Assignment One – LaTeX Palindromes

Daniel Bilotto
danielbilotto1@marist.edu

September 25, 2021

1 Node Class

This class is the base of both the stack and queue class. The node class has a couple of methods like getting and setting data. As well as getting and setting what comes next.

```
public class NodeBilotto
{
        private char myData;
        private NodeBilotto myNext;

        public NodeBilotto ()
        {
                myData = '\0';
                myNext = null;
        }//NodeBilotto

        public NodeBilotto (char newData)
        {
                myData = newData;
                myNext = null;
        }//NodeBilotto

        public char getData()
        {
                return myData;
        }//getData
```

```
        public void setData (char newData)
        {
                myData = newData;
        }//setData

        public NodeBilotto getNext()
        {
                return myNext;
        }//getNext

        public void setNext(NodeBilotto newNext)
        {
                myNext = newNext;
        }//setNext


}//node
```

2 Stack Class

The stack class builds off of the node class, using FILO (first in, last out) we can use this class as the first half of finding the palindromes. The methods here are push to add to a stack; and pop to take away from a stack.

```
public class StackBilotto
{
        private NodeBilotto myTop;

        public StackBilotto()
        {
                myTop = null;
        }//constructor

        public void push (char letter)
        {
                NodeBilotto oldTop = myTop;
                myTop = new NodeBilotto();
                myTop.setData(letter);
                myTop.setNext(oldTop);

        }//push

        public boolean isEmpty()
        {
```

```
            return (myTop == null);
        }//is empty

        public char pop ()
        {
            char retreve = '\0';
            if (!isEmpty())
            {
                retreve = myTop.getData();
                myTop = myTop.getNext();
            }//if
            return retreve;
        }//pop

}//stack
```

3 Queue Class

Queues will be the other half of finding the palindromes because unlike stacks it is FIFO (first in, first out.) This class has the methods enqueue to add and dequeue to take away.

```
public class QueueBilotto {

        private NodeBilotto head;

        private NodeBilotto tail;

        public QueueBilotto()
        {
            head = null;
            tail = null;
        }//Queue

        public boolean isEmpty()
        {
            return (head == null);
        }//is empty


        public void enqueue(char item)
        {
            NodeBilotto oldTail = tail;
```

```
                tail = new NodeBilotto();
                tail.setData(item);
                tail.setNext(null);
                if (isEmpty())
                {
                        head = tail;
                }//if
                else
                {
                        oldTail.setNext(tail);
                }//else

        }//enqueue


        public char dequeue()
        {
                char item = '\0';
                if (!isEmpty())
                {
                        item = head.getData();
                        head = head.getNext();
                }//if

                return item;
        }//dequeue

}//queue
```

4 Main Class

And this is the main class. The main class will read and put the lines from magic items into an array; then it will add the characters from each line to a stack and queue. Once a full line has been added it will then start to pop and dequeue respectively. As it does this it will compare the results and if they match then we have a palindrome!

```
import java.io.*;
import java.util.*;

public class PalindromeBilotto
{
  // Create a new keyboard Scanner object.
  static Scanner keyboard = new Scanner(System.in);
```

```
public static void main(String[] args)
{
  String fileName = null;
  String line = null;
  int size = 666;
  String[] list = new String[size];
  StackBilotto stack = new StackBilotto();
  QueueBilotto queue = new QueueBilotto();

  try
  {
    fileName = "magicitems.txt";

    File theFile = new File(fileName);

    Scanner input = new Scanner(theFile);

    //this will read the file together with the code below

   for (int i = 0; i<size; i++)
   {
               line = input.nextLine().toLowerCase().replaceAll("\\s","");
               list[i] = line;
               //this adds inputs to an array with a size of 665

               String checkOne = "";
               String checkTwo = "";

               for (int j=0; j<list[i].length(); j++)
               {
                       stack.push(list[i].charAt(j));
                       queue.enqueue(list[i].charAt(j));
               }//for

               /*this will take each character of a line in magic item
               and make them into a stack and queue*/

               for (int k=0; k<list[i].length(); k++)
               {
                       checkOne = checkOne + stack.pop();
                       checkTwo = checkTwo + queue.dequeue();

               }//for

               /*this will pop and dequeue the stack and queue
```

```
                    and compare the two lines to see if they match*/

                    if (checkOne.equals(checkTwo))
                            System.out.println(checkTwo);

                    //and if they do it will print out the palindrome

          }//for

            input.close();
            keyboard.close();
          }//try

          catch(Exception ex)
          {
            System.out.println("Oops, something went wrong!");
          }//catch

          /*this sends back a message if something goes wrong in
          importing the text into the array from magic items*/

        }//main

      } //palindrome
```