Homework 3 Problem 1

Glen Madsen

February 22, 2018

Problem 1.
1. add, sub, mul, and div all require that "arg != null". This is because all of the methods access fields of 'arg' without checking if 'arg' is null first. But the methods also access fields of 'this' without checking for null; why is "this != null" absent from the requires-clause for the methods?

If 'this' is equal to null then there is no 'this' to call because the object wouldn't exist, it'd be null! An object would have to be created first.

2. RatNum.div(RatNum) checks whether its argument is NaN (not-a-number).
RatNum.add(RatNum) and RatNum.mul(RatNum) do not do that. Explain.

For add, sub, and mul, if any operation is performed on them with a NaN, the result will remain NaN (which is fine for all test cases). For div however, the test cases check for a specific NaN rather than just anything, so for the div method it needs to have a NaN check such that the input NaN value is not modified.

3. Imagine that the representation invariant was weakened so that we did not require that the numer and denom fields be stored in reduced form. This means that the method implementations could no longer assume this invariant held on entry to the method, but they also no longer would be required to enforce the invariant on exit. The new rep invariant would then be:

// Rep Invariant for every RatNum r: (r.denom $\geq$ 0)

List the method or constructor implementations that would have to change. For each changed piece of code, describe the changes informally, and indicate how much more or less complex (in terms of code clarity and/or execution efficiency) the result would be. Note that the new implementations must still adhere to the given spec; in particular, RatNum.toString() needs to output fractions in reduced form.

1. RatNum

2. ToString
3. equals
4. hashCode
These methods need to be changed because they require the numbers to be in reduced form. So instead of assuming they are reduced on entry, the gcd function would have to be called on entry such that they work as intended. They do not modify anything however so gcd would not have to be called on exit.

4. This would allow the numer or denom variables to be altered outside of the constructor. Since the specifications are for an "immuatable rational number" that number would no longer be immutable and the specification wouldn't hold. That is how it would fail

5. Not only are the elements of the constructors immutable once created, but the object does not yet exist as the constructor is generated. So the representation cannot really be checked until after the data has been arranged as expected.