Object-Oriented Programming: Using Objects

Assignment II: Rational Arithmetic

In this assignment, we shall look at a possible representation of rational numbers in Java using objects. The Java language represents rational numbers using the same representation used for other real numbers. This is the floating-point representation. However, as we may all know, floating-point numbers are quite inaccurate. This means that ½ might actually be represented as 0.49998, which may not be good enough for some applications.

In this assignment, we shall explore a way of representing rational numbers using Java objects. For each rational number, we shall represent/store the numerator and denominator as integers. We shall use an object like this:

```
class Rational {
      int a;
      int b;
}
```

to represent a number $\frac{a}{b}$. This declares a new type, `Rational`, to represent our rational number. We can now define operations on data of this kind. In particular, we need to implement rational number arithmetic. We shall limit ourselves to the operations of addition and multiplication. (Note that subtraction is really addition in disguise and is no more complicated. The same can be said for division vis a vis multiplication)
Your first task will be to implement the constructors and methods:

```
Rational();
Rational(int a, int b);
Rational add(Rational x);
Rational mult(Rational x);
int show();
```

The methods should do the following: The constructor, `Rational`, should initialize a rational representation for $\frac{a}{b}$ where *a* and *b* are the integers (if provided – otherwise it initializes to $\frac{0}{1}$), `add` should add the provide rational number to itself and return the sum, `mult` should return the product of itself and the rational argument it is passed, and `show` should print out the rational number (in the form *a*/*b*, *not* as real decimal).
This part should be pretty straightforward. You should not need to reduce the rational numbers to their lowest form. That is, if you add $\frac{1}{6}$ and $\frac{2}{3}$, it is OK to report 15/18 as the answer, rather than 5/6.

The next part of the assignment requires you to compute a close rational approximation to *e*, which is often approximated to 2.17… Note that *e* is actually $1 + \frac{1}{1} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \ldots$ Compute this sum as far as it will go without giving you integer *overflow* problems. In short, implement the static method:

```
Rational e();
```

This method should compute and return *e* using the rational arithmetic methods you wrote above.

I have written for you a demo class, `RationalDemo`, which you can use to test your rational number arithmetic. The `main()` method should work without any modifications. Try not to create more bugs than you already have to deal with. As you may notice this code will not compile until the `Rational` class is defined and its methods implemented. You may choose to define your `Rational` class in this same file but it is smarter to create another file, `Rational.java` for it, so that your only task in `RationalDemo.java` is to implement the method `e()`.

Challenge for the bored: Modify your rational arithmetic methods to always reduce the rational numbers to their lowest form. In other words, write a private method:

```
gcd(int a, int b);
```

in your `Rational` class and use it to reduce the rational numbers. You will of course have to compute the GCD of *a* and *b* in order to reduce $\frac{a}{b}$. Ask me how if you want more hints.

This assignment is due during the double lecture of the week of 10^{th} – 14^{th} April 2017.