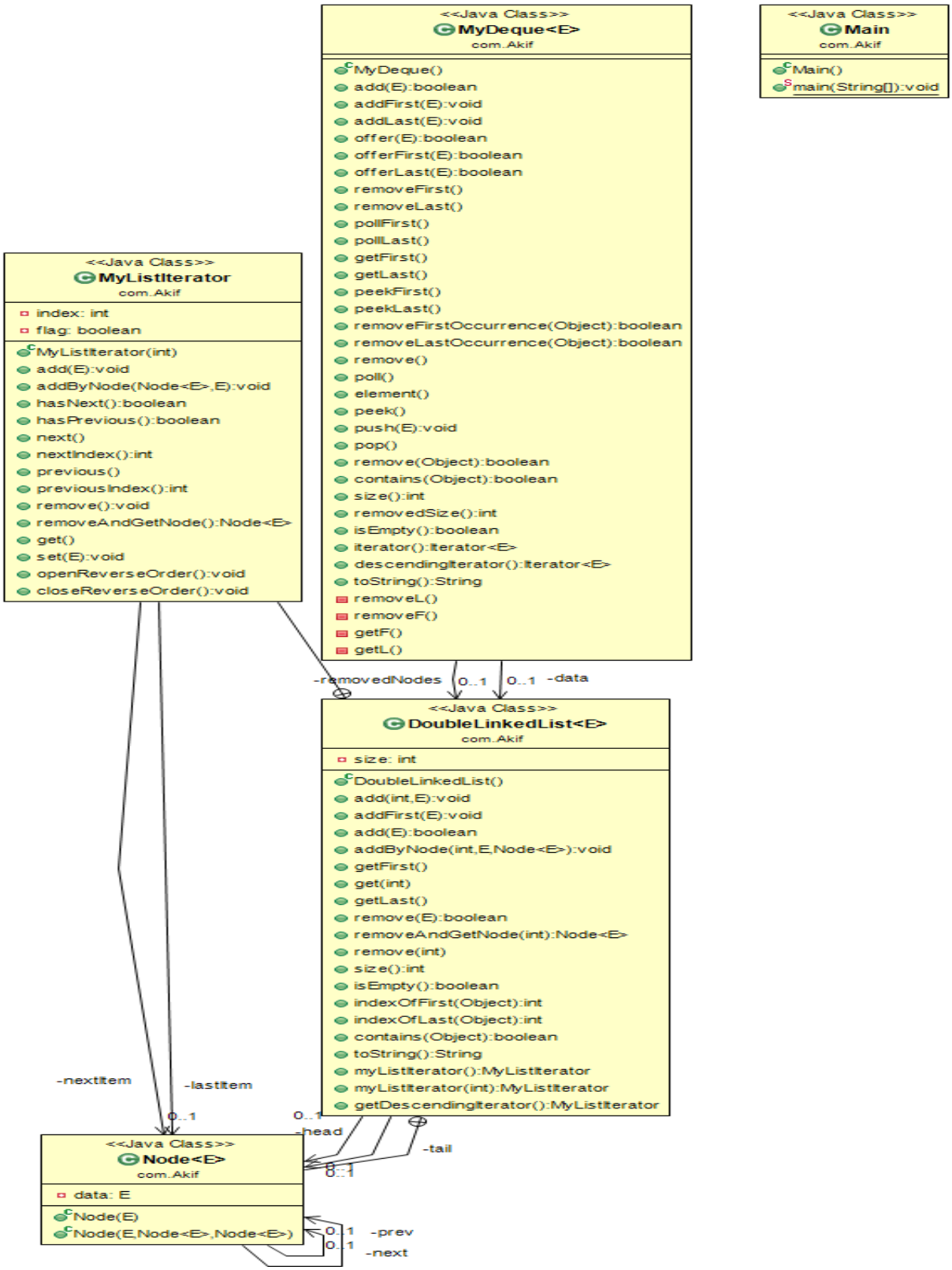
GIT Department of Computer Engineering CSE 222/505 - Spring 2020 Homework 4 Report

Akif KARTAL
171044098

Q2 REPORT

1. CLASS DIAGRAMS



2. PROBLEM SOLUTION APPROACH

My Problem solution steps are;

– Specify the problem requirements
– Analyze the problem
– Design an algorithm and Program
– Implement the algorithm
– Test and verify the program
– Maintain and update the program

1) **Specify the problem requirements :** I understand the problem.
2) **Analyze the problem :** I identify;
   – Input data
   – Output data
   – Additional requirements and constraints

3) **Design an algorithm and Program :** I divide the problem into sub-problems. I listed major steps (sub-problems). I break down each step into a more detailed list. To do these We have to divide this big project into small pieces.

