**Java**

**Rational Numbers**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**You are going to make a rational number calculator. First make a class called RationalNumber with these things in it:**

**Instance Variables:**

**private int numerator -** the numerator of the rational number

**private int denominator -** the denominator of the rational number

**Methods:**

Constructors:

**RationalNumber()** - Creates a RationalNumber with the value 1/1

**RationalNumber(int numerator, int denominator)** - Creates a

RationalNumber with the value numerator/denominator if denominator is nonzero.

Otherwise creates a RationalNumber with the value numerator/1

**RationalNumber(int numerator)** - Creates a RationalNumber with the value

numerator/1.

The standard ones like the methods we have done before:

**public int numerator()** - returns the numerator of the RationalNumber

**public int denominator()** - returns the denominator of the RationalNumber

**public void setNumerator(int numerator)** - updates the numerator of the RationalNumber

**public boolean setDenominator(int denominator)** - if denominator is 0 returns false, otherwise updates the denominator of the RationalNumber and returns true

**public double getDecimalValue()** - returns the value of numerator/denominator

as a double

**public String toString()** - returns a string in the format "1/3" if denominator is

not 1, otherwise returns a string in the format "4" (just the numerator). The result is in

lowest terms.

**public boolean equals(RationalNumber other)** - returns true if this has the same value as other, otherwise returns false.

For the operations there are two of each. These are called overloaded methods, it is fine to have two methods with the same name as long as the parameters are different. Just like the constructors. Don't worry about the lowest terms part yet, wait until after you finish reduce().

**public RationalNumber plus(RationalNumber other) -** returns a new RationalNumber with the result of adding this to other. The result is in lowest terms.

**public RationalNumber plus(int numerator, int denominator) -** returns a new RationalNumber with the result of adding the value of this to numerator/denominator. The result is in lowest terms.

**public RationalNumber times(RationalNumber other) -** returns a new RationalNumber with the result of multiplying this to other. The result is in lowest terms.

**public RationalNumber times(int numerator, int denominator) -** returns a new RationalNumber with the result of multiplying the value of this by numerator/denominator. The result is in lowest terms.

**public RationalNumber minus(RationalNumber other) -** returns a new RationalNumber with the result of subtracting other from this. The result is in lowest terms.

**public RationalNumber minus(int numerator, int denominator) -** returns a new RationalNumber with the result of subtracting numerator/denominator from the value of this. The result is in lowest terms.

**public RationalNumber dividedBy(RationalNumber other) -** returns a new RationalNumber object with the result of dividing this by other. The result is in lowest terms.

**public RationalNumber dividedBy(int numerator, int denominator) -** returns a new RationalNumber with the result of dividing the value of this by the value of numerator/denominator. The result is in lowest terms.

This is a private method. It is just used by other methods of this class to help you organize your code better. It is not accessible outside of the RationalNumber class:

**private void reduce()** - updates this to be in lowest terms.