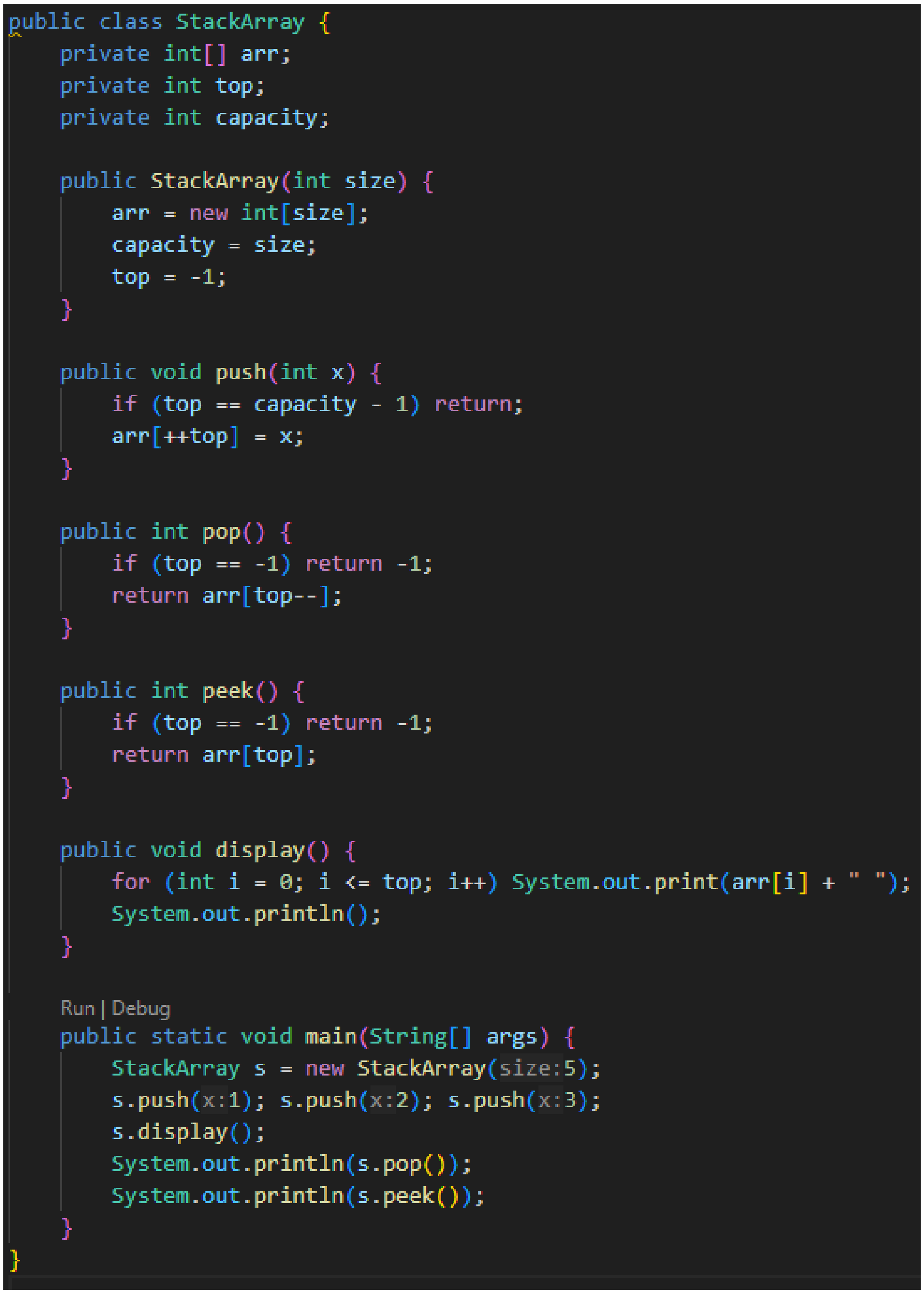
Assignment 4

Submission Date: 28/09/2025

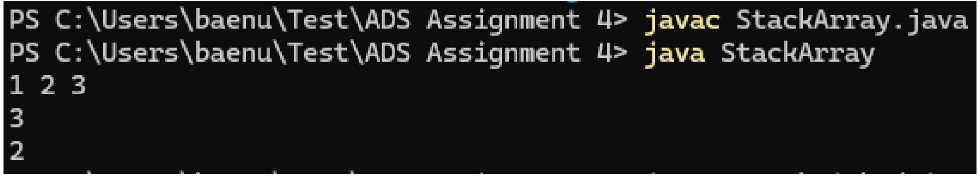
ANSWERS

Stack Coding Questions

1. Implement a Stack using arrays.

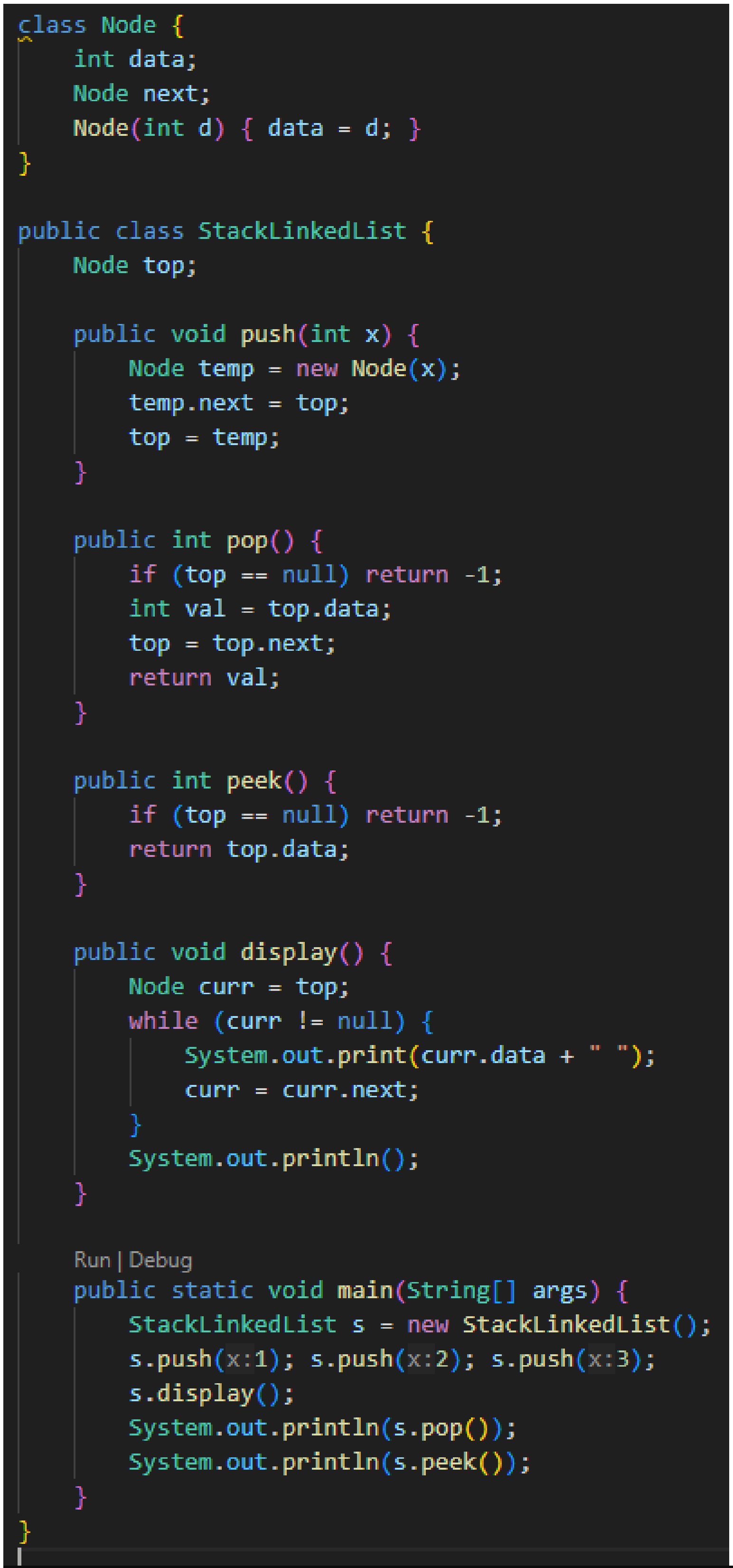


```
public class StackArray {
    private int[] arr;
    private int top;
    private int capacity;

    public StackArray(int size) {
        arr = new int[size];
        capacity = size;
        top = -1;
    }

    public void push(int x) {
        if (top == capacity - 1) return;
        arr[++top] = x;
    }

    public int pop() {
        if (top == -1) return -1;
        return arr[top--];
    }

    public int peek() {
        if (top == -1) return -1;
        return arr[top];
    }

    public void display() {
        for (int i = 0; i <= top; i++) System.out.print(arr[i] + " ");
        System.out.println();
    }

    public static void main(String[] args) {
        StackArray s = new StackArray(size:5);
        s.push(x:1); s.push(x:2); s.push(x:3);
        s.display();
        System.out.println(s.pop());
        System.out.println(s.peek());
    }
}
```

