# Introduction to Comp. Sci., Homework 8

Due at 10pm on Monday, April 22

## Readings from Liang

Read chapter 15, "Abstract Classes and Interfaces."

## Optional readings and exercises from HFJ

Read chapter 10 of Head First Java, "Numbers Matter," and do the exercises.

## To be turned in: Rat

Build a class to represent rational numbers, numbers that can be represented as one integer divided by another, nonzero integer. Start with this definition:

```
package hw8;

public class Rat {
    public final int a;
    public final int b;

    public Rat(int aa, int bb) {
        if (bb == 0) {
            throw new IllegalArgumentException();
        }
        this.a = aa;
        this.b = bb;
    }
}
```

This class represents the fraction $\frac{a}{b}$. So, if `a` is $-5$ and `b` is $-10$, the object would represent one-half, because $\frac{-5}{-10} = \frac{1}{2}$. If `b` is 0, the meaning of the object is undefined.

1. Write a `toString` method for `Rat`. Precisely, if `a` is $-3$ and `b` is $-9$, the method should return "`-3/-9`".
2. Write a static method, `gcd` ("greatest common divisor"). Given two `int` arguments, it should return the largest positive integer that evenly divides both arguments. One evenly divides every integer, including zero, so there is a well-defined answer for any two arguments. Your code should handle the case where an argument is negative. The GCD of zero and any other number is the absolute value of that other number. You do not need to handle the case `gcd(0, 0)` if it is not convenient.

3. Replace the constructor for `Rat` with one that divides out common factors of `aa` and `bb` before assigning them to `a` and `b`. It should also ensure that `b` is positive. If `aa` is 12 and `bb` is $-8$, `a` should be $-3$ and `b` should be 2. The number represented is the same ($\frac{12}{-8} = \frac{-3}{2}$), it is just simplified. You should use `gcd` from the previous part.

4. Write an `equals` method for `Rat`.

5. Add a `doubleValue` method to `Rat` that returns the value of the object as a `double`.

## To be turned in: Largest<T extends Comparable<T>>

Write a parametric class, `hw8.Largest<T extends Comparable<T>>`, that has `put` and `max` methods, where `max` returns the largest value that has been passed to `put` so far. If `put` has not been called, `max` should throw an `IllegalStateException`. `put` should take a single argument of type `T`, which will be some class that implements `Comparable<T>` (so that it is comparable to itself).

```
hw8.Largest<Integer> s1 = new hw8.Largest<Integer>();
s1.put(1);
s1.put(2);
s1.put(-3);
Integer n = s1.max();   // n.equals(Integer.valueOf(2))
```

As usual, there are automated tests, this time called `Hw8Test`.