**Write a java or python implementation of linear quee ,circular quee and Linear quee using stack**

public class LinearQueue {

private int maxSize; // Maximum size of the queue

private int[] queueArray; // Array to store the elements

private int front; // Front of the queue

private int rear; // Rear of the queue

private int currentSize; // Current number of elements in the queue

// Constructor to initialize the queue

public LinearQueue(int size) {

maxSize = size;

queueArray = new int[maxSize];

front = 0;

rear = -1;

currentSize = 0;

}

// Method to check if the queue is empty

public boolean isEmpty() {

return (currentSize == 0);

}

// Method to check if the queue is full

public boolean isFull() {

return (currentSize == maxSize);

}

// Method to insert an element at the rear of the queue

public void enqueue(int item) {

if (isFull()) {

System.out.println("Queue is full. Cannot enqueue element.");

return;

}

rear = (rear + 1) % maxSize;

queueArray[rear] = item;

currentSize++;

System.out.println("Enqueued element: " + item);

}

// Method to remove an element from the front of the queue

public int dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty. Cannot dequeue element.");

return -1;

}

int removedItem = queueArray[front];

front = (front + 1) % maxSize;

currentSize--;

System.out.println("Dequeued element: " + removedItem);

return removedItem;

}

// Method to get the front element of the queue

public int getFront() {

if (isEmpty()) {

System.out.println("Queue is empty. No front element.");

return -1;

}

return queueArray[front];

}

// Method to get the rear element of the queue

public int getRear() {

if (isEmpty()) {

System.out.println("Queue is empty. No rear element.");

return -1;

}

return queueArray[rear];

}

// Method to get the current size of the queue

public int getCurrentSize() {

return currentSize;

}

// Main method

public static void main(String[] args) {

// Create a LinearQueue object

LinearQueue queue = new LinearQueue(5);

// Enqueue elements

queue.enqueue(10);

queue.enqueue(20);

queue.enqueue(30);

// Dequeue elements

queue.dequeue();

queue.dequeue();

// Get front and rear elements

int front = queue.getFront();

int rear = queue.getRear();

// Get the current size of the queue

int size = queue.getCurrentSize();

// Print the results

System.out.println("Front element: " + front);

System.out.println("Rear element: " + rear);

System.out.println("Current size: " + size);

}

}

**Question no 2**

public class CircularQueue {

private int maxSize; // Maximum size of the queue

private int[] queueArray; // Array to store the elements

private int front; // Front of the queue

private int rear; // Rear of the queue

private int currentSize; // Current number of elements in the queue

// Constructor to initialize the queue

public CircularQueue(int size) {

maxSize = size;

queueArray = new int[maxSize];

front = 0;

rear = -1;

currentSize = 0;

}

// Method to check if the queue is empty

public boolean isEmpty() {

return (currentSize == 0);

}

// Method to check if the queue is full

public boolean isFull() {

return (currentSize == maxSize);

}

// Method to insert an element at the rear of the queue

public void enqueue(int item) {

if (isFull()) {

System.out.println("Queue is full. Cannot enqueue element.");

return;

}

rear = (rear + 1) % maxSize;

queueArray[rear] = item;

currentSize++;

System.out.println("Enqueued element: " + item);

}

// Method to remove an element from the front of the queue

public int dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty. Cannot dequeue element.");

return -1;

}

int removedItem = queueArray[front];

front = (front + 1) % maxSize;

currentSize--;

System.out.println("Dequeued element: " + removedItem);

return removedItem;

}

// Method to get the front element of the queue

public int getFront() {

if (isEmpty()) {

System.out.println("Queue is empty. No front element.");

return -1;

}

return queueArray[front];

}

// Method to get the rear element of the queue

public int getRear() {

if (isEmpty()) {

System.out.println("Queue is empty. No rear element.");

return -1;

}

return queueArray[rear];

}

// Method to get the current size of the queue

public int getCurrentSize() {

return currentSize;

}

// Main method for testing

public static void main(String[] args) {

// Create a CircularQueue object

CircularQueue queue = new CircularQueue(5);

// Enqueue elements

queue.enqueue(10);

queue.enqueue(20);

queue.enqueue(30);

queue.enqueue(40);

queue.enqueue(50);

// Dequeue elements

queue.dequeue();

queue.dequeue();

// Enqueue more elements

queue.enqueue(60);

queue.enqueue(70);

// Get front and rear elements

int front = queue.getFront();

int rear = queue.getRear();

// Get the current size of the queue

int size = queue.getCurrentSize();

// Print the results

System.out.println("Front element: " + front);

System.out.println("Rear element: " + rear);

System.out.println("Current size: " + size);

}

}

Question no 3

import java.util.Stack;

public class LinearQueueUsingStack {

private Stack<Integer> enqueueStack;

private Stack<Integer> dequeueStack;

// Constructor to initialize the queues

public LinearQueueUsingStack() {

enqueueStack = new Stack<>();

dequeueStack = new Stack<>();

}

// Method to check if the queue is empty

public boolean isEmpty() {

return enqueueStack.isEmpty() && dequeueStack.isEmpty();

}

// Method to insert an element at the rear of the queue

public void enqueue(int item) {

while (!dequeueStack.isEmpty()) {

enqueueStack.push(dequeueStack.pop());

}

enqueueStack.push(item);

System.out.println("Enqueued element: " + item);

}

// Method to remove an element from the front of the queue

public int dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty. Cannot dequeue element.");

return -1;

}

while (!enqueueStack.isEmpty()) {

dequeueStack.push(enqueueStack.pop());

}

int removedItem = dequeueStack.pop();

System.out.println("Dequeued element: " + removedItem);

return removedItem;

}

// Method to get the front element of the queue

public int getFront() {

if (isEmpty()) {

System.out.println("Queue is empty. No front element.");

return -1;

}

while (!enqueueStack.isEmpty()) {

dequeueStack.push(enqueueStack.pop());

}

return dequeueStack.peek();

}

// Method to get the current size of the queue

public int getCurrentSize() {

return enqueueStack.size() + dequeueStack.size();

}

// Main method for testing

public static void main(String[] args) {

// Create a LinearQueueUsingStack object

LinearQueueUsingStack queue = new LinearQueueUsingStack();

// Enqueue elements

queue.enqueue(10);

queue.enqueue(20);

queue.enqueue(30);

// Dequeue elements

queue.dequeue();

queue.dequeue();

// Enqueue more elements

queue.enqueue(40);

queue.enqueue(50);

// Get front element

int front = queue.getFront();

// Get the current size of the queue

int size = queue.getCurrentSize();

// Print the results

System.out.println("Front element: " + front);

System.out.println("Current size: " + size);

}

}