GTU Department of Computer Engineering CSE 222/505 - Spring 2020
Homework 04 Report

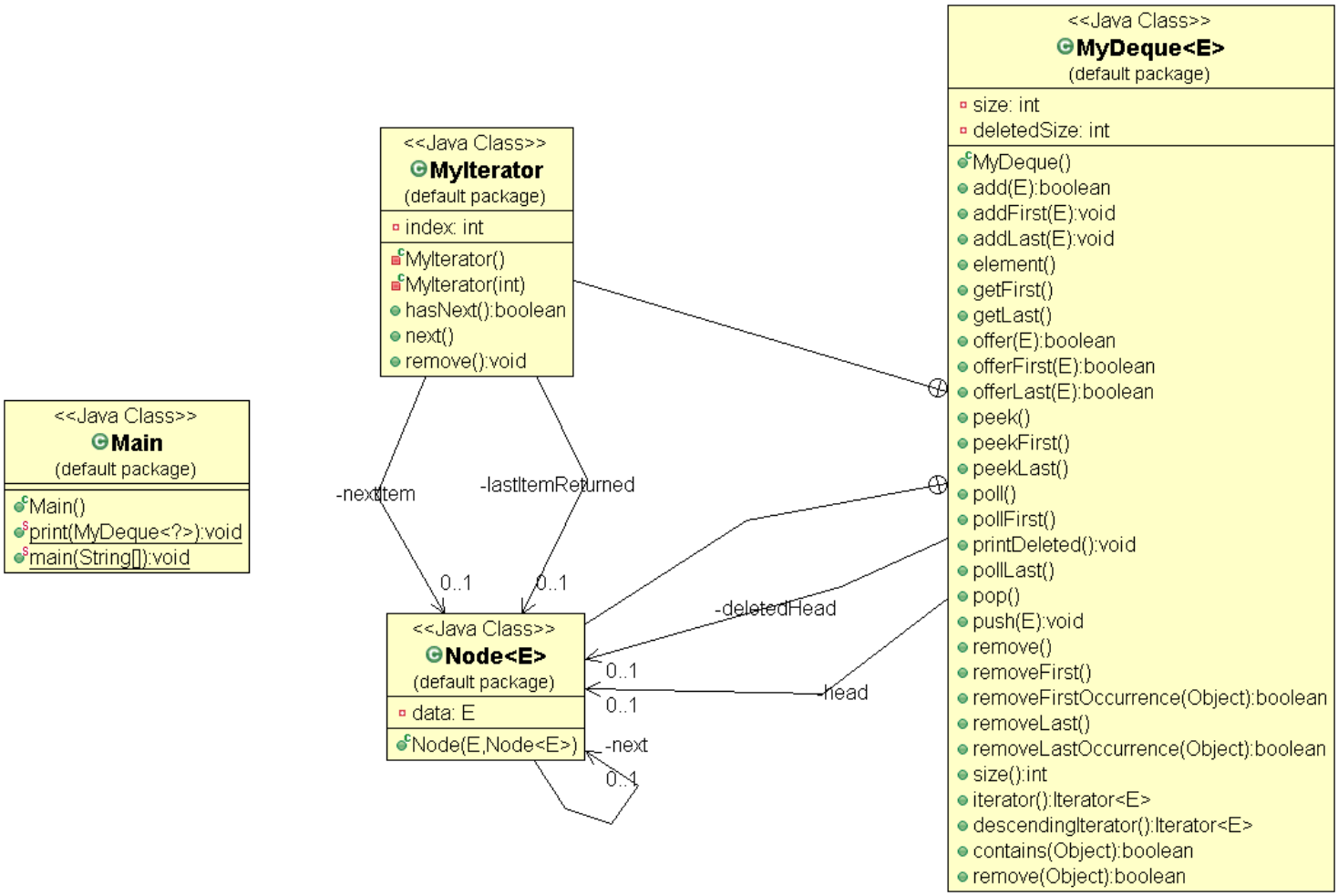
ESRA ERYILMAZ
171044046

-Q1-

All explanations are in 171044046.pdf

-Q2-

1. **CLASS DIAGRAMS**



2. PROBLEM SOLUTION APPROACH

I have MyDeque class. Inside that class I keep 4 fields. I keep ;
deque's size ,
deleted nodes size,
deque' s head which is actually front and
deleted nodes head.