   Implement the algorithm : I wrote the algorithm in Java by converting each step into statements of Java (classes ,methods etc.)

   First I started to write DoubleLinkedList class to keep my data. Then ,In this class I wrote all necessary methods separately and clear.

   In this class to create a full linked list class I wrote MyListIterator and Node class as an inner class. By using this two class I wrote all necessary methods for a linked list and for this question.

   I wrote MyListIterator class which implements ListIterator and Iterator interface. In this class I wrote all necessary method for an original iterator class. In this class I added addByNode() extra method to add new element by using an old node without creating a new node.

   And in DoubleLinkedList class, I did linked list operations by help of MyListIterator and Node class. In this class I added removeByNode() extra method which returns removed node to keep removed node reference to use again.

   Secondly, I wrote MyDeque class which implements Deque interface and extends from AbstractCollection class. In this class I wrote all necessary method for an original Deque class. In this class I keep two DoubleLinkedList class references to keep data. I did all operation of deque my using methods of DoubleLinkedList class.

4) **Test and verify the program:** To test program I wrote the Main class in this class in main method I test each method of MyDeque class by calling the methods and I printed the test results separated and in a good shape.
5) **Maintain and update the program :** I keep the program up-to-date.

3. TEST CASES

Test ID	Test Case	Pass/Fail
T1	Test all “add” methods	Pass
T2	Test all “remove” methods	Pass
T3	Test all “offer” methods	Pass
T4	Test all “poll” methods	Pass
T5	Test all “get” methods	Pass
T6	Test all “peek” methods	Pass
T7	Test “push” and “pop” methods	Pass
T8	Test “element” method	Pass
T9	Test other methods	Pass
T10	Test iterator and descendingIterator methods	

4. RUNNING AND RESULTS

Test ID	Test Result
T1	---------- Add 6(0 to 5) piece element with 'addFirst' method Data: 5==>4==>3==>2==>1==>0 Removed Data: Empty Size: 6 Removed Size: 0 ---------- Add 6(5 to 10) piece element with 'add' method Data: 5==>4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10 Removed Data: Empty Size: 12 Removed Size: 0 ---------- Add 5(10 to 15) piece element with 'addLast' method Data: 5==>4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>11==>12==>13==>14==>15 Removed Data: Empty Size: 17 Removed Size: 0

T2

```
Data: 5==>4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>11==>12==>13==>14==>15
Removed Data: Empty

Size: 17
Removed Size: 0
--------------------------------------------------------------------------------
Remove element with 'remove()' method
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>11==>12==>13==>14==>15
Removed Data: 5

Size: 16
Removed Size: 1
--------------------------------------------------------------------------------
Remove element with 'removeLast' method
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>11==>12==>13==>14
Removed Data: 5==>15

Size: 15
Removed Size: 2
--------------------------------------------------------------------------------
Remove '11' with 'remove(11)' method
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 5==>15==>11

Size: 14
Removed Size: 3
```

```
Data: 3==>4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>5==>15
Removed Data: Empty

Size: 17
Removed Size: 0
--------------------------------------------------------------------------------
Remove '5' with 'removeLastOccurrence()' method
Data: 3==>4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>15
Removed Data: 5

Size: 16
Removed Size: 1
--------------------------------------------------------------------------------
Remove '3' with 'removeFirstOccurrence()' method
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>15
Removed Data: 5==>3

Size: 15
Removed Size: 2
```

T3

```
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 5==>15==>11

Size: 14
Removed Size: 3
--------------------------------------------------------------------------------
Add '5' with 'offer()' method
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>5
Removed Data: 15==>11

Size: 15
Removed Size: 2
--------------------------------------------------------------------------------
Add '15' with 'offerLast()' method
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>5==>15
Removed Data: 11

Size: 16
Removed Size: 1
--------------------------------------------------------------------------------
Add '3' with 'offerFirst()' method
Data: 3==>4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>5==>15
Removed Data: Empty

Size: 17
Removed Size: 0
```

