CSCI 2540 Assignment 2

(100 points)

Due date: Thursday, Feb. 3 (by 11:59pm)

For this assignment, you will be creating a class called **ComplexNum** that will be used to store and manipulate complex numbers. For each complex number in the format of a + bi, you will need two numeric values (using double type) to represent the real and imaginary parts of the number. Below are some examples of complex numbers:

2+3i
5i
6
1.5-2.5i

You will need to write two classes, the **ComplexNum** class and another class called **ComplexNumDemo** which will be used to test the methods in the ComplexNum class.

You will need to write the following methods for the **ComplexNum** class:

Constructors

All constructors should be defined as "public".

- A default constructor should be included. Remember that a default constructor is one that does not use any parameters. Initialize both real and imaginary part to 0 by default.
- A constructor should be included that receives a single double value as a parameter. The parameter will be used to initialize the real part while the imaginary part will be initialized to zero.
- A constructor should be included that receives two double values as parameters. The first parameter represents the real part and the second one represents the imaginary part.

Public Methods

- **getReal -** This is a "get" method used to retrieve the real part.
- **getImaginary -** This is a "get" method used to retrieve the imaginary part.
- **setReal** – This is a "set" method used to modify the real part.
- **setImaginary** – This is a "set" method used to modify the imaginary part.
- **add** - This method should receive a single ComplextNum object as a parameter. It should add the parameter to the current *(this)* object and return the answer as a new ComplexNum object. Neither the current object, nor the parameter should be changed by this method.

For example, suppose x is the complex number "2.2-3.3i", y is the complex number "4.1+1.5i", and z is a complex number. The command `z=x.add(y)` should return the complex number "6.3-1.8i" and store it in z.
In this example, x and y should not be changed by this method. Refer to the formulas listed below to find out how to add two complex numbers.

- **sub** - This method should receive a single complex number as a parameter. It should subtract the parameter from the current *(this)* object and return the answer as a new complex number object. Neither the current object, nor the parameter should be changed by this method.
- **mul** - This method should receive a single complex number as a parameter. It should multiply the current *(this)* object by the parameter and return the answer as a new complex number object. Neither the current object, nor the parameter should be changed by this method.
- **neg** - This method will take no parameters. It should negate the current *(this)* object and return the answer as a new complex number object. The current object should not be changed by this method.
  For example, suppose x is the complex number "2.2-3.5i" and z is a complex number. The command `z = x.neg()` should return the complex number "-2.2+3.5i" and store it in z.

Use the following formulas for the computations performed by the methods.

- Addition: `(a+bi)+(c+di) = (a+c)+(b+d)i`
- Subtraction: `(a+bi)-(c+di) = (a-c)+(b-d)i`
- Multiplication: `(a+bi)*(c+di) = (ac-bd)+(ad+bc)i`
- Negation: `-(a+bi) = -a-bi`

Overridden Methods

The following methods are inherited from the Object class, and you will need to overwrite them.

- **toString** - This method takes no parameters and returns a String object. The purpose of this method is to return a string equivalent of the ComplexNum object. The exact formatting returned should be in the format of a+bi, if a and b are not zero. If a or b is zero, then only the non-zero part should be returned, such as “2.5” or “3.5i”. Also, the positive or negative sign should be displayed properly, such as “2.5-3.6i”.
- **equals** - ~~This method receives an Object as a parameter and returns a boolean value. If the Object that is passed as a parameter is not a ComplexNum object, you should return false. Otherwise,~~ perform a comparison using the current object's real and imaginary part along with the ones passed in through the parameter. The method returns true if the two complex numbers are equal; otherwise it returns false. Two complex numbers are equal if they have the same real part and same imaginary part.

For the **ComplexNumDemo** class, you need to include a main method to test all the methods in the ComplexNum class. You need to create multiple ComplexNum objects to be able to test the methods. Make sure you test each of the constructors and test all other methods. For each method, you will need to test all possible cases, if applicable.

Do not ask user to enter anything from the keyboard. For output, you should print adequate messages to show what you are testing and what the result means.

Technical Notes

- ~~At the top of EVERY Java file you create, you must include a comment which includes your name.~~
- ~~YOU MUST PROVIDE COMMENTS for EVERY method that is included in your class. It does not matter how simple the method is, you must still comment it.~~ **~~Follow the standard "javadoc" commenting style for formatting your comments for each public method.~~**
- You do not need to finish writing the entire class before you can start testing it.
- Programs that do not compile will receive an automatic grade of "F".

Submission instructions:

You need to submit your programs electronically on Canvas.

Please **use a named package for each of your assignment.** For example, for assignment 2, create a new package and **name your package as assg2_yourPirateId** (use lower case for your pirate id), such as assg2_smithj19. You also need to include a statement such as "package assg2_smithj19;" at the beginning of each of your .java file. **Please follow this naming convention exactly for all future assignments. You will be deducted points for not doing so.**

When you submit your files to Canvas, please submit a zip file with your package folder inside the zip file. The package folder should include only .java files (make sure you include .java files, not .class files). The name of the folder should match with your package name. (You can use 7-zip software to zip files/folder).