CSE 114 Fall 2023: Assignment 7

Due: Nov 6, 2023 at 11:59 PM [KST]

Read This Right Away

Directions

- At the top of every file you submit, include the following information in a comment
    - Your first and last name
    - Your Stony Brook email address
- Please read carefully and follow the directions exactly for each problem.
- Your files, Java classes, and Java methods must be named as stated in the directions (including capitalization). Work that does not meet the specifications will not be graded.
- Your source code is expected to compile and run without errors. Source code that does not compile will not be graded.
    - You can get partial credit if your program is incomplete but does compile and run.
- You should create your program using a text editor (e.g. emacs, vim).
- You must use the command-line interface to run your programs. That is, you must use the `javac` and `java` commands to do this assignment. Do not use IntelliJ IDEA yet.
- Your programs should be formatted in a way that is readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.

What Java Features to Use

For this assignment, you are *not* allowed to use more advanced features in Java than what we have studied in class so far.

What to Submit

Combine all your .java files from the problems below into a single zip file named as indicated in the **Submission Instructions** section at the end of this assignment. Upload this file to **Blackboard** as described in that section.

Multiple submissions are allowed before the due date. Only the last submission will be graded.

Please do **not** submit `.class` files or any that I did not ask for.

Naming Conventions In Java And Programming Style In General

Please use these conventions as you establish your *programming style.* Programming professionals use these, too.

- **Names:** Choose informative names,
    - e.g. `hourlyRate` rather than `hr` for a variable name.
    - `CheckingAccount` rather than `CA` for a Java class name.

- **Class names:** We start a class name with an uppercase letter as I have in my examples: `Hello` not `hello` or `HELLO`. For multi-word names you should capitalize the first letter of each word,
    - e.g. `UndergraduateStudent`, `GraduateStudent`, `CheckingAccount`
- **Variable names:** We start a variable name with a lowercase letter,
    - e.g. `count`, `hourlyRate`, averageMonthlyCost
- **Method names:** Naming convention for method names is the same as that for variable names.
- **Use of white space:** Adopt a good indentation style using white spaces and be consistent throughout to make your program more readable. Most text editors (like emacs and vim) should do the indentation automatically for you. If this is not the case, see me and I can help you configure your setup.

Problem 1 (10 points)

In this problem, you will build a simple class that handles complex numbers and arithmetic on complex numbers. Call the class **Complex**.

Complex numbers have the form

$$realPart + imaginaryPart * i$$

where i is $\sqrt{-1}$

Part 1

Your class must contain the following:

Integer members for the real and imaginary parts (you may name these as you please). They should be private. There should be getters and setters for each.
A default constructor that builds a Complex number with default values 1, 1.
A constructor that takes 2 integer arguments, the first being the real part and the latter being the imaginary part.
Dynamic methods to add, subtract, and multiply complex numbers.
A **toString()** method to convert the object to a human readable form for printing. The string should be of the form:
Real +/- Im i
So for a Complex number that has a real part of 5 and an imaginary part of 3, it should print:
5 + 3i
For a Complex number that has a real part of 7 and an imaginary part of -2, it should print:
7 – 2i
Note that it should not print '7 + -2i'! It should print it as shown in the line above.

The add, subtract, and multiply methods are dynamic methods and should update the object from which they are called. The prototypes are:

```
public void add(Complex c);
public void subtract(Complex c);
public void multiply(Complex c);
```

As an example, if we have the complex numbers

Complex c1 = new Complex(3, 5);   // This is : 3 + 5i
Complex c2 = new Complex(7, -2);   // This is 7 – 2i

Then calling:
c1.add(c2);

will change the value of c1 to *10 + 3i*

c1.subtract(c2);

will change the value of c1 to *-4 + 7i*

c1.multiply(c2);

will change the value of c1 to *31 + 29i*

The formulas for computing the sum, difference, and product (multiplication) of two complex numbers can be found here: https://mathworld.wolfram.com/ComplexNumber.html. See formulas (14), (15), and (16).

Part 2

Create a test class called **UseComplex**.
This should exercise the add, subtract, and multiply methods as well as the constructors. At a minimum, demonstrate that you methods can:

Add 3 + 5i and 7 – 2i
Subtract 7 – 2i from 3 + 5i
Multiply 3 + 5i and 7 – 2i

In each case show the result with a System.out.println() command.

Turn in **Complex.java** and **UseComplex.java.**

Problem 2 (15 points)

Create a class called **Rational**. This will perform arithmetic with fractions.
The class must have private integer members to hold the numerator and denominator of a fraction.

Part 1

Your class must contain the following:

Integer members for the numerator and denominator (you may name these as you please). They should be private. There should be getters and setters for each.

A constructor that takes 2 integer arguments, the first being the numerator and the latter being the denominator. The constructor should store the numerator and denominator **in reduced form**. For example, Rational(2, 4) is creating a Rational object for the fraction 2/4 which is equivalent to 1/2 in reduced form.

Dynamic methods to add, subtract, and multiply rational numbers.