T4

```
Data: 4==>3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>15
Removed Data: 5==>3

Size: 15
Removed Size: 2
--------------------------------------------------------------------------------
Remove element with 'poll()' method
Data: 3==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>15
Removed Data: 5==>3==>4

Size: 14
Removed Size: 3
--------------------------------------------------------------------------------
Remove element with 'pollFirst()' method
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14==>15
Removed Data: 5==>3==>4==>3

Size: 13
Removed Size: 4
--------------------------------------------------------------------------------
Remove element with 'pollLast()' method
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 5==>3==>4==>3==>15

Size: 12
Removed Size: 5
```

T5

```
Get methods test results
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 3==>4==>3==>15==>8

getFirst(): 2
getLast(): 14
```

T6

```
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 3==>4==>3==>15==>8

peek(): 2
peekFirst(): 2
peekLast(): 14
```

T7

```
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 5==>3==>4==>3==>15

Size: 12
Removed Size: 5
------------------------------------------------------------
Add '8' with 'push()' method
Data: 8==>2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 3==>4==>3==>15

Size: 13
Removed Size: 4
------------------------------------------------------------
Remove element with 'pop()' method
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 3==>4==>3==>15==>8

Size: 12
Removed Size: 5
```

T8

```
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 3==>4==>3==>15==>8

element(): 2
```

T9

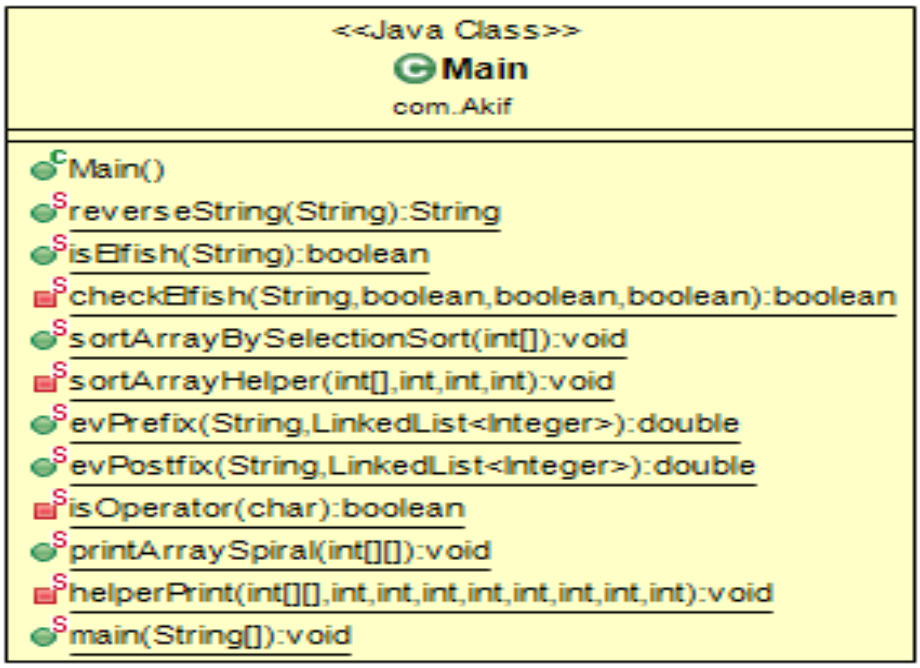
```
Other test results
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 3==>4==>3==>15==>8

contains(111): false
isEmpty(): false
```

T10

```
Data: 2==>1==>0==>5==>6==>7==>8==>9==>10==>12==>13==>14
Removed Data: 3==>4==>3==>15==>8
-----------------------------------------
Print all list with iterator
2 1 0 5 6 7 8 9 10 12 13 14
-----------------------------------------
Print all list with descending iterator
14 13 12 10 9 8 7 6 5 0 1 2
-----------------------------------------
```

Q3 REPORT

1. CLASS DIAGRAMS



2. PROBLEM SOLUTION APPROACHES

1. Reversing a string
   - **Identify the base case (or base cases) that stops the recursion;**
     I identify two base case ;
     a. If the string is null or empty stop the recursion
     b. If the string contains only one word stop the recursion

- **Define the smaller problem (or problems);**

  The sub problem is reverse just one word to do this split the word from spaces.
- **Explain how to combine solutions of smaller problems to get the solution of original problem;**

  To reverse the string split the string from spaces and when splitting is done, start to return each separated word. While returning combine all word in one string.

