Willy Husted

3/19/13

Ginormint Lab Report

**Problem Statement**

**Overview**

* For this assignment you will develop public class Ginormint, whose objects model arbitrarily large, signed integers. These objects/numbers will have instance methods for adding, subtracting, multiplying, dividing, remaindering, and more. To save you some typing, here's some skeletal code.
* Note that there are some required elements in addition to the five methods just mentioned. Read the code, or, equivalently, the specifications, carefully.

**Ground rules**

* You must implement your program from first principles, e.g., you may not use *java.math.BigInteger*.
* The numbers involved in each computation may consist of millions of (decimal) digits.
* Your unfinished methods **must** *throw new UnsupportedOperationException()* .
* Your *main()* method should implement a kind of command-line calculator; for example, if I were to invoke your program like this *java Ginormint product 135531123487687632457563755619283 857093475579357093874532400394*, it should output the resulting product. The permissable operator-words are *sum*, *difference*, *product*, *quotient*, and *remainder*.

**Major design and implementation issues**

One design issue I faced was simply determining how to represent the Ginormints. It came down to using arrays or strings, and I chose strings because they made more sense to me in my head than visualizing the Ginormints as arrays.

My greatest downfall in designing Ginormint was not looking ahead to multiplication and division, and thus didn’t use binary but decimal when constructing Ginormints. If I were to do this problem again, I would convert Ginormints to binary in the backend of things in order to perform multiplication and division much quicker, without having to do repeated addition and subtraction.

**Problems (bugs, deviations from the specifications)**

The most glaring issues are that the multiplication, division, and remainder methods all throw exceptions.

Subtraction doesn’t work properly when both Ginormints are negative. Addition doesn’t work properly when the first Ginormint is positive and the second is negative. Instead, the difference is found (correctly, but that’s beside the point).

Lastly, my program does very little to check for bad input. The operations are accounted for, but nothing stops the user from putting in characters instead of only numbers when entering a Ginormint.