Seneca College **May 26, 2017**

Applied Arts & Technology

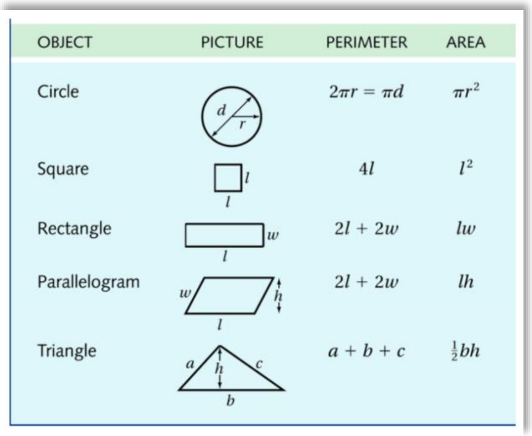
SCHOOL OF COMPUTER STUDIES

JAC444 **Due date: June 16**

Assignment 1

Description:
The first assignment lets you practice basic concepts such as encapsulation and abstraction, inheritance, polymorphism, exceptions, File IO, and lambda expressions.

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:



OBJECT	PICTURE	PERIMETER	AREA
Circle	d, r	$2\pi r = \pi d$	πr^2
Square	l, l	$4l$	l^2
Rectangle	w, l	$2l + 2w$	lw
Parallelogram	w, h, l	$2l + 2w$	lh
Triangle	a, h, c, b	$a + b + c$	$\frac{1}{2}bh$

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()`***, ***`equals()`***, ***`hashCode()`***, at least one constructor, setters/getters, and the documentations for the entire class, preferably in javadoc style.

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:

```
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 (ex.: `Circle{r=1.0} perimeter = 6.28319`) *polymorphically*.

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 shapes that you have deleted, then print the number of shapes that have remained, and finally, print all the shapes and their perimeters *polymorphically*.

Task 3: Calculate and print the total perimeter of all parallelograms and the total perimeter of all triangles.

Task 4: Your code would be tested to check whether it works correctly when we are checking equality between any two shapes and whether it produces the same hash codes for two equal shapes (hint: be careful in the cases of parallelograms, rectangles, and triangles!)

(Bonus) Task 5: Define a functional interface to calculate the area of a shape. Create an object of that interface in **`Circle, Square, Rectangle`** using lambda expressions (to be covered in week 6). Print all the shapes and their perimeters polymorphically and in cases that the shape being printed is an object of these classes (i.e. **`Circle, Square, Rectangle`**), print their areas (using the object you created) as well.

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 marks**.

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

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

4. Task 3: **2 marks**.

5. Task 4: **2 marks**.

6. Task 5: **2 marks**.

Deliverables:
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 called **`Savage354.zip`**. Upload the zip file to Blackboard.

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.

- 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. The assignments should be submitted online.