  To get all word in the string call reverse function again with substring of original string such that a string from first whitespace in the original.

2. Determining whether a word is elfish or not.
   - **Identify the base case (or base cases) that stops the recursion;**

     I identify one base case ;
     a. If the string is null or empty stop the recursion
   - **Define the smaller problem (or problems);**

     The sub problem is check just one character of the string whether it is e or l or f or not.
   - **Explain how to combine solutions of smaller problems to get the solution of original problem;**

     While checking just one character of the string, If it finds the necessary letter for an elfish save it and after all string is done check the saved values.
3. Sorting an array of elements using selection sort algorithm
   - **Identify the base case (or base cases) that stops the recursion;**

     I identify one base case ;
     a. If it traverses all element of the array stop the recursion
   - **Define the smaller problem (or problems);**

     The sub problem is finding the minimum element every recursive call then after finding, swap it as leftmost element in the array as in original selection sort algorithm.
   - **Explain how to combine solutions of smaller problems to get the solution of original problem;**

     After checking for just one element in array then swap the values as in selection algorithm after swapping continue to sorting(call again) until all element is done.

4. Evaluating a Prefix expression
   - **Identify the base case (or base cases) that stops the recursion;**

     I identify one base case ;
     a. If the expression is null or empty stop the recursion.
   - **Define the smaller problem (or problems);**

     The sub problem is evaluate only one string from separated the expression from spaces.

Explain how to combine solutions of smaller problems to get the solution of original problem;

To evaluate the expression first reverse the expression by using 1th recursive method that is written after this use the recursive method that is written to evaluate postfix expression because process is same after reversing.

In the postfix recursive method split the expression from spaces and evaluate each separated string to determine whether it is an operand or operator

After all evaluation return the result.

5. Evaluating a Postfix expression
   - **Identify the base case (or base cases) that stops the recursion;**

     I identify one base case ;

     a. If the expression is null or empty stop the recursion.
   - **Define the smaller problem (or problems);**

     The sub problem is evaluate only one string that is separated the expression from spaces.

     Explain how to combine solutions of smaller problems to get the solution of original problem;

     In each recursive call first split the expression from spaces and evaluate each separated string to determine whether it is an operand or operator and do necessary operation according given token. To make evaluation use a helper method called “isOperator” to determine it is an operator or not. After determining use a linked list to keep the result as a stack.

     After all evaluation return the result from list.
6. Printing the elements of an array on the screen
   - **Identify the base case (or base cases) that stops the recursion;**

     I identify one base case ;

     a. If it prints all element of the 2D array stop the recursion
   - **Define the smaller problem (or problems);**

     The sub problem is printing just one element as expected.

- **Explain how to combine solutions of smaller problems to get the solution of original problem;**

  In each recursive call by using index values of array and some other values it prints correct value on the screen. To do this we keep some values and by playing and checking these values by using observation and simple mathematic we solve the asked problem. When we change the numbers, we are checking the directions. Directions are important for this question.

3. TEST CASES

Test ID	Test Case	Pre-Condition	Pass/Fail
T1	Test 1th method.	- Provides parameters	**Pass**
T2	Test 2th method.	- Provides parameters	**Pass**
T3	Test 3th method.	- Provides parameters	**Pass**
T4	Test 4th method.	- Provides parameters	**Pass**
T5	Test 5th method.	- Provides parameters	**Pass**
T6	Test 6th method.	- Provides parameters	**Pass**

4. RUNNING AND RESULTS

Test ID	Test Result
T1	---------- Test 1th method with 'this function writes the sentence in reverse' sentence. Result: reverse in sentence the writes function this ----------
T2	---------- Test 2th method with 'whiteleaf' word. Result: true Test 2th method with 'result' word. Result: false ----------
T3	---------- Test 3th method with '10 8 5 11 0' array Result: 0 5 8 10 11 ----------
T4	---------- Test 4th method with '+ 7 + / - 10 * 5 3 10 - 2 / 4 8' prefix notation Result: 9.0 ----------
T5	---------- Test 5th method with '4 7 * 20 -' postfix notation Result: 8.0 ----------
T6	---------- Test 6th method with following array 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 Result: 1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10 ----------