ICT-4361 Homework 10

Purpose

This assignment provides you the opportunity to work with fractions as a new abstract data type!
You should be able to help third and fourth graders with their homework when you complete this exercise!

What to Hand In

For this, as well as future programming exercises, hand in a listing of the program (which must be formatted in a reasonable style), and sample run(s) of the program.

Assignment submission instructions:
Please combine multiple files into a single "zip" archive, and save it in a location that you will remember. Please ensure you have collected all the necessary files and pictures requested. Please preserve the directory structure for the Java source if at all possible.
When you are ready to submit the assignment, go to the assignment page, and click on the "Submit Assignment" link.
The section you see requires you to check the Plagiarism policy checkbox, and provides a button to choose the file you saved the assignment in.
Click the Submit Assignment button at the bottom of the form to complete your submission.

Please pay attention to your code indentation to ensure your paper is easy to read and understand.

Problems

I. Derive a new kind of Number called Fraction.
   1. Give it appropriate methods to add, subtract, multiply and divide fractions. Also, provide the required methods for a Number
   2. Determine some appropriate way to convert a Fraction to a String, and implement the appropriate method. Simply outputting the double equivalent is not a satisfactory answer.
   3. You must provide a method (which could be private) to simplify your Fraction into lowest terms. That method could be used after every operation changing the Fraction, and certainly is required to provide a reasonable comparison operation.
   4. Make sure to implement equals so you have a way to compare Fractions.
   5. Create a main method or test class with a main method with adequate number of test cases to demonstrate that your Fraction behaves appropriately.

II.

Notes

- A. A fraction has a numerator and a denominator (the part above the line and the part below the line).
- B. The denominator may not be zero (0).
- C. If the denominator is one (1), the Fraction represents the integer in the numerator, and should be converted to a String just like an integer.
- D. Remember the rules for addition, subtraction, multiplication, and division of fractions:

   1. $\frac{a}{b} + \frac{c}{d} = \frac{a \times d + c \times b}{b \times d}$

   2. $\frac{a}{b} - \frac{c}{d} = \frac{a \times d - c \times b}{b \times d}$

   3. $\frac{a}{b}$ x $\frac{c}{d} = \frac{a \times c}{b \times d}$

   4. $\frac{a}{b} / \frac{c}{d} = \frac{a \times d}{b \times c}$

- E. To simplify a Fraction, one divides the numerator and denominator by the greatest common divisor, which is the largest number that divides each of them evenly. 1. If you are not using BigInteger, you can use the following gcd method. It can be modified to use long instead of int: /* As a static method using int */

```
public static int greatestCommonDivisor (int int1, int int2)
{
  int temp;
  if ( int1 < 0 )
    int1 = -int1;
  if ( int2 < 0 )
    int2 = -int2;
  while ( int2 != 0)
```

```
  {
    temp = int1 % int2;  // % is the modulus operator
    int1 = int2;
    int2 = temp;
  }
  return int1;
}
```

2. Additionally, one should ensure the denominator is positive. If you are using BigInteger to represent the components, you compare to zero with compareTo and use negate to reverse the sign. 3. Once you simplify, you can compare Fractions by manipulating denominators and numerators. Hint: For compareTo you can subtract them and look at the result.

- F. If you use BigInteger to represent these components, you will get a gcd method to go along with it! 1. BigInteger also provides a satisfactory set of name expectations for the operations one might expect on a Fraction. These are worth consulting!
- G. To create fractions, it is sensible to have a String constructor. A sample constructor is provided below (this one uses the BigInteger data members; a somewhat simpler, but similar one would work with long data members):

```
public Fraction (String fractionString) throws NumberFormatException {
  String[] parts = fractionString.split("/");
  denominator = BigInteger.ONE;
  numerator = BigInteger.ZERO;
  if ( parts.length == 0 || parts.length > 2 ) {
    throw new NumberFormatException("Illegally formatted Fraction");
  }
  // The BigInteger constructor may throw NumberFormatExceptions
  numerator = new BigInteger(parts[0].trim());
  if ( parts.length == 2 ) {
    BigInteger d = new BigInteger(parts[1].trim());
    if ( d.equals(BigInteger.ZERO))
      throw new NumberFormatException("Fraction may not have zero denominator");
    denominator = d;
  }
  simplify();
}
```

- H. Remember: Delegate, delegate, delegate!

Evaluation

Criteria	Weight
Creation of the Fraction type and its components, including a String constructor.	20
Add, Subtract, Multiply, and Divide operations	20
All required Number methods	10
simplify and toString methods	10
Method for comparison of Fractions (equals, compareTo)	20
Main program with an adequate number of test cases to demonstrate the above	20