CS5004-5005-SV – Lab 11 – July 16, 2020
ArrayLists, For-Each, and Exception Handling

This week's lab provides an opportunity to explore and practice ArrayLists and Exception handling. For each **question** in the steps below, add the question and answer as **comment lines** in your code file. (You may find it helpful to use **jshell** to explore some of the ArrayList behaviors.)

0. Make a copy of the **ArrayListDemo.java** file from the lecture slides. (Having to re-type this code is actually good practice and helps memory retention.) Include additional print statements to show the **size** of the list. Add a traditional **for** loop with a **counter** to show the **index** of each element as well as the element itself, when printing the elements (but keep the for-each loop, for comparison -- it is OK to let this sample program print everything twice). Q1: When is a traditional for loop better? Q2: Why use the for-each construct instead of regular for loop?

1. Try using the get(index) method to obtain an element that is at a position higher than you have entered but lower than the initial capacity of the ArrayList. Q1: what was the result? Q2: Is **size()** the same as **capacity**? Q3: Is there a way to look up **capacity** without violating source code privacy? Q4: Since ArrayLists automatically re-size, anyway, why bother providing an initial capacity?

2. Try changing the **main** method to use the **add** method to insert elements at the *front* of the list instead of at the end. (Comment out the code you are changing, do not delete it.) Your existing print statements should demonstrate the result. After completing, comment this change out (don't delete) and restore the original version -- after testing it.

3. Now use the **set** method to manually **set** a few String elements in the ***middle*** of the list, between the two versions of your **for** loop that do the printing, so that you can easily see the effects. You can comment these out after testing.

4. Create a try block and handler for **IndexOutOfBoundsException** occurring when your main program attempts to **get** an element from the toDoList that is larger than the current **size** of the list. It should "do

something reasonable," such as printing a polite warning message (followed by the error instance object on the next line) but then go ahead and return the highest numbered ***actual*** element, if any. *However,* if the calling program requests an index less than 0, or there are no elements at all, then your catch block should just ***throw*** the error upward. Q1: Could you accomplish the same behavior, without using a **try** block? Q2: Since your code may throw an exception, even though you have a **catch** block, do you need a "throws" declaration here? (Why or why not?)

5. Make a new version of your program, ArrayListDemo2, by making a copy and renaming the class and its filename. This one should user a Scanner to read the toDoList elements from a text file, instead of from the keyboard. There should be one toDoList String per line in the file. Use the **try-catch** method to handle the case where the file is not found. Your program should give the user 3 chances to provide a valid, existing filename within the current directory before propagating the exception upward. Q1: Since it is still possible to eventually throw a **FileNotFoundException**, even though you have a **catch** block for it, does that mean you need a "throws" declaration here? (Why or why not?) Q2: Instead of having to keep asking the user whether there are more elements, how do you know when you have read all of the elements? Be sure to submit both your code and your sample .txt file.

6. In a separate file called requests.txt, please list from 1 to 3 activities that you would find helpful in the next few labs. Our default is to practice concepts in OOD and new constructs in Java that are being covered in the lectures, of course.

Submit your answers to above items to Canvas for CS5005 *(due next Thursday by midnight). Only your .java files and any associated .txt files for the above need to be uploaded. (Often you can just add comments within your code files as needed to identify the problems and to answer questions about the code or clarify your approach.) Timely submission encouraged, but late is better than never. Then work on any unfinished labs or CS5004 homework assignments. Remember to submit* ***homeworks to Canvas-for-CS5004*** *and* ***labs to Canvas-for-CS5005****.*