**Task 3: Queue Sorting with Limited Space**

**You have a queue of integers that you need to sort. You can only use additional space equivalent to one stack. Describe the steps you would take to sort the elements in the queue.**

**Sol:**

**Steps to Sort the Queue**

**Initialize the Stack**:

* Create an empty stack to be used for sorting.

**Sorting Process**:

* Find the minimum element in the queue and push it onto the stack.
* Remove the found minimum element from the queue.
* Repeat the above steps until the queue is empty.

**Restoring Sorted Elements to the Queue**:

* Pop elements from the stack back into the queue. Since the stack has elements in sorted order (smallest at the top), the queue will now be sorted.

**Program:**

import java.util.LinkedList;

import java.util.Queue;

import java.util.Stack;

public class SortQueueUsingStack {

public static void sortQueue(Queue<Integer> queue) {

Stack<Integer> stack = new Stack<>();

int n = queue.size();

for (int i = 0; i < n; i++) {

// Find the minimum element in the queue and move elements to the stack

int min = Integer.MAX\_VALUE;

int countMin = 0;

int size = queue.size();

while (size-- > 0) {

int curr = queue.poll();

if (curr < min) {

min = curr;

countMin = 1;

} else if (curr == min) {

countMin++;

}

stack.push(curr);

}

// Move elements back to the queue, skipping the min elements

while (!stack.isEmpty()) {

int curr = stack.pop();

if (curr != min) {

queue.offer(curr);

}

}

// Reinsert the min elements at the back of the queue

while (countMin-- > 0) {

queue.offer(min);

}

}

}

public static void main(String[] args) {

Queue<Integer> queue = new LinkedList<>();

queue.offer(34);

queue.offer(3);

queue.offer(31);

queue.offer(98);

queue.offer(92);

queue.offer(23);

System.out.println("Original Queue: " + queue);

sortQueue(queue);

System.out.println("Sorted Queue: " + queue);

}

}