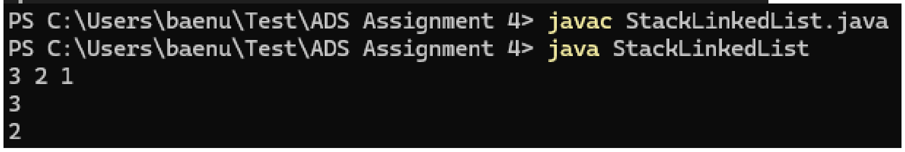


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac StackArray.java
PS C:\Users\baenu\Test\ADS Assignment 4> java StackArray
1 2 3
3
2
```

2. Implement a Stack using linked list.

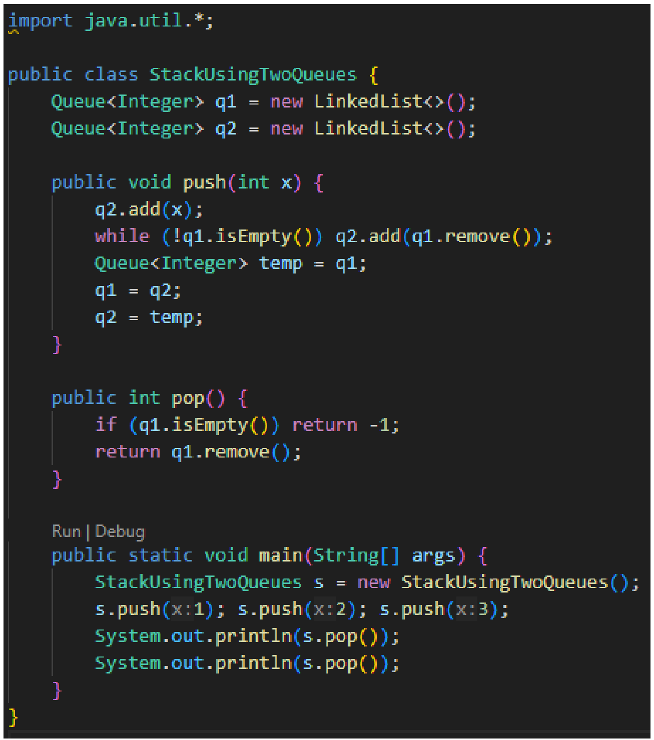


```java
class Node {
    int data;
    Node next;
    Node(int d) { data = d; }
}

public class StackLinkedList {
    Node top;

    public void push(int x) {
        Node temp = new Node(x);
        temp.next = top;
        top = temp;
    }

    public int pop() {
        if (top == null) return -1;
        int val = top.data;
        top = top.next;
        return val;
    }

    public int peek() {
        if (top == null) return -1;
        return top.data;
    }

    public void display() {
        Node curr = top;
        while (curr != null) {
            System.out.print(curr.data + " ");
            curr = curr.next;
        }
        System.out.println();
    }

