Assignment 2

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 programming to an interface and an alternative implementation of the List abstract data type. Your list will build on your knowledge of the `Player` class and allow you to store player objects in your linked list. Your assignment is to implement the interface defined in `PlayerList.java` as a doubly-linked list in the class `PlayerLinkedList.java`.

Quick Start:

1) Download `a2tester.java`, `PlayerLinkedList.java`, `PlayerList.java`, `PlayerNode.java` and `PlayerArrayList.java`

2) Read `PlayerList.java` carefully – this is where the documentation including preconditions and return values for each method are described.

3) `PlayerArrayList.java` is the array implementation of the interface and is provided for your reference. You can compile and run the tester to see how it works.

   a) Make sure `testArraySolution` is set to `true` on line 20 of `a2tester.java`:

   ```
   public static Boolean testArraySolution = true;
   ```

   a) Compile:

   ```
   javac a2tester.java
   java a2tester
   ```

4) Start implementing each function in PlayerArrayList.java. Repeat the following until no errors are reported…

   a) Make sure `testArraySolution` is set to `false` on line 20 of `a2tester.java`:

   ```
   public static Boolean testArraySolution = false;
   ```

   b) Compile:

   ```
   javac a2tester.java
   java a2tester
   ```

5) If no errors reported by test program, see the Grading section of this document

Understanding the test program: a2tester.java

You've been given a program that will test your linked-list implementation of the `PlayerList` interface.

One of the first things you should do after downloading the source code files is to run the test program.

When you compile the test program you should see the following output:

```
Basic testing of toString, size, addAt
Failed test: 0 at line 51
```

The tester is reporting that your implementation is failing the very first test, 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` . This is the benefit of programming to an interface – a clear separation of what is being done from how it is being done.

Linked Lists

Your implementation must be a doubly-linked list that maintains both a head and a tail reference. The stub code you've been provided in `PlayerLinkedList.java` already includes the appropriate instance variables. You've also been provided with a node class (`PlayerNode.java`) that includes `prev` and `next` references.

Tips

Start EARLY!!!

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.

Start EARLY!!!

It is very helpful to ***draw*** pictures of what a list looks like before, during after an operation to help you understand what the code you are writing is trying to accomplish.

Start EARLY!!!

Develop incrementally. Implement one method and then re-run the test program.

Submission

Submit only your `PlayerLinkedList.java` using conneX. **Please be sure you submit your assignment, not just save a draft**. Submit as many times as you want, your last submission will be the one graded.

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	10
1 mark for each test case you pass	70
Style	20
Total	**100**