.java*. To complete the assignment, implement the methods in all three class files. Do not modify the name of the methods in any of the files or the automated testing will not recognize it. Feel free to write unit tests to test any of your method implementations.

You will submit at least three files for this assignment: *Triangle.java*, *EquilateralTriangle.java*, *RightTriangle.java*, and any other optional unit testing files you create. Make sure your program and any optional unit testing files run without errors! Submit the completed program using the steps outlined in the assignment in Coursera.

Evaluation

Points:

- Triangle Class (5 pts)
  - hasValidSize(double sideA, double sideB, double sideC) - 2 pts
  - getPerimeter() - 3 pts
- EquilateralTriangle Class (5 pts)
  - EquilateralTriangle constructor - 2 pts
  - getArea() - 3 pts
- RightTriangle Class (7 pts)
  - RightTriangle constructor - 2 pts
  - getHypotenuse(double sideA, double sideB) - 2 pt
  - getArea() - 3 pts