    Run | Debug
    public static void main(String[] args) {
        StackLinkedList s = new StackLinkedList();
        s.push(x:1); s.push(x:2); s.push(x:3);
        s.display();
        System.out.println(s.pop());
        System.out.println(s.peek());
    }
}
```

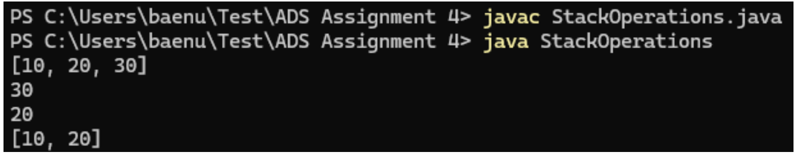


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac StackLinkedList.java
PS C:\Users\baenu\Test\ADS Assignment 4> java StackLinkedList
3 2 1
3
2
```

3. Write a program to push, pop, peek, and display elements of a stack.

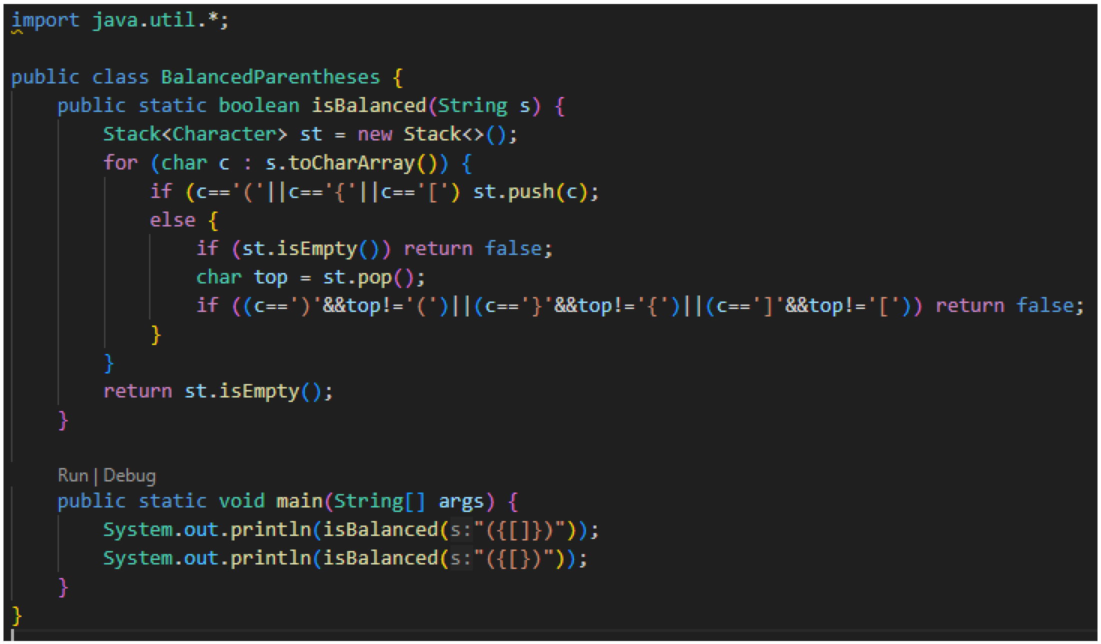


```java
import java.util.*;

public class StackUsingTwoQueues {
    Queue<Integer> q1 = new LinkedList<>();
    Queue<Integer> q2 = new LinkedList<>();

    public void push(int x) {
        q2.add(x);
        while (!q1.isEmpty()) q2.add(q1.remove());
        Queue<Integer> temp = q1;
        q1 = q2;
        q2 = temp;
    }

    public int pop() {
        if (q1.isEmpty()) return -1;
        return q1.remove();
    }

    Run | Debug
    public static void main(String[] args) {
        StackUsingTwoQueues s = new StackUsingTwoQueues();
        s.push(x:1); s.push(x:2); s.push(x:3);
        System.out.println(s.pop());
        System.out.println(s.pop());
    }
}
```