While implement deque I use singly linked list.
Inside that class I implement all java Deque interface methods.
But I didn't implement some methods because these are not compatible with deque ADT.
And also I write my own Node and Iterator class.

While removing nodes I didn't delete node. I break connection than connect this removing node to the deleted linked list.
Also while adding nodes if there is a node inside deleted linked list I use that node without creating new node. I break connection with deleted linked list than connect to the my actual linked list.

3. TEST CASES

Firstly , I create MyDeque object and with that object I call all deque methods.

```
MyDeque<Integer> test = new MyDeque<Integer>();
```

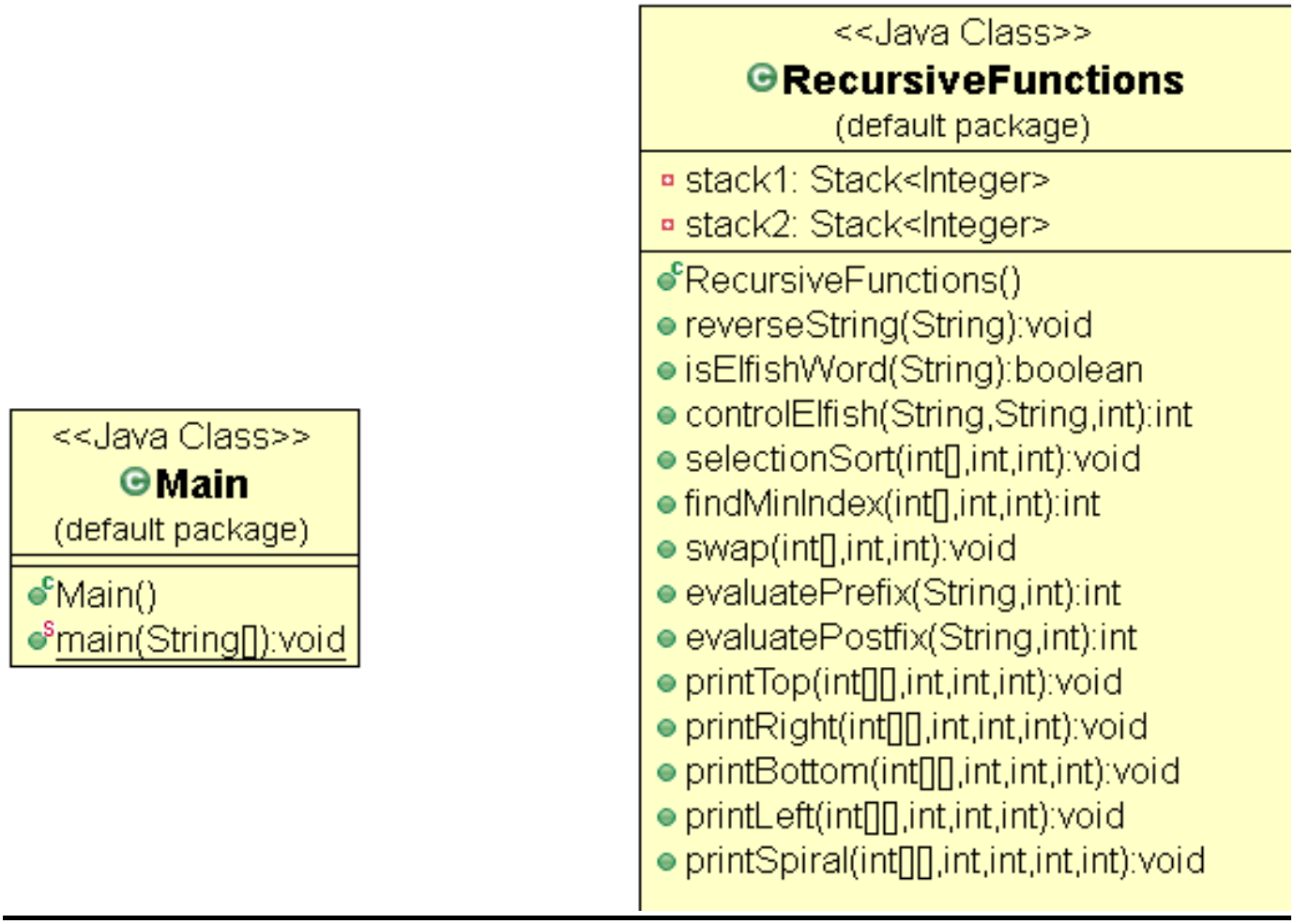
```
System.out.println("add(4)");       test.add(4);
System.out.println("addFirst(1)");  test.addFirst(1);
System.out.println("addLast(7)");   test.addLast(7);
print(test);

System.out.println("element()\t" + test.element());
System.out.println("getFirst()\t" + test.getFirst());
System.out.println("getLast()\t" + test.getLast());
print(test);

System.out.println("offer(5)  \t" + test.offer(5));
System.out.println("offerFirst(2)\t" + test.offerFirst(2));
System.out.println("offerLast(3)\t" + test.offerLast(3));
print(test);

System.out.println("peek()  \t" + test.peek());
System.out.println("peekFirst()\t" + test.peekFirst());
System.out.println("peekLast()\t" + test.peekLast());
print(test);


  System.out.println("poll()  \t" + test.poll());
  System.out.println("pollFirst()\t" + test.pollFirst());
  System.out.println("pollLast()\t" + test.pollLast());
  print(test);
  test.printDeleted();

  System.out.println("addLast(8)");   test.addLast(8);
  print(test);
  test.printDeleted();

  System.out.println("pop()   \t" + test.pop());
  System.out.println("push(9)");    test.push(9);
  print(test);

  System.out.println("remove()\t" + test.remove());
  System.out.println("removeFirst()\t" + test.removeFirst());
  System.out.println("removeLast()\t" + test.removeLast());
  print(test);

  System.out.println("size()  \t" + test.size());

  System.out.println("\n[I used iterator while printing deque.]");
```

