Gebze Technical University
Department of Computer Engineering
CSE 222/505 - Spring 2020

Homework 2 Report

Alper YAŞAR
151044072

1 – Q2

1. SYSTEM REQUIREMENTS

Deque is a queue that supports adding or removing items from both ends of the data structure. To implement it, your class should keep two linked list. One for the elements in the deque and the other to keep the nodes removed. You should use a removed node, if any available, when a new node is needed instead of creating a new node. With this type of operation you can save time for constructing new nodes and garbage collection the nodes that are not used anymore.

- Constructor
  First create new node : front, rear and removeElement
  Front : hold the first element in deque
  Rear : hold the last element in deque
  removeElement : hold the removed elements from deque

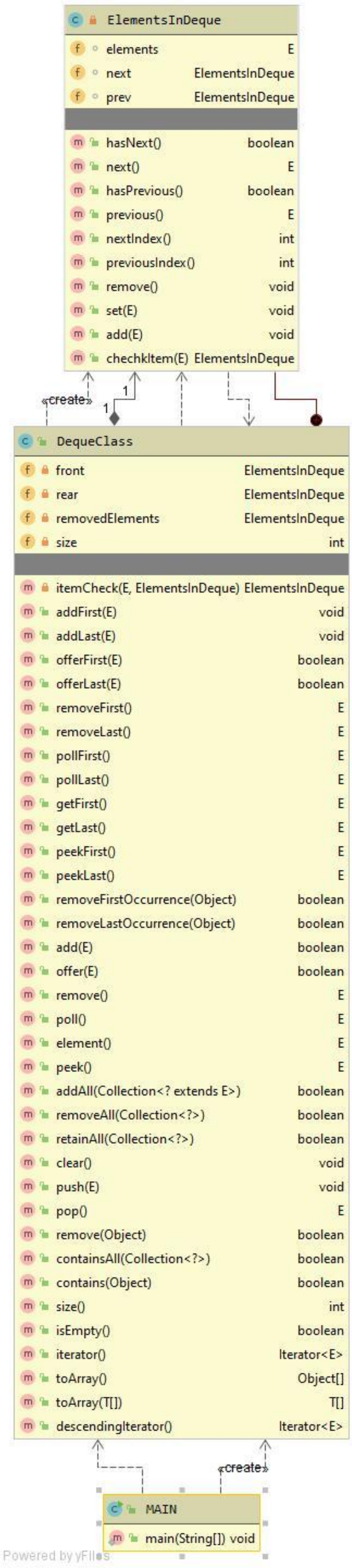
- addFirst(item)
  Firstly check item is removed before in :
  itemCheck(element, node)
  search iteratively in this method and if found return node and remove in removedElement. And assign to front. If front empty after assign; assign front to rear.
  If itemCheck return null create new node and assign front. If front is empty, assign front to rear with new node.
- addLast(item)
  Firstly check item is removed before in :
  itemCheck(element, node)
  search iteratively in this method and if found return node and remove in removedElement. And assign to rear. If front empty after assign; assign rear to front.
  If itemCheck return null create new node and assign front. If front is empty, assign front to rear with new node.
- PoolFirst()
  if front is empty return null.
  Else get first element with getFirst() method and assign it to new element
  After that add first element to removedElement node and assign front next node to front node

- PoolLast()

  if rear is empty return null.
  Else get first element with getLast() method and assign it to new element
  After that add last element to removedElement node and assign rear prev node to rear node

