GIT Department of Computer Engineering

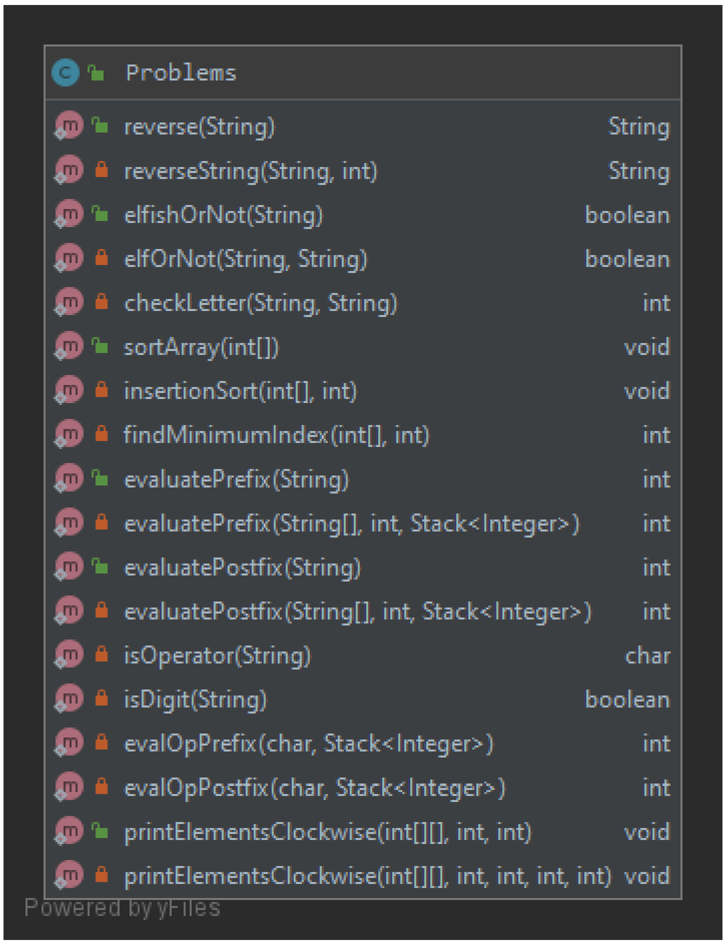
CSE 222/505 – Spring 2020

Homework #04 Part 3 Report

Abdullah ÇELİK

171044002

Class Diagram



Problems	
reverse(String)	String
reverseString(String, int)	String
elfishOrNot(String)	boolean
elfOrNot(String, String)	boolean
checkLetter(String, String)	int
sortArray(int[])	void
insertionSort(int[], int)	void
findMinimumIndex(int[], int)	int
evaluatePrefix(String)	int
evaluatePrefix(String[], int, Stack<Integer>)	int
evaluatePostfix(String)	int
evaluatePostfix(String[], int, Stack<Integer>)	int
isOperator(String)	char
isDigit(String)	boolean
evalOpPrefix(char, Stack<Integer>)	int
evalOpPostfix(char, Stack<Integer>)	int
printElementsClockwise(int[][], int, int)	void
printElementsClockwise(int[][], int, int, int, int)	void

Problem Solution Approach

1. Problem

Definition: 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”.

Method signature :

String reverseString(String str, int index)

Base Case:

If string does not contain any spaces from the index to the end, returns the part from index to the end.

Smaller Problems:

a. append reverse string to the part of the string from index to space character
b. The index parameter of the next recursion call is one more than the index of the space character

Solution:

Method finds the first space character starting from the first index of the string. After finding, the b smaller problem works. This process continues until base case and base case returns the last word. After this, every recursion call apply the a smaller problem. Finally, method returns reverse of the string.

2. Problem

Definition: 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.

Method signature:
    boolean elfOrNot(String str, String letter)
Helper method signature:
    int checkLetter(String str, String letter)

Base Cases:

a. If the size of the letter string in the parameter of the elfOrNot method is 0, returns true.
b. If the size of the str in the parameter of the elfOrNot method is 0, returns false.

Smaller Problems:

a. Checking whether the first character of the str is one of the character sougth.
b. If the first character of the str is the searched characters, substracting that searched character from the letter.

Solution:

The method applies the a smaller problem for str. It uses helper method fort his. If first character of str is a searched characters, returns that searched character's index. Otherwise returns -1. If returned value is not equal -1, method applies the b smaller problem. Until any of the base cases next recursion calls as subtracting first character of the str.

3. Problem

Definition: Sorting an array of elements using selection sort algorithm.

Method signature:
    void insertionSort(int[] arr, int index)
Helper method signature:
    int findMinimumIndex(int[] arr, int index)

Base Case:

The index in the parameter of the insertionSort method is greater or equal than size of the array

Smaller Problems:

a. Finding index of the minimum number from the index in the parameter of the findMinimumIndex method to end.
b. Replacing the number in the index with smallest number.

Solution:

Method applies the a and b smaller problems respectively as long as base case is provided. Then the next recursion is called by increasing the index parameter one.

4. Problem

Definition: Evaluating a Prefix expression

Method signature:

int evaluatePrefix(String[] prefix, int index, Stack<Integer> stack) throws Exception

Helper methods signature:

char isOperator(String str)
boolean isDigit(String str)
int evalOpPrefix(char operator, Stack<Integer> stack)

Base Case:

If the index in the parameter of the evaluatePrefix method is lower than 0, returns the top of the stack.