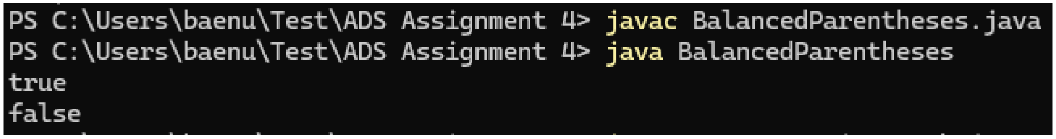


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac StackOperations.java
PS C:\Users\baenu\Test\ADS Assignment 4> java StackOperations
[10, 20, 30]
30
20
[10, 20]
```

4. Check if a string of parentheses is balanced using a stack.
* Example: "({[]})" → Balanced.

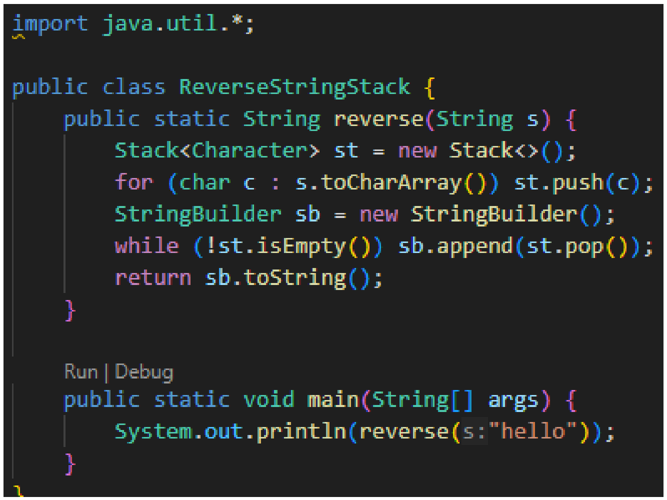


```java
import java.util.*;

public class BalancedParentheses {
    public static boolean isBalanced(String s) {
        Stack<Character> st = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c=='('||c=='{'||c=='[') st.push(c);
            else {
                if (st.isEmpty()) return false;
                char top = st.pop();
                if ((c==')'&&top!='(')||(c=='}'&&top!='{')||(c==']'&&top!='[')) return false;
            }
        }
        return st.isEmpty();
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(isBalanced(s:"({[]})"));
        System.out.println(isBalanced(s:"({[})"));
    }
}
```

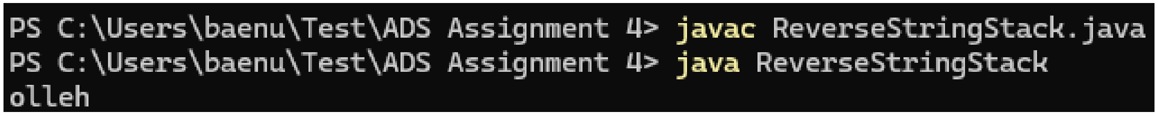


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac BalancedParentheses.java
PS C:\Users\baenu\Test\ADS Assignment 4> java BalancedParentheses
true
false
```