- peekFirst()
  Return the entry at the front of the deque without removing

- peekFirst()
  Return the entry at the rear of the deque without removing

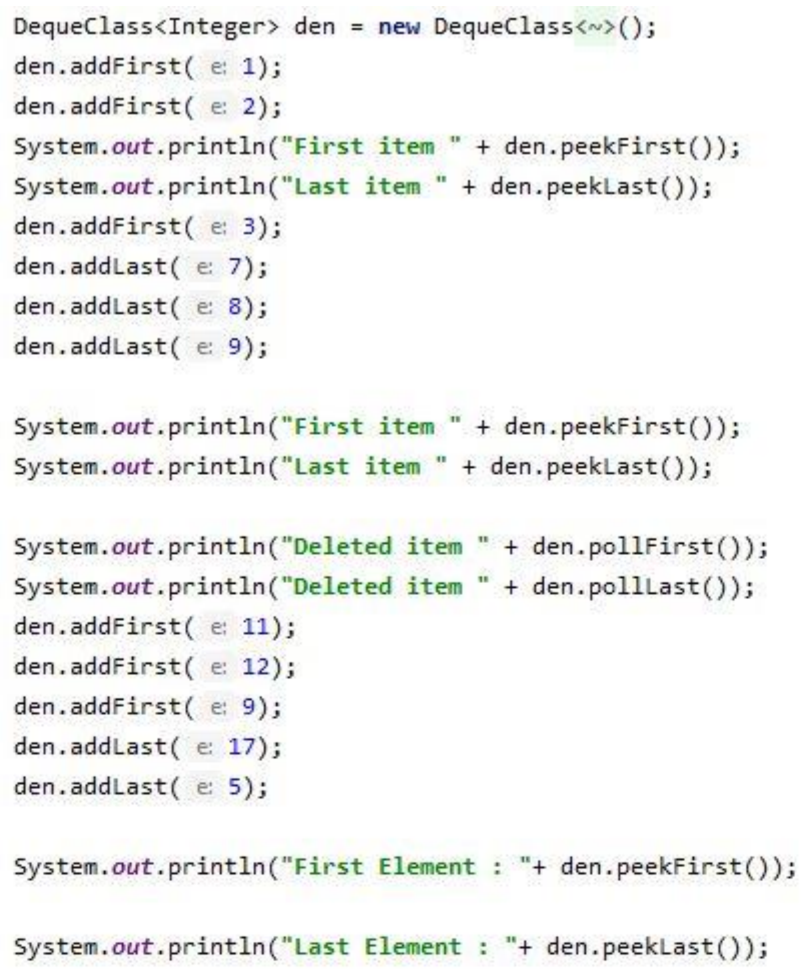


CLASS DIAGRAM

4. PROBLEM SOLUTION APPROACH

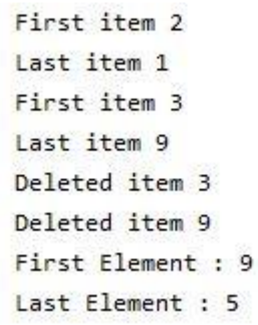
Add new element to deque and check front and rear element with peek and remove element in front or rear with poll

5. TEST CASES



```
DequeClass<Integer> den = new DequeClass<~>();
den.addFirst( e: 1);
den.addFirst( e: 2);
System.out.println("First item " + den.peekFirst());
System.out.println("Last item " + den.peekLast());
den.addFirst( e: 3);
den.addLast( e: 7);
den.addLast( e: 8);
den.addLast( e: 9);

System.out.println("First item " + den.peekFirst());
System.out.println("Last item " + den.peekLast());

System.out.println("Deleted item " + den.pollFirst());
System.out.println("Deleted item " + den.pollLast());
den.addFirst( e: 11);
den.addFirst( e: 12);
den.addFirst( e: 9);
den.addLast( e: 17);
den.addLast( e: 5);

System.out.println("First Element : "+ den.peekFirst());

System.out.println("Last Element : "+ den.peekLast());
```

2. RUNNING AND RESULTS



```
First item 2
Last item 1
First item 3
Last item 9
Deleted item 3
Deleted item 9
First Element : 9
Last Element : 5
```

1 – Q3

1. SYSTEM REQUIREMENTS

Solving special problems with recursive method.

1. Reversing a string. For example, if the input is “this function writes the sentence in reverse”, then the output should be “reverse in sentence the writes function this”.
   First split string to words and assign to array them. After that send the send the reverse(string [],int) method. Check the array size equal to index return last word. Else return method. When the method is done return words.
   ```
   reverse(string, ++index)  +" "+ x[index-1];
   ```

2. Determining whether a word is elfish or not. A word is considered elfish if it contains the letters: e, l, and f in it, in any order. For example, whiteleaf, tasteful, unfriendly, and waffles are some elfish words.
   Check each character in array equal to e,l or f. İf found 3 of them return true or false.

3. Sorting an array of elements using selection sort algorithm.
   Sort first n-1 elements
   Insert last element at its correct position in sorted
   Move elements of arr[0..i-1], that are greater than key, to one position ahead of their current position

4. Evaluating a Prefix expression
   If character is operate push it to stack else return numeric value of first character.control is string empty. If character is numeric value return value op next character recursively.

