COE318 – Lab 1: ComplexNumber objects

Objectives

- Implement a ComplexNumber class.
- Learn how immutable objects work.
- Create a project with more than one class.

Duration: one week.

Grading Scheme:

50% submitted source code

25% in-class demonstration and questions (during week 3 lab hours)

25% in-class quiz – Held during the first 5 mins of the lab class (during week 3 lab hours)

Overview

In mathematics, complex numbers combine two real numbers and can be thought of as specifying a point on a plane. The most common ways to express the two components of the complex number are:

- *rectangular*: where the two numbers represent the x- and y-components of the point;
- *polar*: where one number represents the distance between the origin and the point and the other represents the angle between the x-axis and the line connecting the origin and the point.

In this lab, we only consider the rectangular version.

The Design of ComplexNumber

The design of a class means specifying all of its public members. Usually, only some methods and constructors are public (and hence designed); it is highly unusual for instance variables to be public.

In Java, a design is expressed in javadocs: specially formatted comments in the source code that describe the API for the class.

You are given the design for the `ComplexNumber` class below. The design specifies methods such as `getX()`, `add(ComplexNumber z)`, etc. The source code can be more conveniently accessed here

The source code provided consists mainly of *method stubs*: methods that compile but produce dummy results. A notable exception is the method `toString()`; this method, which gives a String representation of a complex number, does work and should not be modified. (This method is implicitly invoked in the testing class that produces output.)

The main objective of the lab requires that you fix the method stubs so that the program works.

Source Code

The skeleton code is provided with the lab handout in the file named `ComplexNumber.java`. Copy the code and paste it into your own ComplexNumber class. You are also provided with another class, `ComplexTry.java`, which includes a main method that you can use to test your implementation of `ComplexNumber.java`.

Step 1: Create a Netbeans Project

1. Create a Netbeans project called **`Lab2ComplexNumber`**.
2. Create a Java file (class library type) called `ComplexNumber` specifying the package as `coe318.lab1` and copy and paste the provided source code.
3. Similarly, create the Java file `ComplexTry.` Ensure that Netbeans sets the package to `coe318.lab1`.
4. Generate the javadocs and compile and run the project.
5. It should compile correctly and produce output. Unfortunately, the output is incorrect and you have to fix it.

Step 2: Add instance variables and fix constructor and getters

1. Add instance variables for the two components of a complex number.
2. Modify the constructor so that they are properly initialized.
3. Fix the getReal() and getImaginary() methods so that they return the appropriate component.
4. Compile and run your project. The statement

```
System.out.println("a = " + a)
```

should now produce the correct output.

Step 3: Fix remaining methods

1. Fix the remaining methods.
2. Suggestion: fix them in the order they are used in `ComplexTry`.
3. Hint: `subtract()` and `divide()` are easier to write by using previously fixed methods.

Step 4: Submit your lab

You must submit your lab electronically on D2L. Please make sure you hand over the quiz answer sheet to the TA at the end of the in-class quiz.

Please zip up your NetBeans project containing all source files and submit to the respective assignment folder on D2L.