Assignment 5: Fraction

Instructions

For this assignment, you will create a class which must be named `Fraction` to store fractions. The class will hold two integer values: a numerator and a denominator. For your class, we will stick to positive fractions, meaning the numerator and the denominator must both be greater than 0. Your `Fraction` class will also keep track of the number of `Fraction` objects created by a program. You will need to create appropriate variables to store the information in these fields, all of which should be private.

You will also need to add all of the public constructors and methods which are described below. You may choose to add other methods to help you implement these.

Constructors

- `Fraction()`: default constructor which creates a fraction 1/1
- `Fraction(int n, int d)`: If `n` is positive, set numerator to `n`. Otherwise, set numerator to 1. If `d` is positive, set denominator to `d`. Otherwise, set denominator to 1.

Accessors

- `int getNumerator()`: Returns the numerator of the fraction.
- `int getDenominator()`: Returns the denominator of the fraction.
- `static int getNumFractions()`: Returns the number of fractions created in a program so far.
- `String toString()`: Returns the fraction as a string in the format “numerator/denominator”. For example 1/2 or 5/3.
- `String mixedNumber()`: Returns any improper (top-heavy) fraction as a mixed number, for example, 2 3/5. If the numerator of the fraction part is 0, return only the integer part of the mixed number. If the fraction is proper, return only the fraction part.

Mutators

- `void setNumerator(int n)`: Sets the numerator of the fraction to `n` if it is positive.
- `void setDenominator(int d)`: Sets the denominator of the fraction to `d` if it is positive.
- `void add(int n, int d)`: If `n` and `d` are both positive, add the fraction n/d to this fraction. Otherwise, leave the fraction unchanged. In general the sum of the fractions a/b and c/d is(a*d + c*b)/(b*d).
- `void add(Fraction other)`: Add the fraction represented by `other` to this fraction. In general the sum of the fractions a/b and c/d is(a*d + c*b)/(b*d). Postcondition: the `other` Fraction is left unchanged.

- `void multiply(int n, int d)`: If `n` and `d` are both positive, multiply the fraction n/d by this fraction. Otherwise, leave the fraction unchanged. Unsimplified is ok in this case, the formula for multiplying fractions a/b and c/d is (a * c)/(b*d).
- `void multiply(Fraction other)`: Set this fraction to the product of this fraction and `other`. Unsimplified is ok in this case, the formula for multiplying fractions a/b and c/d is (a * c)/(b*d). Postcondition: the `other` Fraction is left unchanged.

To test your code, you will need to run the `runner_Fraction` class main method. Don't add a main method to your Fraction class as your code will not be checked or scored correctly.

This runner class is a little more advanced than some of the others you have seen so far. The runner first creates an initial fraction by using the default constructor which you write in the Fraction class. You then have the option to test each method individually by typing the name of the method, or to create a new Fraction using the other Fraction constructor. When a method requires parameters you'll be asked to input these, and if there is an overloaded method you'll be asked to specify which method signature you want to use by typing 1 or 2.

As the runner code has calls to all the required public Fraction methods, it won't compile or run until it can see headers for all these methods. So it's a good idea to add headers for each method you have to write in the Fraction class when you first start. You can leave the method bodies blank until you are ready to write them, or if a return is required add a dummy return like `return 0;` or `return "";`. This way you'll be able to test the methods you write as you go along.

Milestones

As you work on this assignment, you can use the milestones below to inform your development process.

Milestone 1: Write the method and constructor headers for all the public methods/constructors described, and add dummy return statements (e.g. return ""; or return 0;) to any non-void methods. This will allow you to compile your runner class and test methods as you go along. Create private variables to store the class information (numerator, denominator and number of fractions).

Milestone 2: Write the `setNumerator`, `setDenominator` methods and set up the two class constructors to initialize the numerator and denominator of each new Fraction. Write the accessor methods `getNumerator`, `getDenominator` and `toString` methods and test all these methods using the runner class.

Milestone 3: Add the `mixedNumber` method to the class. Add code to the constructors to keep track of the number of fractions, then implement the static `getNumFractions` method. Test your code using the runner.

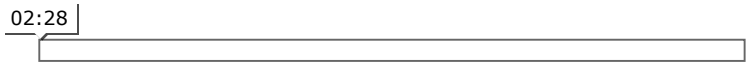
Milestone 4: Write the mutator method `add(int n, int d)` and then overload this by writing the `add(Fraction other)` method. Repeat this process for both of the `multiply` methods. Test all methods thoroughly using the runner class.

Optional square() method

After you have completed implementing all the public methods for your Fraction class, there's one more method you should consider adding. This is an optional part of the assignment, and will not impact your score or grade. But, you might want to try it out as it demonstrates an interesting thing you can do in Java: pass an object to its own method. If you want to test this method out, feel free to modify the runner file so you can call this method too.

Fraction Assignment Extension

from **Edhesive**



edhesive

Download Video (https://videos.edhesive.com/courses/AP_CSA/Unit_5/Fraction_Assignment_Extension.mp4)

Files

Fraction.java

runner_Fraction.java

STATUS SUBMITTED 100% SAVE SUBMIT

```
5   private int denominator;
6   private static int counter = 0;
7
8   public Fraction() {
9     numerator = 1;
10    denominator = 1;
11    counter++;
12  }
13
14  public Fraction(int n, int d) {
15    if (n > 0) {
16      numerator = n;
17    } else {
18      numerator = 1;
19    }
20    if (d > 0) {
21      denominator = d;
22    } else {
23      denominator = 1;
24    }
25    counter++;
26  }
27
28  public int getNumerator() {
29    return numerator;
30  }
31
32  public int getDenominator() {
33    return denominator;
34  }
35
36  public static int getNumFractions() {
37    return counter;
38  }
39
40  public String toString() {
41    return (numerator + "/" + denominator
42  }
43
44  public String mixedNumber() {
```

INSTRUCTIONS RUN CODE GRADING HISTORY

For this assignment, you will creat a class which must be named `Fraction` to store fractions. The class will hold two integer values: a numerator and a denominator. Fo your class, we will stick to positive fractions, meaning the numerator and the denominator must both b greater than 0. Your `Fraction` class will also keep track of the number of `Fraction` objects created by a program. You will nee to create appropriate variables to store the information in these fields, all of which should be private.

You will also need to add all of the public constructors and methods which are described below. You ma choose to add other methods to help you implement these.

Constructors

- `Fraction()`: default