Unit 6 Project – Fraction Calculator

Introduction

This project is designed to get you practice building your own object class and testing it with a client class. You will be creating two classes, one called `Fraction` and the other called `FractionCalculator`. The `Fraction` class is an object that holds information about a fraction (numerator and denominator). It will have several constructors and both private and public methods implementing the behavior of a fraction. The `FractionCalculator` class is a class that will allow the user to enter in fractions and operations, calculating and displaying the result. It will run until the user tells it to quit.

When this program is complete, you won't have to second guess your fraction arithmetic ever again!

Part 1 - Fraction Class

Private Variables, Constructors and Getters

Your `Fraction` class must have two private instance variables holding the numerator and denominator. They must be integers.

TODO: Create two private instance variables for the Fraction class.

Constructors

Constructor with Two Parameters

This constructor is the typical fraction where the first parameter initializes the numerator and the second parameter initializes the denominator. It should throw an `IllegalArgumentException` if the denominator is zero.

For example:

```
Fraction frac = new Fraction(4, 5);
```

Would create a `Fraction` with numerator equal to 4 and denominator equal to 5.

It would be nice if the user enters a negative denominator for us to bump the negative sign to the numerator. This simplifies a few things for us, like the toString method and determining if a Fraction is positive or negative. For example, -3/-2 looks silly and should be converted to 3/2. Likewise, 5/-3 should be converted to -5/3. Have your two parameter constructor check if the denominator is negative and fix it so that the denominator is always positive.

Lastly, since 0=0/4=0/-2=0/1, we should have a unique representation for 0, so we might as well always write 0 as 0/1. Have your constructor convert the denominator to 1 if the numerator is zero.

Constructor with One Parameter (Integer Fraction)

This constructor takes a single parameter and creates a `Fraction` object equal in value to the integer parameter. That is, if we were to write:

```
Fraction frac = new Fraction(3);
```

It would create a `Fraction` with numerator equal to 3 and denominator equal to 1. You must call the constructor with two parameters to solve this.

Constructor with No Parameters (Zero Fraction)

This would construct the Fraction equal to zero. In this case, you must call the constructor with one parameter to solve this. In other words, `new Fraction()` and `new Fraction(0)` should be equal.

TODO: Write 3 constructors for your `Fraction` class.

1. The first constructor has two parameters and:
    a. Throws an IllegalArgumentException when the denominator is zero.
    b. Moves the signs around if the denominator is negative.
    c. Every fraction equal to zero has a denominator of 1.
2. The second constructor has 1 parameter and calls the first constructor using `this`.
3. The third constructor has no parameters and calls the second constructor using `this`.

Getters

Write two public methods called `getNumerator` and `getDenominator` that return the values of the private variables. Due to private access, other classes cannot directly access these variables, so these getters enable any client class to access these variables.

TODO: Write getters for your private instance variables.

toLowestTerms() Method

When we perform operations with fractions we should reduce our answers to lowest terms. We want to have a method that converts **`this`** to lowest terms. To convert a fraction to lowest terms we have to determine the greatest common divisor (factor) between the numerator and denominator.

Euclidean Algorithm for GCD

The greatest common divisor of two numbers a and b, is the largest number that evenly divides both a and b. For example, the GCD of 54 and 24 is 6, since all the factors of 54 are 1, 2, 3, 6, 9, 18, 27, 54 and all the factors of 24 are 1, 2, 3, 4, 6, 8, 12, 24. The common factors are 1, 2, 3, and 6, so the GCD is 6. If you're interested in reading more on GCD, go to https://en.wikipedia.org/wiki/Greatest_common_divisor.

Determining the GCD is important for reducing fractions, since if my fraction was 24/54, I would want to divide the numerator and denominator by 6 to reduce it to 4/9.

The Euclidean Algorithm is a fast method for determining the GCD of two numbers. Here is pseudocode for its implementation:

```
while a and b are not zero
      find the remainder of a divided by b
      set a to b
      set b to the remainder you found
return a
```

Here is an example of how it would work if a is 105 and b is 147.

Loop Iteration	Value of a at end of loop run	Value of b at end of loop run	a % b
Pre-Loop	105	147	
1	147	105	105 % 147 = 105
2	105	42	147 % 105 = 147 - 105 = 42
3	42	21	105 % 42 = 105 - 84 = 21
4	21	0	42 % 21 = 0

This shows that the GCD of 105 and 147 is 21, since 105 = 5 * 21 and 147 = 7 * 21. For more info on the Euclidean Algorithm for determining GCD go to https://en.wikipedia.org/wiki/Euclidean_algorithm.

