Programming Assignment 4

Submission Date : 26.08.2021
Due Date : 02.09.2021 (23:59)
Subject : Stack and Queue Operations

1 Problem

In this assignment, you will introduce with data structures such as queue, stacks, etc. You will take an input file (command.txt) as an argument from the command line. According to this input file, you will make some operations and print the output of stack and queue files at the end of the input file operations.

You will be given only one input files namely command.txt. In this assignment, you are expected to create one stack and queue file and perform various operations with these created files. The first operation in command.txt always will be the "insertNumbers" for each stack and queue (each stack and queue can contain up to 100 elements). You will create the required stack.txt and queue.txt files with this operation.You will update the data in the stack and the queue files according to the each operations that are included in the command.txt file. If the commands in the text file start with "S", that command is performed for the stack, but if it starts with "Q", it is performed for the queue. After each of the operations performed, the results also are printed into the stackOut.txt file for the stack and the queueOut.txt file for the queue.

1. Insert Given Numbers

In this operation, given numbers will be inserted within given order into the required queues or stacks according to given command. The number to be added can be any number between 1 and 100.

- Command format in the Command.txt file for this operation:

S[space] insertNumbers [space] s (for stack)

Q[space] insertNumbers [space] q (for queue)

- Format of the s and q is like that:

number*[space]*number*[space]*number*[space]*....number

2. Clear Stack or Queue

In this operation, you will remove all elements from queues or stacks and show the content of stack or queue.

- Command format in the Command.txt file for this operation:

S[space] clear (for stack)

Q[space] clear (for queue)

3. Check Number

In this operation, a positive k number is read from the input file. If this number is even, then add this number to the queue or stack after removing maksimum number in stack or queue. If this number is odd, then add this number to the queue or stack after removing min number from the queue or stack. After this operation, the final version of the stack or queue is printed on the their output files.

- Command format in the Command.txt file for this operation:

S[space] checkNumber [space] k (for stack)

Q[space] checkNumber [space] k (for queue)

You must use ONLY queue and stack, don't use other data structures such as pure array structure, string, etc.

4.Calculate Fibonacci Numbers

In this operation, you will insert first k fibonacci number starting with 1 and 1 number as fib(0) and fib(1) to the stack or queue, and update them as reverse order. After this operation, the final version of the stack or queue is printed on the their output files.

- Command format in the Command.txt file for this operation:

S[space] calculateFib [space] k (for stack)

Q[space] calculateFib [space] k (for queue)

5. Reverse Elements

In this operation, according to integer k that is given in command.txt. If the integer k is positive, then reverse the first k elements of stack or queue. If the integer k is negative, then reverse the last k elements of stack or queue. After this operation, the final version of the stack or queue is printed on the their output files.

- Command format in the Command.txt file for this operation:

S[space] reverse [space] k (for stack)

Q[space] reverse [space] k (for queue)

6. Existence of Specified Number

In this operation, you will print 0 or 1 by considering integer k. If integer k exist in the queue or stack, then print 1, if it does not exist in the queue or stack, then print 0.

- Command format in the Command.txt file for this operation:

S[space] isExist [space] k (for stack)

Q[space] isExist [space] k (for queue)

7. Summation of Distinct Elements

In this operation, you are expected to find the summation of distinct elements there are in stack and queue files.

- Command format in the Command.txt file for this operation:

S[space] sumDistinctElements(for stack)

Q[space] sumDistinctElements(for queue)

8. Reverse Even Numbers

In this operation, according to given numbers, you are expected to adding these numbers to stack or queue, and updating stack or queue by reversing the order of the even numbers in stack or queue, and leaves the odd numbers in place. The number to be added can be any number between 1 and 100.

- Command format in the Command.txt file for this operation:

  S[space] evenReverse s (for stack)

  Q[space] evenReverse q (for queue)

- Format of the s and q is like that:

  number[space]number[space]number[space]....number

2 Input Files

Below, you can find an example of input and output file format.

