Queue To Stack Adapter - Push Efficient

Try First, Check Solution later

1. You should first read the question and watch the question video.  
2. Think of a solution approach, then try and submit the question on editor tab.  
3. We strongly advise you to watch the solution video for prescribed approach.

Question

1. You are required to complete the code of our QueueToStackAdapter class.   
2. As data members you've two queues available - mainQ and helperQ. mainQ is to contain data and helperQ is to assist in operations. (This is cryptic - take hint from video)  
2. Here is the list of functions that you are supposed to complete  
2.1. push -> Should accept new data in LIFO manner  
2.2. pop -> Should remove and return data in LIFO manner. If not available, print "Stack underflow" and return -1.  
2.3. top -> Should return data in LIFO manner. If not available, print "Stack underflow" and return -1.  
2.4. size -> Should return the number of elements available in the stack  
3. Input and Output is managed for you.  
  
Note -> push and size should work in constant time. pop and top should work in linear time.

Input Format

Input is managed for you

Output Format

Output is managed for you

Constraints

Note -> push and size should work in constant time. pop and top should work in linear time.

Sample Input

push 10  
push 20  
push 5  
push 8  
push 2  
push 4  
push 11  
top  
size  
pop  
top  
size  
pop  
top  
size  
pop  
top  
size  
pop  
top  
size  
pop  
top  
size  
pop  
top  
size  
pop  
quit

Sample Output

11  
7  
11  
4  
6  
4  
2  
5  
2  
8  
4  
8  
5  
3  
5  
20  
2  
20  
10  
1  
10

import java.io.\*;

import java.util.\*;

public class Main {

public static class QueueToStackAdapter {

Queue<Integer> mainQ;

Queue<Integer> helperQ;

public QueueToStackAdapter() {

mainQ = new ArrayDeque<>();

helperQ = new ArrayDeque<>();

}

int size() {

// write your code here

return mainQ.size();

}

void push(int val) {

// write your code here

mainQ.add(val);

}

int pop() {

if(mainQ.size()==0){

System.out.println("Stack underflow");

return -1;

}

while(mainQ.size() > 1){

helperQ.add(mainQ.remove());

}

int ans=mainQ.remove();

Queue <Integer> temp=mainQ;

mainQ=helperQ;

helperQ=temp;

return ans;

}

int top() {

if(mainQ.size()==0){

System.out.println("Stack underflow");

return -1;

}

while(mainQ.size() > 1){

helperQ.add(mainQ.remove());

}

int ans=mainQ.peek();

helperQ.add(mainQ.remove());

Queue <Integer> temp=mainQ;

mainQ=helperQ;

helperQ=temp;

return ans;

}

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

QueueToStackAdapter st = new QueueToStackAdapter();

String str = br.readLine();

while (str.equals("quit") == false) {

if (str.startsWith("push")) {

int val = Integer.parseInt(str.split(" ")[1]);

st.push(val);

} else if (str.startsWith("pop")) {

int val = st.pop();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("top")) {

int val = st.top();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("size")) {

System.out.println(st.size());

}

str = br.readLine();

}

}

}