Lab Practice Week 3

You need to show working versions of your answers to all questions to your tutor. Your tutor will expect to see them by your next session.

What to submit: your answers to exercises 1, 2, 4, 5, and 6.

Note: even though you only need to submit those exercises mentioned above, you should attempt all exercises in each lab practice to help broaden your understanding; this may also help with your assignment work.

Do all the programs in NetBeans IDE.

NOTE: Include internal documentation in your code, if you are not sure, read chapter 2.4 in your textbook and talk to your lecturer

---

Before starting the exercises, make sure you have read the relevant material.

1. Write a class called *Fraction.java*; this class will be designed to handle fractions. The Fraction class has only **two** instance variables:

   - An integer numerator
   - An integer denominator

   The two instance variables can be publicly accessible (at this stage). Supply an input method for the Fraction class; this method will be used by the client to get a fraction (as two separate integer inputs) from the user. The input method should ensure that the denominator cannot be 0. Supply an output method for the Fraction class; this method will be used by the client to print out a fraction (as two separate integers) to the screen.

   An output fraction would be in the form:

   numerator / denominator

   An example: the fraction for a half would be input as 1 for the first input and 2 for the second input. The output should be printed out as 1 / 2.

   Once this is done, write a client class program called *TestFraction.java*; this class is to be used to test your Fraction class. The program should loop around getting fractions from the user and displaying them to the screen. Stop when the numerator is negative.

   Note that the client program should use the Fraction class methods (via the dot notation) to read and display fractions. For example, *pseudo-code* to create a Fraction object and then use the input and output methods:

```
class TestFraction
public static void main(String[] args)
        // create new fraction
        Fraction frac = new Fraction()
        // use dot notation to access methods
        frac.input()
        frac.display()
```

2. This exercise requires you to ensure that the input values representing fractions are stored with denominators that are positive integers. You **cannot** require the user to only enter a positive denominator value; the user should not be inconvenienced by such a restriction. For example, whilst values of 1 / -2 are acceptable inputs for a fraction, the output representation should be -1 / 2. Your solution should check the denominator input; if it is negative, swap the sign of **both** numerator and denominator instance variables. Test this functionality with your *TestFraction.java* class.

3. Now change your Fraction class so that its instance variables (numerator and denominator) are hidden (i.e. private). Check that your client class (after this change) will now **not** compile. That is, if the client class previously accessed the Fraction class instance variable numerator to test the Fraction class in question 2, why does it now not compile?

4. To rectify the problem identified in Q3, you will add a public method called *isZero*() to your Fraction class. This method will determine if a Fraction represents a zero fraction. A zero fraction has a numerator == 0, no matter what the denominator is. Your *isZero*() method should return a Boolean result indicating a zero fraction or otherwise. The method will be used by the client class (*TestFraction.java*) to test whether the 'calling fraction' is equal to the number zero. Modify your old client class so that it now loops until a fraction representing zero is entered.

5. It is a very common task with O-O to compare two objects of the same class, to see if they are equivalent. So, now you will add another public method called *isEqual()*. This method will determine if two fractions represent the same number (i.e. fraction), and return a Boolean result to indicate this. The method will take a Fraction class object as a parameter to compare with the 'calling object'. As we are not dealing with mathematical fractions, we need to find a way to equate fractions represented as two integers. Modify your client class (*TestFraction.java*) to test the Fraction class functionality that has been just added.

6. Provide the UML class diagram for your Fraction class after completing question 4 (refer lecture notes topic 2).