Assignment 1: Data Structures

Robert Perrone
robert.perrone1@Marist.edu

September 17, 2020

1 Main Class

The main class it where all the magic happens. First the magicitems.txt file is scanned/read (line 13) and then put into an array through a for loop (lines 18-22). A for loop is used to iterate through each item in the array (lines 28-79). A nested for loop is then used to add each character of each magicitem onto both the stack and queue through push and enqueue, respecitvely (lines 42-49). Another nested for loop is used to remove each character of the magicitem from stack and queue using pop and dequeue, respectively (lines 52-67). The last few steps involve check if the original magicitem (dequeued from the queue) and the revered magicitem (popped from the stack) is the same or not. If both are the same, then it is a palindrome. If the original and reveresed are different, the magicitem is not a palindrome (lines 70-74). Finally, if it's a palindrome, then print out the statement to the system (line 79).

```
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.util.Scanner;
4
5  public class Main {
6      public static void main(String[] args) throws FileNotFoundException {
7          //Initialize Linked List, Stack, Queue
8          LinkedList linkedlist = new LinkedList();
9          Stack stack = new Stack();
10         Queue queue = new Queue();
11
12         //Initialize scanner to scan through magicitems.txt file
13         Scanner input = new Scanner(new File("src/magicitems.txt"));
14
15         final int arrSize = 666;
16
17         //Initialize array to hold the magicitems
18         String[] magicitemsList = new String[arrSize];
19         //Add each item in magicitems to the array
20         for(int i = 0; input.hasNextLine(); i++) {
21             magicitemsList[i] = input.nextLine();
22         }//end for i
```

```

        //Print out the array to check
        //System.out.println(magicitemsList);

        //For loop to go through each magicitem in the array
        for(int x = 0; x < magicitemsList.length; x++) {
            String stackPop = "";
            String queueDeq = "";
            String reverse = "";
            String original="";
            boolean palindrome = false;

            String item = magicitemsList[x];
            item = item.replaceAll(" ", "");
            item = item.replaceAll("'","");
            item = item.toUpperCase();
            //System.out.println(item);

            //For loop to go through each magicitem character in the string
            for(int j = 0; j < item.length(); j++) {
                if(!item.equals("")) {
                    Character chr = item.charAt(j);
                    //Add item's characters to stack and queue
                    stack.push(chr.toString());
                    queue.enqueue(chr.toString());
                }//end if
            }//end for j

            //For loop to go through each magicitem character in the string
            for(int y = 0; y < item.length(); y++) {
                if(!item.equals("")) {
                    //Remove each item's character from stack and queue
                    stackPop = stack.pop().item;
                    queueDeq = queue.dequeue().item;
                    reverse= reverse + stackPop;
                    original = original + queueDeq;

                    /*if(stackPop.equals(queueDeq)) {
                        palindrome = true;
                    }else {
                        palindrome = false;
                    }*/ //end if

                }//end if
            }//end for i
            //System.out.println(original + " " + reverse);

            if(original.equals(reverse)) {
                palindrome = true;
            }else {
                palindrome = false;
            }

            if (palindrome == true) {
                System.out.println("The magic item: " + original + " is a palindrome.");
            }//end if
        }//end for x
    }//end main
}//end Main Class
```

2 Stack Class

The Stack class has three different functions. The first function delcared is the isEmpty() function, which returns a boolean value. This function determines if the stack is empty or not, in which it returns true if empty and false if not empty (lines 3-15). The next function is the push() function. This function adds a new item to the stack and then changes the pointer to the old top if there was something on the stack already (lines 16-27). The last function is the pop() function. This function returns null if empty but otherwise removes the item from the top of the stack and then makes the value that the popped item referenced as the next Node into the new top (lines 28-44).

```
 1  public class Stack {
 2      public Node top;
 3
 4      boolean isEmpty() {
 5          /*Stack-Empty(S)
 6          * if S.top == 0
 7          *   return TRUE
 8          * else return FALSE */
 9
10          if (top == null) {
11              return true;
12          } else {
13              return false;
14          }
15      }
16      void push(String elem) {
17          /*Push(S,x)
18          * S.top = S.top + 1
19          * S[S.top] = x */
20
21          Node n = new Node();
22          n.item = elem;
23
24          Node prevTop = this.top;
25          this.top = n;
26          n.next = prevTop;
27      }
28      Node pop() {
29          /*Pop(S)
30          * if Stack-Empty(S)
31          *   error "underflow"
32          * else S.top = S.top - 1
33          *   return S[S.top + 1] */
34
35          if (this.isEmpty()) {
36              System.out.print("Stack is empty.");
37              return null;
38          }else {
39              Node retVal = this.top;
40            this.top = this.top.next;
41            return retVal;
42        }
43    }
44 }
```

3 Queue Class

The Queue class also contains three functions. The first being the same isEmpty() function as detailed in the stack class. This returns true if the queue is empty and false otherwise (lines 5-11). The enqueue() function addes the new item as the new tail, which it then makes a pointer from the old tail to the new tail (lines 12-31). The dequeue() function returns null if the queue is empty, but otherwise removes the head from the queue and makes the reference to the next Node the new head (lines 32-49).

```
1  public class Queue {
2      public Node head;
3      public Node tail;
4
5      boolean isEmpty() {
6          if (head == null) {
7              return true;
8          } else {
9              return false;
10         }
11     }
12     public void enqueue(String elem) {
13         /*Enqueue(Q,x)
14         * Q[Q.tail] = x
15         * if Q.tail == Q.length
16         *   Q.tail = 1
17         * else Q.tail = Q.tail + 1 */
18
19         Node n = new Node();
20         n.item = elem;
21
22         Node prevBack = this.tail;
23         this.tail = n;
24         //n.next = this.tail; <- this ONE line ruined everything and left me puzzled for hours
25
26         if(isEmpty()) {
27             this.head = this.tail;
28         }else {
29             prevBack.next = tail;
30         }
31     }
32     Node dequeue() {
33         /*Dequeue(Q)
34         * x = Q[Q.head]
35         * if Q.head == Q.length
36         *   Q.head = 1
37         * else Q.head = Q.head + 1
38         * return x */
39
```

```
        if (this.isEmpty()) {
            System.out.print("Queue is empty.");
            return null;
        }else {
            Node retVal = this.head;
            this.head = this.head.next;
            return retVal;
        }
    }
}
```