Smaller Problems:

a. Check whether the index of the prefix is the operator
b. Check whether the index of the prefix is the digit

Solution:
Method applies the a smaller problem. If, as a result, the expression in the index is an operator, the evalOpPrefix method is called. This method calculates according to the operator and pushes the result into the stack. If the expression is not an operator, method applies the b smaller problem. isDigit method is called. If this is a number, the stack is pushed. If not, an exception is thrown. The next recusion call is made by reducing the index by one until base case is provided.

5. Problem

Definition: Evaluating a Postfix expression

Method signature:

int evaluatePostfix(String[] postfix, int index, Stack<Integer> stack) throws Exception

Helper methods signature:

char isOperator(String str)
boolean isDigit(String str)
int evalOpPostfix(char operator, Stack<Integer> stack)

Base Case:

If the index in the parameter of the evaluatePostfix method is greater or equal than postfix expression length, returns the top of the stack.

Smaller Problems:

a. Check whether the index of the prefix is the operator
b. Check whether the index of the prefix is the digit

Solution:

Method applies the a smaller problem. If, as a result, the expression in the index is an operator, the evalOpPostfix method is called. This method calculates according to the operator and pushes the result into the stack. If the expression is not an operator, method applies the b smaller problem. isDigit method is called. If this is a number, the stack is pushed. If not, an exception is thrown. The next recusion call is made by increasing the index by one until base case is provided.

6. Problem

Definition: Printing the elements of an array on the screen as in the example below

Method signature:

void printElementsClockwise(int[][] arr, int xPos, int yPos, int row, int column)

Base Case:

xPos is greater or equal than the column or yPos is greater or equal than the row.

Smaller Problems:

a. Printing on the screen first row from xPos to end
b. Printing on the screen last column from yPos to end
c. Printing on the screen if xPos is not equal last row from end to yPos
d. Printing on the screen if yPos is not equal last column from end to xPos

Solution:

Method applies the a,b,c and d smaller problems until base case is provided. The next recusion call is made by increasing the xPos and yPos by one and decreasing the row and column by one.

Test Cases

Test ID	Scenerio	Test Data	Expected Results	Actual Results	Pass/ Fail
TEST01	Testing reverse method when string size is 0	String size : 0	Successfully reversed string	As expected	Pass
TEST02	Testing reverse method when string size is 4 and string contains only one word	String size : 4	Successfully reversed string	As expected	Pass
TEST03	Testing reverse method when string size is 44 and string contains several words	String size : 44	Successfully reversed string	As expected	Pass
TEST04	Testing elfishOrNot method when word is not elfish	Word : "elise"	Successfully returned false	As expected	Pass
TEST05	Testing elfishOrNot method when word is elfish	Word : "tasteful"	Successfully returned true	As expected	Pass
TEST06	Testing sortArray method when array is unsorted	Array size : 13	Successfully sorted array	As expected	Pass
TEST07	Testing sortArray method when array is sorted	Array size : 7	Successfully sorted array	As expected	Pass
TEST08	Testing evaluatePrefix method	Prefix : + 2 + / - 8 * 2 3 2 - 8 / 10 5	Successfully returned right result	As expected	Pass
TEST09	Testing evaluatePrefix method with invalid character	Prefix : + 2 + / - 8 < 9	Expected Exception	As expected	Pass
TEST10	Testing evaluatePostfix method	Postfix : 2 8 2 3 * - 2 / + 8 + 10 5 / -	Successfully returned right result	As expected	Pass
TEST11	Testing evaluatePostfix method with invalid character	Postfix : 2 8 2 3 < - 2 / + 8 + 10 5 / -	Expected Exception	As expected	Pass
TEST12	Testing printElementsClockwise method when array has only one row	Array length: Row : 1 Column : 4	Successfully printed clockwise on the screen	As expected	Pass
TEST13	Testing printElementsClockwise method when array has only one column	Array length: Row : 5 Column : 1	Successfully printed clockwise on the screen	As expected	Pass

TEST14	Testing printElementsClockwise method when array has some rows and columns	Array length: Row : 5 Column : 4	Successfully printed clockwise on the screen	As expected	Pass

Running and Results

```
TEST01

When string's length is 0
Before reverse method
String :   Size : 0
After reverse method

TEST02

When string's length is greater than 0 but contains one word
Before reverse method
String : DATA  Size : 4
After reverse method
DATA

TEST03

When string's length is greater than 0 and contains several words
Before reverse method
String : this function writes the sentence in reverse  Size : 44
After reverse method
reverse in sentence the writes function this

TEST04

When word is not elfish
Word : elise
It is elfish ? false

TEST05

When word is elfish
Word : tasteful
It is elfish ? true

TEST06
When array is unsorted
Array size : 13
5 2 6 9 -1 4 -4 0 3 12 1 8 -10
After sortArray method
-10 -4 -1 0 1 2 3 4 5 6 8 9 12

TEST07
When array is sorted
Array size : 7
0 1 3 5 6 7 9
After sortArray method
0 1 3 5 6 7 9

TEST08
Prefix expression : + 2 + / - 8 * 2 3 2 - 8 / 10 5
Result : 9
```

```
TEST09
When prefix expression contains invalid characters
Prefix expression : + 2 - / - 8 < 9
Exception was caught!
Invalid character!

TEST10
Postfix expression : 2 8 2 3 * - 2 / + 8 + 10 5 / -
Result : 9

TEST11
When postfix expression contains invalid characters
Postfix expression : 2 8 2 3 < - 2 / + 8 + 10 5 / -
Exception was caught!
Invalid character!

TEST12
When array has only one row
Array :
1 2 3 4
After printElementsClockwise method
1 2 3 4

TEST13
When array has only one column
Array :
1
2
3
4
5
After printElementsClockwise method
1 2 3 4 5

TEST14
When array has some rows and columns
Array :
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
After printElementsClockwise method
1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10
```