TODO: Implement a **public static** method called `gcd` that takes two ***positive*** integers as parameters and returns an int that is their greatest common divisor.

TODO: Implement a public method called `toLowestTerms` that uses the `gcd` method to reduce **`this`** fraction to lowest terms.

toString() Method

All classes inherit the `Object` toString() method so we should override it to print out something that is not gobbledygook. Currently, if we try to print out a `Fraction` we get something like this: `Fraction@154617c`. This isn't very helpful for showing printing out our fraction. Override the `toString` method by returning a String of the form: [numerator]/[denominator]. If the fraction's denominator is 1, `toString` must return a String that is just the numerator.

For example, if we had the following lines of code:

```
Fraction fOne = new Fraction(2,3);

Fraction fTwo = new Fraction(4,-1);

System.out.println("fOne is " + fOne + " and fTwo is " + fTwo);
```

The program should output:

```
fOne is 2/3 and fTwo is -4
```

TODO: Override `toString` so that it returns a String (numerator/denominator or just numerator) representation of your `Fraction` object.

toDouble() Method

It may be helpful when checking our `Fraction` math if we can get a double representation of a `Fraction`. Write a `toDouble` method that returns a double approximately equal to **`this`** `Fraction`. For example,

```
Fraction fOne = new Fraction(1,2);

Fraciont fTwo = new Fraction(-2,3);

System.out.println(fOne.toDouble() + " " + fTwo.toDouble());
```

would output:

```
0.5 -0.6666666666666666
```

TODO: Write a `toDouble` method that returns a double representation of **`this`** `Fraction`.

add(), subtract(), multiply(), and divide() Methods

Implement add, subtract, multiply and divide methods for your `Fraction` objects. Each method takes another `Fraction` as a parameter and returns a new `Fraction` that is the result of that operation with **`this`** `Fraction`.

Suppose we have the following fraction objects defined by the left hand column. The results of these sample calls are fractions with the following numerator and denominator.

Once the operation has been performed, please call your `toLowestTerms()` method to reduce the resulting Fraction.

The only method to be wary of is the divide method. If the user tries to divide by zero, it should throw a `new IllegalArgumentException()` and print `"Error: Cannot divide by zero."`.

`Fraction`	Sample Calls	Return

`Fraction fOne = new Fraction(1, 2)`	`fOne.add(fTwo)`	new Fraction(0,1)
`Fraction fTwo = new Fraction(1, -2)`	`fTwo.subtract(fOne)`	new Fraction(-1,1)
`Fraction fThree = new Fraction(2, 3)`	`fThree.multiply(fFour)`	new Fraction(2,1)
`Fraction fFour = new Fraction(3, 1)`	`fFour.divide(fThree)`	new Fraction(9,2)

TODO: Implement `add`, `subtract`, `multiply` and `divide` methods. Each method should take a `Fraction` as a parameter and return a `Fraction` equal to the sum, difference, product or quotient, with **`this`** `Fraction` respectively.

Reduce all answers to lowest terms.

The `divide` method should throw an `IllegalArgumentException` if the client tries to divide by the zero `Fraction`.

equals() Method

We should have a way of determining whether or not two fractions are equal. Override the `Object` equals() method so that it accurately determines whether or not two fractions are equal.

In order to have it override, it has to take an `Object` as a parameter. Your method should check whether or not the parameter is an `instanceof Fraction`, since if it is not a `Fraction` it cannot be equal.

Don't forget to cast the parameter to a `Fraction` after you check if it is an Object of type `Fraction` so that you can access its variables.

Two fractions are equal if they represent the same number (i.e. 3/6 = 1/2 and -2/3 = 2/-3).

TODO: Override `equals` so that it returns true if the `Object` parameter is a `Fraction` equal in value, false otherwise.