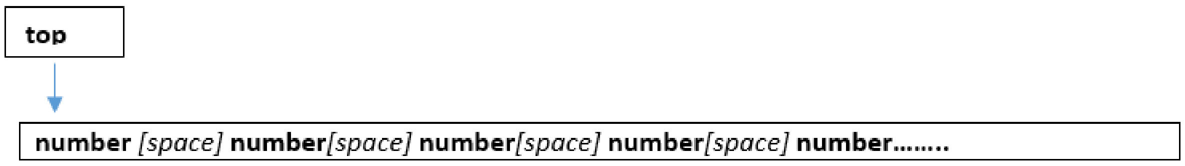

top
number [space] number[space] number[space] number[space] number........

Figure 1: format of stack.txt

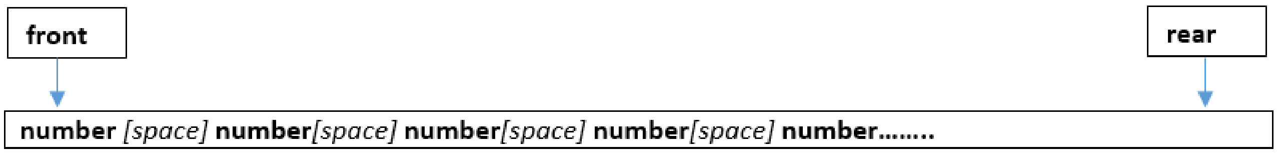
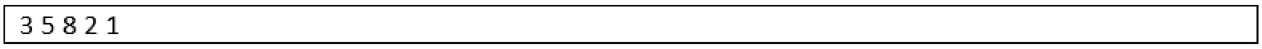
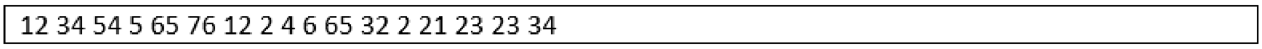

front
rear
number [space] number[space] number[space] number[space] number........

Figure 2: format of queue.txt



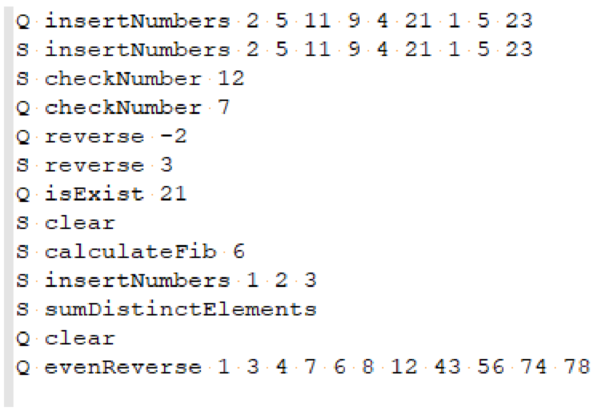
```
3 5 8 2 1
```

Figure 3: example of stack.txt



```
12 34 54 5 65 76 12 2 4 6 65 32 2 21 23 23 34
```

Figure 4: example of queue.txt

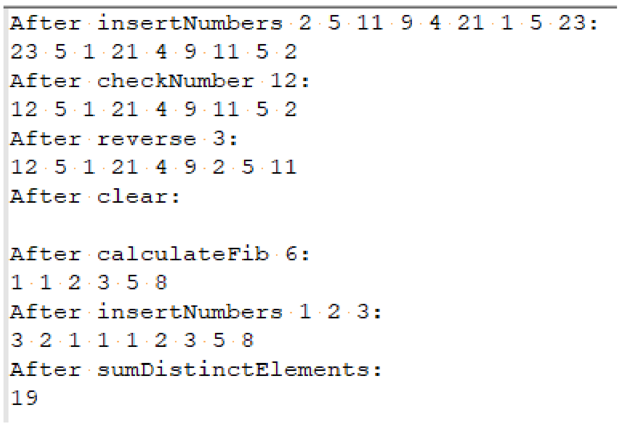


```
Q insertNumbers 2 5 11 9 4 21 1 5 23
S insertNumbers 2 5 11 9 4 21 1 5 23
S checkNumber 12
Q checkNumber 7
Q reverse -2
S reverse 3
Q isExist 21
S clear
S calculateFib 6
S insertNumbers 1 2 3
S sumDistinctElements
Q clear
Q evenReverse 1 3 4 7 6 8 12 43 56 74 78
```