5. Evaluating a Postfix expression
   If first character is number push to stack and return next character to string. If first character is an operater pop one element and assign to new value and po the other element to new second value and operate them .

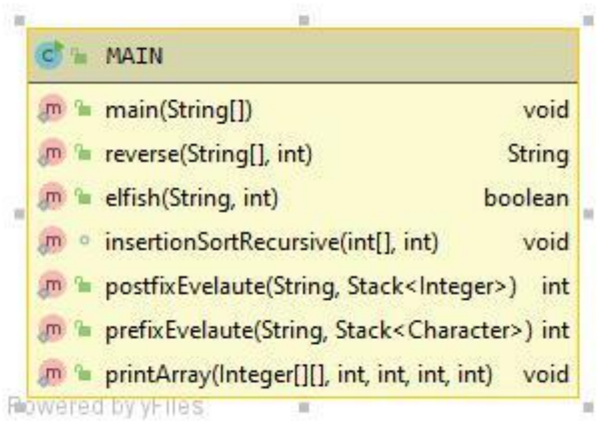
6. Printing the elements of an array on the screen as in the example below.

Write input array like a maze. Return to equal to size. If passed one element do not write this element again. Assign to its 0. Return next element.

Input : 1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20

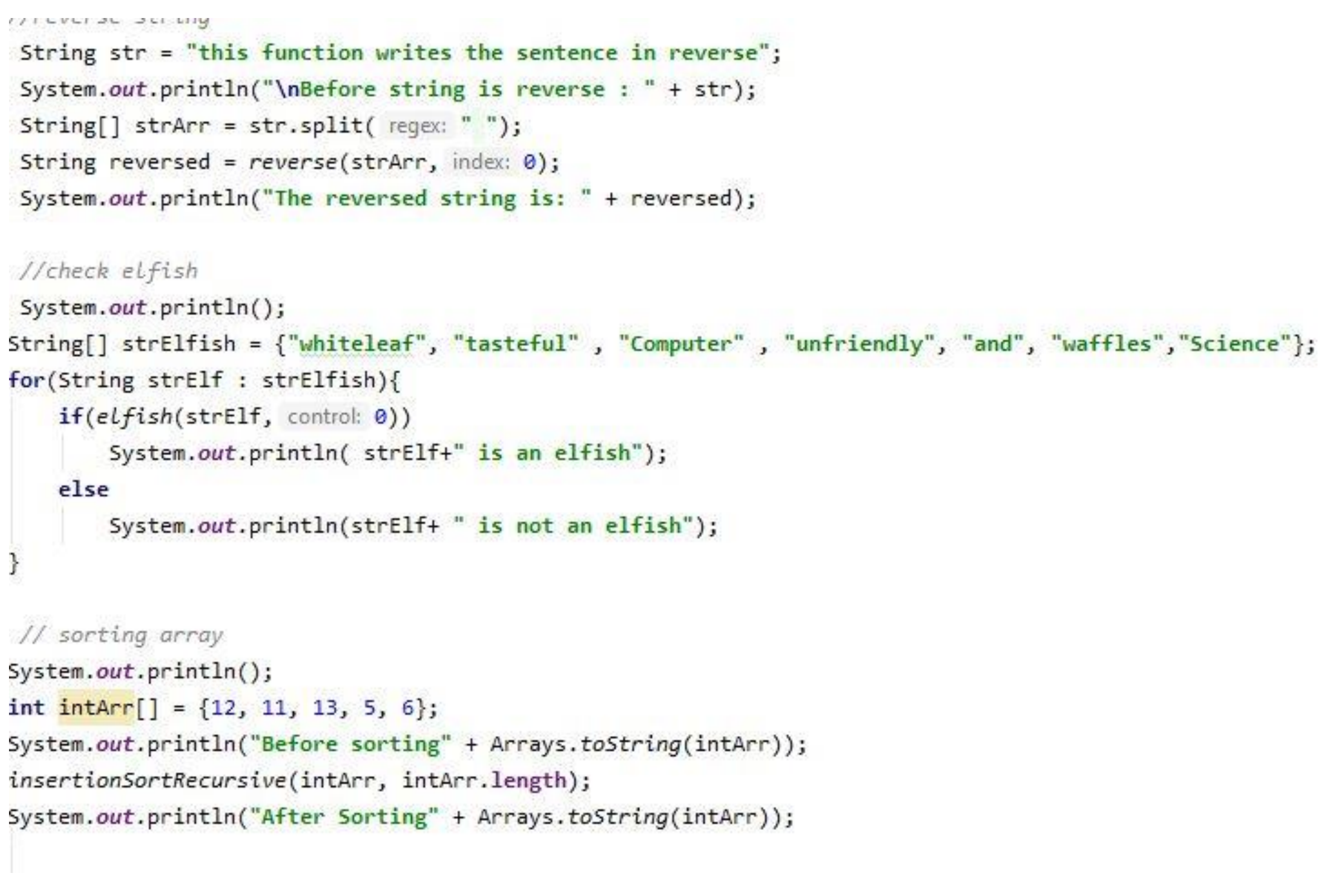
Output : 1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10 11

