AP Computer Science — Project 21 - Fraction — Name:

Dr. Paul L. Bailey — Monday, October 23, 2017

The completed project is due Sunday, October 29, at 11:59 PM. Zip and send source code of all Java programs via email to `paul.bailey@basised.com` before that time. Please rename the zip file so it has your name on it. Also, put your name in a comment on the top of each program.

If you have any questions, you may ask the question in an email. Do not copy code from other students. Do not type any code that you have received advice on, unless you understand what you are typing. You will be tested on your understanding of the code you submit, without having access to that code.

The purpose of this project is get more acquainted with object oriented programming by creating a new "type", that being rational numbers.

Create a new folder called `P09_Fraction`. This will hold all three classes created in this project.

Program 1. Create a source file called `Number.java`. In this source file, create the class `Number`. In this class, place the following static method.

```
public static int gcd(int m, int n)
{
    if (n == 0 || m == 0) return 0;
    if (n < 0) n = -n;
    if (m < 0) m = -m;
    int r = 0;
    while ((r = n % m) > 0)
    {
        n = m;
        m = r;
    }
    return m;
}
```

This method computes the greatest common divisor of two integers.

Create a source file called `Program.java`. In this source file, create a `main` method, which calls a `test1` method, which tests the `Number.gcd` method.

Program 2. Create a source file called `Fraction.java` In this source file, begin your `Fraction` class. The two private instance fields represent the numerator and denominator of the fraction.

```
public class Fraction
{
    private int top = 0;
    private int bot = 1;

    public Fraction()
    { }

    public Fraction(int a)
    {
        top = a;
    }

    public Fraction(int a, int b)
    {
        install(a, b);
    }

    public void install(int a, int b)
    {
        top = a;
        bot = b;
    }

    public String toString()
    {
        return "Fraction";
    }
}
```

Complete the `install` method to guarantee that the denominator is positive and the numerator and denominator have no common prime factors. Use the `Number.gcd` method to do this. Complete the `toString` method for testing; it should produce the format `top/bot` for toString, unless the fraction is an integer; in this case use the format `top`. For example, the code

```
    Fraction x = new Fraction(42, 15);
    Fraction y = new Fraction(11, 11);
    System.out.printf("x = %s and y = %s.\n", x, y);
    System.out.printf
```

should output

```
x = 14/5 and y = 1.
```

In your `Program` class, create and execute a `test2` method to test the fraction code you have so far.

Program 3. Enhance your `Fraction` class to include the following methods.

```
public float toFloat()
public static Fraction negative(Fraction x)
public static Fraction reciprocal(Fraction x)
public static Fraction add(Fraction x, Fraction y)
public static Fraction subtract(Fraction x, Fraction y)
public static Fraction multiply(Fraction x, Fraction y)
public static Fraction divide(Fraction x, Fraction y)
public static boolean equals(Fraction x, Fraction y)
```

In your `Program` class, create and execute a `test3` method to test each method after you write it. Don't write too much code without compiling and testing; doing so is a time-wasting rookie mistake.

Just for example, this is my `test3` method.

```
public static void test3()
{
    x = new Fraction(50, 18);
    y = new Fraction(96, 15);
    z = new Fraction(75, 27);

    System.out.printf("0 - %s = %s\n", x, Fraction.negative(x));
    System.out.printf("1 / %s = %s\n", x, Fraction.reciprocal(x));
    System.out.printf("%s + %s = %s\n", x, y, Fraction.add(x,y));
    System.out.printf("%s - %s = %s\n", x, y, Fraction.subtract(x,y));
    System.out.printf("%s * %s = %s\n", x, y, Fraction.multiply(x,y));
    System.out.printf("%s / %s = %s\n", x, y, Fraction.divide(x,y));
    System.out.printf("equals(%s, %s) = %s\n", x, y, Fraction.equals(x,y));
    System.out.printf("equals(%s, %s) = %s\n", x, z, Fraction.equals(x,z));
}
```

It produces this output.

```
0 - 25/9 = -25/9
1 / 25/9 = 9/25
25/9 + 32/5 = 413/45
25/9 - 32/5 = -163/45
25/9 * 32/5 = 160/9
25/9 / 32/5 = 125/288
equals(25/9, 32/5) = false
equals(25/9, 25/9) = true
```

Program 4. Enhance you `Fraction` class to put an ordering on the fractions. This will allow you to sort them later. To do you, you need to "implement the `Comparable` interface". This requires you to conform to the Comparable contract, which is to write a `compareTo` method. Every new class you create should implement an instance method called `equals`; do this as indicated below.

```
public class Fraction implements Comparable<Fraction>
{
    public boolean equals(Fraction that)
    {
        return equals(this, that);
    }

    public int compareTo(Fraction that)
    {
        return compare(this, that);
    }

    public static boolean equals(Fraction x, Fraction y)
    {
        return compare(x, y) == 0;
    }

    public static int compare(Fraction x, Fraction y)
    {
        // if x <  y, return a negative number
        // if x == y, return zero
        // if x >  y, return a positive number
        return 0;
    }
}
```

In your `Program` class, create and execute a `test4` method to test these methods.

Program 5. In the `Fraction` class, create a static method to generate an array of random fractions whose numerators and denominators are integers between 0 and 100. Modify the code below to do this. Check to make sure that the length is positive.

```
    public static Fraction[] generate(int length)
    {
        Fraction[] a = new Fraction[length];
        // loop to fill array with random fractions
        return a;
    }
```

In your `Program` class, create and execute a `test5` method which using this to generate an array of 20 random fractions, and then sorts them. Use the `Arrays.sort` method to sort the array of fractions. This method, supplied by the makers of Java, will call your `compareTo` method to sort the array. Use the `Arrays.toString` method to print the sorted array.

Create a parallel array of twenty `float`'s; loop through the sorted array of fractions, and use `toFloat` to store the fractions value as a float into the parallel array. Then print the array of floats, to see that they are in the correct order.

Program 6. (Bonus) Let x be a positive real number. The Babylonians the approximated $\sqrt{x}$ using the recursive sequence

$$a_1 = 1 \quad \text{and} \quad a_{n+1} = \frac{1}{2}\Big(a_n + \frac{x}{a_n}\Big).$$

Use your Fraction class and five iterations of this algorithm to find a rational approximation for $\sqrt{5}$. Then take this approximation, square it, and subtract it from 5 to see how close it is.