Assignment 2 Collections

Objectives

- Practice creating class that *implements* an interface in Java
- Practice using the Node class
- Practice using references
- Practice reading and understanding specifications
- Exposure to more complete testing

Introduction

This assignment introduces the idea of a Container—a data structure that is used to store a collection of items. For simplicity, our containers will store only integers. Once a data structure works with simple integers, it can easily be modified to work with more complex objects.

For this assignment, you will implement a class that implements the completed interface described in the file IntegerList.java. Your source code will implement the Abstact Data Type (ADT) using a doubly-linked list, illustrated below on the next page.

Quick Start

1) Download and store the java files provided.

2) Read IntegerList.java carefully, or run

   javadoc IntegerList.java IntegerArrayList.java

   in the directory where all the java files are stored. Open the file called index.html, by double-clicking on it.

3) Compile and run ArrayIntegerList.java. Study how this works and how it was implemented. Note its relationship to IntegerList.java.

4) Type in the method headers that are in IntegerList.java into IntegerLinkedList.java and create *stub* methods (empty or with a single return statement), until it compiles without errors.

5) Compile and run A1Tester.java.

6) When the test program reports no errors, see the Grading section of this document. Otherwise continue.

7) Implement one of the methods in IntegerLinkedList.java.

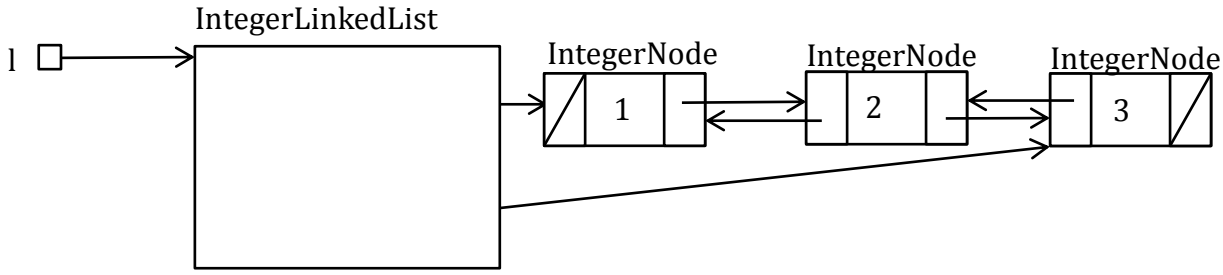
8) Go back to step 3.

Linked Lists

Your implementation must be a doubly-linked list that maintains both a head and a tail reference. The shell you've been provided in IntegerLinkedList.java already includes the appropriate instance variables.

You've also been provided with a node class (IntegerNode.java) that includes a prev and next references.

If your implementation is correct, the in memory picture of the list l containing {1,2,3} will be:



Your implementation of addFront and addBack must be O(1).

Understanding the test program: A2Tester.java

A2Tester.java will test your implementations of IntegerLinkedList.java

Once you have filled in the basic stubs for IntegerLinkedList.java and it compiles, then compile and run the test program.

Compile the test program by typing:

```
javac A2Tester.java
```

Run the test program by typing:

```
java A2Tester
```

You should see the following output:

```
Basic testing of size, addFront, addBack, get
Failed test: 0 at line 53
```

The tester is reporting that your implementation is failing the very first test. This is hardly surprising, since you haven't written any code yet!

The tester is written to be able to test both an array implementation (provided to you in the file IntegerArrayList.java) and the linked list implementation that you are required to implement in IntegerLinkedList.java

To see the result of testing your instructor's array based solution, type:

```
java A2Tester array
```

You will see 21 test cases (numbered 0 to 20) and the array based solution passes them all. Your goal is to make the doubly-linked list implementation pass all 21 test cases.

What to do

The only file you need to modify is IntegerLinkedList.java.

Before writing any code, you must understand the idea of a linked list and the purpose of the node class. There will be examples and code regarding linked lists in the labs and lectures to help your understanding. The textbook also has many examples. When inserting and removing nodes, draw the picture of what is happening to the links in sequence; this helps prevent surprising (and common) NullPointerExceptions.

Implement your code incrementally. Work on one method and test thoroughly before continuing.

In order to pass the first set of cases, the methods size, insertFirst, insertLast and get will need to be completed. Then move on to toString and removeAll. Complete remove after everything else works.

Submission

Submit the completed IntegerLinkedList.java using conneX. **Please be sure you submit your assignment, not just save a draft**.

A reminder that it is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

We will be using plagiarism detection software on your assignment submissions.

Grading

If you submit something that does not compile, you will receive a grade of 0 for the assignment. It is your responsibility to make sure you submit the correct files.

Requirement	Marks
You submit something that compiles	1
The source code follows the coding_conventions posted on conneX	2
1 mark for each test case you pass	Up to 21