3. CLASS DIAGRAM

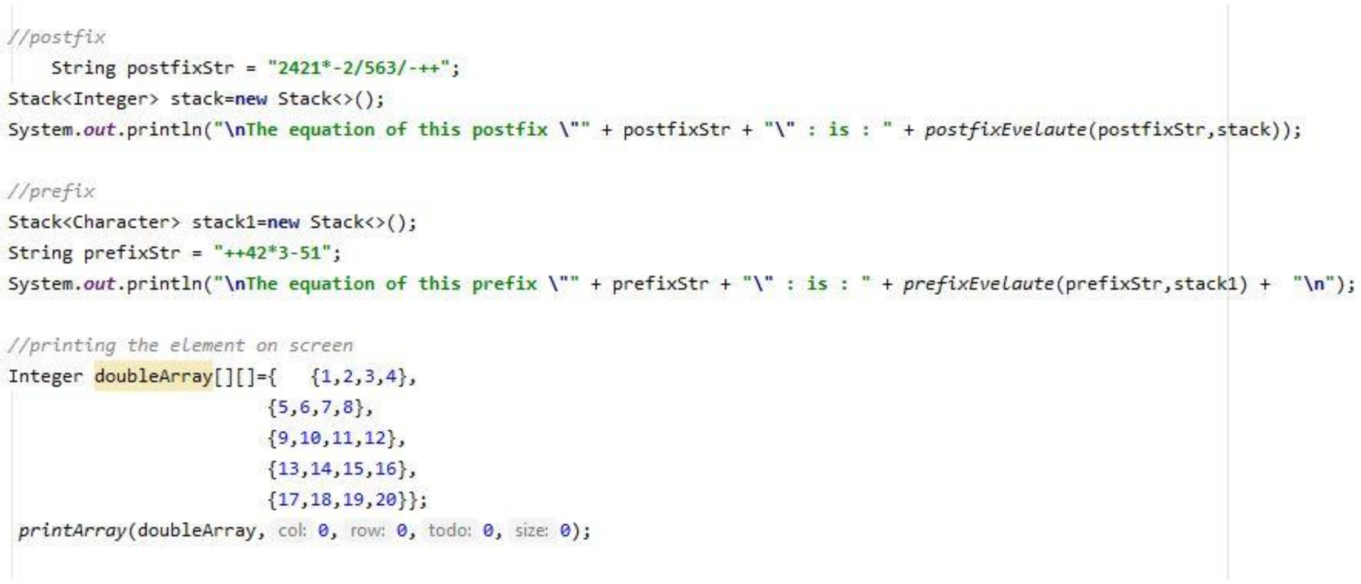


MAIN
main(String[]) void
reverse(String[], int) String
elfish(String, int) boolean
insertionSortRecursive(int[], int) void
postfixEvelaute(String, Stack<Integer>) int
prefixEvelaute(String, Stack<Character>) int
printArray(Integer[][], int, int, int, int) void
Powered by yFiles

