**ListOfNotes Lab**

**ArrayLists**

## Objectives:

To get practice with collections and loops.

## Directions:

1. Copy the ListOfNotes class into jGrasp. (DO NOT USE JGRASP, USE ECLIPSE. Note: this class is included in this repository)
2. Take a moment to familiarize yourself with the Java code in the ListOfNotes class. The collection of notes is represented by an ArrayList of Strings (created in the constructor), and there are methods that add Strings to the list, remove Strings, and a loop that displays all of the Strings in the collection. Create an instance of ListOfNotes and experiment with its methods to get a feel for how it works.
3. The storeANote method always adds new items at the *end* of the list. Users may prefer to add new items to the beginning of the list instead — especially if the notes are Important. Add a new method called storeImportantNote that works just like storeANote, but adds the new note to the front of the collection. It should prepend "Important: " to the front of the note string before adding it, to make clear that this is a high-priority item, as illustrated in the **partial** tester statementsbelow.

> ListOfNotes book;

> book = new ListOfNotes();

> book.storeANote("This is my first note.");

> book.storeANote("Second note.");

> book.printNotes()

This is my first note.

Second note.

> book.storeImportantNote("This is apparently Important.");

> book.printNotes()

Important: This is apparently Important.

This is my first note.

Second note.

1. Positions within an ArrayList are zero based. That is, the first item is at position zero, the next item is at position 1, etc. The removeANote method assumes users will pass in the appropriate zero-based number when specifying which note to remove, but users may not be familiar with this convention. Let's make it easier on the user and show item numbers when *displaying* the list. Change the code in the printNotes method so that it maintains a counter (an integer variable), and displays this number along with the notes as shown below. **Note:** You should *not* alter the basic loop. Instead, add extra statements to the existing loop to implement the numbers.

> ListOfNotes book;

> book = new ListOfNotes();

> book.storeANote("one");

> book.storeANote("two");

> book.storeANote("three");

> book.printNotes();

0) one

1) two

2) three

> book.removeANote(0);

> book.printNotes();

0) two

1) three

1. Write a printImportantNotes method that only displays the Important messages in the list. In other words, it only displays the notes that have "Important: " at the beginning. (There is a startsWith method in the String class that you may find useful.)

> ListOfNotes book;

> book = new ListOfNotes();

> book.storeANote("This is my first note.");

> book.storeANote("Second note.");

> book.storeImportantNote("This is apparently Important.");

> book.storeImportantNote("This is Important too.");

> book.printNotes()

0) Important: This is Important too.

1) Important: This is apparently Important.

2) This is my first note.

3) Second note.

> book.printImportantNotes()

0) Important: This is Important too.

1) Important: This is apparently Important.

6. Write a method called endNote that takes a note number, just like removeANote does. Instead of removing the note from the list, endNote should move it to the *end* of the list. There's no ArrayList method for performing exactly this operation, but you can accomplish it by doing a remove and an add. To keep things simple, you may assume that the user passes in a valid, zero-based note number. Note that if you move an Important note, you must remove the "Important: " prefix from the moved note, if it's present. You can use startsWith to determine if the prefix is present, and substring from the String class to extract everything but the prefix.

> ListOfNotes book;

> book = new ListOfNotes();

> book.storeANote("this is a non-Important note");

> book.storeANote("this note is also not Important");

> book.storeImportantNote("Remember to study for exam!");

> book.printNotes();

0) Important: Remember to study for exam!

1) this is a non-Important note

2) this note is also not Important

> book.endNote(0);

> book.printNotes();

0) this is a non-Important note

1) this note is also not Important

2) Remember to study for exam! // delete Important:

1. Write an exchangeNotes method that takes a note number like endNote does, but *exchanges* the specified note with its successor. In the example below, the note at position 1 is being swapped with the note "below" it:

> ListOfNotes book;

> book = new ListOfNotes();

> book.storeANote("one");

> book.storeANote("two");

> book.storeANote("three");

> book.printNotes();

0) one

1) two

2) three

> book.exchangeNotes(1);

> book.printNotes();

0) one

1) three

2) two

8. You need to write a ListOfNotesTester that tests each method in the ListOfNotes class.