4. RUNNING AND RESULTS

```
               TEST STARTING...
-----------------------------------------------
add(4)
addFirst(1)
addLast(7)
[PRINT DEQUE]   : 1  4  7

element()       1
getFirst()      1
getLast()       7
[PRINT DEQUE]   : 1  4  7

offer(5)        true
offerFirst(2)   true
offerLast(3)    true
[PRINT DEQUE]   : 2  1  4  7  5  3

peek()          2
peekFirst()     2
peekLast()      3
[PRINT DEQUE]   : 2  1  4  7  5  3

poll()          2
pollFirst()     1
pollLast()      3
[PRINT DEQUE]   : 4  7  5

[PRINT DELETED NODES: 2  1  3]

addLast(8)
[PRINT DEQUE]   : 4  7  5  8

[PRINT DELETED NODES: 1  3]

pop()           4
push(9)
[PRINT DEQUE]   : 9  7  5  8

remove()        9
removeFirst()   7
removeLast()    8
[PRINT DEQUE]   : 5

size()          1

[I used iterator while printing deque.]


-----------------------------------------------
               TEST FINISHED...
```

-Q3-

1. **CLASS DIAGRAMS**



2. PROBLEM SOLUTION APPROACH

- *Problem 1*

Firstly , I take the string and then with help of the java String methods ; I find last index of the space then print after that space, then I update my string then find last index of the space again then print it again recursively.
My base case is ; if my string is null or there is no space anymore.

- *Problem 2*

I take index(initially 0) word and control word which is "elf". With that word I control 'e' if my word contains then increase index with 1. Then I control 'l' and 'f' recursively.
My base case is : if control word has no more characters inside then return the index. With that index I return true if index is 3 and false otherwise. Because "elf" has 3 characters.

- *Problem 3*

I take array of elements and with starting with first index then with helper function I find minimum element's index ,if my index is not equal min index then I swap these elements.
With increasing index I call my function again and again recursively.
My base case is if index is equal to the array length than return because at that time array is sorted.

