Introduction to Lists

Purpose:
The purpose of this lab is to introduce list container classes.

Containers and lists:
A container is an object whose function is to contain a collection of other objects. A list is a container with the following properties:

- contains any (finite) number of objects;
- contains only objects of one given class; i.e., all objects on the list must be instances of the same class;
- allows access to a contained object by means of an integer index denoting the object’s position on the list.

(We will see later that with sub-classing, the restriction that all objects on the list must be of the same class is not limiting.)

The class RetailItemList:
In this lab, you will use the class RetailItemList. A RetailItemList can contain, surprise, only RetailItems. The class is specified here.

Note in particular the following methods:

- size returns the number of objects on the list. The size of a list can be 0, in which case the list is empty: it contains no elements.
- get returns the object at the specified position (index). Indexing starts at 0. If a list contains 10 objects, the first object is at index 0 and the last object is at index 9.
- append adds an object to the end of the list.
- remove removes the object at the specified index position. This does not leave a “hole” in the list. For instance, if the object with index 3 is removed, the object that was at index 4 “moves up” to 3.

The class Inventory:
To see how a list can be used, we examine a simple class that uses a RetailItemList.

- Copy the files from ~labCourse/Labs/Lab13/inventory/ into a subdirectory of your Java directory named inventory.
- Open and read the definition of the class Inventory in file Inventory.java.
- Open and read the test code in InventoryTest.java.
• Compile and run `inventory.InventoryTest`.

Now you will add two methods to the class `Inventory`. Examining the methods already defined in the class, particularly the method `numberOutOfStock`, should give you clues as to how to proceed.

• First, implement a method `totalUnitsOnHand`. This method returns the total number of units on hand for all retail items in the inventory.

• Second, implement a method `outOfStock`. This method should return a `RetailItemList` containing all of the items that are currently out of stock: that is, items for which there are no units on hand. Note that this method should create a new `RetailItemList`, and append `RetailItems` to it.

• Modify the test code to test your methods.

• Compile and test.

**Post-lab:**

Submit listings of the modified `Inventory` class and modified test code. Submit a script listing showing the results of your test.