Programming Project 7 – Rational Number

Point Value – 100 points

Note: When you turn in an assignment to be graded in this class you are making the claim that you neither gave nor received assistance on the work you turned in (except, of course, assistance from the instructor).

Program Names: Rational.java and **RationalDriver.java**

Define a class for rational numbers. A rational number is a number that can be represented as the quotient of two integers. For example, ½, ¾, and so forth are all rational numbers (by ½ and so forth, we mean the mathematical meaning of the fraction, not the integer division this expression would produce in a Java program.) Represent rational numbers as two values of type `int`, one for the numerator, and one for the denominator. Your class should have two instance variables of type `int`. The name of the class is **`Rational`**. Include a default constructor and an overloaded constructor with two arguments that are used to set the instance variables of an object of the **`Rational`** class to any two integer values. Note that the numerator, the denominator, or both may be negative.

You should include a method to normalize the sign of the rational number so that the denominator is always positive and the numerator is either positive or negative. For example, after normalization, 4/-8 would be the same as –4/8.

Define the following methods (using the names in parenthesis) for addition (**add**), subtraction (**subtract**), multiplication (**multiply**), and division (**divide**) of objects of your class, **`Rational`**. Each of these methods must accept a **`Rational`** object as the single parameter. Arithmetic operations on fractions are defined by the following rules:

```
a/b + c/d = (ad + bc) / bd
a/b –c/d = (ad-bc) / bd
a/b * c/d = ac/db
(a/b) / (c/d) = ad / bc, where c/d ≠ 0
```

These are instance methods with a single argument that do not return a value. The result is a change in the state of the value of the data members of the calling object. For example, the add method (for addition) has a calling object and one argument. Therefore, if **`rationalNum1`** has a **`numerator`** value of **1** and a **`denominator`** value of **2** and **`rationalNum2`** has a **`numerator`** value of **1** and a denominator value of **4**, the method call,

```
rationalNum1.add(rationalNum2);
```

changes the values of the instance variables of **`rationalNum1`** so they represent the result of adding **`rationalNum2`** to the original version of **`rationalNum1 (numerator`** is **3** and **`denominator`** is **`4)`**.

Define observer (getter) and mutator (setter) methods. In addition, define the following method:

- **`public boolean equals(Object other)`**
- **`public String toString()`**

Hint: Two rational numbers a/b and c/d are equal if a*d equals c*b. Guidelines for overriding the **`equals`** method:

- Use the == operator to check if the argument is a reference to itself.
- Use the **instanceof** operator to check if the argument has the correct data type.
- Cast the argument to the correct type.
- For each significant data member, test for equality.
- Test these three properties:
    - Is it symmetric?
    - Is it transitive?
    - Is it consistent?

Write a second class called **`RationalDriver`** containing a main method. Your main method will allow the user to thoroughly test all of the methods in your **`Rational`** class. The main method should take keyboard input from the user; instantiate two **`Rational`** objects, and then ask the user which operations to perform. Allow the user to continue performing operations on rational numbers until the user elects to quit. For example:

```
Enter the numerator for rational number #1:
Enter the denominator for rational number #1:
Enter the numerator for rational number #2:
Enter the denominator for rational number #2:
Enter the corresponding number for the desired action,
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Test for Equality
6. Change 1st rational number
7. Change 2nd rational number
8. Exit
```

Develop this program in stages:

1. Default and Parameterized Constructors
2. **`toString()`** to display fraction - print out fraction to test it
3. Normalize fraction after constructor executes
4. Implement method – print out results
5. Repeat…. for the remaining project requirements
6. Loop to allow multiple rational numbers to be input

Your file names should be **`Rational.java`** and **`RationalDriver.java`**.

Test your program with several different data sets. Document your tests on the attached table and submit the plan with copies of your source code files.

Make sure your program is well documented - including the comment block header in the source code file; include a **`printHeading()`** method that is called from the main method; and follow all previous instructions about method and file names, identifiers, etc. Submit your source code to the assignment link in Blackboard by midnight on the due date.

Test Plan:

rationalNum1	rationalNum2	Action Performed	Expected Output	Actual Output

CSC 201 Computer Science I

Programming Project 7 – Rational Number Grading Rubric

Rational class:

Instance data members are correctly declared private (2 pts.)	__________
Default constructor included with appropriate assignments (5 pts.)	__________
Two-parameter constructor correctly written with call to normalization method (5 pts.)	__________
Accessor and mutator methods included for private data members (8 pts.)	__________
Normalization method written correctly (5 pts.)	__________
Add method written correctly (5 pts.)	__________
Subtract method written correctly (5 pts.)	__________
Multiply method written correctly (5 pts.)	__________
Divide method written correctly (5 pts.)	__________
Equals method written correctly (5 pts.)	__________
toString() method written correctly (5 pts.)	__________

RationalDriver class:

printHeading() method included and called from main() (2 pts.)	__________
Prompts for input for each of two rational numbers and allows for creation of a default instance (5 pts.)	__________
Menu option allows user to instantiate a new rational number to replace either of the two existing Rational objects (5 pt.)	__________
Menu correctly implements options to add, subtract, multiply, divide and quit (10 pts.)	__________
Method and identifiers are self-documenting (2 pts.)	__________
Files are named and submitted as specified (3 pts.)	__________
Appropriate use of comments (including header comment) (5 pts.)	__________
Submitted Test Plan (10 pts.)	__________
Total (100 pts.)	__________