RationalNumber Challenge
COSC 3327 Algorithms and Data Structures
Due Dates: See Canvas Assignments

The set of **rational numbers** consists of all quotients of the form a/b, where a and b are each an integer. So, the following are all rational numbers: 3/2, 5/23, 7/22, -9/9, 6/4, 14/1. The following are not: $\pi/3$, the square root of 2, and the number *e*.

The operations on the rational numbers that we will be concerned with are:

- Addition: a/b + c/d = (ad + bc)/bd
- Subtraction: a/b – c/d = (ad – bc)/bd
- Multiplication: (a/b)*(c/d) = (ac/bd)
- Division: (a/b)/(c/d) = (ad/bc)
- Negation: -(a/b) = (-a/b)
- Equality: Left to student

The **reduced form** of the rational number a/b is defined to be c/d provided:

- a/b = c/d
- c and d have no common positive integer divisors other than the integer 1
- d > 0

So, for example, 1/2 is the reduced form of 3/6 and 4/7 is the reduced form of 24/42.

We'll learn later how to write a better specification (and what the issues are with the specification below), but for now we'll define our rational number abstract data type **very informally** as follows:

Abstract Data Type RatNum

Distinct instances: all rational numbers

Construction: create(a, b) creates the rational number a/b

Functions:

getNumerator(R)
Returns the integer value of the numerator of the reduced form of R
getDenominator(R)
Returns the integer value of the denominator of the reduced form of R
getValue(R)
Returns the real value of R
toString(R)
Returns the string "a/b", where a = getNumerator(R) and
b = getDenominator(R)

Operations:

```
equal(RatNum r1, RatNum r2)  //Returns true if and only if r1 = r2 as
                             //rational numbers
add(RatNum r1, RatNum r2) //Returns the RatNum r1 + r2
subtract(RatNum r1, RatNum r2) //Returns the RatNum r1 –  r2
multiply(RatNum r1, RatNum r2) //Returns the RatNum r1 * r2
divide(RatNum r1, RatNum r2) //Returns the RatNum r1 / r2
```

negate(RatNum r1) //Returns the RatNum –r1

Notice that getNumerator(create(2,4)) = 1 and getDenominator(create(2,4)) = 2.

Your assignment is to create three properly documented constructs in Java:

- a RationalNumber interface (see below) which models the Functions of the RatNum ADT above (just copy the code below; leave the name of the interface, *RationalNumber*, alone). **You will not be submitting this file – I will use my own.**
- a RationalNumberImpl_*LastName* class (e.g. RationalNumberImpl_Kart) which implements the RationalNumber interface
- a RationalNumberUtils_*LastName* class (e.g. RationalNumberUtils_Kart) which contains all of the operations of the RatNum ADT above

Java Interface

```
public interface RationalNumber
{
	//rv is the numerator of the reduced form of this rational number
	//Ex: since 5/3 is the reduced form of 10/6, (10/6).getNumerator() = 5
	public int getNumerator();

	//rv is the numerator of the reduced form of this rational number
	//Ex: since 5/3 is the reduced form of 10/6, (10/6).getDenominator() = 3
	public int getDenominator();

	//rv is the double equivalent of this rational number
	//Ex: (5/10).getValue() = 0.5
	public double getValue();
}
```

Java Utils Class

```
//rv = r1+r2, where + is regular numerical addition
public static RationalNumber add(RationalNumber r1, RationalNumber r2)

//rv = r1-r2, where - is regular numerical subtraction
public static RationalNumber subtract(RationalNumber r1, RationalNumber r2)

//rv = r1*r2, where * is regular numerical multiplication
public static RationalNumber multiply(RationalNumber r1, RationalNumber r2)

//rv = r1/r2, where + is regular numerical division
public static RationalNumber divide(RationalNumber r1, RationalNumber r2)

//rv = -r1, where - is regular numerical negation
public static RationalNumber negate(RationalNumber r1)
```

Deliverables

- A .zip file containing RationalNumberImpl_*LastName*.java and RationalNumberUtils_*LastName*.java uploaded to Canvas:
  (Look for "Rational Numbers" assignment or similar)

Rules

- **USE THE PACKAGE 'rationalnumbers' for all of your files!**
- Ensure that I, with only modest effort, can understand your code
- Ensure that the code is properly documented
- Ensure that the code is properly formatted
- Your RationalNumberUtils methods should not change the rational numbers r1 and r2. For instance, this implies that for any RationalNumber r3, the values of RationalNumber.equal(r1, r3) and RationalNumber.equal(r2, r3) should not change after the call RationalNumberUtils.add(r1, r2).
- Test your code! (What test cases can you think of? [Straightforward, extreme, bizarre/exotic])
- Test your code some more! (What other test cases can you think of?)
- Code that doesn't compile will not pass any tests and receive a score of 0
- **Ensure that your files follow the naming convention under Deliverables**
- **WARNING: This specification may be misleading or incomplete! Part of the assignment is to read the assignment early, think about it, and ask any clarifying questions next class!**