5. Reverse a string using stack.

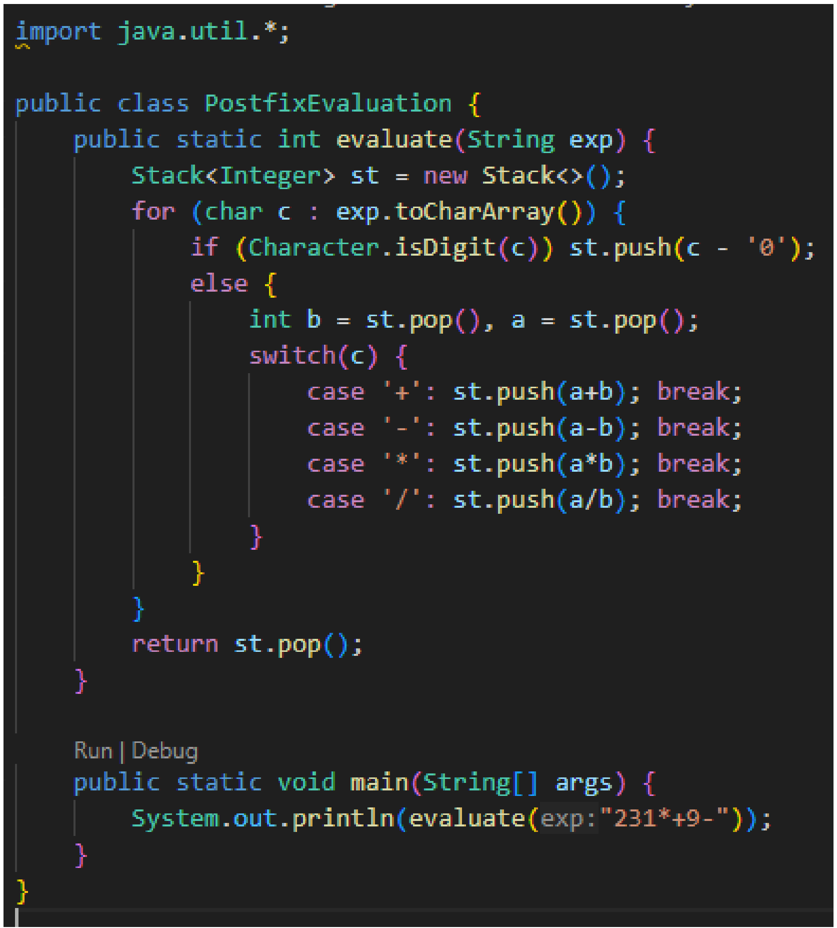


```java
import java.util.*;

public class ReverseStringStack {
    public static String reverse(String s) {
        Stack<Character> st = new Stack<>();
        for (char c : s.toCharArray()) st.push(c);
        StringBuilder sb = new StringBuilder();
        while (!st.isEmpty()) sb.append(st.pop());
        return sb.toString();
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(reverse(s:"hello"));
    }
}
```

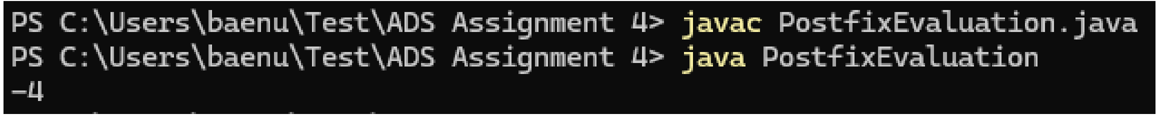


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac ReverseStringStack.java
PS C:\Users\baenu\Test\ADS Assignment 4> java ReverseStringStack
olleh
```

6. Evaluate a postfix expression using stack.
Example: 231*+9- → -4.

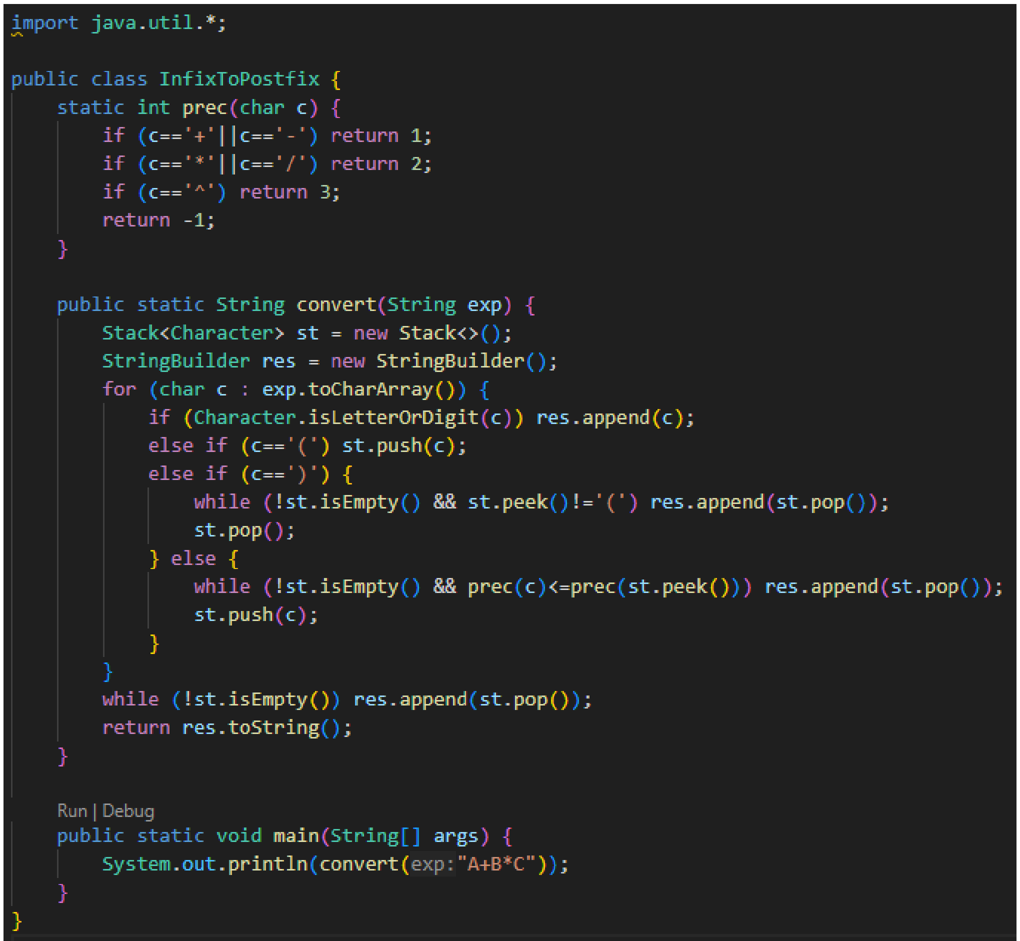


