

Quiz 4.a (Take Home)

Q1. Define a class for rational numbers as described below:

A rational number is a number that can be represented as the quotient of two integers. For example, $1/2$, $3/4$, $64/2$, and so forth are all rational numbers. (By $1/2$ and so forth, we mean the everyday 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.

Part One: Your class should have two instance variables of type `int`. Call the class `Rational`. Include a constructor with two arguments that can be used to set the instance variables of an object to any values. Also include a constructor that has only a single parameter of type `int`; call this single parameter `wholeNumber` and define the constructor so that the object will be initialized to the rational number `wholeNumber/1`. Also include a no-argument constructor that initializes an object to 0 (that is, to $0/1$). Note that the numerator, the denominator, or both may contain a minus sign. Define methods for addition, subtraction, multiplication, and division of objects of your class `Rational`. These methods should be static methods that each has two parameters of type `Rational` and return a value of type `Rational`. For example, `Rational.add(r1, r2)` will return the result of adding the two rational numbers (two objects of the class `Rational`, `r1` and `r2`). Define accessor and mutator methods as well as the methods `equals` and `toString`. You should include a method to normalize the sign of the rational number so that the denominator is positive and the numerator is either positive or negative. For example, after normalization, $4/-8$ would be represented the same as $-4/8$.

Hint: Two rational numbers a/b and c/d are equal if $a*d$ equals $c*b$.

Part Two: Add a second version of the methods for addition, subtraction, multiplication, and division. These methods should have the same names as the static version but should use a calling object and a single argument. For example, this version of the add method (for addition) has a calling object and one argument. So

`r1.add(r2)` returns the result of adding the rationals `r1` and `r2`. Note that your class should have all these methods; for example, there should be two methods named `add`.

Part Three: Add another version of the methods for addition, subtraction, multiplication, and division. These methods should not have the same names as the static version (do you know why?) and should use a calling object and a single argument. The methods should be **void** methods. Method names should be `plus`, `minus`, `times`, and `over`. The result is given as the changed value of the calling object. For example, this version of the `plus` method (for addition) has a calling object and one argument. Therefore, `r1.plus(r2);` changes the values of the instance variables of `r1` so they represent the result of adding `r2` to the original version of `r1`.

Q2. Write a Calculator User Interface as Follows:

The program should be a menu driven program that works infinitely or tell a user requests quitting. Main menu should list the possible types of numbers and the quitting option (i.e. 1.Add, 2.Subtract, 3.Multibly, 4.Divide, 5.Quit) Under each of the first four options a submenu containing the type of the number (Real or Rational) should be provided. The program should be fully implemented having a clear dialogue with the user and providing results in a nice format (i.e. $4 + 3 = 7$, not only 7).

DELIVERY GUIDELINES:

1. Fully working softcopy version should be submitted in WebCT.
2. Report containing the problem definition, screen snapshots, and how valuable you found the Quiz in learning Java.
3. A test class must be provided to *each* Part of Q1.
4. Plagiarism will lead to F in the whole lab grade.
5. You have to drop by my office to discuss your working code.
6. [further modifications might be announced, keep in touch!]

~ Good Luck! ~