5. TEST CASES

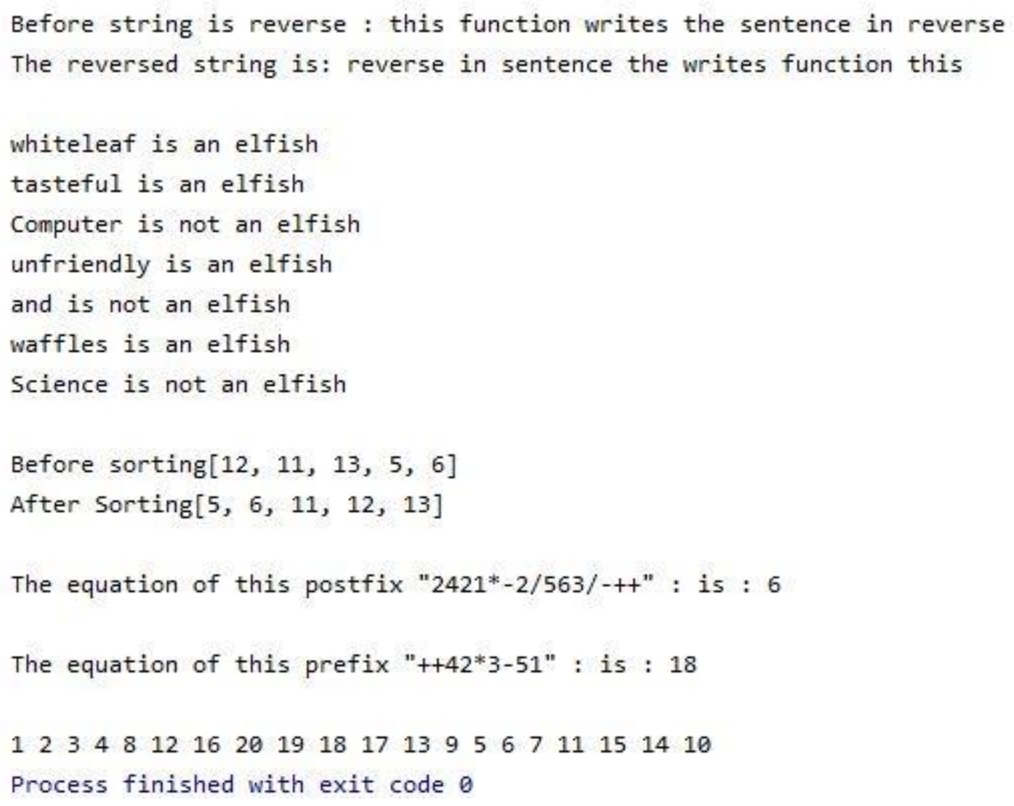


```
//reverse string
String str = "this function writes the sentence in reverse";
System.out.println("\nBefore string is reverse : " + str);
String[] strArr = str.split( regex: " ");
String reversed = reverse(strArr, index: 0);
System.out.println("The reversed string is: " + reversed);

//check elfish
System.out.println();
String[] strElfish = {"whiteleaf", "tasteful" , "Computer" , "unfriendly", "and", "waffles","Science"};
for(String strElf : strElfish){
    if(elfish(strElf, control: 0))
        System.out.println( strElf+" is an elfish");
    else
        System.out.println(strElf+ " is not an elfish");
}

// sorting array
System.out.println();
int intArr[] = {12, 11, 13, 5, 6};
System.out.println("Before sorting" + Arrays.toString(intArr));
insertionSortRecursive(intArr, intArr.length);
System.out.println("After Sorting" + Arrays.toString(intArr));
```



```
//postfix
    String postfixStr = "2421*-2/563/-++";
Stack<Integer> stack=new Stack<>();
System.out.println("\nThe equation of this postfix \"" + postfixStr + "\" : is : " + postfixEvelaute(postfixStr,stack));

//prefix
Stack<Character> stack1=new Stack<>();
String prefixStr = "++42*3-51";
System.out.println("\nThe equation of this prefix \"" + prefixStr + "\" : is : " + prefixEvelaute(prefixStr,stack1) +  "\n");

//printing the element on screen
Integer doubleArray[][]={   {1,2,3,4},
                        {5,6,7,8},
                        {9,10,11,12},
                        {13,14,15,16},
                        {17,18,19,20}};
 printArray(doubleArray, col: 0, row: 0, todo: 0, size: 0);
```

RUNNING AND RESULTS



```
Before string is reverse : this function writes the sentence in reverse
The reversed string is: reverse in sentence the writes function this

whiteleaf is an elfish
tasteful is an elfish
Computer is not an elfish
unfriendly is an elfish
and is not an elfish
waffles is an elfish
Science is not an elfish

Before sorting[12, 11, 13, 5, 6]
After Sorting[5, 6, 11, 12, 13]

The equation of this postfix "2421*-2/563/-++" : is : 6

The equation of this prefix "++42*3-51" : is : 18

1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10
Process finished with exit code 0
```