Assignment 2

Handout: **Monday, 27 September 2021**
Due: **23:59, Wednesday, 6 October 2021**

Goals:

- To understand the importance of information hiding;
- To design and implement Java classes;
- To get used to the IntelliJ Idea IDE.

Attention

1. You should make sure your code compiles. Compilation errors may lead to 0 points!

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, as is done in the released sample tests. Failed tests due to changes to the file names or method signatures/return types will be treated the same as failed tests due to incorrect implementations.

1. Rational Numbers (28 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.

Note:

- 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.
- Tests in RationalTest.java should all pass after you've finished Tasks 1 and 2.

What to do: In Rational.java

[Task 1] Add the missing fields to class Rational;

[Task 2] Complete the constructor as well as the methods add, subtract, multiply, divide, simplify, and toString;

2. Complex Numbers (28 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).

Note:

- 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;
- Tests in ComplexTest.java should all pass after you've finished Tasks 3 and 4.

What to do: In Complex.java

[Task 3] Add the missing fields to class Complex;

[Task 4] Complete the constructor as well as the methods add, subtract, multiply, divide, and asString;

3. CharIntMap (44 points)

In this task you are going to implement a class called `CharIntMap` which abstracts simple mappings from an English letter (i.e., one of the 26 letters in the English alphabet) to an integer. A map is a collection of distinct key-value pairs, e.g., key 'a' is paired with 1 and key 'b' is paired with value 2. The key is case-sensitive. Every key of a map must be unique in that map, but different keys may be paired with the same value. The order in which the keys are included in the `CharIntMap` is irrelevant.

Two fields have been given in class `CharIntMap` and you must NOT add other fields. A class `KeyValueEntry` has been given which needs no further modifications. The `CharIntMap` class should have:

1. A `constructor` with one parameter to specify the maximum number of unique keys the map may have. The maximum number may not be larger than 52 for the purpose of this assignment.
2. Ten methods:

   Method `isFull` takes no parameter and returns true if and only if the map contains already the maximum number of keys;

   Method `getValue` takes a key of type char as parameter and returns the value if the key is contained in the map. Note: It is ***not necessary*** to check whether the key is contained in the map or not for the purpose of this assignment because only valid keys are tested for this method.

   Method `containsKey` takes a key of type char as parameter and returns true if and only if this map contains a mapping for the specified key;

   Method `containsValue` takes a value of type int as parameter and returns true if and only if this map maps one or more keys to the specified value;

   Method `put` takes a key of type char and a value of type int as parameters and associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced by the specified value;

   Method `remove` takes a key of type char as parameter and deletes the key-value pair from this map if it is present;

   Method `replace` takes a key of type char and a value of type int as parameters and replaces the entry for the specified key if and only if the key is in the map;

   Method `merge` takes a key of type char and a value of type int as parameters. If the specified key is not in this map, this method associates the key with the given value. Otherwise, replaces the associated value with the addition of the associated value and the given value.

   Method `keyString` takes no parameter and returns a String which is concatenation of the keys in this map (keys may appear in arbitrary order in the String).

   Method `hasSameKeys` takes another `CharIntMap` as the only parameter and returns true if and only if the parameter `CharIntMap` contains the same keys as `this`.

Note:

- You may assume the following when completing the class: 1) The parameter of the constructor is always positive; 2) Method `put` and `merge` are not allowed to add a new key-value pair to a `CharIntMap` when the map is already full; 3) All the given parameters in the test phase are valid, which means that you do not need to check the validness of the parameters for all the methods; 4) you may define additional methods.
- Tests in `CharIntMapTest.java` should all pass after you've finished Task 5.

What to do:

[Task 5] Complete the constructor as well as the nine methods in `CharIntMap.java`.

What to hand in

The whole Assignment2 folder with the completed methods in a ZIP file. *Do NOT hand in just the Java files!!*