**CSE 274**

**Homework #1**

**Big Integer**

**Objectives.** Create a ADT, use exception handling, and perform unit testing.

**Description.** In this assignment you will implement an ADT that will replicate some of the capabilities of Java’s BigInteger class. This is NOT a research assignment; you are not to search for, look at, or use any source code except for that created by yourself. You may not use any Java classes such as BigInteger, BigDecimal, etc. You may use generic container classes such as ArrayList.

**Class definition.** In the future, you may be asked to design the interface, or may contribute to the interface’s design. For this particular homework assignment, you are given the required methods (see below). You may add other private helper methods if you wish.

// Your class should behave as Java's BigInteger class does. The majority of

// the methods can be studied using Java's documentation.

public class MyBigInt {

public MyBigInt () // **(5)**

public MyBigInt(long value) // **(5)**

public MyBigInt(String str) // **(5)**

public String toString() // **(5)**

public MyBigInt add(MyBigInt other) // **(20)**

public MyBigInt subtract(MyBigInt other) // **(10)**

public MyBigInt negate() // **(5)**

public int compareTo(MyBigInt other) // **(5)**

public MyBigInt max(MyBigInt other) // **(2)**

public MyBigInt min(MyBigInt other) // **(2)**

public int signum() // **(2)**

public static MyBigInt valueOf(long value) // **(4)**

}

**Notes.**

* Use the exact spellings (including the case of letters) shown in the class definition above.
* Provide JUnit tests to exercise your all the constructors, add, and subtract. Of course, you should test all your methods and JUnit is one way to do that. However, the grader will examine only your JUnit tests for the constructors, add, and subtract.
* When creating your JUnit tests, write the test with the idea that you were testing another person’s MyBigInt class and you are highly motivated to identify any errors present.
* Your class definition will be tested with positive and negative big integers. You should implement a solution for positive values first. Then, add the code to handle negative values.
* You should strive to reuse method whenever possible, even at the expense of efficiency.
* The functionality of your code should mirror Java’s BigInteger.
* The development approach is important. You should work in small steps. One such approach would be:  
  + Phase 1 - Constructors. toString. valueOf. Handle positive values at this stage.
  + Phase 2 - Implement add.
  + Phase 3 - Extend Phase 1 methods to allow negative values.
  + Phase 4 - Implement compareTo, max, min, signum to work with positive and negative values.  
    Phase 5 - Extend add to handle negative values.
  + Phase 6 - Implement subtract and negate.

The idea is to implement, test, and have confidence in your work before moving on to the next phase.

**Requirements and point distribution.** Grading will be based on the following point-distribution:

*70 Correctness of MyBigInt*

*15 JUnit tests.* See above for the tests that need to be performed.

*8 JavaDoc.* Provide JavaDoc comments for all of your MyBigInt methods. Generate the documentation before submitting.  
 **Extra Credit.**

*7 Programming style* – Your code must adhere to good practices such as: using good variables, using consistent indentation,  
 declaring data members private, etc.

**What to turn in.** Submit a zipped up version of your project folder. In addition include a PDF file that contains a brief paragraph describing what aspects of your program works and what does not. This could look like one of the following:

* “All aspects of aspects of my code works.”
* “All aspects of aspects of my code works for positive big integers. Negative values do not.”
* “All methods work, except for the following: multiply, compareTo, and equals.”