Assignment One – LaTeX Palindrome Check

Tyler Vultaggio

September 2021

1 My Stack

The implementation of MyStack was done through the usage of a node that I generated. MyStack used the node in order to maintain positions of the items in the stack, but MyStack implemented all pop and push methods fully even though the node already had implementations that could be used Figure 1.

2 Push

The push method, used to add a new node to the top of MyStack, specifically takes in a char which it then uses to generate a new Node which is given a pointer to the old top of MyStack. Figure 2.

3 Pop

The pop method of MyStack, which is used to remove the top-most value of MyStack and returns its value, initially checks if the stack is empty to make sure it is not attempting to remove from an empty stack. Then it locally saves the the value of the old head and then sets the head to the next item in MyStack. Figure 3.

4 My Queue

The implementation of MyQueue was done the same as with MyStack, which was by using a Node as an internal structure. This Node only served as a place

to hold the data, as both the enqueue and dequeue methods were taken care of through MyQueue Figure 4.

5 enqueue

The method enqueue is intended to add a new item to MyQueue, which it does by appending a new Node to the tail of the internal Node, and then setting that node to be the new tail of the Node. Figure 5.

6 dequeue

The dequeue method of MyQueue removes the first item from MyQueue and returns it. To do this it just sets the head value to the next item past head in the internal Node. Figure 6.

7 fileReader

The fileReader part takes a text file in this case magicitems.txt and it takes a scanner and a reader and turns each line into a string and puts it into an array once the string is in the array we clean up the string by removing all of the spaces, number, and special characters in the sting. Figure 7.

8 isPalindrome

The isPalindrome method takes the string that is given from the file reader and then loops through that string and puts each of its character's into a stack and a queue. Then those stack and queues are looped through and each element is compared to each-other through stack's pop and queue's dequeue. If there is a character that doesn't match the other the stack and queue are emptied and the method returns false. However if the loop fully goes through the stack and queue then the word is a palindrome and true is return, and since the whole word was went through stack and queue are already empty. Figure 8.

Figure 1: Example Stack class.

```
/** @author Tyler Vultaggio
* Assignment 1
* Due Friday 9/24/2021
* Algorithms
*/


public class MyStack extends Node
{
        public static Node top;

        public MyStack()
        {

        }
```

Figure 2: Example of my push method.

```
//Push adds and element to the top of the stack
public void push(char newLetter)
{
        Node oldTop = top;
        top = new Node();
        top.letter = newLetter;
        top.next = oldTop;
}
```

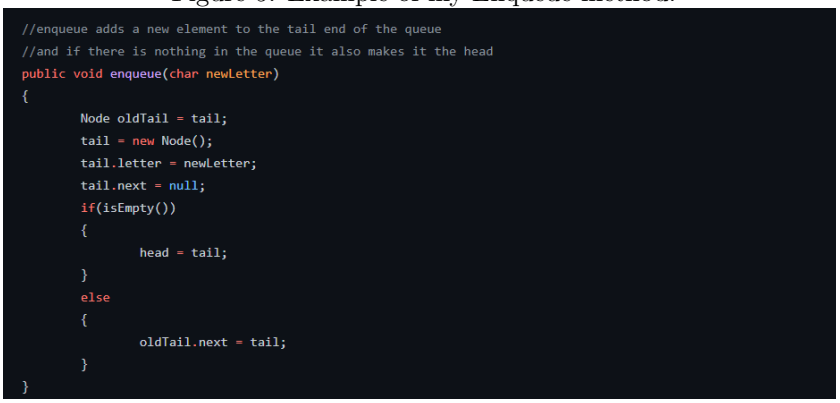
Figure 3: Example of my pop method.

```
//Pop takes the top element off and returns it
public char pop()
{
        if(this.isEmpty())
        {
                throw new IllegalArgumentException("Stack is empty");
        }

        else
        {
                char poppedChar = top.letter;
                top = top.next;
                return poppedChar;
        }
}
```

Figure 4: Example Queue class.

