Assignment 2

Due: February 19th, 2017 before 11:59pm

Objectives

- Practice implementing an interface in Java
- Practice with a Node class and references
- Practice reading and understanding specification
- 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 only store integers. Later in the course we will encounter a mechanism that Java supports to store items of an arbitrary type: generics.

Your task is to implement the interface described in the file `IntegerList.java` using a doubly-linked list in the file `IntegerLinkedList.java`.

Quick Start:

1. Download `a2tester.java, IntegerLinkedList.java, IntegerList.java, IntegerNode.java and IntegerArrayList.java`
2. Read `IntegerList.java` carefully
3. Compile and run the test program `a2tester.java`
4. If no errors reported by test program, see the Grading section of this document
5. Implement one of the methods in `IntegerLinkedList.java`
6. goto 3

Understanding the test program: a2tester.java

You've been given a program that will test your implementation of the IntegerList interface.

One of the first things you should do after downloading the source code files is to 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 there are 36 test cases (numbered 0 to 35), and the array based solution passes them all.

Your goal is to make your linked list based implementation pass all 36 test cases.

Bonus:

Jason (i.e., instructor for the Tuesday/Wednesday/Friday course section) will give out a chocolate bar for the first student to report:

- a bug that is found in `IntegerArrayList.java`
- an ambiguity in `IntegerList.java`

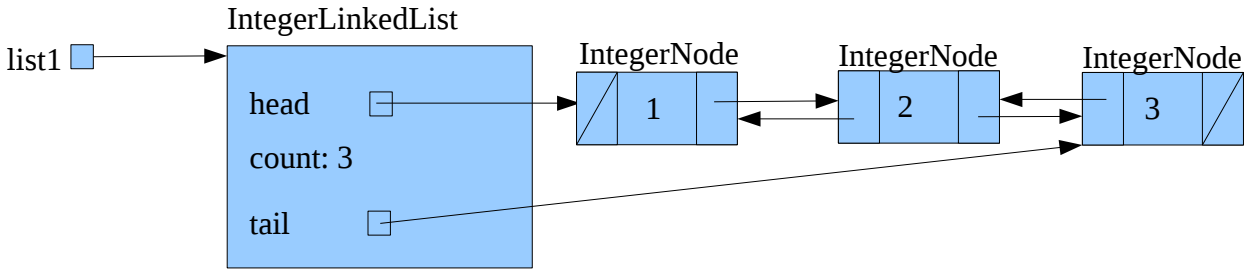
To claim your chocolate bar, send an email to jcorless@uvic.ca with the bug (include a test case that demonstrates the bug) or the ambiguity (include a fix to the specification).

Linked Lists

Your implementation must be a doubly-linked list that maintains both a `head` and a `tail` reference.

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 list1 containing {1,2,3} will be:



What to do

All the coding you have to do in this assignment will be contained in the file named `IntegerLinkedList.java`. This code already compiles and runs, it just doesn't do anything useful without your additional code.

You should be sure that you understand the idea of a linked list and the purpose of the node class before you start writing code. There will be examples and code regarding linked lists in the labs and lectures this week to help your understanding.

It is very helpful to draw pictures of what a list looks like before and after an operation (such as `addFront`). This will assist you in understanding what your intended code (i.e., what you are writing) is trying to accomplish in modifying the structure of some existing list.

You must do your development incrementally, that is, implement one method and then re-run the test program.

- In order to pass the first set of test cases, you will have to implement the methods: `size`, `addFront, addBack`, and `get`.
- Once you've completed those methods, move on to `toString` and `clear`.
- Then implement `addAt` and `removeAt`. These will be the most difficult to complete.
- Finally, implement `remove`.

Submission

Submit your `IntegerLinkedList.java` using Connex. **When doing this, please be sure you complete the last "confirm your submission" step; do not just save a draft assignment submission**.

We remind you that it is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together. However, each student must implement their own solution. Sharing of code is not permitted. (We will use code-similarly software tools on submitted assignments.)

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.

You must implement doubly-linked lists with a head and tail reference. Your implementation of `addFront` and `addBack` must be O(1).

Requirement	Marks
You submit something that compiles	1
Your code passes the first set of test cases: "Basic testing of size, addFront, addBack, get" (tests 0 to 6, inclusive)	3
Your code passes the second set of test cases: "Testing toString() and clear" (tests 7 to 11, inclusive)	2
Your code passes all the remaining test cases (tests 12 to 35, inclusive)	6
Total	12

And to emphasize a point about the above guidelines – in order to obtain a passing grade on the assignment, you must at least satisfy the first three requirements by passing all of their test cases.