Hacettepe University: Computer Science and Engineering Department

Name and Surname: Mert Tazeoğlu
Identity Number: 21946606
Course: BBM104
Subject: Stack and Queue Operations By Java
Due Date: 10.06.2020 – (23:59)
Advisors: Bahar Gezici – Nebi Yılmaz
E-Mail: b21946606@cs.hacettepe.edu.tr
Main Program: HU_CENG Submit System > b21946606 > Main.Txt

Table Of Contents **Page Number**

2. Software Using Documentation

2.1 Software Usage: Our program uses one input file (command.txt) in argument form and two input files (stack.txt and queue.txt) in program directory for applying commands in input file. At the end, our program makes required operations and write 2 new output files.

2.2 Error Messages: There two error messages:

1) `"An Error Occured While Reading File! Please Try Again!"`

This error message occurs when reading input file. For handling this situation, you only need to be sure that input file is in correct format.

2) `"Queue Is Underflow!"`

This error message occurs when queue.peek() operation, if queue is empty. For handling this situation you must be sure that there isn't any logical errors in command file operations.

3. Software Design Notes

3.1 Description Of Problem

3.1.1 Problem: We need to take data with input files, save it in data structures of stack and queue. Also we need to make some operatons on data.

3.1.2 Solution: With using data structures like stack and queue, we can save data. Also with data structures, we are able to make different operations on data.

3.2 Algorithm

1. Read input files of stack.txt and queue.txt, and save their data in stack and queue.
2. Read input file of command.txt and determine which operations will be executed.
3. Execute required operations in command.txt.
4. Update stack.txt and queue.txt with updates stack and queue files.
5. Write logs of executed operations in stackOut.txt and queueOut.txt files.

3.3 Special Design Notes

* My stack class is extending Vector<E> class. Because it makes implementing class and processing our operations easier. Also i can have automatically overrided functions like stack.get(index) and stack.remove(index).
* Also my queue class is implemented with using nodes.

4. Algorithm Complexity Analysis

distinctElements() method finds number of unique elements in a stack. This method has got basically 2 for loops. Because of that, complexity of this algorithm is O(n*m).
(n=StackSize; m=Second Loops Repeat Time)

If we implement same algorithm not only with stack and queues, but also with a hashset (for removing duplicates); complexity of algorithm could be O(2n). My algorithm, which has O(n*m) complexity is implemented like that:

```java
public static void distinctElements(){
    // This method finds number of unique elements in stack.
    int res = 1;
    for (int i = 0; i < stackSize; i++){
        int j = 0;
        for (j = 0; j < i; j++)
            if (stack.get(i) == stack.get(j))
                break;
        if (i == j)
            res++;
    }
    FileOperations.stackOut.add("After distinctElements:");
    FileOperations.stackOut.add("Total distinct element=" + res);
}
```

5. References

<Problem Definition Reference>

By BBM104 Team
Assignment4.pdf
Published On Piazza Platform
Published On 22.05.2020