The Fraction Class

The Fraction Class is the definition for a Fraction object. Classes contain Attributes (the data) and Behaviors (the methods).

The Fraction Attributes

A fraction is an object that contains a numerator and denominator. Your Fraction class definition must therefore contain two private attributes to represent the numerator and denominator. These attributes must be integers.

The Fraction Behaviors

Your Fraction class must contain public definitions for the following methods. *The methods are the public interface to a Fraction object so you are NOT allowed to change the return type, method name, or what is sent to the method* (notice that nothing is sent to any of the methods).

public String toString( )

This method returns a String that looks like the fraction (ex. 34/250). This method does NOT display anything on the screen.

public double getDecimal( )

This method returns the decimal equivalent of the Fraction by dividing the numerator by denominator.

public void reduce( )

This method potentially changes the numerator and denominator of the object by reducing the object to lowest terms. This is the only method that can change the numerator or denominator. reduce( ) is the only mutator in the Fraction class definition.

public String toMixed( )

This method returns a String which includes the whole number and a "reduced" fractional component. NOTE: this method DOES NOT MODIFY the numerator or denominator of the fraction – it simply returns a String. For example, if the object is the Fraction 35/10, the toMixed( ) method will return the String: "3 1/2" . Remember, the numerator will remain 35, and the denominator will remain 10, after this method is called.

If there is no whole number part, do not display 0.

You must also account for negative values in the numerator and denominator position when the toMixed() method is called.

The Constructor Methods:

Along with the above methods, you must also be able to **instantiate**, or create, a new Fraction object. This is done by using the new operator and calling the constructor method. Define the following overloaded constructor:

Fraction(int n, int d)

This constructor initializes the numerator and denominator of the object to the values (n and d) that are sent to the constructor.

Fraction( )

This constructor initializes the numerator and denominator to 1.

The Test Program (main method)

Test your Fraction class. You must also create a small test program (main method) in a separate java class file to test your fraction class. Your program should do the following: create 5 different Fraction objects. Hard code the values of the numerator and denominator so that the user does not have to enter these values. Part of your evaluation on this assignment is based on the quality of the Fractions that you instantiate to test your code.

Next, allow the user to choose which method to perform. You can use the Netbeans output window or dialog boxes for the I/O in this program. Display a set of numbered choices to the user which includes a choice to allow the user to test every Fraction behavior:

1. Test the toString() method
2. Test the reduce() method
3. Test the toMixed() method
4. Test the getDecimal( ) method
5. Quit

When the user makes the choice, invoke that method on all 5 fraction objects, displaying the returned value of the fraction objects on the screen. Do not display anything if the method is void (in the case of reduce( )). Redisplay the numbered choices so the user may choose other options before quitting the program.

Hint: For ease of coding and for full credit, in the main method I strongly recommend that you create an array of 5 Fraction objects. If you don't use an array, you will not receive full credit for this assignment. You can create and instantiate your Fraction array with the following code. Choose your fractions carefully in order to fully test your code. Does your code work for negative numerators and denominators?

Fraction [ ] myFracs = { new Fraction(12, 18) , new Fraction( ), new Fraction(125, 30), … };