CS212: Paradigms
Dr. Gross, Spring 2016

Lab 7 - Linked Lists

1. Create a new package called edu.blackburn.cs.cs212sp16.linkedlist.lastnamelastname
2. Copy the Account class from the Lab07 directory, and change the package name if necessary
3. Make a Runner in this package
4. Create a main() method in Runner

Due Thursday, March 24 before lab

Create Your Initial LinkedList and ListElement Classes

You will create a LinkedList class and a ListElement class in your package. Only LinkedList will ever use ListElement. I know I told you not to use a specific type for inputs and outputs in a data structure, but I'm going to teach you the wrong (and easier) way and leave that problem for Dr. Coogan to clean up next spring.

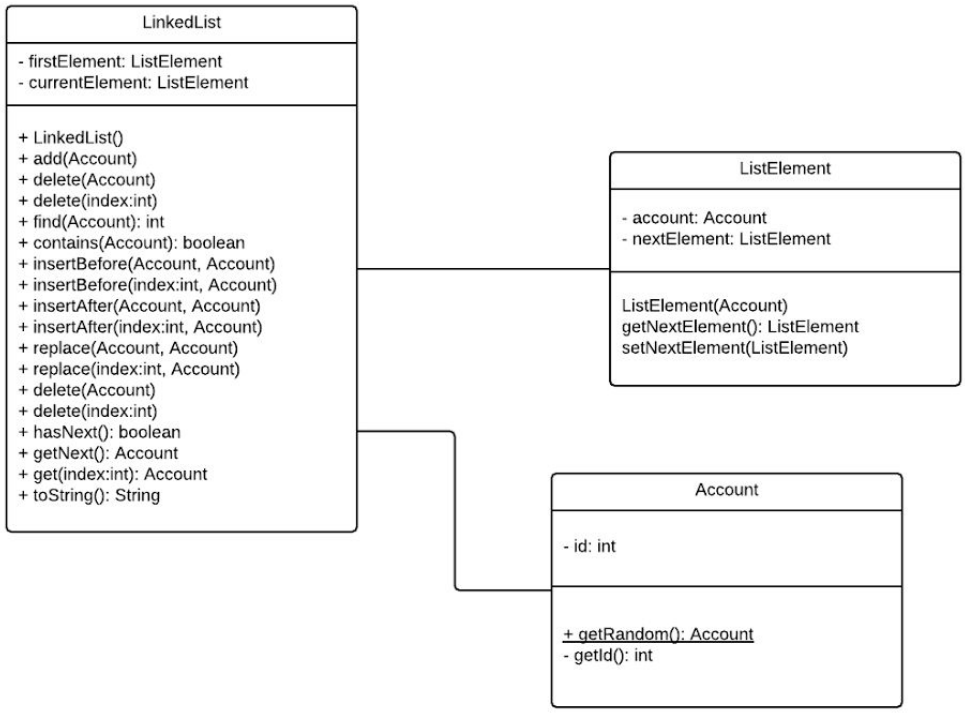

Figure 1. Class Diagram for Account and LinkedList

Yes, I'm giving you the design. Aren't I nice?

Account has just two methods: a static method called getRandom() that returns an Account object, and a getId() method that returns the account's ID. What kind of Account is it? The kind that has an integer ID. A positive integer id.

Internally, the *class* Account maintains a list of Account objects it has created. When you call getRandom(), it will return a new Account object 2/3 of the time, and an existing one 1/3 of the time. This isn't so important now but it will be later.

Your initial version of LinkedList will need to have the following methods completed:

```
LinkedList() //constructor
add() // add new Account to the end
hasNext() // determine if there's a next element, used while looping through
getNext() // return the next element, used while looping through
```

Obviously you'll need an internal reference for the first object and another to the current object (used while looping through) but that's the only ListElement that the LinkedList will ever know about it. You will need to figure out where to start that first element and what to do when you get to the end of the line.

Now, using a loop, test out adding elements and looping through your LinkedList. Once you know those are working, go on.

Contains and Find and Get

The methods contains() and find() will need to internally loop through the LinkedList (without disturbing the currentElement used for hasNext() and getNext()!) to determine, respectively, whether or not a specific Account object is already in the LinkedList and what the position of the that Account object is, if it exists (what should find() return if the Account is not in the LinkedList?).

The get() method simply returns the Account at the position of the index passed in (what should it return if the LinkedList doesn't have that many elements?).

Add more tests to Runner to show that this code works. You'll need to use another loop and print out the entire LinkedList again.

Delete

There are two delete() methods; one that takes an Account object, and the other than takes an index.

Add more tests to Runner. You should delete Account objects using both methods at the beginning, middle, and end of the LinkedList, and you should also delete Account objects that aren't in the LinkedList and indexes bigger than the LinkedList. (If you want to make a getSize() method, that's fine).

Inserts: Before and After

As above, there are two variants for both insertBefore() and insertAfter(). One each finds the place to insert with an index, and one each finds the place to insert with an Account object. Yep, you'll need to test this at the beginning, the middle, and the end, and probably repeatedly print out the whole LinkedList and show what you expect vs. what you get. You'll also need to handle the error situations discussed above. In all cases, the second argument is the Account to insert.

Replace

This is very similar to the Inserts. Again, test at multiple areas and with bad inputs. Again, the second argument is the Account to replace the old account with.

Nearing Done

Now we need to make this a bit more annoying. LinkedList now can only hold one instance of each unique Account object. If you try to insert, replace, or add an Account object that's already in the LinkedList, you'll need to remove it from its old spot.

Finally, read through the rubric and make sure you aren't missing anything!

Rubric

Standards/comments	10
Time estimate/accounts	5
All methods implemented	10
Clear tests for each method, documented in output and comments	10
Tests explain what is expected and how to read the output	10
All methods implemented correctly (not counting autoreplace)	25
Exceptions generated and handle as appropriate	10
All other error conditions accounted for and properly documented	10
Autoreplace working for all necessary methods	10
Total	**100**