Lab One

Reece Schenck
Reece.Schenck@Marist.edu

February 20, 2023

1 Program

There are some imports needed for my code, more detail is presented in the explanation of Main

1.1 Node

I tweaked this class from one of my prior SD1 project. This node holds a string(I only use one character) and a reference to the next node. The string is taken in as a parameter.

See Lines 5-8 of the Node class(Find under Appendix/Code/Node).

1.2 LinkedList

I tweaked this class from one of my prior SD1 project. This uses the node class by declaring a Node root that the linked list is build off of. Root is null by default. The linked list is made by adding nodes onto the list either at the front or end. To add at the end, the linked list is traversed until the last node is found, then the node with the given parameter is added at the end.

See Lines 12-22 of the LinkedList class(Find under Appendix/Code/LinkedList)

.

To add at the front the process is slightly different. The node with the given parameter replaces the root and the rest of the nodes are shifted over accordingly.

See Lines 28-39 of the LinkedList class(Find under Appendix/Code/LinkedList)

.

To delete the node at the end of the list, the list is traversed and when the last node is found the second to last node is set as the last node(it's pointer to the previous last node was set to null). If the root was never set then nothing will be deleted.

See Lines 45-56 of the LinkedList class(Find under Appendix/Code/LinkedList)

.

To delete the node at the front(the root node), the root is set to the next node. The the list is traversed to ensure all the nodes are still properly connected. If the root is null then nothing is deleted.

See Lines 60-71 of the LinkedList class(Find under Appendix/Code/LinkedList)

.

To get the length of the list. The root is counted, and then the rest of the list is traversed, incrementing the counter with each pass until the end is found, in which the final value is returned.

See Lines 76-86 of the LinkedList class(Find under Appendix/Code/LinkedList)

.

To print the list, first the root is printed before traversing the list printing each node as it goes to completion.

See Lines 89-98 of the LinkedList class(Find under Appendix/Code/LinkedList)

.

1.3 Stack

Using the LinkedList class, the stack class can push items(LinkedList's addEnd), pop items(LinkedList's deleteEnd), and get the length(LinkedList's length). I didn't end up implementing peek.

See Lines 8-17 of the Stack class(Find under Appendix/Code/Stack)

1.4 Queue

Using the LinkedList class, the stack class can enqueue items(LinkedList's addEnd), dequeue items(LinkedList's deleteFront), and get the length(LinkedList's length).

See Lines 8-17 of the Queue class(Find under Appendix/Code/Queue)

1.5 Main

1.5.1 Imports

These are the imports I needed for my code in Main: import java.util.ArrayList; import java.util.Arrays; import java.util.Objects; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException;

These Are used for reading/writing from the text file into and Array

See Lines 1-6 of the Main class(Find under Appendix/Code/Main)

1.5.2 Read/Write Text File

To read/Write the text file into an array I first got the filename of the text file and made an array list. I then use the file reader to to read the file line by line adding them to the array list. I ended up using ChatGPT to help me with this part.

See Lines 18-38 of the Main class(Find under Appendix/Code/Main)

1.5.3 Palindrome Check

After writing the text file into the array, I then traverse the array. Each string is split into characters and then pushed to a Stack and Queue(This gave me some errors that I haven't been able to resolve, I go into more details under 'Unresolved Issues and Errors'). After the stack and queue are filled, they are popped and dequeued one by one, each time checking to see if the they match. If so it is a palindrome, otherwise it is not, the stack and queue are emptied and the loop is ended, then the next string goes through the same process. The items that are palindromes are printed out(using a temp stack which gets emptied after).

See Lines 44-86 of the Main class(Find under Appendix/Code/Main)

2 Unresolved Issues and Errors

After spending numerous hours over multiple days I ended up with a few errors That I could not fix. The central error being: Exception in thread "main" java.lang.NullPointerException:Cannot invoke "Stack.push(String)" because "Main.lifo" is null at Main.main(Main.java:52). I tried using multiple different resources from ChatGPT, StackOverflow, other comp sci major students/friends, and other resources on google to try and fix the errors, (As I had the errors after the time you would realistically respond to an email) but nothing worked. Every change I implemented caused more errors to appear that I tried to fix as much as possible, but I ran out of time.

3 Appendix

3.1 Code

3.1.1 Node

```
1  public class Node {
2      String item;
3      Node nextNode;
4
5      //taken from my previous project in SD1
6      public Node(String item) {
7          this.item = item;
8          this.nextNode = null;
9      }
10 }
```

