COMP 250 Assignment 1

Prepared by Prof. Michael Langer

Posted: Thurs. Sept. 20, 2012
Due: Sun, Sept. 30, 2012 at 23:59.

General Instructions

- The T.A.s handing this assignment are Masoud, Danesh, Feras, and Mathieu. Their office hours will be posted on the public web page. Office hours will be in Trottier 3110.

- Use the mycourses discussion boards for clarification questions only. Note that Prof. Langer will monitor the discussion boards up until Fri. Sept 28 only.

- Do not change any of the given code, and add code only where instructed. If you have any doubts, then ask.

- Do not change the package name **(a1posted)**, as this slows down the grading. If you are unsure what a package is, see page 4 of (click here).

- The starter code includes a **Tester** class that you should start with to test that your methods are correct. Your code will be tested using a more elaborate set of test cases. Do ***not*** hand in your **Tester** Class. If you do hand it in, the grader will ignore it.

- You must use Java naming conventions for variable names (click here).

- You must comment your code so that the grader can easily follow what the code is doing. Points will be removed for poor style (improper indentation, non-existing or unhelpful comments).

- Hand in your code (one file only for this assignment) using the myCourses assignment dropbox. Include your name and student ID number at the top of the file and read the comment written there. Also, if you have any issues that you wish the TAs (graders) to be aware of, then include them as a comment at the top of your file.

- **Late assignment policy:** Late assignments will be accepted up to only 3 days late and will be penalized by 20 points per day. If you submit one minute late, this is equivalent to submitting 23 hours and 59 minutes late, etc. So make sure you are nowhere near that threshold when you submit.

Introduction

Computers represent integers as binary numbers, typically using a relatively small number of bits e.g. 16, 32, or 64. Operations are performed on integers using specialized hardware which you will learn about later (COMP 273). In Java, integer type **short**, **int** and **long** use a fixed number of bits and so there are severe limits on the how big these integers can be.

In lectures 2 and 3, we discussed how to represent positive and negative integers in binary. The concepts were similar to representing numbers in base 10. More generally, for any base, we can represent any positive integer *p* uniquely as a sum of powers of the base:

$$p = \sum_{i=0}^{n-1} a_i \ base^i = a_0 + a_1 \ base + a_2 \ base^2 + \ldots + a_{n-1} base^{n-1}$$

where the coefficients a_i satisfy

$$0 \leq a_i < base.$$

Such a number can be represented as a list of coefficients $(a_0, a_1, a_2, \ldots a_{n-1})$. Notice that the ordering of the coefficients is opposite to the usual ordering for reading numbers, e.g. the integer 35461 would be represented as the list (1,6,4,5,3).

In this assignment, you are required to represent and perform standard operations on arbitrarily large positive integers. Java has class for doing so, called **BigInteger**. You will implement your own class that allows you to represent arbitrarily big positive integers. In particular, you will implement basic arithmetic algorithms that you learned in grade school. Your representation will allow you to perform these operations in any base.

You are given a partially implemented **NaturalNumber** class. The class has two private fields: **base** and **coefficients.** The coefficients are represented using the **LinkedList<Integer>** class, with coefficients ordered as described above. The class also contains:

- code “stubs” of the methods that you are required to implement.

- helper methods that are implemented for you and that you are free to use, namely **clone(), multiplyByBaseToThePower()**, **compareTo()**, **toString()**. You are not allowed to modify these helper methods.

- a **Tester** class with a simple example of how the methods that you will implement could be tested.

Your Task

Implement the following methods. The signatures of the methods are given in the file **NaturalNumber.java**. *You are not allowed to change the signatures*.

1) **add( )**
2) **multiply( )**
3) **subtract( )**
4) **divide( )**

Each of these method is worth 25 points, for a total of 100 points. Note that the first one is relatively easy. The second and third are a bit tricky. The fourth one is the most difficult. You need to think carefully about how grade school long division works in order to do the fourth one.

We suggest that you begin by testing your code on numbers that are written in base *10*.

The code should work on any base up to $2^{15} - 1$, though we will only test it on relatively small bases.

You may write your own helper methods, but if you do then you must be sure to document them, so that the TA grading it can easily follow what you are doing. *Please keep in mind that there are a large number of students in the class and so the TA will only have a few minutes to go over your code. If your code is difficult to read (poor formatting, mysterious variable names, poorly commented, etc) then you are likely to be penalized.*

What to submit

The file **NaturalNumber.java** which contains the code provided to you along with the code you have added.

Get started early! Good luck!