Lab Assignment 1: Linear Data Structures [30 Marks]

SE2205: Data Structures and Algorithms using Java – Fall 2022

Open Day: October 3, 2022; **Cut off time**: October 9, 2022, Sunday @11pm

Demo Days During Lab Time: Week of 10/Oct/2022 Exception: Monday's sections: 17/Oct/2022 (Reason: Thanksgiving Holiday: 10/Oct/2022)

Prepared by Dr. Quazi Rahman (qrahman3@uwo.ca).

A. Rationale and Background

In this lab Assignment you may reuse the code for Singly Link-List, given in the class handout, and add few more methods to realize Stack and Queue data structures using Singly Linked List. Also, while building the Singly linked list from the given ArrayList<>, you will demonstrate your understanding of the Singly Linked List.

B. Evaluation and Submission Instructions

Submit your Lab-Assignment online by carrying out the following instructions:

1. Create a Project with the following name: *username_LabAssignment1*
2. For this question create a package called LA1Q1.
3. Use the names for each class and the associated variables as shown in the UML diagram.
4. For this question, use the static header and footer methods your created before and modify it as you see fit.
5. Comments: **Writing short but clear comments for Lab Exercises is mandatory**. If the comments are clear, full credit will be given to the written comments.
6. Once the assignment is completed, go to your 'Assignments' folder. Select the project folder (e.g. *username_LabAssignment1*). Right-click to select 'Send to' 'Compressed zipped folder'. Make sure it has the same naming convention (e.g., *usernames_LabAssignment1*.zip). Upload this file on OWL as your submission.
7. You will be graded on this lab Assignment based on the comments, running code and your understanding of the code (demonstration). Comments: 15%, Running Code: 25%, Demonstration: 60%. You will do individual-demonstration for this lab.
8. If you are absent during your demo-time, you will get a zero grade, even if you submit the properly commented lab that runs.
9. If you can't answer to any question, your TA asks you on the lab, you will get a maximum overall (including all components) grade of 40%
10. Extension for submission: Up to three days with 10% penalty on each day with the cut off time @11pm on each day.

C. Lab Question

1. Question 1 [30 Marks]

Create a Singly link-list data structure and use that to realize a stack and a queue.

Here you need to create three classes as shown in the UML diagram below, and these classes are a generic **Node<T>** class, a generic Singly Link List Class called **SinglyLinkList<T>**, and a driver class called **DemoSinglyLinkedList**. Unlike the nested Node<T> class given in the handout, here, you need

to create an independent Node<T> class. You will use the same identifiers as shown in the UML Diagram. The specifications follow:

(a) Define a generic class called *Node<T>* with the following specifications (see the class diagram below):

i) Two private fields: T type field called element and Node<T> type field called next.

ii) Two constructors: without parameter and with parameter. For without-parameter one, you can leave the method-body empty.

iii) Two accessor/ getter methods and one mutator/ setter method as shown in the UML diagram.

(b) Define a second generic class called *SinglyLinkList<T>* with the following specifications:

i) Node<T> type two private fields called head and tail, and an int type private field called size, which will keep track of the number of elements in the list.

ii) Two constructors: without parameter (you can keep it empty) and with parameter.

iii) A getter method for the field 'size'.

iv) isEmpty() method: returns true if the list is empty.

v) toString() method: Override the Object's toString() method which should return the list as a string as shown in the sample output. (Hint: **String.format(//any print statement format)**; returns a string according to the format that you use inside any print statements. Talk to your instructor if you need any help.)

vi) addFirst() method: adds a node at the head of the linked list, and makes it a new head.

vii) addLast() method: adds a node at the tail of the linked list, and makes it a new tail.

viii) removeFirst(): removes the head and returns the element, and then assigns the next one as the new head.

ix) removeLast(): removes the tail and returns the element, and then assigns the node before the current tail as its new tail.

x) pushAtHead(): Define this method to realize the link-list as a stack with the head as the top of the stack. This method will push an element to the top of the stack. This definition will be very simple as below:

```
public void pushAtHead(T element){
    addFirst(element);
}
```

xi) popFromHead(): Define this method to realize the link-list as a stack with the head as the top of the stack. This method will pop an element from the top of the stack, and then return it. You can write this simple definition with the help of removeFirst() method.