A `toString()` method to convert the object to a human readable form for printing. The string should be of the form ***numerator / denominator***. So if a Rational object has a numerator of 3 and a denominator of 4, it should print as:

```
3 / 4
```

Provide methods to do the following:

1. Add a Rational to the current rational. Make sure that the resulting fraction is stored in reduced form. The method header is:

   ```
   public void add(Rational r);
   ```

2. Subtract a Rational from the current Rational. Make sure that the resulting fraction is stored in reduced form. The method header is:

   ```
   public void subtrace(Rational r);
   ```

3. Multiply a Rational by the current Rational. Make sure that the resulting fraction is stored in reduced form. The method header is:

   ```
   public void multiply(Rational r);
   ```

4. Divide a Rational into the current Rational. Make sure that the resulting fraction is stored in reduced form. The method header is:

   ```
   public void divide(Rational r);
   ```

As a reminder, division on fractions can be performed by multiplying by the reciprocal (denominator / numerator). As an example

4/5 / 3/10 is the same as:

4/5 * 10/3 = 40/15 = 8/3.

These methods will modify the object from which they are called. So if there are two rational numbers as follows:
Rational r1 = new Rational(3, 4); // This is: 3 / 4
Rational r2 = new Rational(2, 3); // This is: 2 / 3

And you call

```
r1.multiply(r2);
```

Then r1 is updated with the new value:

1 / 2.

Note: 2*3=6, 3*4 = 12, 6/12 reduces to 1 / 2

Part 2

Create a test class called UseRational.   This should exercise the add, subtract, multiply, and divide methods as well as the constructor. At a minimum, demonstrate that you methods can:

Add 3 / 4 and 2 / 3
Subtract 1 / 2 from 3 / 4
Multiply 2 / 3 and 4 / 5
Divide 1 / 5 into 3 / 5

In each case show the result with a System.out.println() command.

Turn in **Rational.java** and **UseRational.java.**

Problem 3 (10 points)

Create a class called **Circle2D**. This will hold a circle with a specific center given by X and Y coordinates and having a radius..

Part 1

Define the Circle2D class that contains:

Two fields of type double named x and y that specify the center of the circle.
A field of type double named radius
A no-arg constructor that creates a default circle with (0, 0) for (x, y) and 1 for radius.
A constructor that creates a circle with the specified x, y, and radius. (So the constructor takes 3 arguments)
Getter methods for x,y, and radius

Add methods to do the following:

1. A method that returns the area of the circle. The method header is:

   public double getArea();

2. A method that returns the perimeter of the circle. The method header is:

   public double getPerimeter();

3. A method that returns true if the specified point (x, y) is inside the circle (see the figure below). The method header is:

   public boolean contains(double x, double y);

4. A method that returns true if the specified circle is inside this circle (see the figure below). The method header is:

   public boolean contains (Circle2D circle);

5. A method that returns true if the specified circle overlaps with this circle (see the figure below). The method header is:

   public boolean overlaps(Circle2D circle);

For methods in 3-5 above, see the figures below in part 2.

Part 2

Write a test program named **UseCircle2D** that creates a Circle2D object c1 with its x, y coordinates of 2, 2 and the radius as 5.5.

Display the area and the perimeter of c1.

Display the result of:

```
c1.contains(3, 3)

c1.contains (new Circle2D(4, 5, 10.5))

c1.overlaps(new Circle2D(3, 5, 2.3))

c1.contains(new Circle2D(3, 5, 2.3))
```

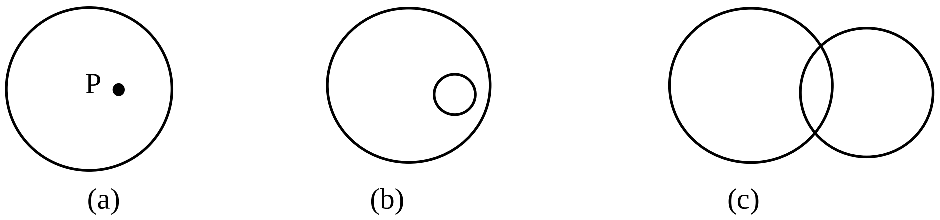

Figure: (a): A point inside the circle. (b): A circle inside another circle. (c): A circle overlaps another circle.

Turn in **Circle2D.java** and **UseCircle2D.java**.

Submission Instructions

Please follow this procedure for submission:

1. Place the deliverable source files into a folder by themselves (**Complex.java**, **UseComplex.java**, **Rational.java**, **UseRational.java**, **Circle2D.java**, **UseCircle2D.java**). The folder's name should be CSE114_HW7_<yourname>_<yourid>. So if your name is Alice Kim and your id is 12345678, the folder should be named 'CSE114_HW7_AliceKim_12345678'

2. Compress the folder and submit the zip file.
   a. On Windows, do this by clicking the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Blackboard.
   b. On Mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Blackboard.
3. Navigate to the course Blackboard site. Click **Assignments** in the left column menu. Click **Assignment7** in the content area. Under **ASSIGNMENT SUBMISSION**, click **Browse My Computer** next to **Attach Files.** Find the zip file and click it to upload it to the web site.