CSC 103 Programming Assignment #2 Spring 2022

Due date – **Sun March 20th at 11:59pm**

Each student must do the work on this project by themselves unless they have notified and received permission to work with another by the instructor. You are only allowed to work in pairs or alone. You are not allowed to use a previous student’s work, share a current students work, or find a solution online and use it. Doing so will have consequences far worse than just a zero on this assignment. Taking credit for work you have not done is unethical. Note: making minor changes to an existing solution is not considered doing the work. Any work that ends up in your projects that is not your own, must be cited as such. You will lose credit for parts of your projects that are not yours, but you will not be accused of plagiarism so long as you do not take credit for it. Any plagiarism will result in an automatic zero on this assignment, and possible withdrawal from this class.

**Project # 2 Implementation of UnboundedInt class**

Your task is to create the class that is discussed in chapter 4, programming project 10 on page 249 of your textbook. You must use a linked list of integers to store the numbers. The use of the **IntNode** class (unaltered) that we discussed in class from chapter 4 is required. There is a fresh copy of this class in with this assignment. The **UnboundedInt** class will contain three or four instance variables:

* An integer called **manyNodes** that equals the number of Nodes
* Link to front of list, called **front**
* Link to back of list, called **back**
* Optional – a **cursor** that points to an IntNode within the list
* NO OTHERS

The idea of this class is to allow numerical values(integers) that are of any size and not limited to 32 bits(int) or 64 bits(long) of storage. To do this we will use a linked list of IntNode objects. Each Node will contain an integer value from zero to 999. When values of all nodes are listed together, this will allow us to store values with almost unlimited size (unless computer memory is used up). **NOTE: We can assume that we will only be storing positive numbers for this project.**

Although you can store the representation of your numbers in either way, it makes more sense if you do the lower value terms at the front of the list. For example, to represent the number 12,453,075 you would put a 75 in the first Node, then a 453 in the second and a 12 in the third. This order will help you when you are attempting to add or multiply two numbers of unknown length. Note: each node is assumed to store 3 digits of the full number, so values of less than 100 are still storing place holders.

Example:

front

6

23

447

5

= 6,023,447,005

Your class must have the following methods at minimum:

**Constructor(String)**

This constructor will take a string of digits(no commas) and turn it into an UnboundedInt object (MUST BE STRING INPUT)

**UnboundedInt add (UnboundedInt )**

A method that adds the current UnboundedInt with a passed in one. The return is a new UnboundedInt.

**UnboundedInt multiply (UnboundedInt ) - do this one last!**

A method that multiplies the current UnboundedInt with a passed in one. The return is a new UnboundedInt.

**void addEnd ( int ) -optional method (helpful)**

A method to add a new element at the end of the sequence , used for building up each higher term in a single sequence. (i.e. adding a new IntNode to the linked list)

**UnboundedInt clone( )**

a method that returns a copy of the original structure

**boolean equals ( Object )**

a method that returns true if linked list represents the same numerical number as the input parameter. False otherwise. Overrides method in Object class.

**String toString ( )**

creates a string of all elements in order separated by commas, making sure leading zeros are added when needed. (i.e. 12,005,016 or 34,000 )

Throw an IllegalStateException if the sequence is empty

**void start( ) –optional (useful if you add a cursor variable into class)**

set the cursor to the front of the list

**void advance( ) –optional (useful if you add a cursor variable into class)**

move the cursor along the list

Throw an IllegalStateException if the cursor is null

**int getNodeValue ( ) –optional (useful if you add a cursor variable into class)**

a method that returns the integer value of the Node that is pointed to by the cursor.

Throw an IllegalStateException if the cursor is not pointing to a Node

**String toStringNoCommas( ) –optional (may be helpful)**

same as toString but no commas in string.

For each method added you need to also add the specifications in the Javadoc comments. You can have more methods than this if needed but must document them well.

DO THIS FIRST: Create your constructor and your toString( ) methods first because without these it would be difficult to test any parts of your class.

VERY IMPORTANT: The purpose of this class is to allow us to store numbers that are larger than the standard types. If at any time you are asking user to input integers, storing the entire number as an **int** or **long**, or having your constructor read in something as an **int** or **long** you are doing it wrong. There also exists a class in the java library called **BigInteger** which does some of what I am asking you to do. I do **not** want you to use this, your job is to create the class from scratch. Finally, this is a project asking you to manipulate values stored within linked lists. You should never need to use arrays to store the values from the linked lists. If you do so, I will take points off. (Moving a number stored in a linked list temporally into an array to do calculations is an inefficient use of our processor)

Lastly, you must create a Test (driver) class that allow the user to create large numbers and test your arithmetic using your class.

The Test program will ask the user to input, without commas, two large numbers. You will read them in as strings and then use your constructor to make then into stored **UnboundedInt** objects. Then the program will have a menu output to the screen with the following choices:

1. Display both numbers (with commas)
2. Input two new numbers (without commas)
3. Check if numbers are equal
4. Report the sum of the two numbers
5. Report the multiplication of the two numbers
6. Create and output the clone of the first number
7. Quit

These choices will allow the user and you to test your class well. After each choice is finished the menu should pop up again to reshow the choices.

General rules for programs:

* Running your programs should be obvious- I should be given clear directions for any inputs the programs is waiting for.
* Make sure you use comments, some of your grade will depend on program style. You could lose points for things like not indenting, or naming variables in non-descriptive ways. Do no leave in debugging code, or commented out code.
* Cite any help you received and be specific on the portions of the project helped and the location of the sources.
* Make sure your name and is included at the top of all files submitted.
* Javadoc comments must be included for all methods in the ADT.
* I use jGrasp and a recent java version. So make sure that your programs work with this even if you do most of the programming using another IDE.
* Test your own projects thoroughly before you hand them in.
* Late projects will not be accepted so plan ahead.
* All students in a group need to submit the same set of files.
* Any exceptions that are possible to be thrown from the class must be caught in the driver class. There should not be any occasion where the class crashes and ends the program.
* Javadoc comments must be throughout your ADT class (UnboundedInt) but is not needed in the driver class. (But regular comments are still needed there!)

Each of these classes must be done in a separate file. Name them **UnboundedInt.java,** and **LargeNumberTest.java**. (**IntNode.java** should be there as well – but not altered). If you do not name these files correctly, you will lose points. Use Javadoc to create the documentation for your ADT class (UnboundedInt.html). Javacdoc comments are not required for your test(driver) program.

Submission details:

If students got permission to work as a group, each student must submit the assignment. If students within a group hand in different files, I will grade only the first one that I see from you or your partner. You will both receive the same grade, see below for submission instructions. Blackboard must be used, I will not accept email submissions!

In Blackboard:

1. Make sure that you have created and tested all files to submit: **UnboundedInt.java**, **LargeNumberTest.java**, and **IntNode.java**
2. Using the **zip** utility on your computer place all four files into a zipped folder called Project2\_lastname.zip (using your real lastname)
3. Open Blackboard for this course
4. Click on Programming Projects (left side of screen)
5. Click on correct Project to submit
6. Click in Attach files box – Browse my computer (Click “Start New” first if this is not your first submission)
7. Locate zipped folder and Attach
8. Click Submit on bottom right
9. You will get a confirmation e-mail
10. You may submit as often as you want, I will only grade the last submission that happens before the due date/time.