Seneca College

January 31, 2018

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

JAC444	**Due dates: February 6 and February 13, 2018**

Workshop 1

Description:

The first assignment lets you practice basic concepts such as encapsulation and abstraction, inheritance, polymorphism, and exceptions.

In this assignment, you will be working with geometrical shapes such as: **Circle, Square, Rectangle, Parallelogram,** and **Triangle.** You will need to create geometrical shapes and calculate their perimeters. Use the following figure as a reference (you don't need to calculate shapes' area for this assignment):

Therefore, you must develop java classes for all the geometrical shapes mentioned above (put all your classes in a package named **shapes**). Each class must be able to calculate the shape's perimeter. For each class, you must provide implementations for **toString(),** at least one constructor, setters/getters, and the documentations for the entire class.

You should also provide an interface **Shape** as the root of the inheritance hierarchy, and design the appropriate "is-a" relationships among the above-mentioned classes.

A text file named *shapes.txt* has been given to you. The file contains the definitions of some geometrical shapes. Each line might define a geometrical shape with the following format: *name,x,y,z.* The *name* is the shape's name and *x,y,z* are the values needed to define the shape's dimensions. The empty line signals the end of the file.

If a line is not properly formatted or it does not contain the necessary number of values to correctly describe a shape, your program must ignore that line. If you cannot build the

geometrical shapes with the given values (ex. zero or negative values for dimensions, wrong values for three sides of a triangle, etc.), you should throw an exception. Therefore, there is a need to define some custom Exception classes for some of your classes. Your program must be capable of storing all the geometrical shapes read from the file in one data structure. Since arrays are the only data structure that we have covered so far, *you must use just one array to contain all the shapes in this assignment*. In future, while we cover collections in Java, you will figure out that other data structures could ease your programming tasks a lot!

Copy the *shapes.txt* file in the root folder of your Java project in Eclipse. We would cover exceptions in week 4 and File IO in week 5, but if you want to start ahead of time, you could use the following code snippet to read all the lines from the file. You could also consider using the **split()** method of class **String** to split each line to its composing items:

```java
try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
      while ((s = br.readLine()) != null) {
            String[] tokens = s.split(",");

            //your code

      }
} catch (IOException e) {
      System.out.println(e.getMessage());
}
```

Do the following tasks in different modules (methods) of the **Main** class of your project:

Task 1: Read the file *Shapes.txt,* create the shapes and store them in your data structure. Then print the number of shapes you created, and finally, print all the shapes and their calculated perimeters *polymorphically*. For the sample input file, sample output could be:

```
------->JAC 444 Assignment 1<-------
------->Task 1 ... <-------
Invalid radius!
Invalid side(s)!
Invalid side!
Invalid side(s)!

32 shapes were created:
Circle {r=1.0} perimeter = 6.28319

Circle {r=2.111} perimeter = 13.2638

Circle {r=1.1} perimeter = 6.91150

Triangle {s1=3.0, s2=4.0, s3=5.0} perimeter = 12.0000

Triangle {s1=3.9, s2=4.0, s3=5.9} perimeter = 13.8000

Square {s=3.0} perimeter = 12.0000

Parallelogram {w=4.0, h=9.0} perimeter = 26.0000

Rectangle {w=3.0, h=5.1} perimeter = 16.2000
```

```
Square {s=5.0} perimeter = 20.0000

Parallelogram {w=3.9, h=9.2} perimeter = 26.2000

Triangle {s1=4.9, s2=5.0, s3=8.9} perimeter = 18.8000

Rectangle {w=8.0, h=2.1} perimeter = 20.2000

Circle {r=3.8} perimeter = 23.8761

Parallelogram {w=3.1, h=9.8} perimeter = 25.8000

Triangle {s1=3.1, s2=4.1, s3=5.1} perimeter = 12.3000

Parallelogram {w=0.1, h=0.2} perimeter = 0.600000

Triangle {s1=4.0, s2=5.0, s3=6.0} perimeter = 15.0000

Rectangle {w=3.1, h=5.2} perimeter = 16.6000

Circle {r=10.0} perimeter = 62.8319

Square {s=0.1} perimeter = 0.400000

Triangle {s1=3.1, s2=4.0, s3=5.0} perimeter = 12.1000

Circle {r=2.0} perimeter = 12.5664

Parallelogram {w=3.9, h=9.3} perimeter = 26.4000

Parallelogram {w=1.1, h=1.2} perimeter = 4.60000

Rectangle {w=3.0, h=5.2} perimeter = 16.4000

Square {s=100.1} perimeter = 400.400

Square {s=100.2} perimeter = 400.800

Circle {r=10.1} perimeter = 63.4602

Triangle {s1=3.0, s2=4.0, s3=5.0} perimeter = 12.0000

Triangle {s1=3.9, s2=4.8, s3=5.7} perimeter = 14.4000

Parallelogram {w=1.2, h=2.1} perimeter = 6.60000

Square {s=1.0E-5} perimeter = 4.00000e-05
```

