Principles of Java Language with Applications, PIC20a
D. Hyde
Spring 2020

UCLA

Homework 3
Due Monday, May 4th, 2020

Problem 1: (Rational)

Write a `class` named `Rational` that represents rational numbers.

Make `Rational` inherit `java.lang.Number`.
`https://docs.oracle.com/javase/9/docs/api/java/lang/Number.html`.

Make `Rational` immutable.

Provide the following constructors

```
public Rational(int numerator, int denominator)
public Rational(BigInteger numerator, BigInteger denominator)
```

where `BigInteger` is `java.math.BigInteger`.

Override/implement

- all `abstract` methods inherited from `java.lang.Number`,
- `equals` and `toString` inherited from `java.lang.Object`,
- appropriate member functions named `add`, `subtract`, `multiply`, and `divide`, and
- the getter methods `getNumerator` and `getDenominator`.

`Rational` must be able to represent an arbitrarily large numerator and denominator.

Two `Rational` objects must be the same if they represent the same number, regardless of how the numerator and denominator were provided. For example,

```
Rational r1 = new Rational(1,2);
Rational r2 = new Rational(-2,-4);
System.out.println(r1);
System.out.println(r2); //output should be the same.
System.out.println(r1.equals(r2)); //output true
```

You may want to use `gcd` of `BigInteger` for this.

Write the factory methods with signature

```
public static Rational intToRational(int num);
public static Rational BigIntegerToRational(BigInteger num);
```

which convert the input (a whole number) into a `Rational` that represents the same number.

`Rational` and its methods must work when the object represents `0`.

In principle, you should handle division by `0` by `throw`ing an exception. However, don't worry about this since we haven't covered this topic yet.