Assignment 1

Handout: **Friday, 29 September 2023**
Due: **23:59:59, Monday, 9 October 2023**

Goals:

- To practice the use of Java data types and control structures.
- To understand the importance of information hiding.
- To design and implement Java classes.
- To get used to the IntelliJ Idea IDE.

NOTES

1. JDK 17[1] and IntelliJ IDEA Community Edition Version 2023.2[2] will be used in grading your assignments. Make sure you use the same versions of tools for your development.

2. You MUST NOT change the names of the Java files or the signatures/return types of the methods in those files. We use tests to grade your assignments, and the tests refer to the classes and methods defined in the given Java files. Failed tests due to changes to the file names and/or method signatures/return types will be treated the same as failed tests due to incorrect implementations.

[1] https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

[2] https://www.jetbrains.com/idea/download/other.html

1. Base7 (15 points)

Given an integer value X, you need to return the base-7 representation of X as a string. In a base-7 representation, each digit is between 0 and 6, both inclusive, and the value of a digit is based on its position in the representation. For example, given a base-7 representation $r_1 = (d_n \ldots d_3 d_2 d_1 d_0)_7$ $(0 \leq d_i \leq 6, 0 \leq i \leq n)$, the decimal value of r_1 is calculated as $d_n*7^n + \ldots + d_3*7^3 + d_2*7^2 + d_1*7^1 + d_0*7^0$.

What to do: In Base7.java

[Task 1] Complete method convertToBase7 in class Base7 so that the method will take an int value as the argument and return the argument's String representation based on 7. For example, when the input value is 100, the method should return a string "202".

Notes:

- You may not add new methods to class Base7.
- You may right-click on class Base7Test and select "Run 'Base7Test'" to execute the tests we prepared for class Base7. If any test fails, your implementation is buggy. Note, however, that since the provided tests only check a small number of input/output pairs, passing all those tests does *not* mean your implementation is correct.

2. Rational Numbers (15 points)

In mathematics, a rational number is any number that can be expressed as the quotient or fraction p/q of two integers, a numerator p and a non-zero denominator q. Since q may be equal to 1, every integer is a rational number.

-- Wikipedia

Write a Java class for rational numbers. The class should have

1. two fields of type `int`, one for the numerator and the other for the denominator.
2. a constructor with two parameters, for the numerator and denominator, respectively.
3. four methods called `add`, `subtract`, `multiply`, and `divide`, respectively; Each method takes another rational number as the parameter, does the calculation using `this` and the parameter rational number, and returns the result rational.
4. a `simplify` method that simplifies `this` rational number. Each simplified rational number should satisfy the following three requirements:
    a. Common factors between the numerator and the denominator should be cancelled out; For example, `12/30` should become `2/5` after simplification.
    b. The denominator should always be positive, while the numerator could be positive, negative, or zero;
    c. The denominator should always be 1 when the rational number is an integer.
5. a `toString` method which returns the string representation of `this` in the form `numerator/denominator`.

Notes:

- You may assume the following when completing the class: 1) The constructor will never be used to instantiate a rational number with denominator equal to 0; 2) The parameters of `add`, `subtract`, `multiply`, and `divide` will never be null; 3) The parameter of `divide` will never be equal to 0.
- You may define additional methods when you see fit.
- Tests in `RationalTest.java` should all pass after you've completed the class.

What to do: In Rational.java

[Task 2] Add the missing fields to class `Rational`.

[Task 3] Complete the constructor and the methods `add`, `subtract`, `multiply`, `divide`, `simplify`, and `toString`.

3. Complex Numbers (15 points)

> A complex number is a number that can be expressed in the form a + bi, where a and b are real numbers and i is the imaginary unit, which satisfies the equation $i^2 = -1$. In this expression, a is the real part and b is the imaginary part of the complex number.
>
> -- Wikipedia

Write a Java class for complex numbers, but with both the real and the imaginary parts of type `Rational`. The class should have

1. two fields of type `Rational`, one for the real part and the other for the imaginary part.
2. a constructor with two parameters, one for the real part and the other for the imaginary part.
3. four methods called `add`, `subtract`, `multiply`, and `divide`, respectively; Each method takes another complex number as the parameter, does the calculation using `this` and the parameter, and returns the result complex.
4. a `simplify` method that simplifies the real and imaginary parts of `this`.

