Manipulating data using objects

Objective:

This lab shows how Java can uses objects to store and manipulate data instead of taking the data one by one or in a group (like arrays).

Introduction:

As you have learned from other programming courses, we use arrays to unite a group of data. The main reason behind grouping a number of data in one unit is to make searching and data analysis easier. However, arrays alone don't provide any more advantage than grouping data together. Therefore, using Object Oriented Programming (OOP), we can add more features for every data grouping.

In OOP, to create a new data unit design you should create a new data type. In other words, gathering a number of data -each with specific type- is what we call creating a new data type or a new class. Writing a class does not mean that your data is ready to use!! you have to create a class instance (object) that has the shape of the class you wrote.

This lab uses fractions addition to explain how we can use objects to achieve data manipulation. It starts by creating a "fraction_addition" method that takes the data separately and ends by using the objects to pass data to the method.

Part#1 Fraction addition:

Create a class called Fraction1 that has two methods. The first method is the "fraction_addition" that takes 4 integers and return one integer. The 4 inputs are for the numerator and denominator of the two fractions you want the method to add. Write the equation for adding two fractions and return the result in an integer form. Use the main method to show the results on the console. You should get wrong results!! Write down why you got wrong results. Finally, fix the mistake and explain how you did it.

Part#2 Changing the output shape:

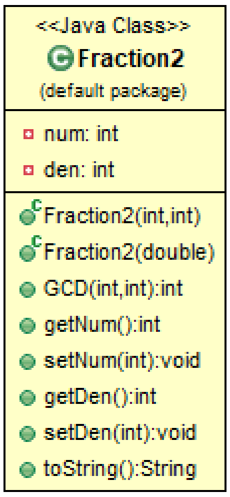
After you solve the problem in part#1, change the return of the method to String. Now, change the output of the method from decimal to fraction (ex. from 0.5 to 1/2). In addition, add a third method called GCD (greatest common devisor) and use it to show the fraction in the simplest form.

Part#3 Using arrays:

Now that you have the results of the code in a fraction form, try to add the result of one fraction addition with another fraction. I am sure that you can achieve this task, but it will be difficult if you still using the String output. Instead, you can use arrays to return two numbers from the method. Of course, the output is not in a fraction shape, but you still have the numerator and denominator. The caller of the method can take the output and shape them in the way he wants. Finally, in the main method calls the "fraction_addition" twice and adds the results from the first call to the third input in the second call.

Part#4 Using OOP:

You saw in the last part that you can join the numerator and denominator in one unit (array), but this is not what Java was built for!! Java uses object as a data unit. Therefore, you have to create a new class called Fraction2 that looks just like the following UML.



You can see that this class holds two data fields (numerator and denominator). Those two fields have to have an initial value that is been taken care off by the two constructors. The first one, simply takes the numerator and denominator and store them. The second constructor, takes a decimal number and converts it to a fraction. Then stores the figured numerator and denominator. There is a method for GCD, getters and setters. Finally, the toString method shows the values that are stored in any object in the following form.

```
1/2 | 0.5
```

After you are done from creating the Fraction2 class, it is time to use it. Create a new driver class that has two methods, the "fraction_addition" and the main methods. The fraction addition method should have the following signature.

```
public static Fraction2 fraction_addition(Fraction2 frac1,Fraction2 frac2) {
```

As you can see that the "fraction_addition" deals with object from the Fraction2 class that not only holds the numerator and denominator, but also provide different constructors and methods that improve the efficiency of your program. Finally, copy/paste the following main method and run the code.

```
public static void main(String[] args) {
    Fraction2 frac1 = new Fraction2(0.5);
    System.out.println(frac1);

    Fraction2 frac2 = new Fraction2(1,8);
    System.out.println(frac2);

    Fraction2 result = fraction_addition(frac1,frac2);
    System.out.println(result);
}
```

Part#5 Considering the caller object as the first operand:

Move the "fraction_addition" method inside the Fraction2 class. Change the methods signature line to be like the following.

```
public Fraction2 fraction_addition(Fraction2 frac) {
```

Now the "fraction_addition" method is taking one fraction only. The other fraction is the same object calling this method. Did you notice that the method is not static anymore? Do you know why? Run the same main method -after you make a minor change to make it work- and check the results.

Part#6 Using the caller object as the data holder:

```
public void fraction_addition2(Fraction2 frac) {
```

Add a new method called "fraction_addition2" -with the same signature above- to the Fraction2 class that does the same tasks as the "fraction_addition" method. However, this new method does not return anything. The results are kept in the caller object. Update the main method with the following code and run the code.

```
public static void main(String[] args) {
        Fraction2 frac1 = new Fraction2(0.5);
        System.out.println(frac1);

        Fraction2 frac2 = new Fraction2(1,8);
        System.out.println(frac2);

        frac1.fraction_addition2(frac2);
        System.out.println(frac1);
}
```

What to turn in

You should turn your work in a **MS Word** file that has your codes for each part as specified below. All the codes should be yours. **NO COPY/PASTE IS ALLOWED**. You should always copy/paste the code from your IDE and take a screenshot of the console.

Part#1:
- The Fraction1 class before fixing the wrong results and screenshot of the console after you run the code.
- The Fraction1 class after fixing the wrong results and screenshot of the console after you run the code.
- Why you got the wrong results and how did you fix it?

Part#2&3:
- The Fraction1 class update with a screenshot of the console after you run the code.

Part#4:
- The Fraction2 class update
- The driver class with a screenshot of the console after you run the code.

Part#5:
- The Fraction2 class update
- The driver class with a screenshot of the console after you run the code.

Part#6:
- The Fraction2 class update
- The driver class with a screenshot of the console after you run the code.