Lab 03 Report: To Do List

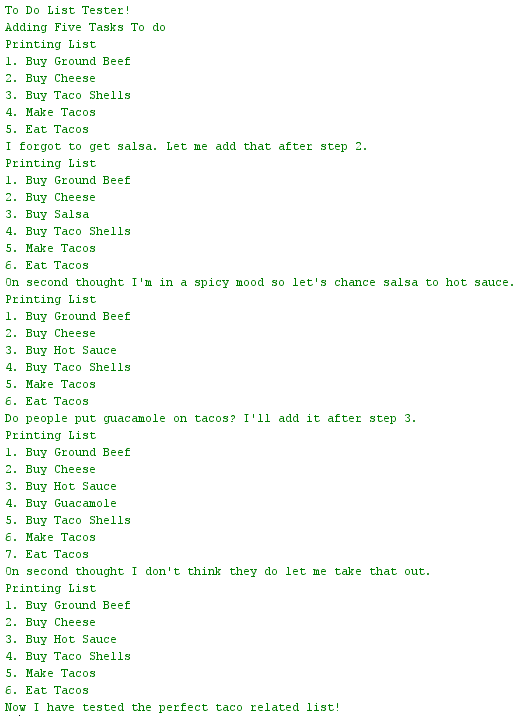
Problem

We had to write a program which implemented LinkedLists with an internal class called ListNode that had instance variables data of type String and link of type ListNode. The external class ToDoList would contain instance variables head current and previous to keep track of where the program is within the ListNode. We then had to implement different methods which would move current with goToNext and goToPrev, getDataatCurrent and setDataAtCurrent which would just use the current instance variable, addItem which would start the ListNode, insertAfterCurrent which would add an item after current if it was null, deleteCurrentNode by resetting links, and showList which would print the entire ListNode as long as the next spot in the ListNode was not null. Then we had to write a frontend driver which would demonstrate all of these methods being used.

Proposed Solution

1. Write a class ToDoList
   1. Implement an Internal class ListNode
      1. Include instance variables data of type String, and link of type ListNode
      2. Include a default and parameterized constructor
   2. Three instance variables, head, current, and previous all of type ListNode
   3. Default constructor which initializes head to an empty LsitNode, setting current and previous to point at head
   4. Write 8 methods:
      1. goToNext- moves current forward, can’t move forward if next is null
      2. goToPrev- moves backwards, can’t move back if current is the head therefore previous = null.
      3. getDataAtCurrent- returns the current node current.data if current != null.
      4. setDataAtCurrent- takes in a String and sets current.data to String unless current is null.
      5. addItem- adds a new node at end of list, if head is null then it starts the list.
      6. insertAfterCurrent- creates a new node based on String passed in, and creates node after current position.
      7. deleteCurrentNode- removes current node by resetting links
      8. showList- prints out the contents of the list line-by-line by using a while loop checking if the next spot is not null.
2. Test all methods in a driver class which implements an instance of ToDoList.
   1. Call all methods using appropriate parameters to demonstrate that each of the 8 methods mentioned earlier work.

Tests and Results



Problems Encountered

The only issue I see with my code is how I decided to do it. You can write LinkedList in many different ways, but with my addItem I decided to use it to start off the list, not to add any item as I could not get it to add at the end of the list, so I used insertAfterCurrent for each new entry after the very first.

Conclusions and Discussion

This lab was very helpful in learning LinkedList and how pointers (.link) works in this lab to see where everything goes or what it is referring to. However, it is still challenging to write a GenericDoubledLinkedList do to the completion of this lab not requiring a full understanding of LinkedLists.

While this solution works, my addItem method was not properly written as I did not understand how to use a loop in order to look for the first null element at the time, but have learned how to do this after the homework of the same week. This lab was farely straight forward and easy to complete as long as the student followed the example of TacoDatabase which JJ wrote during this week’s Tuesday lecture.

Additional Questions

1. Name some advantages of using a linked list instead of an array.

Linked lists are a dynamic data structure which can change size over time, and insertion/deletion operates can be easily implemented. Data structures such as stacks and queues which use these features can be implemented by using a linked list.

1. Name some disadvantages of using a linked list instead of an array.

Linked lists do not allow random access to elements, are time consuming to setup but worth it, and traversing backwords through the linked list is much harder when compared to using an array.