CSC 115 A01
Fall 2013 L.Jackson

Assignment #1

Due

Before 8 pm on Monday September 23, 2013. You can only consider you assignment submitted when you have received a message from Connex system that indicates your assignment has been submitted. Save that message.

Description – Part 1

Learning Outcomes: Upon successful completion of Assignment #1 Part 1 you will be able to:

- Use a Java class to instantiate objects and manipulate its attributes using the class methods.
- Write Java programs that are able to input data from a text file.

Your Tasks:

1. Carefully review the **`Complex1.java`** class provided. In part 1 (this part) of Assignment 1 you are not expected to alter the coding in this file in any way. However, you need to add the following documentation to **`Complex1.java`**:
   - Immediately prior to each of the *constructor* methods add (appropriately) one of the following comments:

```
// Default Constructor

// Constructor for a new complex number, with:
//   - real component = r
//   - imaginary component = i
```

   - Immediately prior to each of the *accessor* methods add a comment with the format below, replacing the `XXX` with the appropriate attribute name:

```
// Accessor for the XXX attribute
```

   - Immediately prior to each of the *mutator* methods add a comment with the format below, replacing the `XXX` with the appropriate attribute name:

```
// Mutator for the XXX attribute
```

2. Using the **`Complex1.java`** class provided write a Java program, called **`ComplexExerciser.java`**, that instantiates complex numbers (ie, calls the constructor of the Complex class) and then outputs them, as follows:

- one with value `2 + 4i`,
- an other with value `4 + -5i`, and
- an array of complex numbers that contain the values that are found in the file `ComplexData.txt`. The file will contain a single integer on the first line indicating the number of lines that follow, then each of the following lines will contain two integers, the first one will be used for the real part of a complex number, the second one will be used for the imaginary part of the same complex number. (An example input file is provided.)
- After all of the numbers have been created, print them to the output screen, using calls to the `toString()` method of the `Complex1` class.
- Add documentation to the file includes a name, purpose, author, edit date, list of credits

The solution to the problem (stated above) must be contained in one Java file, called **`ComplexExerciser.java`**.

The **`Complex1.java`** class must be used, including using its methods, wherever possible but none of the attributes nor method code should altered in any way.

Submitting your Solution:

When complete (including suitable documentation*) submit your **`Complex1.java`** and **`ComplexExerciser.java`** files to the CSc 115 Connex Site using the Assignments: Assignment 1 link before 8 pm on Monday September 23, 2013.

Before you go on: In case you don't remember complex numbers from your previous math courses:
A description of the theory and some exercises in provided in the Resources link under
Assignments: Assignment 1.

Description - Part 2

Learning Outcomes: Upon successful completion of Assignment #1 Part 2 you will be able to:

- Discern the difference between *static* methods and *instance* methods
- Add instance methods to a Java class.
- Test the functionality of the created class by instantiating objects and manipulating attributes using instance methods.

Your Tasks:

First, copy the content of **`Complex1.java`** into a file called **`Complex.java`** and adjust the code so that the result compiles.

Adjust the **`toString()`** method such that it improves the appearance of complex numbers with imaginary parts that are `0` or negative, as follows:

Correct output: `3 - 2 i`	Incorrect output: `3 + -2 i`
Correct output: `7`	Incorrect output: `7  + 0 i`

Add and test 5 new methods for the `Complex` class (ie, include new instance methods in **`Complex.java`**), as follows:

- Create one more constructor that has only one integer parameter. It will be used for the real attribute; the corresponding imaginary attribute will be equal to 0.
- Test the functionality of this new method by adding a main (static) method to your `Complex` class. When a main is used in a class that defines an object, it is only used to test the various methods. For example, you might use the following lines to your new main method, to test constructor method:

```
Complex aReal = new Complex(12);
System.out.print("Should Output 12 + 0i : ");
System.out.println(aReal.toString());
```

  This main method will be referred to as the *tester*.
- Write method called `add` with the following method signature:

```
public Complex add(Complex val)
```

  The method you write need to be able to return a complex number whose value is `this + val`.
- Test the functionality of this new method by adding lines to `Complex.java`'s (tester) main method. For example, you might use the following lines to test your new method: `Complex aValue = new Complex(1,2);`

```
Complex anotherValue = new Complex(2,-1);
System.out.print("Should Output 3 + 1i : ");
System.out.println(aValue.add(anotherValue));
```

- Write a method called subtract with the following method signature:

```
public Complex subtract(Complex val)
```

  This method will return a complex number whose value is `this - val`.
- Test the functionality of this new method by adding lines to `Complex.java`'s tester.
- Write method called multiply with the following method signature:

```
public Complex multiply(Complex val)
```

  This method will return a complex number whose value is `this * val`.
- Test the functionality of this new method by adding lines to `Complex.java`'s tester.
- Write method called divide with the following method signature:

```
public Complex divide(Complex val)
```

  This method will return a complex number whose value is `this / val`.

- Test the functionality of this new method by adding lines to `Complex.java`'s tester.
- Extend your program from part 1 (in the file ComplexExerciser.java) such that it calculates the sum of all the numbers created using the data in the file ComplexData.txt then subtracts from that sum the number `2 + 4i` and multiplies the result by `4 – 5i`. Print the final result to the output screen.
- Add a comment just above your *tester* method that explains the difference between a *static* method (example: main) and the instance methods (example: add, subtract, multiply, divide, . . .) .

* Suitable documentation for this assignment

- ✓ Every file needs a name, purpose, author, edit date, list of credits
- ✓ Every method needs a description, input, output
- ✓ Throughout the code, every 3-6 lines, that form a functional group, require a descriptive comment.

Submitting your Solution:

- When complete (including suitable documentation) submit your files to the CSc 115 Connex Site using the Assignments: Assignment 1 link.

Due: before 8 pm on Monday September 23, 2013.

Observe: Your *complete* Assignment 1 submission requires 3 java files: `Complex1.java`, `ComplexExerciser.java` and `Complex.java`.