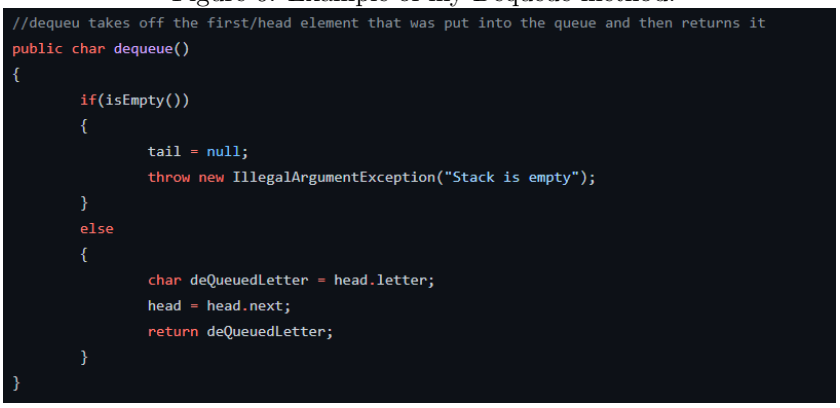
```
/** @author Tyler Vultaggio
 * Assignment 1
 * Due Friday 9/24/2021
 * Algorithms
 */

public class MyQueue extends Node
{
        public static Node head;
        public static Node tail;

        public MyQueue()
        {

        }
```

Figure 5: Example of my Enqueue method.



```
//enqueue adds a new element to the tail end of the queue
//and if there is nothing in the queue it also makes it the head
public void enqueue(char newLetter)
{
        Node oldTail = tail;
        tail = new Node();
        tail.letter = newLetter;
        tail.next = null;
        if(isEmpty())
        {
                head = tail;
        }
        else
        {
                oldTail.next = tail;
        }
}
```

Figure 6: Example of my Dequeue method.



```
//dequeu takes off the first/head element that was put into the queue and then returns it
public char dequeue()
{
        if(isEmpty())
        {
                tail = null;
                throw new IllegalArgumentException("Stack is empty");
        }
        else
        {
                char deQueuedLetter = head.letter;
                head = head.next;
                return deQueuedLetter;
        }
}
```

Figure 7: Example of my fileReader in main.

```
//This puts the file (magicitems into an array of cleaned strings)
Scanner scanner = new Scanner(new File("magicitems.txt"));
String[] myArray = new String[666];
int index = 0;


//This while loop puts each line into an array string and then gets rid of any special chars and spaces
while(scanner.hasNextLine())
{
        myArray[index] = scanner.nextLine();
        myArray[index] = myArray[index].replaceAll("[0-9+,_()/.]", "");
        myArray[index] = myArray[index].replaceAll("\\s", "");
        myArray[index] = myArray[index].replaceAll("'", "");
        myArray[index] = myArray[index].replaceAll("-", "");
        myArray[index] = myArray[index].toLowerCase();

        index = index + 1;
}
scanner.close();

//This loops through the whole array to check for Palindromes in the array
for(int i = 0; i < myArray.length; i++)
{
        if(isPalindrome(myArray[i]))
        {
                System.out.println(myArray[i]);
        }
}
```

Figure 8: Example of my isPalindrome method.

```
public static boolean isPalindrome(String word)
{

        //All of the print statements where to show me what was happening at each stage so I could debug it
        //I thought it was good to leave them in but comment them out.

        //System.out.println(word);
        MyStack stack = new MyStack();
        MyQueue queue = new MyQueue();

        char letter;

        for(int i = 0; i < word.length(); i++)
        {
                letter = word.charAt(i);
                //System.out.println(letter);
                stack.push(letter);
                queue.enqueue(letter);
        }
        for(int i = 0; i < word.length(); i++)
        {
                //System.out.println(stack.isEmpty());
                //System.out.println(queue.isEmpty());
                char letter1 = stack.pop();
                char letter2 = queue.dequeue();
                //System.out.println(letter1 + " " + letter2);
                if(letter1 != letter2)
                {
                        //This clears stack and queue when they are not a palindrome
                        for(int j = i; j < word.length()-1; j++)
                        {
                                stack.pop();
                                queue.dequeue();
                        }
                        return false;
```