```
// class Queue {

//     private int front;
//     private int rear;
//     private int maxSize;
//     private int arr[];

//     Queue(int maxSize) {
//         this.front = 0;
//         this.rear = -1;
//         this.maxSize = maxSize;
//         this.arr = new int[this.maxSize];
//     }

//     public boolean isFull() {
//         if (rear == maxSize - 1) {
//             return true;
//         }
//         return false;
//     }

//     public boolean enqueue(int data) {
//         if (isFull()) {
//             return false;
//         } else {
//             arr[++rear] = data;
//             return true;
//         }
//     }

//     public void display() {
//         if(isEmpty())
//             System.out.println("Queue is empty!");
//         else {
//             for (int index = front; index <= rear; index++) {
//                 System.out.println(arr[index]);
//             }
//         }
//     }

//     public boolean isEmpty() {
//         if (front > rear)
//             return true;
//         return false;
//     }

//     public int dequeue() {
//         if (isEmpty()) {
//             return Integer.MIN_VALUE;
//         } else {
//             int data = arr[this.front];
//             arr[front++] = Integer.MIN_VALUE;
//             return data;
//         }
//     }

//     public int getMaxSize() {
//         return maxSize;
//     }

//     public static void  Leftqueue(){

//     }

//     public static void Rightqueue(){

//     }
```

```
// }


// class Tester {

//     public static void main(String[] args) {

//         Queue queue = new Queue(7);
//         queue.enqueue(2);
//         queue.enqueue(7);
//         queue.enqueue(9);
//         queue.enqueue(4);
//         queue.enqueue(6);
//         queue.enqueue(5);
//         queue.enqueue(10);

//         Queue[] queueArray = splitQueue(queue);

//         System.out.println("Elements in the queue of odd numbers");
//         queueArray[0].display();

//         System.out.println("\nElements in the queue of even numbers");
//         queueArray[1].display();

//     }

//     public static Queue[] splitQueue(Queue queue) {
//         // Implement your code here and change the return value accordingly







//         return null;
//     }
// }


class Stack {

    private int top;
    private int maxSize;
    private int[] arr;
    private int i = 0;

    Stack(int maxSize) {
        this.top = -1;
        this.maxSize = maxSize;
        arr = new int[maxSize];
    }

    public boolean isFull() {
        if (top >= (maxSize - 1)) {
            return true;
        }
        return false;
    }

    public boolean push(int data) {
        if (isFull()) {
            return false;
        }
        else {
            arr[++top] = data;
            return true;
        }
```

```
    }

    public int peek() {
        if (isEmpty())
            return Integer.MIN_VALUE;
        else
            return arr[top];
    }

    public void sum() {
        int temp = 0;
        if (isEmpty())
            System.out.println("null");

        else {
            for (int index = top; index >= 0; index--) {
                    temp = temp + arr[index];

                    // accessing element at position index
                }

            }
            System.out.println(temp);



    }

    public void display() {
        if (isEmpty())
            System.out.println("Stack is empty!");
        else {
            System.out.println("Displaying stack elements");
            for (int index = top; index >= 0; index--) {
                System.out.println(arr[index]); // accessing element at position index
            }
        }
    }

    public boolean isEmpty() {
        if (top < 0) {
            return true;
        }
        return false;
    }

    public int pop() {

        if (isEmpty())
            return Integer.MIN_VALUE;
        else

            return arr[top--];
    }




}


class Tester {

    public static void main(String args[]) {

        Stack stack = new Stack(10);
        stack.push(15);
```

```
        stack.push(20);
        stack.push(30);
        stack.push(40);



        System.out.println("Updated stack");
        stack.display();
        calculateSum(stack);
        // stack.sum();
    }
    public static void calculateSum(Stack stack) {

             stack.sum();



        }
}
```

```
      // int sum = 0;
        // Stack stack2 = new Stack(10);

        // for(int i = 0; i < 10 ; i++ ){
        //    stack2.sum();
        // }
```

```
        //Implement your code here
```