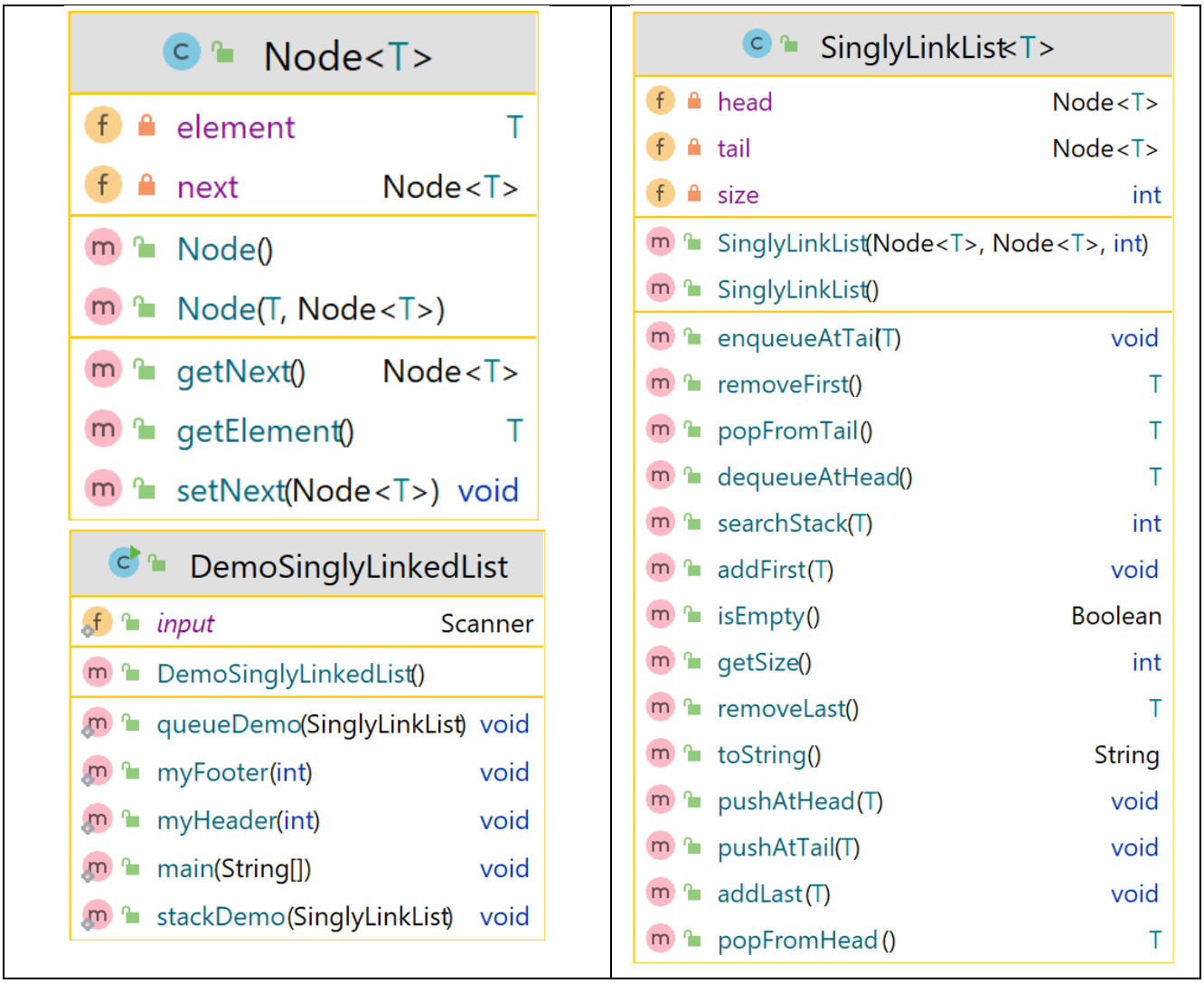
xii) pushAtTail(): Define this method to realize the link-list as a stack with the tail as the top of the stack.

xiii) popFromTail(): Define this method to realize the link-list as a stack with the tail as the top of the stack. It will return the popped value.

xiv) enqueueAtTail(): Define this method to realize the link-list as a queue with the tail as the rear of the queue. (Hint: you need to use addLast() method inside)

xv) dequeueAtHead(): Define this method to realize the link-list as a queue with the head as the front of the queue. (Hint: you need to use removeFirst() method inside). It will return the dequeued value.

xvi) searchStack(): This method will search an element in the stack from the head-side of the link-list and return an integer value based on the position of the element (e.g., if the element is found in the first position it will return 1). If the element is not found, it will return 0.



