CPE 102 Project 5
Doubly Linked List and Iterators

Ground Rules

No collaboration is allowed on this program assignment. Your program must be an individual and original effort. Except for any situations explicitly identified in this assignment, if any, you may only receive help from your instructor, TA, or the tutors provided by the Computer Science Department.

Due Date and Submission Instructions

Submit the following files in PolyLearn:

Files: BasicList.java BasicListIterator.java BasicLinkedList.java

Testing With the Provided Test Driver

1. You should develop and use your own tests prior to using the provided test driver.
2. Do not use the provided test driver until your solution is complete and you believe it is correct.
3. Using the ***save-as feature of your browser, not cut-and-paste,*** save P5TestDriver.java (to be published at 6am on the first due date) in the same directory as all of the source files (.java files)
4. Compile the P5TestDriver.java, all of your source files (.java files) and run P5TestDriver. Remember that your code will be graded one of the CSL servers (unix1-unix4) so, to avoid unpleasant grading surprises, be sure to test on one of the CSL servers at before handing it in!

Objectives

1. To understand the implementation details of a doubly linked-list data structure.
2. To understand the implementation details of iterators on data structures.
3. To appreciate the performance characteristics of a linked list data structure and iterators.

Orientation

You will be implementing a linked list class that mimics the design of the LinkedList class found in the Java Standard Library. So that you can complete the assignment in a reasonable amount of time you will only be implementing a subset of the functionality found in the standard library version. Note that some of the methods have an O(1) performance requirement (see specifications). You will also be supporting forward and backward iteration via methods by implementing an interface that extends the Iterator<E> interface from the Java Standard Library. To achieve efficient iteration both forward and backward you will implement the list as a doubly-linked list. Finally, to support the Java for-each statement, you will implement the Iterable<E> interface from the Java Standard Library.

Specification

1. Your source code must meet the Programming/Coding style guidelines required by your instructor.

2. Do not suppress any compiler warnings.
3. Note that there is an existing LinkedList class in the Java Standard Library - ***you may not use it*** or any other data structure from the Java Standard Library in your implementation.
4. Write an interface called ***BasicListIterator <E>*** to this specification.
    1. Notice that this interface extends java.util.Iterator <E> (yes, an interface can ***extend*** another interface).
5. Write an interface called ***BasicList <E>*** to this specification. Recall that making this interface generic will affect many of the parameters and return types of the methods specified in the interface.
    1. Notice that this interface extends java.lang.Iterable<E> (yes, an interface can ***extend*** another interface).

5. Write a class called ***BasicLinkedList<E>*** to this specification.

    1. This class is required to have no more than three instance variables (all of which must be private). This count does not include the instance variable(s) found in the private inner classes you will be writing.
    2. Some methods have performance requirements - be sure to read the detailed specifications!
    3. ***Do not*** use the iterators in your solution to the methods of BasicLinkedList. Such a dependency is unnecessary and will make

development, testing, and debugging more difficult. I ***strongly suggest*** that you implement the iterators ***after*** completely and correctly completing all of the prior steps and testing the functionality of your BasicLinkedList thoroughly.

4. You will need to write a private inner class that implements the *BasicListIterator* interface to support the *iterator*( ) and *basicListIterator*( ) methods.
    1. Note that the *next*( ) and *previous*( ) methods ***must have O(1) performance.*** (We will discuss this in class.)
    2. The ***remove method*** specified by java.util.Iterator should be implemented as an ***unsupported operation*** as specified in the java.util.Iterator Java Documentation.