3.1.2 LinkedList

```
1  public class LinkedList {
2      //singly linked list
3      Node root;
4      //Node tail;
5
6      public LinkedList(){
7          this.root = null;
8      }
9
10     //traverses list and adds at the end
11     //Took this code from a previous project in SD1 and slightly modified it
12     public void addEnd(String item) {
13         if (this.root == null) {
14             this.root = new Node(item);
15             System.out.println("The item " + item + " has been added successfully.");
16         } else {
```

```
17              Node tempNode = this.root;
18              while (tempNode.nextNode != null) {
19                  tempNode = tempNode.nextNode;
20              }
21              tempNode.nextNode = new Node(item);
22              System.out.println("The item " + item + " has been added successfully.");
23          }
24      }
25
26      //Replaces root
27      //Used code from previous add and heavily changed it
28      public void addFront(String item) {
29          if (this.root == null) {
30              this.root = new Node(item);
31              System.out.println("The item " + item + " has been added successfully.");
32          } else {
33              Node tempNode = this.root;
34              this.root = new Node(item);
35              while (tempNode.nextNode != null) {
36                  tempNode = tempNode.nextNode;
37              }
38              tempNode.nextNode = null;
39              System.out.println("The item " + item + " has been added successfully.");
40          }
41      }
42
43      //Used previous project's delete method as a base and heavily modified for
44      //both deleteEnd and deleteFront
45      public String deleteEnd() {
46          if (this.root == null) {
47              return("The root hasn't been set yet.");
48          } else {
49              Node tempNode = this.root;
50              while (tempNode.nextNode != null) {
51                  tempNode = tempNode.nextNode;
52              }
53              String deleted = tempNode.item;
54              System.out.println("The item " + deleted + " has been deleted");
55              tempNode = null;
56              return deleted;
57          }
58      }
59
60      public String deleteFront() {
61          if (this.root == null) {
62              return("The root hasn't been set yet.");
63          } else {
64              String deleted = this.root.item;
65              Node tempNode = this.root;
66              this.root = tempNode.nextNode;
67              while (tempNode.nextNode != null) {
68                  tempNode = tempNode.nextNode;
69              }
70              System.out.println("The item " + deleted + " has been deleted");
71              return deleted;
72          }
73      }
74
75      //I took this code from a project I did in SD1
76      public int length() {
77          int length = 0;
78          if (this.root != null) {
79              Node tempNode = this.root;
80              length++;
```

```
            while (tempNode.nextNode != null) {
                tempNode = tempNode.nextNode;
                length++;
            }
        }
        return length;
    }

    public void print(){
        if (this.root == null) {
            System.out.println("There is nothing in the linked list.");
        } else {
            System.out.println("");
            Node tempNode = this.root;
            System.out.print(tempNode);
            while (tempNode.nextNode != null) {
                tempNode = tempNode.nextNode;
                System.out.print(tempNode);
            }
        }
    }
}
```

3.1.3 Stack

```
public class Stack {
    LinkedList list;

    public Stack(){
        this.list.root = null;
    }

    public void push(String value) {
        list.addEnd(value);
    }

    public String pop() {
        return list.deleteEnd();
    }

    public int length(){
        return list.length();
    }

    //public String peek(){
    //
    //}
}
```

3.1.4 Queue

```
public class Queue {
    LinkedList list;

    public Queue(){
        this.list.root = null;
    }

    public void enqueue(String value) {
        list.addEnd(value);
    }
```

```
11
12      public String dequeue() {
13          return list.deleteFront();
14      }
15
16      public int length(){
17          return list.length();
18      }
19  }
```

3.1.5 Main

```
1   import java.util.ArrayList;
2   import java.util.Arrays;
3   import java.util.Objects;
4   import java.io.BufferedReader;
5   import java.io.FileReader;
6   import java.io.IOException;
7
8   public class Main {
9
10      public static Stack lifo;
11      public static Queue fifo;
12      public static boolean palindrome;
13      public static LinkedList temp;
14
15      public static void main(String[] args) {
16          //Read and Write text file into an array
17          //I was having some trouble getting the read/write to work so I tweaked
18          //some code from ChatGPT
19          String filename = "C:\\Users\\goldh\\OneDrive\\Documents\\GitHub\\RSchenck-435\\Lab 1\\magicite
20          ArrayList<String> lines = new ArrayList<String>();
21
22          try {
23              BufferedReader reader = new BufferedReader(new FileReader(filename));
24              String line = reader.readLine();
25              while (line != null) {
26                  lines.add(line);
27                  line = reader.readLine();
28              }
29              reader.close();
30          } catch (IOException e) {
31              e.printStackTrace();
32          }
33
34          String[] linesArray = lines.toArray(new String[lines.size()]);
35          //prints out the read lines
36          //System.out.println("Lines read from file:");
37          //for (String l : linesArray) {
38          //    System.out.println(l);
39          //}
40
41          //Splits up the strings to single characters
42          //Then checks for palindromes
43
44          System.out.println("Here are all the palindromes:");
45
46          for (String l : linesArray) {
47              String[] splitted = l.split("");
48              for(String s : splitted){
49                  if(!" ".equals(s)){
50                      //IDK why I'm getting this error:
51                      //Exception in thread "main" java.lang.NullPointerException:
```

```
52                      //Cannot invoke "Stack.push(String)"
53                      //because "Main.lifo" is null at Main.main(Main.java:52)
54                      lifo.push(s);
55                      fifo.enqueue(s);
56                      temp.addFront(s);
57                  }
58              }
59              //The length of both lifo and fifo will be the same
60              int len = lifo.length();
61
62              for (int i=0;i<len;i++){
63                  String popped = lifo.pop();
64                  String dequeued = fifo.dequeue();
65                  if(Objects.equals(popped, dequeued)){
66                      palindrome = true;
67                  }else{
68                      //The item won't be counted as a palindrome and ends the loop
69                      //but the stack and queue will be fully deleted for the next
70                      //pass
71                      palindrome = false;
72                      for (int j=i;j<len;j++){
73                          lifo.pop();
74                          fifo.dequeue();
75                      }
76                      i = len;
77                  }
78              }
79
80              //prints temp only if the item is a palindrome
81              if (palindrome == true){
82                  temp.print();
83              }
84              //Clears temp
85              for(int i=0;i<temp.length();i++){
86                  temp.deleteFront();
87              }
88
89          }
90      }
91  }
```

3.1.6 Text File (magicitems)

I'm not putting all of the words here because its 666 lines of text/code