CS 2401 Assignment #5

Due Date: Monday October 22, 07:00 AM
(See the syllabus for late policy)

Objective: The goal of this assignment is to practice linked list of objects.

Background: El Paso Packaging and Supply Co. comes back to you (again). They are very happy with your linked list based program. They would like to add more linked list functionality in the program.

As you know from Lab Assignment 4, an input file may look like the following one.

```
20 10 8
4.5 8.45 12.2
8.0 2.5 4.0
1.0 15.0 18.0
3.5 3.5 3.5
6.0 5.0 10.0
```

The input filename must be `input.txt`. Each line contains the width, height and length of a box. The dimensions are separated by spaces. In Lab Assignment 4, you created your own linked list; you were not allowed to use any java.util lists for this revised software. The same conditions apply for this new assignment.

Assignment: All the following conditions from the previous assignment are valid in this assignment. You will not make any change to the `Box` class.

1. You are allowed to keep only the `next` variable `public`. The rest of the status variables of the `Box` class must be `private`.
2. Write no more than two constructors in the `Box` class.
3. The `Box` class must have a public method named `getVolume()` that will return the volume of the box.
4. The `Box` class must have a public method named `isCube()` that will return true if the box is cubic, false otherwise.
5. The `Box` class must NOT contain any main method.

Feel free to write any additional method in the `Box` class, as you see fit.

In this assignment, you will write two additional java files `LinkedList.java` and `Runner.java`. Notice that in the previous assignment you coded all the linked list operations, including iteration of the linked list, in the `Runner.java` file. In the new assignment, you will write the basic linked list operations in the `LinkedList.java` file. You will use a `LinkedList` object in the `Runner.java` file as your linked list.

The skeleton of `LinkedList.java` is provided below.

```java
public class LinkedList {
  private Box head;
  private Box iterator;

  LinkedList(){}

  /* Add b as the last node of the linked list.*/
  void addNode(Box b){
      // WRITE YOUR CODE HERE.
  }

  /* Insert b in position pos. If insertion is successful
  * return true, otherwise return false.
  */
  boolean insertNode(Box b, int pos){
      // WRITE YOUR CODE HERE.
  }

  /**Print width, height, length, and volume of each of the boxes in
   * this linked list.   */
  void printAllBoxes(){
      // WRITE YOUR CODE HERE.
  }

  /** Remove the box in position pos. Return true if removal
   * is successful, otherwise return false.*/
  boolean removeNode(int pos){
      // WRITE YOUR CODE HERE.
  }

  /** Return the box in position pos. Return null when pos is
   * invalid.*/
  Box getBox(int pos){
      // WRITE YOUR CODE HERE.
  }

  /**Print width, height, length, and volume of each of the boxes in
   * this linked list in reverse order.*/
  void printReverse(){
      // WRITE YOUR CODE HERE.
  }

  /**Initiate the iterator variable*/
  void initiateIterator(){
      // WRITE YOUR CODE HERE.
  }

  /**
   * Return the box in the current iterator position.
   */
  Box getNextBox(){
      // WRITE YOUR CODE HERE.
  }

}
```

Fill out the methods in the `LinkedList` class. The functionality of each of the methods is provided as comments in the code-skeleton above. Note that, for `LinkedList.java`, you must only write inside the given methods of `LinkedList`. You are not allowed to write anything outside the body of the given methods of `LinkedList`.

From `Runner.java`, whenever you need to iterate over the `Box` objects of the `LinkedList` object, you will use the method **`getNextBox`**. Notice that you will need to call **`initiateIterator`** from `Runner` before you start iterating. This is because **`getNextBox`** will use the status variable `iterator`, which is initiated by method **`initiateIterator`**.

You will use `Runner.java` to read the input file and construct the `LinkedList` object. The `LinkedList` object must hold all the box objects. **You are not allowed to use the `next` variable of any `Box` object directly in `Runner.java`. That is, the use of anything like `temp=temp.next` in `Runner.java` is strictly prohibited.**

`Runner.java` must provide the following functionalities in separate methods.

1. Read the input text file provided by the client and create a `LinkedList` object. Show that your **`printAllBoxes`** method in the `LinkedList` class can print all the boxes in the original sequence they were written in the input file. Also demonstrate that the **`printReverse`** method in the `LinkedList` class prints the boxes in reverse order of their appearance in the linked list.
2. Find the smallest box in the `LinkedList` object. Report the position, dimensions, and volume of the smallest object. Position of an object in a linked list is equivalent of index in an array. You must call the method **`getNextBox`** of the `LinkedList` object to iterate over the linked list.
3. Find the largest box in the `LinkedList` object. Report the position, dimensions, and volume of the largest box. You already know what a position in a linked list means. You must call the method **`getNextBox`** of the `LinkedList` object to iterate over the linked list.
4. How many cubic boxes are there in the linked list? Notice that this will require iteration.
5. Demonstrate that your **`removeNode`** method of `LinkedList` object works properly when called from Runner.java.
6. Demonstrate that your **`insertNode`** method of `LinkedList` object works properly when called from Runner.java.

Note: *It is ok to use a one-dimensional array of length three to keep width, length, and height of a box for string-splitting purpose. You cannot use any array of size larger than three. If you use the split method then this array of size three is inevitable.*

Deliverables: Submit three Java files (`Box.java`, `LinkedList.java`, and `Runner.java`) via Blackboard. You have to demo your programs within one week after the due date. Your demo will be based on your last submission in the Blackboard. Your TA will instruct you with further details.