Part 2 – FractionCalculator Class

In this section, you will implement a FractionCalculator class that has a main method and three helper methods. Here is a screenshot from a sample run:

```
This program is a fraction calculator
It will add, subtract, multiply and divide fractions until you type Q to quit.
Please enter your fractions in the form a/b, where a and b are integers.
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): +
Please enter a fraction (a/b) or integer (a): 1/2
Please enter a fraction (a/b) or integer (a): 1/3
1/2 + 1/3 = 5/6
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): -
Please enter a fraction (a/b) or integer (a): 4
Please enter a fraction (a/b) or integer (a): 3/2
4 - 3/2 = 5/2
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): /
Please enter a fraction (a/b) or integer (a): 3/12
Please enter a fraction (a/b) or integer (a): 1/4
3/12 / 1/4 = 1
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): *
Please enter a fraction (a/b) or integer (a): -2
Please enter a fraction (a/b) or integer (a): 5/2
-2 * 5/2 = -5
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): =
Please enter a fraction (a/b) or integer (a): 50/100
Please enter a fraction (a/b) or integer (a): 1/2
50/100 = 1/2 is true
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): q

Process finished with exit code 0
```

Your program should be robust so that it if the user enters invalid input it will continue to re-prompt them until it is valid. Here is an example run where the user is confused and enters invalid input:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_65.jdk/Contents/Home/bin/java ...
This program is a fraction calculator
It will add, subtract, multiply and divide fractions until you type Q to quit.
Please enter your fractions in the form a/b, where a and b are integers.
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): foo
Invalid input (+, -, /, *, = or Q to quit): operation
Invalid input (+, -, /, *, = or Q to quit): 1/2
Invalid input (+, -, /, *, = or Q to quit): +-
Invalid input (+, -, /, *, = or Q to quit): / *
Invalid input (+, -, /, *, = or Q to quit): /
Please enter a fraction (a/b) or integer (a): three
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: one/2
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: a/b
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: 1 3
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: 3/0
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: 1/3
Please enter a fraction (a/b) or integer (a): 0/2
1/3 / 0 = Undefined
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): q

Process finished with exit code 0
```

getOperation() Method

Implement a method called `getOperation` that takes a Scanner as input and returns a String representing a valid operation. If they enter anything except "+", "-", "/", "*", "=", "q", or "Q" it will re-prompt them until it is a valid input. Here is example output from a call to `getOperation`.

```
Please enter an operation (+, -, /, *, = or Q to quit): plus
Invalid input (+, -, /, *, = or Q to quit): 1 + 2
Invalid input (+, -, /, *, = or Q to quit): + or -
Invalid input (+, -, /, *, = or Q to quit): gimme a break!
Invalid input (+, -, /, *, = or Q to quit): fine...
Invalid input (+, -, /, *, = or Q to quit): *
```

At the end of this run, `getOperation` would have returned "*".

TODO: Implement `getOperation` method that takes a Scanner as input and prompts the user for a valid operation (+, -, /, *, =, or Q) and returns that operation as a String.

validFraction() Method

Implement a method called `validFraction` that take a String as input and returns `true` if the parameter is of the form "a/b" where a can be any integer and b is any positive integer, and `false` otherwise. Here are some example calls and returns:

Call	Return
`validFraction("1/4")`	`true`
`validFraction("-2/1")`	`true`
`validFraction("21")`	`true`
`validFraction("2/-1")`	`false`
`validFraction("/1")`	`false`
`validFraction("foo")`	`false`
`validFraction("1.5/2")`	`false`
`validFraction("3 1")`	`false`

Some things to be mindful of when implementing this:

- The first character may or may not be a '-' character. If a negative shows up anywhere else, then it is not a valid fraction. It may be helpful to remove the '-' character if there is one.
- If there is no '/' character, then every character in the string must be a number (if you removed the '-' sign).
- If there is a '/' character, then it may be helpful to create substrings for the numerator and denominator.
    - Both substrings must be non-empty.
    - Both must be entirely made of numbers.
    - The denominator cannot be "0".

Hint 1: It may be useful to create a helper method `isNumber` that takes a String as input and returns true if every character in the String is a number 0-9 and false otherwise. This method can also check for empty strings.

Hint 2: Once you determine whether or not the Strings are numbers, you may find the Integer.parseInt() method helpful.

http://stackoverflow.com/questions/5585779/converting-string-to-int-in-java/

TODO: Implement `validFraction` method to determine if a String is of the form "a/b" or "a" where a is any integer and b is any positive integer.

getFraction() Method

Implement a method called `getFraction` that takes a Scanner as input and returns a `Fraction` object. It prompts the user for a String that is a `validFraction`. If they enter any thing that is not a valid Fraction, it should re-prompt them until it is valid. Here is an example call of `getFraction`:

