SYSC 2004
Fall 2018
Lab 2

Objectives

1. Practice writing classes
2. Learn how to include a JUnit test class within Netbeans
3. Experience test-driven development
4. First experience in writing arrays
5. Be prepared for Assignment 1

Provided Files: CircleTest.java, TriangleTest.java, RectangleTest.java
Submit Files: Circle.java, Triangle.java, Lab2c.java
Part 1

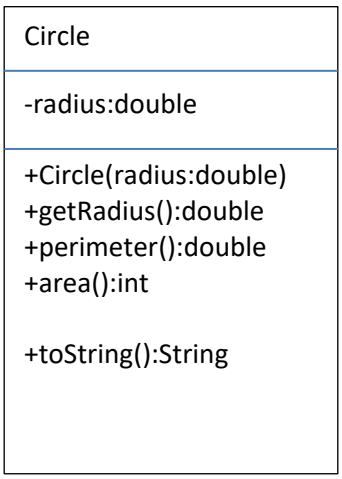
You need to simply first experience running a JUnit test using existing code. In a new project called Lab 2, you will re-use the same Rectangle class from Lab 1 and pair it with a newly posted `RectangleTest` class.

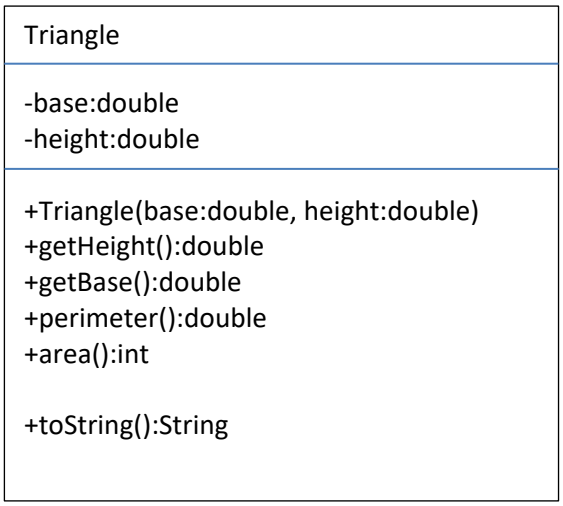
1. Under the CULearn section called **Lab Software**, please read the PowerPoint presentation on ***Setting up a JUnit Test in Netbeans***.
2. Open Netbeans and create a new project called **Lab2**.
    - Do NOT create a main class.
3. Forget about testing for a minute and add **`Rectangle`** to your new project, same as in Lab 1
    - **Create a New Java Class**
        - Make sure the name of the class is Rectangle
        - Make sure the name of the package is Shapes
    - Copy-paste the posted `Rectangle` code into the empty class
    - Make sure everything compiles.
4. Follow the guidance in the PowerPoint presentation to create a JUnit class called `RectangleTest`.
5. Copy-paste the posted code for **RectangleTest.java** into your empty `RectangleTest`.
6. Run `RectangleTest`. Hopefully you see the GREEN bar indicating that all tests pass successfully.

Part 2

You now need to experience test-driven development. Test-driven means that the test code is written before – or at least, the same time – as the actual source code. I have provided two JUnit test classes called `TriangleTest` and `CircleTest`.

1. Add these two test files to the same project, to the same place as `RectangleTest`
2. You will have compiler errors simply because you have not yet written the needed code, `Triangle` and `Circle`.
    - Pick one class, either is fine. I will pick `Triangle`, for example.
3. Using the UML below, write the public interface only for `Triangle`. Your goal is to write enough so that the test code compiles even though the test runs fail.
    - Systematically, translate each line of the UML diagram into Java code.
    - For each method, write the **public** call signature but leave the private implementation empty
        i. i.e. Simply write { }
4. Only after the code compiles, incrementally write the implementations of the `Triangle` methods
    - Write the code for one method. Study `Rectangle` to figure out how to implement these methods.
    - Run the JUnit test. One less test should fail.
    - Repeat, until one-by-one all the tests pass.
5. Repeat for the other class.





Tip: Finding the perimeter of a triangle is technically not possible if you only know the base and the height. For our solution, see: https://math.stackexchange.com/questions/80397/can-we-find-the-perimeter-of-a-triangle-given-only-its-base-and-height

Part 3

Your final task is a small exercise in array programming.

1. Within your Netbeans project, add a new file called **Lab2c.java**
2. This client class will contain only the main() method

```
public static void main(String args[]) {

}
```

3. Within the `main`() method, write code to accomplish the following task:
   - Construct an array of 10 `Circle` objects (called `circles`), each one individually initialised to a random radius.
   - Construct two arrays of 10 `doubles` (called `areas` and `perimeters`) , filling their values with the area and perimeter of the corresponding `circle`, in the `circles` array.
     - Note: It is possible to do the next task without these arrays, but by following these instructions you are gaining practice in arrays-of-primitives
   - For each `Circle` object, print out its radius, area, and perimeter
4. Run your program. Its output should resemble the following

```
run:
Circle 0(32.0) : Area = 3216.98816, Perimeter = 201.06176.
Circle 1(78.0) : Area = 19113.433559999998, Perimeter = 490.08804.
Circle 2(48.0) : Area = 7238.22336, Perimeter = 301.59263999999996.
Circle 3(12.0) : Area = 452.38896, Perimeter = 75.39815999999999.
Circle 4(63.0) : Area = 12468.97071, Perimeter = 395.84033999999997.
Circle 5(41.0) : Area = 5281.01279, Perimeter = 257.61037999999996.
Circle 6(29.0) : Area = 2642.07719, Perimeter = 182.21222.
Circle 7(84.0) : Area = 22167.05904, Perimeter = 527.78712.
Circle 8(41.0) : Area = 5281.01279, Perimeter = 257.61037999999996.
Circle 9(84.0) : Area = 22167.05904, Perimeter = 527.78712.
BUILD SUCCESSFUL (total time: 0 seconds)
```