5. a toString method which returns the string representation of this in the form (real, imaginary).

Notes:

- You may assume the following when completing the class: 1) The parameters of the constructor are never null; 2) The parameters of add, subtract, multiply, and divide will never be null; 3) The parameter of divide will never be equal to 0+0i or 0-0i.
- You may define additional methods when you see fit.
- Tests in ComplexTest.java should all pass after you've completed the class.

What to do: In Complex.java

[Task 4] Add the missing fields to class Complex.

[Task 5] Complete the constructor as well as the methods add, subtract, multiply, divide, and asString.

4. XY-Rectangle (20 points)

A point P in a Cartesian coordinate system can be denoted using its x-coordinate and y-coordinate as (x_p,y_p), and an xy-rectangle, i.e., a rectangle with its four sides in parallel or overlapping with the x- and y-axes, can be represented using its top-left vertex P_{tl} and bottom-right vertex P_{br} as $<P_{tl},P_{br}>$. For example, xy-rectangle R_1 in Figure 1 (in solid line) can be represented as $<P_1,P_2>$, while xy-rectangle R_2 (in dotted line) can be represented as $<P_3,P_4>$. Given class Point as defined in XYRectangle.java, please complete class XYRectangle so that each XYRectangle object represents a valid xy-rectangle and supports the following operations:

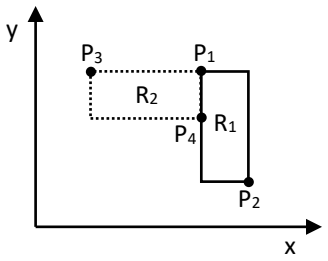

Figure 1. Two xy-rectangles.

1. A constructor that takes two points as arguments and initializes the newly created xy-rectangle in such a way that the line segment between the two points forms a diagonal of the rectangle; You may assume that the line passing both argument points always intersects with both the x and the y axis.
2. A method toString that returns a String representation of the xy-rectangle in the form $<P_{tl},P_{br}>$, where a point P with coordinates P_x and P_y is denoted as (P_x,P_y). For example, given an xy-rectangle with the top-left and bottom-right vertexes at positions (2,3) and (4,1), invoking toString on the xy-rectangle will return "<(2,3),(4,1)>".
3. A method area that takes no argument and returns the area of the xy-rectangle.
4. A method rotateClockwise that takes no argument and rotates the current xy-rectangle clockwise by 90 degrees around its top-left vertex; For example, xy-rectangle R_1 in Figure 1 will be at the position of R_2 after the method is invoked on R_1.
5. A method move that takes two arguments deltaX and deltaY and moves the xy-rectangle horizontally by deltaX and vertically by deltaY.
6. A method contains that takes a point P as the argument and returns true if and only if the point is within or on the border of the xy-rectangle.
7. A method contains that takes an xy-rectangle R as the argument and returns true if and only if every point contained in R is also contained in the current xy-rectangle.
8. A method overlapsWith that takes an xy-rectangle R as the argument and returns true if and only if at least one point is contained in both R and this xy-rectangle.

Note:

- You may assume that the reference-typed parameters of the constructor or methods are never `null`.
- You may define additional methods when you see fit.
- Tests in `XYRectangleTest.java` should all pass after you've completed the class.

What to do: In XYRectangle.java

[Task 6] Complete the constructor as well as the other methods in class `XYRectangle`.

What to hand in:

The whole **Assignment1** folder, including the completed methods, compressed into a ZIP file.

Note:

You may invoke methods defined in the String class in your code, but no methods from the other library classes should be invoked.

The detailed documentation of the String class is at

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html.