Assignment 1 - ITSC 2214

Due October 26th, midnight.

Table of contents

Objectives

At the completion of this assignment, you will have completed an `ArrayList` implementation of a `ShoppingList` interface, and completed unit testing of the implementation. By completing this project, you will have gained experience in understanding the difference between interface and implementation, a critical part of encapsulation. You will get experience writing JUnit test cases for methods, to make sure the methods work as expected. You will also become accustomed to the Web-Cat system for auto-grading of code projects and tests. This system may seem frustrating at first, but in industry you will have to check in code that you write to similar systems that will run your code against a bunch of test cases. You can also observe the use of exceptions, how they are passed and handled and how you work with them when writing test cases.

Instructions

During this assignment, you will be working with an interface for shopping lists. We provide a completed array implementation of this interface, and an incomplete `ArrayList` implementation. Your job is to complete the `ArrayList` implementation. Both implementations adhere to the `ShoppingListADT` interface. The shopping lists hold Grocery items, and there is a `Grocery` class that defines what a Grocery item is. While duplicate grocery items are not allowed in the list, the quantities of a grocery item are combined when an attempt is made to add a duplicate `Grocery`. Through the `Grocery` class's `compareTo` method, entries are determined as duplicates if their name and category match, **regardless of letter case (ie. "bread" is the same as "Bread")**.

Requirements

You must fulfill the following requirements in your implementation.

1. Complete the three empty methods (`remove()`, `find()`, and `contains()`) in the `ShoppingListArrayList.java` file. These methods are already implemented in the `ShoppingListArray` class.
2. Ensure all methods in both implementations have appropriate JavaDoc comments.
3. Generate the JavaDoc html files.
4. Complete the `ShoppingListArrayListTest.java` (For many tests, we now just throw a false statement which will cause the test cases to fail. You need to complete them.)

Submit your project to Web-Cat.

Files needed

The files needed to get started are available on Canvas inside of a zipped project called `DS_Assignment1_Shopping.zip`. Download this, but don't unzip it. Import the project into NetBeans. In your NetBeans, you will get a project called `Shopping`. When you first import the project, it will run using the array implementation of `ShoppingList`. After completing the methods described in the first requirement, the project can be run using the `ArrayList` implementation. Commented code is provided for switching to this version of the simulation in `ShoppingSimulation.java`.

What you must test

Think of your shopping list as having "states." When your program starts and you create the shopping list, it is in its initial state. At this state, the shopping list has nothing in it, so size should be 0. It should also not allow you to remove any element (since it is empty). Once you start adding elements, then the shopping list is not empty anymore and with every element you add, the shopping list's size should reflect one more and you should be able to find the element within it. As you remove elements, the shopping list size should decrease and you will no longer be able to find elements that have been removed.

In the function that you implement, make sure that you can handle special cases. For example, can you double-remove an entry? Can you remove an entry that does not exist in the `ArrayList`?

A test file for the `ArrayList` implementation is included in the project. Some method tests are provided, but you must implement the tests that are empty. You can run your tests by right-clicking on either the class or test file in NetBeans and selecting 'test file'.

> **Code coverage note**: Web-cat will run the JUnit tests that you submit and will basically count the percentage of lines of code in your classes that get executed as a result of your test cases. So, you want to try to write test cases that cause all lines of code in all branches of your code, to be executed. (So, if you have an if-else statement, write code in your test case that runs the branch of the if statement when the expression is true, and write other lines of code in your test case that will cause the else clause part of the if statement to be run).
>
> **ShoppingSimulation note**: The project contains a class called `ShoppingSimulation.java`. This simulation class also has the main method that executes when you run the project. You can run the simulation using the `ShoppingListArray` class or the `ShoppingListArrayList` class, depending on which line of code you leave uncommented in the main method. It is important for you to understand that this simulation class is there to give you a sense for how the other classes could be used in a real application. When you get everything working, this simulation will run. But just because this simulation runs, this does NOT mean your code is working perfectly. This simulation does not thoroughly test your code at all! You need to write good test cases to make sure your code is working. Also, note that you don't need to write a test file to test this simulation.

Submission

- Submit your assignment to Web-Cat.
- Submit Honor Code declaration to Canvas. More on the same later in this document.

Grading

1. TA Grade on quality of your JavaDoc and algorithmic clarity (20%)
2. Web-cat style testing* (20%)
3. Web-cat unit testing, code coverage, and instructor generated test cases (60%)

Honor Code

Sign and submit the following declaration on Canvas. (Your code is submitted on web-cat, but this signed document must be submitted on Canvas – printing it, signing it and uploading a picture of it is sufficient). Copy-paste the declaration to Word and digitally sign it, or print it out, sign it, and upload a scan of it.

Without this declaration, you will not receive a grade for Assignment 1.

```
I promise that I completed this assignment on my own and did not look at, borrow from or copy assignment code from a classmate or anyone v

Describe any help you received on this assignment (from a professor or TA or the tutoring center):



Signature:                                          Date:
```