Doubly Linked List

1 Overview

In this lab we will implement a linked list capable of traversing both forward and backwards between nodes. By utilizing an abstract base classes, we can define a fully featured Java list implementation by providing a handful of central methods. In addition we will continue to practice unit testing in which we write our linked list code to the requirements defined in our tests.

2 Learning Outcomes

By the end of this project students should be able to:

- write generic classes;
- write custom collections;
- utilize abstract classes for code reuse;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

This project will require you to use Java generics, implement linked list logic and use built in abstract classes

- Read Big Java chapters 16 and 18
- Java's collection interfaces have several built in abstract implementations, named with the word Abstract at the beginning of the class name. These classes provide implementations for the majority of methods specified by the interface to reduce the repeating of simple code when creating a collection.
  - List all such abstract interfaces for the List, Queue, Set and Map interfaces.

  – Which would be useful for implementing a linked list? Justify your answer.

4 Lab Instructions

Do this part in lab:

Step 1

This lab requires you to build unit tests for your code as you write it, in the spirit of test driven design. Our first step is to set up our DoublyLinkedList class file as well as a DoublyLinkedListTest class file. Create an empty class for DoublyLinkedList implementing List, and extending the AbstractSequenceList class. Provide stubs for each function, such that it and your test class both compile. Start with a single tests that ensures that you can instantiate a DoublyLinkedList object.

Step 2

Now we will implement our linked list one requirement at a time. Create unit tests to evaluate the following requirements. After adding a test, verify that it fails, and then implement just enough code in your linked list class to get the tests working. Once you verify that all tests pass, ensure that you aren't repeating code, or storing data inefficiently. Refactor if necessary and test again. The following list provides the requirements you must test in a recommended order. Some requirements may require multiple unit tests.

- You must be able to retrieve an iterator object with listIterator(int)
- You must be able to add to an empty list using an iterator
- You must be able to read the first element of a list using an iterator
- You must be able to move forward and backwards through your list using an iterator
- You must be able to add arbitrary elements using an iterator
- You must be able to read arbitrary elements using an iterator
- You must be able to remove elements using an iterator
- You must be able to replace a value in the list using an iterator
- You must be able to determine if you are at the beginning or end of a list using an iterator
- You must be able to determine your current index using an iterator

- You must be able to retrieve iterator objects at a non-zero index from listIterator(int)
- You must be able to correctly retrieve the size of the list after adding and removing elements correctly
- You must be able to use addAll(Collection) to add every item in an existing collection.
- You must be able to read from the list by index
- You must be able to remove from the list by index
- You must be able to add to the list by index
- You must be able to replace an element by index
- Trying to iterate past the end of the list, or previous to the beginning should throw a NoSuchElementException

5 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical. In order to recieve points, you MUST make a BBLearn submission.

Your lab report should begin with a preamble that contains:

- The lab assignment number and all partner names
- Your name(s)
- The date
- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list. It is recomended that you complete this section before you begin coding the problem, as gathering these requirements may help you organize your thoughts before you begin your soulation.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why. How do you plan on breaking up the problems into functions, classes and methods?

3. Implementation and Testing

In the third section you should demonstrate your implementation. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as .java files, should not be in your report document)
- a screen shot of your running application / solution. This should include a screenshot of the output, be it a terminal window or graphical interface
- results from testing. This is specific to the lab, and not every lab will include this element.

4. Reflection

In the last section you should reflect on the project. What part of the projec was the most difficult? Where there other solutions to this problem that you considered? Where there any new solutions you can think of now that you couldn't then? Do you think the way you chose was the best option, or would you go about the problem differently if you had to start over? How might the problem had been broken up into easier problems? If you had one more day to work this problem, what improvements might you make? Every problem has alternative solutions and solution has tradeoffs or improvements, you are required to identify some of these elements.

5. Partner Rating

Every assignment you are required to rate your partner with a rating between 1 and 10. For each patener a name and rating should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating 8 or more indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 5 indcates that your partner met the class expectations of them. Rating your partner 3 or less means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. In the case where you give a very low rating, please explain why in the comment section of the submission. Those who recieve low scores may be asked to explain their actions to the lab staff, and additional low ratings after a warning could lead to losing a letter grade, or even failing the course. Consistant high ratings from partners are noted durring the grading process, and may be taken into account when rounding your final grade.

Colophon

This project was developed by Richard Lester and Dr. James Dean Palmer of Northern Arizona University. Except as otherwise noted, the content of this document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.