Assignment 4 – Implementing Lists with Performance Requirements

Assigned: Tuesday February 26, 2013
Due: Wednesday March 6, 2013 11:59 PM
Assignment Type: Individual

Problem Description

This assignment focuses on implementing two very different List collections that are specified in a small interface hierarchy. In addition to implementing the functional specification (i.e., the method behavior), you must also meet non-functional performance constraints.

As always, you must closely adhere to the API as specified in the provided interfaces. Deviation from any aspect of the API specified in these interfaces, or the items listed below, will result in a significant deduction of points. Other specific requirements and notes:

- You must not change (add, delete, modify) either interface in any way.
- You must not add any public methods to either the RandomizedList class or the DoubleEndedList class. You are free to add any private methods that you think appropriate, but you can't add any public methods.
- You may add any private fields that you need to both the RandomizedList class and the DoubleEndedList class. Do not add any public fields to either class.
- You may have any number of constructors that you wish, but you must provide a parameterless constructor for both the RandomizedList class and the DoubleEndedList class that correctly initializes an empty list.
- You will need to create additional helper classes (e.g., for iteration, etc.), but they must be implemented as private nested classes inside the RandomizedList class and/or the DoubleEndedList class. No additional top-level classes can be specified.
- You may not use anything from the java.util package other than the Iterator interface, the Random class, and the NoSuchElement class.
- You may not use any pre-built collection (e.g., java.util.LinkedList, or a List collection implemented in a textbook or online). The use of any pre-built collection will result in a grade of zero points. This assignment is all about you learning to build collections from first principles (using arrays and pointer chains).
- The remove method of an iterator must not remove any element of the underlying list but should throw an UnsupportedOperationException.
- The next method of an iterator must throw a NoSuchElementException if there are no more elements in the iteration.
- The only warning that your code is allowed to generate is the warning resulting from casting a newly allocated array to a generic type. You may use the @SuppressWarnings annotation to keep this from being reported, if you wish. No other warnings are allowed.
- You must not put your source code in a Java package.
- Record your name and TigerMail in the provided @author tag.
- Replace the date in the @version tag with the current date each time you work on the file.
- Your submission will be graded against a test suite written by the course staff. If your submission does not compile with the test suite, you will receive a score of zero points for the assignment.
- Your submission must be solely your own work. Discussing ideas and general approaches to a problem is fine, but sharing source code, even small sections of it, is considered a violation of the academic honesty policy.
- You are encouraged to ask and answer questions on Piazza regarding the assignment.

- Good coding style is expected. An excellent discussion on style can be found in the CACM article Coding Guidelines: Finding the Art in the Science by Green and Ledgard (http://cacm.acm.org/magazines/2011/12/142527-coding-guidelines-finding-the-art-in-the-science/fulltext).

Assignment Submission

To submit your solution, you must turn in a single zip file that contains **only** RandomizedList.java and DoubleEndedList.java. If any other files are in the zip file, they will be ignored and not used in grading. You must upload this single zip file to Canvas no later than the date and time indicated. If your submission does not compile with the assignment test suite, you will receive a score of zero points for the assignment.