(c) Now define the driver class called *DemoSinglyLinkedList* with the following specifications:

i) A public static field of Scanner type.

ii) An empty parameter constructor: this is optional.

iii) The header and the footer methods: You should copy-paste these methods from your lab exercise solution and modify it according to the current problem

iv) stackDemo() method: It is a static void method that will accept a SinglyLinkList type parameter and demonstrate the stack operations. Check the specifications in section (d).

v) queueDemo() method: It is a static void method that will accept a SinglyLinkList type parameter and demonstrate the queue operations. Check the specifications in section (e).

vi) main() method: See the specifications Check the specifications in section (f).

(d) Specifications of stackDemo() method:

i) Once called from main(), this method will print the message – "Which end of the Linked List you would like to use as the stack top? "

ii) Get the user's choice, and validate that the user enters either 1 or 2, but nothing else (similar to what you did in Lab exercise 2).

iii) Based on the user's validated choice (see the sample output), it will either demonstrate stack operation considering the head as the top or the tail as the top of the stack.

iv) In the demo, as shown in the sample output, when the user enters 1, it will pop all the elements of the linked list one at a time in sequence and display the stack content, with the aid of the toString() method, after every pop operation. (Hint: you need to use a loop here.)

v) After that it will push two numbers as shown in the sample output. You need to use the same integer values that have been used in the sample output.

vi) If the user chooses 2, it will pop three elements from the link list and later push two more as shown in the sample output.

vii) Once done, this method will end its execution, and the control will go back to the main() method.

(e) Specifications of queueDemo() method (See the method signature and the return type in the UML diagram):

i) Once called it will display a message – "Note: Head is the Q-front, and Tail is the Q-rear"

ii) After that it will enqueue couple of strings (you need to use the same string values that have been used in the sample output) in the queue and then dequeue first three strings from the queue. After every enqueue and deque operation, it has to print the content of the list with the aid of the toString() method.

iii) Once done, this method will end its execution, and the control will go back to the main() method.

(f) Specifications of the main () method See the method signature and the return type in the UML diagram):

i) Call the header method.

ii) Declare an ArrayList<> of Integer with values: 56, -22, 34, 57, 98.

iii) Declare an ArrayList<> of String with values: "Griffin", "Will", "Isra", "Delaney", "Madison"

iv) Print these two array lists on the screen

v) Declare a linked list of Integers and populate this with the Integer ArrayList<> values you declared above. In this case, to demonstrate the linked-list structure add the first three elements from the ArrayList<> at the header side of linked list, and last two elements from the end of the ArrayList<> at the tail side of the linked list. Print the linked list with the aid of the toSting() method you already defined (see the sample output).

vi) Repeat (iv) for a linked list of Strings with the String ArrayList<> values and print the linked list.

vii) Print the message – "Stack demo with the Integer linked list ..." and then call the stackDemo() method and pass the Integer linked list.

viii) Once that demo is done in the stackDemo() method, print the message " Queue demo with the String linked list ..." and then call the queueDemo() method and pass the String linked list.

ix) Once done, search the stack for an element chosen by the user. For this section (only), set up the search in a way that it continues until the user wants to stop (same as you did in Lab Exercise 2). Based on the user's choice, the search will stop.

x) Call the footer method.