Figure 5: example of command.txt

3 Output Files

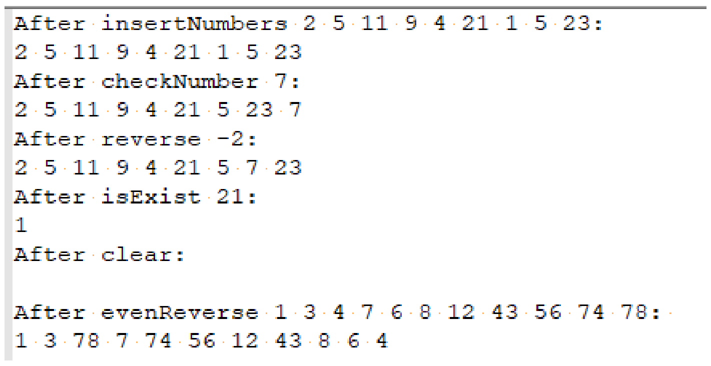
Output files will be produced separately for the stack and queue as stackOut.txt and queueOut.txt.



```
After insertNumbers 2 5 11 9 4 21 1 5 23:
23 5 1 21 4 9 11 5 2
After checkNumber 12:
12 5 1 21 4 9 11 5 2
After reverse 3:
12 5 1 21 4 9 2 5 11
After clear:

After calculateFib 6:
1 1 2 3 5 8
After insertNumbers 1 2 3:
3 2 1 1 1 2 3 5 8
After sumDistinctElements:
19
```

Figure 6: example of stackOut.txt



```
After insertNumbers 2 5 11 9 4 21 1 5 23:
2 5 11 9 4 21 1 5 23
After checkNumber 7:
2 5 11 9 4 21 5 23 7
After reverse -2:
2 5 11 9 4 21 5 7 23
After isExist 21:
1
After clear:

After evenReverse 1 3 4 7 6 8 12 43 56 74 78:
1 3 78 7 74 56 12 43 8 6 4
```

Figure 7: example of queueOut.txt

4 Execution and Test

- Upload your java files to your server account (dev.cs.hacettepe.edu.tr)
- Compile your code (javac *.java)
- Run your program (java Main command.txt)
- Control your output (stackOut.txt, queueOut.txt).

5 Submit Format

- File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

- ⟨*studentid*⟩*.zip*
– *src*(*Main.java*, ∗*.java*)

6 Grading Policy

Task	**Point**
Submit	1
Compiled	10
Coding standard (clean code, comment line, name convention)	15
Output	74
Total	100

7 Notes

- **You MUST implement your own queue and stack classes, don't use other data structures such as array, string, arraylist, linked list etc in your implementation. If you use other data structures, you will not get any point (your grade will be 0 point).**

- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.
- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.
- You can ask your questions through course's piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports .
- With this assignment which covers the subjects of data structures like stacks, queue, etc; you are expected to gain practice on the basics of these structure. Therefore, you will not be graded if any paradigm other than these is developed!
- Don't forget to write comments of your codes when necessary.
- The names of classes', attributes' and methods' should obey to Java naming convention.
- Save all work until the assignment is graded.
- Do not miss the deadline. Submission will be end at 02/09/2021 23:59, the system will be open 23:59:59. The problem about submission after 23:59 will not be considered.
- There will be again 3 days extensions (each day degraded by 10 points) in this project.