```
Please enter a fraction (a/b) or integer (a): fraction!
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: three
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: two/one
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: 3/two
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: 1 / 2
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: 0.5
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: 2/-1
Invalid fraction. Please enter (a/b) or (a), where a and b are integers and b is not zero: -31
```

This call would return a new `Fraction` object equal to -31/1.

No user input should throw an exception! If you are getting exceptions, then it is likely your `validFraction` method isn't correct.

TODO: Implement `getFraction` method that takes a Scanner as input and returns a `Fraction` object as long as the input is valid.

Putting it all together!

Write a short introduction method that describes the calculator program. Call this method in your main method and then get an operation from the user. As long as they enter something that's not "q" or "Q" you should run the calculator. Get two fractions from the user and then perform whichever operation they ask for, printing the result of the operation. If they ever try to divide by zero, then you should print `Undefined`. Print **80** '-' characters on a line before each operation.

Here is an example run of the entire program:

```
This program is a fraction calculator
It will add, subtract, multiply and divide fractions until you type Q to quit.
Please enter your fractions in the form a/b, where a and b are integers.
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): +
Please enter a fraction (a/b) or integer (a): 1/2
Please enter a fraction (a/b) or integer (a): 1/3
1/2 + 1/3 = 5/6
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): -
Please enter a fraction (a/b) or integer (a): 4
Please enter a fraction (a/b) or integer (a): 3/2
4 - 3/2 = 5/2
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): /
Please enter a fraction (a/b) or integer (a): 3/12
Please enter a fraction (a/b) or integer (a): 1/4
3/12 / 1/4 = 1
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): *
Please enter a fraction (a/b) or integer (a): -2
Please enter a fraction (a/b) or integer (a): 5/2
-2 * 5/2 = -5
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): =
Please enter a fraction (a/b) or integer (a): 50/100
Please enter a fraction (a/b) or integer (a): 1/2
50/100 = 1/2 is true
--------------------------------------------------------------------------------
Please enter an operation (+, -, /, *, = or Q to quit): q

Process finished with exit code 0
```

TODO: Write an introduction method describing the program. Get an operation from the user, while the operation is not "q" or "Q" get two Fractions and then print the result of the operation between the two Fractions.

Hacker Problem - FractionCalculatorAdvanced

Create another class called `FractionCalculatorAdvanced`. You may cut and paste useful methods from FractionCalculator. The key difference between FractionCalculator and FractionCalculatorAdvanced is that the user can enter in their operations on a single line.

Allow the user to enter their input onto a single line. Your program must be robust so that if the user enters in invalid input, it will re-prompt them until they either enter a q to quit or a valid operation. It is possible to do this without try/catch, but it is quite difficult. You may read about try/catch blocks here: http://beginnersbook.com/2013/04/try-catch-in-java/.

Here is sample output from a run of `FractionCalculatorAdvanced`:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_65.jdk/Contents/Home/bin/java ...
This program is a fraction calculator
It will add, subtract, multiply and divide fractions until you type Q to quit.
Valid operations are of the form "[FRAC] [OPERATION] [FRAC]".
[FRAC] can be either a single integer or two integers separated by "/".
[OPERATION] can be +, -, *, / or =.
Enter an operation (q to quit): this
Invalid operation. Must be "[FRAC] [OPERATION] [FRAC]".
Enter an operation (q to quit): isn't
Invalid operation. Must be "[FRAC] [OPERATION] [FRAC]".
Enter an operation (q to quit): a valid
Invalid operation. Must be "[FRAC] [OPERATION] [FRAC]".
Enter an operation (q to quit): op + ation
Invalid operation. Must be "[FRAC] [OPERATION] [FRAC]".
Enter an operation (q to quit): 2 + foo
Invalid operation. Must be "[FRAC] [OPERATION] [FRAC]".
Enter an operation (q to quit): 3 - zee
Invalid operation. Must be "[FRAC] [OPERATION] [FRAC]".
Enter an operation (q to quit): 45 / 22
45 / 22 = 45/22
Enter an operation (q to quit): 33 / 11
33 / 11 = 3
Enter an operation (q to quit): 1/2 * 4/3
1/2 * 4/3 = 2/3
Enter an operation (q to quit): 11/14 / 7/11
11/14 / 7/11 = 121/98
Enter an operation (q to quit): q

Process finished with exit code 0
```