Sample Output:
Sample Run 1:

```
=====================================================
Lab Assignment 1
Prepared By: Quazi Rahman
Student Number: 999999999
Goal of this Exercise: ........!
=====================================================

The given Integer array: [56, -22, 34, 57, 98]
The given String array [Griffin, Will, Isra, Delaney, Madison]
Your Integer List: [34, -22, 56, 98, 57]
Your String List: [Isra, Will, Griffin, Madison, Delaney]

Stack demo with the Integer linked list ...
Which end of the Linked List you would like to use as the stack top?
Enter 1 for head or 2 for tail: 1

Stack Top: Head of the linked list

Let's pop all the elements from the stack in sequence:
The current stack: [34, -22, 56, 98, 57]

34 has been popped! The Revised stack: [-22, 56, 98, 57]
-22 has been popped! The Revised stack: [56, 98, 57]
56 has been popped! The Revised stack: [98, 57]
98 has been popped! The Revised stack: [57]
57 has been popped! The Revised stack: []

Let's push 39 and -58 in the stack ….
The current stack: []
After 39 is pushed, the revised stack: [39]
After -58 is pushed, the revised stack: [-58, 39]

Queue demo with the String linked list ...
Note: Head is the Q-front, and Tail is the Q-rear

Let's enqueue Joelle and Lukas in the queue in sequence ....

The current Queue: [Isra, Will, Griffin, Madison, Delaney]

After Joelle is enqueued, the revised Queue: [Isra, Will, Griffin, Madison, Delaney, Joelle]
After Lukas is enqueued, the revised Queue: [Isra, Will, Griffin, Madison, Delaney, Joelle, Lukas]

Let's dequeue first three elements from the queue in sequence ....
```

```
The current Queue: [Isra, Will, Griffin, Madison, Delaney, Joelle, Lukas]

Isra has been dequeued! The revised queue: [Will, Griffin, Madison, Delaney, Joelle, Lukas]
Will has been dequeued! The revised queue: [Griffin, Madison, Delaney, Joelle, Lukas]
Griffin has been dequeued! The revised queue: [Madison, Delaney, Joelle, Lukas]

Let's search the Stack...
The current stack: [-58, 39]

Enter the value you are searching for: 70
The value is not found!
Do you want to continue? (enter 'y' for yes, or enter any other key to discontinue): y
Enter the value you are searching for: 39
The value 39 is found in location 2 from the top of the stack
Do you want to continue? (enter 'y' for yes, or enter any other key to discontinue): y
Enter the value you are searching for: -58
The value -58 is found in location 1 from the top of the stack
Do you want to continue? (enter 'y' for yes, or enter any other key to discontinue): n

=========================================================
Completion of Lab Assignment 1 is successful!
Signing off - Quazi
=========================================================
```

Sample Run 2:

```
=========================================================
Lab Assignment 1
Prepared By: Quazi Rahman
Student Number: 999999999
Goal of this Exercise: ........!
=========================================================
The given Integer array: [56, -22, 34, 57, 98]
The given String array [Griffin, Will, Isra, Delaney, Madison]
Your Integer List: [34, -22, 56, 98, 57]
Your String List: [Isra, Will, Griffin, Madison, Delaney]

Stack demo with the Integer linked list ...
Which end of the Linked List you would like to use as the stack top?
Enter 1 for head or 2 for tail: 2

Stack Top: Tail of the linked list

Let's pop first three elements from the stack in sequence:
The current stack: [34, -22, 56, 98, 57]
```

57 has been popped! The revised stack: [34, -22, 56, 98]
98 has been popped! The revised stack: [34, -22, 56]
56 has been popped! The revised stack: [34, -22]

Let's push 39 and -58 in the stack ....
The current stack: [34, -22]
After 39 is pushed, the revised stack: [34, -22, 39]
After 58 is pushed, the revised stack: [34, -22, 39, -58]

Queue demo with the String linked list ...
Note: Head is the Q-front, and Tail is the Q-rear

Let's enqueue Joelle and Lukas in the queue in sequence ....

The current Queue: [Isra, Will, Griffin, Madison, Delaney]

After Joelle is enqueued, the revised Queue: [Isra, Will, Griffin, Madison, Delaney, Joelle]
After Lukas is enqueued, the revised Queue: [Isra, Will, Griffin, Madison, Delaney, Joelle, Lukas]

Let's dequeue first three elements from the queue in sequence ....

The current Queue: [Isra, Will, Griffin, Madison, Delaney, Joelle, Lukas]

Isra has been dequeued! The revised queue: [Will, Griffin, Madison, Delaney, Joelle, Lukas]
Will has been dequeued! The revised queue: [Griffin, Madison, Delaney, Joelle, Lukas]
Griffin has been dequeued! The revised queue: [Madison, Delaney, Joelle, Lukas]
Let's search the Stack...
The current stack: [34, -22, 39, -58]

Enter the value you are searching for: 76
The value is not found!
Do you want to continue? (enter 'y' for yes or enter any other key to discontinue): y
Enter the value you are searching for: 39
The value 39 is found in location 3 from the top of the stack
Do you want to continue? (enter 'y' for yes or enter any other key to discontinue): n
==========
Completion of Lab Assignment 1 is successful!
Signing off - Quazi
==========