**JAVA – LAB**

**ETCS-357**



**NAME:** SHIVAM SINGH

**ENROLLMENT NO:** 07313302720

**CLASS & SEC:** CSE-5A

**GROUP:** 2

**SUBMITTED TO: RASHMI TIWARI**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | | | | |
| **S.NO** | **AIM** | **DATE OF EXPERIMENT** | **DATE OF**  **SUBMISSION** | **REMARKS** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Experiment-1**

**Aim 🡺** Create a java program to implement stack and queue concept

**Code 🡺**

**//**Stack Implementation

public class stack {

    private int[] arr;

    private int capacity;

    private int top;

    stack(int size) {

         arr = new int[size];

         capacity = size;

         top = -1;

    }

    public void push(int x) {

        if(isFull()) {

            System.out.println("Overflow\nExiting Program\n");

            System.exit(-1);

        }

        System.out.println("Inserting element "+ x);

        arr[++top] = x;

    }

    public int pop() {

        if(isEmpty()) {

            System.out.println("Underflow\nExiting Program\n");

            System.exit(-1);

        }

        System.out.println("Removing the element " + arr[top]);

        return arr[top--];

    }

    public int peek() {

        if(!isEmpty()) {

            return arr[top];

        }else {

            System.out.println("Stack is Empty\n");

            System.exit(-1);

        }

        return -1;

    }

    public int size() {

        return top+1;

    }

    public boolean isEmpty() {

        return top == -1;

    }

    public boolean isFull() {

        return top == capacity - 1;

    }

}

//Queue Implementation

public class queue {

    private int[] qarr;

    private int front;

    private int rear;

    private int capacity;

    private int count;

    queue(int size) {

        qarr = new int[size];

        front = 0;

        rear = -1;

        capacity = size;

        count = 0;

    }

    public int size() {

        return count;

    }

    public boolean isEmpty() {

        return count == 0;

    }

    public boolean isFull() {

        return count == capacity;

    }

    public void enqueue(int x) {

        if(isFull()) {

            System.out.println("Overflow\nExiting Program\n");

            System.exit(-1);

        }

        System.out.println("Inserting " + x);

        rear = (rear+1)%capacity;

        qarr[rear] = x;

        count++;

    }

    public int dequeue() {

        if(isEmpty()) {

            System.out.println("Underflow\nExiting Program\n");

            System.exit(-1);

        }

        System.out.println("Removing Element " + qarr[front]);

        int x = qarr[front];

        front = (front+1)%capacity;

        count--;

        return x;

    }

    public int qfront() {

        if(!isEmpty()){

            return qarr[front];

        }

        return -1;

    }

    public int qback() {

        if(!isEmpty()){

            return qarr[rear];

        }

        return -1;

    }

}

//Main Code

public class ExpOne {

    public static void main(String[] args) {

        System.out.println("Shivam Singh\n07313302720\nCSE-5A\n");

        //-----------------------Stack Implementation------------------------------------//

        System.out.println("\nStack Implementation\n");

        stack s1 = new stack(3);

        s1.push(1);

        s1.push(2);

        s1.pop();

        s1.pop();

        s1.push(3);

        System.out.println("The top element is " + s1.peek());

        System.out.println("The stack size is " + s1.size());

        //-----------------------Stack Implementation------------------------------------//

        //-----------------------Queue Implementation------------------------------------//

        System.out.println("\nQueue Implementation\n");

        queue q = new queue(5);

        q.enqueue(1);

        q.enqueue(2);

        q.enqueue(3);

        System.out.println("The front element is " + q.qfront());

        q.dequeue();

        System.out.println("The rear element is " + q.qback());

        System.out.println("The queue size is " + q.size());

        q.dequeue();

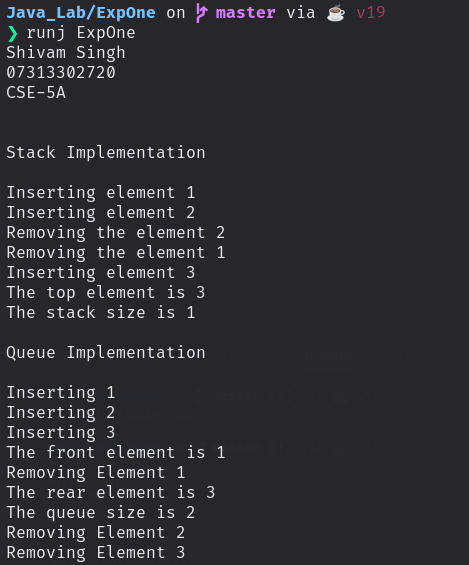
        q.dequeue();

        //-----------------------Queue Implementation------------------------------------//

    }

}

//Output



**Experiment-2**

**Aim 🡺 WAP to implement Method Overloading**

**Code 🡺**

class ExpTwo {

    public int sum(int a, int b) {

        System.out.println("Method with two arguments\n");

        return a+b;

    }

    public int sum(int a, int b, int c) {

        System.out.println("Method with three arguments\n");

        return a+b+c;

    }

    public float sum(float a, float b, float c, float d) {

        System.out.println("Method with four arguments\n");

        return a+b+c+d;

    }

    public static void main(String[] args) {

        System.out.println("Shivam Singh\n07313302720\nCSE-5A\n");

        ExpTwo obj1 = new ExpTwo();

        System.out.println(obj1.sum(5,2));

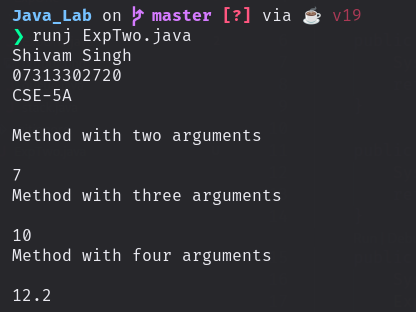
        System.out.println(obj1.sum(5,2, 3));

        System.out.println(obj1.sum(5.1f,2.3f, 3.3f, 1.5f));

    }

}

//**Output**

****