- *Problem 4*

I take the string prefix expression then ,to differentiate spaces I make some operations. I'm getting after the last space.Then evaluate this string contains numbers or operator. If number than push to the stack if operator after evaluation I push to the stack. With updating string and looking at the last space recursively I reach the result.
My base case is; if there is no space in my string than pop to the stack and return it because the stack have only one element and that is result.

(For testing this method in the given string ; there must be a space between each element and at the beginning there must be space too.
*For exp: " + 9 * 20 6")*

- *Problem 5*

  I take the string postfix expression then ,to differentiate spaces I make some operations. I'm getting after the first space.Then evaluate this string contains numbers or operator. If number than push to the stack if operator after evaluation I push to the stack. With updating string and looking at the first space recursively I reach the result.

  My base case is; if there is no space in my string than pop to the stack and return it because the stack have only one element and that is result.

  (For testing this method in the given string ; there must be a space between each element and at the end there must be space too.
  *For exp: "4 50 7 20 + - * ")*

- *Problem 6*

  I take mxn matrix and than print first row and right side and bottom and left side , then updating rows columns and some helper variables then print it again recursively .While printing I use some helper functions.

3. TEST CASES

Firstly , I create RecursiveFunctions object and with that object I call all methods.

```
RecursiveFunctions obj = new RecursiveFunctions();
```

The test of all recursive problems is respectively below

```
System.out.println("\n...PROBLEM 1...");
String input = "this function writes the sentence in reverse";
System.out.println("Given input :\t\"this function writes the sentence in reverse\"");
System.out.print("Output :\t");
obj.reverseString(input);


System.out.println("\n...PROBLEM 2...");
System.out.println("Is whiteleaf elfish?\t" + obj.isElfishWord("whiteleaf"));
System.out.println("Is stayhome elfish?\t" + obj.isElfishWord("stayhome"));
System.out.println("Is unfriendly elfish?\t" + obj.isElfishWord("unfriendly"));
System.out.println("Is television elfish?\t" + obj.isElfishWord("television"));


System.out.println("\n...PROBLEM 3...");
int[] arr = {4, 2, 5, 1, 7};
obj.selectionSort(arr, arr.length, 0);
System.out.println("Unsorted array :   4 2 5 1 7");
System.out.print("Sorted array   :   ");
//printing sorted array
for(int temp : arr)
    System.out.print(temp + " ");


System.out.println("\n\n...PROBLEM 4...");
String prefix = " + 9 * 20 6";  //result 129
System.out.println("Prefix expression :  + 9 * 20 6");
System.out.println("Evaluate prefix  :\t" + obj.evaluatePrefix(prefix, prefix.lastIndexOf(" ")));


System.out.println("\n...PROBLEM 5...");
String postfix = "4 50 7 20 + - * ";    //result 92
System.out.println("Postfix expression : 4 50 7 20 + - * ");
System.out.println("Evaluate postfix  :\t" + obj.evaluatePostfix(postfix,postfix.indexOf(" ")));


System.out.println("\n...PROBLEM 6...");
System.out.println("Input:\n\t1  2  3  4 \n\t5  6  7  8 \n\t9  10 11 12 \n\t13 14 15 16 \n\t17 18 19 20");
System.out.println("Output:");
int[][] matrix = {  {1, 2, 3, 4},
                    {5, 6, 7, 8},
                    {9, 10, 11, 12},
                    {13, 14, 15, 16},
                    {17, 18, 19, 20}
                 };

obj.printSpiral(matrix, 5, 4, 0, 0);
```

4. RUNNING AND RESULTS

```
                TEST STARTING...
------------------------------------------------------------

...PROBLEM 1...
Given input :   "this function writes the sentence in reverse"
Output :        reverse in sentence the writes function this

...PROBLEM 2...
Is whiteleaf elfish?    true
Is stayhome elfish?     false
Is unfriendly elfish?   true
Is television elfish?   false

...PROBLEM 3...
Unsorted array :   4 2 5 1 7
Sorted array   :   1 2 4 5 7

...PROBLEM 4...
Prefix expression :  + 9 * 20 6
Evaluate prefix  :      129

...PROBLEM 5...
Postfix expression : 4 50 7 20 + - *
Evaluate postfix  :     92

...PROBLEM 6...
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

------------------------------------------------------------
                TEST FINISHED...
```