LAB EXERCISE

OrderedList

Background:

This lab will use a text file, *'file20.txt'*, which is similar to the one used in the binary search lab in Lesson 27. The file has been saved in random order by `id` number. Your program must build an ordered linked list based on the `id` field. The ordered list should be implemented as a `SinglyLinkedList` of type `ListNode`.

Assignment:

1. Here are some of the specifications for the methods to be added to the `SinglyLinkedList` class:

   a. You are to write a method `insert` that builds the linked list in order based on the `id` value.

   b. A `find` method will check the list for a specific id value, returning a reference to such a node if the value exists in the list. If the value is not found, the method should return a **`null`** value.

   c. A `remove` method will remove unwanted data from the linked list.

   d. Write a `clear` method that clears the entire list.

   e. Write a `size` method that returns the number of nodes in the list. Avoid copying this code from the last lesson and write it from scratch.

   f. Reading the data file is an identical process to that used in *Store.java* in Lesson 26.

   g. Printing the list involves the same code as used in the previous lesson on linked lists.

   h. Code a recursive `printBackward` method that prints out the linked list contents in reverse order.

2. If your instructor chooses, you will be provided with a program shell consisting of a *main* menu, testing methods, and stubbed methods. Here are some of the specifications of this program shell.

   a. A method `addLast` is provided which builds the list just as in lab L.A.29.1, `List1`. The list will be built in the same order as it exists in the data file. The routine that reads the data from the text file has been separated from the method to insert the information into the linked list. See the code in the program shell.

   b. The `find` method returns a **`null`** value.

   c. The `remove` method returns a **`null`** value.

   d. The `clear` method is stubbed out as a print statement.

   e. The `printBackwards` method is stubbed out as a print statement.

   f. The `size` method returns 0.

   g. Methods to read the data file and print the list are provided.

<u>Instructions:</u>

1. Modify and write code as necessary to satisfy the above specifications.

2. Display the source code.

3. Display a run output of the following:

    a. The entire list printed in ascending order. Include a call of the `size` method and print the number of nodes.

    b. Print the list backward using the recursive `printBackward` method. Do not worry about printing a line number in this algorithm.

    c. The `id` 's you searched for and their corresponding inventory amounts. Your instructor will specify which `id` values to search for.

    d. Delete `id` values as specified by your instructor. Print the abbreviated list after values have been deleted and include a call to `size`.

    e. Clear the entire list of all nodes. Call the `size` method one last time.

    f. Call your instructor to your workspace for scoring.