|  |  |
| --- | --- |
| Student Name | Uzair Hussain |
| Roll Number | 21SW085 |
| Section # | 3rd or III |
| Lab # | 11th - Queue |

**Task#01**

**Code: (Using Array)**

public class Task1\_ArrayQueue {

    int[] queue;

    int front; // index of the front element

    int rear;  // index of the rear element

    int size;  // current size of the queue

    public boolean isEmpty() {

        return size == 0;

    }

    public boolean isFull() {

        return size == queue.length;

    }

    public int size() {

        return size;

    }

    public Task1\_ArrayQueue(int capacity) {

        queue = new int[capacity];

        front = 0;      // starting element index

        rear = -1;      // last element index

        size = 0;

    }

    public void enqueue(int value) {             /// add method

        if (isFull()) {

            throw new RuntimeException("Queue is full");

        }

        rear = (rear + 1) % queue.length;

        queue[rear] = value;

        size++;

    }

    public int dequeue() {              ///remove method removes from front

        if (isEmpty()) {

            throw new RuntimeException("Queue is empty");

        }

        int item = queue[front];

        front = (front + 1) % queue.length;

        size--;

        return item;

    }

    public static void printQueue(Task1\_ArrayQueue queue) {

        System.out.print("Queue: ");

        for (int i = 0; i < queue.size(); i++) {

            System.out.print(queue.queue[(queue.front + i) % queue.queue.length] + " ");

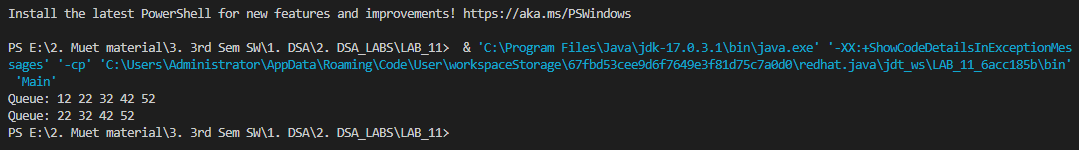
        }

        System.out.println();

    }

}

**Output 1:**

****

**Code: (Using Linked List)**

import java.util.concurrent.ThreadPoolExecutor.DiscardPolicy;

interface Queue{

    public void add(Object obj);

    public Object first();

    public Object remove();

    public int size();

}

public class Task1\_LinkedQueue implements Queue{

    private Node head=new Node(null);

    private int size;

    static class Node{

        Object object;

        Node prev=this;

        Node next=this;

        Node(Object obj){

            this.object=obj;

        }

        Node(Object obj, Node n,Node p){

            object=obj;

            next=n;

            prev=p;

        }

    }

    @Override

    public int size(){

        return size;

    }

    @Override

    public Object first(){

        if(size==0){

            throw new IllegalStateException();

        }

        return head.next.object;

    }

    @Override

    public void add(Object obj){

        head.prev.next=new Node(obj,head,head.prev);

        head.prev=head.prev.next;

        size++;

    }

    @Override

    public Object remove(){

        if(size==0){

            throw new IllegalStateException();

        }

        Object temp=head.next.object;

        head.next=head.next.next;

        head.next.prev=head;

        size--;

        return temp;

    }

    public boolean isEmpty(){

        return (head==null);

    }

    public void displayTasks() {

        Node current = head.next;

        while (current != head) {

            System.out.println(current.object + " ");

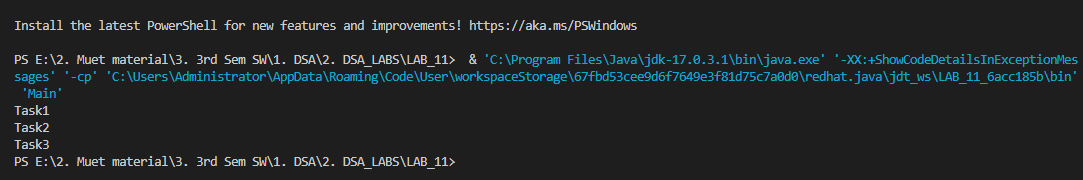
            current = current.next;

        }

    }

}

**Output:**

****

**Main Class:**

public class Main {

    public static void main(String[] args) {

        /// Array Queue

        // Task1\_ArrayQueue aq= new Task1\_ArrayQueue(5);

        // aq.enqueue(12);

        // aq.enqueue(22);

        // aq.enqueue(32);

        // aq.enqueue(42);

        // aq.enqueue(52);

        // aq.printQueue(aq);

        // aq.dequeue();

        // aq.printQueue(aq);

        /// Linked Queue

        Task1\_LinkedQueue lq = new Task1\_LinkedQueue();

        lq.add("Task1");

        lq.add("Task2");

        lq.add("Task3");

        lq.displayTasks();

    }

}

**GitHub Repository for all Lab Tasks: (from lab 1 to continue)**

[**https://github.com/UzairHussain193/DSA\_LABS\_21SW**](https://github.com/UzairHussain193/DSA_LABS_21SW)

**The End!**