Task 2: Delete the triangle with the minimum perimeter (there could be more than one minimum) and the circle with the maximum perimeter (there could be more than one maximum) from the shapes. Print the all the remaining shapes and their perimeters *polymorphically*. For the sample input file, sample output could be:

```
-------->Task 2 ... <-------
Circle {r=1.0} perimeter = 6.28319
```

```
Circle {r=2.111} perimeter = 13.2638

Circle {r=1.1} perimeter = 6.91150

Triangle {s1=3.9, s2=4.0, s3=5.9} perimeter = 13.8000

Square {s=3.0} perimeter = 12.0000

Parallelogram {w=4.0, h=9.0} perimeter = 26.0000

Rectangle {w=3.0, h=5.1} perimeter = 16.2000

Square {s=5.0} perimeter = 20.0000

Parallelogram {w=3.9, h=9.2} perimeter = 26.2000

Triangle {s1=4.9, s2=5.0, s3=8.9} perimeter = 18.8000

Rectangle {w=8.0, h=2.1} perimeter = 20.2000

Circle {r=3.8} perimeter = 23.8761

Parallelogram {w=3.1, h=9.8} perimeter = 25.8000

Triangle {s1=3.1, s2=4.1, s3=5.1} perimeter = 12.3000

Parallelogram {w=0.1, h=0.2} perimeter = 0.600000

Triangle {s1=4.0, s2=5.0, s3=6.0} perimeter = 15.0000

Rectangle {w=3.1, h=5.2} perimeter = 16.6000

Circle {r=10.0} perimeter = 62.8319

Square {s=0.1} perimeter = 0.400000

Triangle {s1=3.1, s2=4.0, s3=5.0} perimeter = 12.1000

Circle {r=2.0} perimeter = 12.5664

Parallelogram {w=3.9, h=9.3} perimeter = 26.4000

Parallelogram {w=1.1, h=1.2} perimeter = 4.60000

Rectangle {w=3.0, h=5.2} perimeter = 16.4000

Square {s=100.1} perimeter = 400.400

Square {s=100.2} perimeter = 400.800

Triangle {s1=3.9, s2=4.8, s3=5.7} perimeter = 14.4000

Parallelogram {w=1.2, h=2.1} perimeter = 6.60000

Square {s=1.0E-5} perimeter = 4.00000e-05
```

<u>Task 3:</u> Calculate and print the total perimeter of all parallelograms and the total perimeter of all triangles. For the sample input file, sample output could be:

```
------->Task 3 ... <-------
Total perimeter of Parallelogram is: 116.19999999999999
Total perimeter of Triangle is: 86.4
```

Marking criteria:
1. Having appropriate indentation, having proper file structures and modularization, following Java naming conventions, documenting all the classes properly, not having debug/useless code and/or file(s) left in assignment submission, and good intra and/or inter class designs: **2.5 marks**.

2. Building, storing, calculating, and printing all the geometrical shapes, correctly (Task 1): **2.5 marks**.

3. Task 2: **2.5 marks**.

4. Task 3: **2.5 marks**.

Deliverables:
You are supposed to deliver your project (run the solution and/or answer related Qs) in the labs 3 and 4. **Task 1 should be delivered in lab 3 and the remaining tasks should be delivered in lab 4**.

Since we have not covered Exceptions in Java so far, you could ignore it for lab 3 and add your Exception classes (and related exception handling code) for lab 4.

In case you don't show up in the labs to deliver the required segments/tasks, you could submit your final solution (described below) but note that there would be a 50% penalty!

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 email your zip file to me at reza.khojasteh@senecacollege.ca

Reminding Some Important Notes:
•   Each assignment should be submitted before/on its due date. The deduction for late submission will be 10% each day or part of it. No assignment will not be accepted after week 12.

•   All the assignments should be done satisfactorily to pass the course. Moreover, to obtain a credit in this course, a student must achieve a weighted average of 50% or better on workshops.

•   Students 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.