```java
import java.util.*;

public class PostfixEvaluation {
    public static int evaluate(String exp) {
        Stack<Integer> st = new Stack<>();
        for (char c : exp.toCharArray()) {
            if (Character.isDigit(c)) st.push(c - '0');
            else {
                int b = st.pop(), a = st.pop();
                switch(c) {
                    case '+': st.push(a+b); break;
                    case '-': st.push(a-b); break;
                    case '*': st.push(a*b); break;
                    case '/': st.push(a/b); break;
                }
            }
        }
        return st.pop();
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(evaluate(exp:"231*+9-"));
    }
}
```

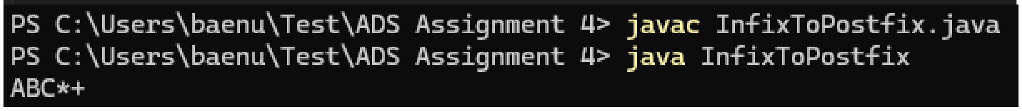


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac PostfixEvaluation.java
PS C:\Users\baenu\Test\ADS Assignment 4> java PostfixEvaluation
-4
```

7. Convert an infix expression to postfix using stack.
Example: A+B*C → ABC*+.

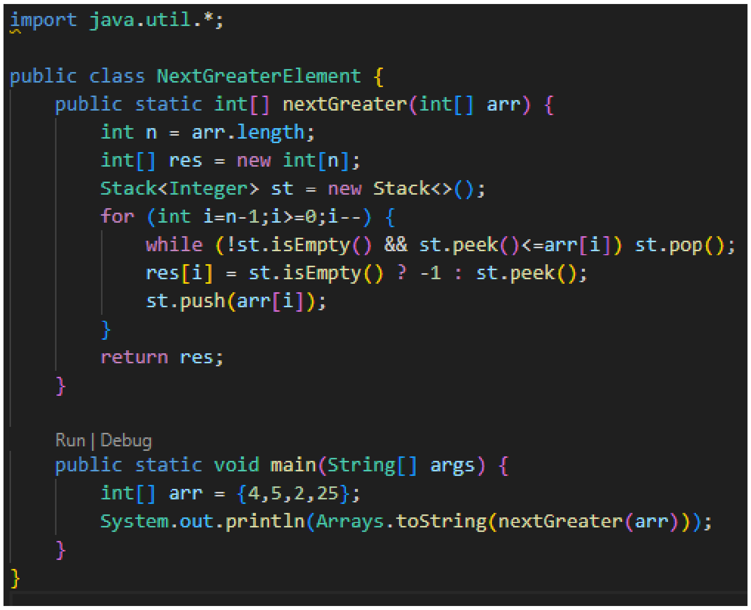


```
import java.util.*;

public class InfixToPostfix {
    static int prec(char c) {
        if (c=='+'||c=='-') return 1;
        if (c=='*'||c=='/') return 2;
        if (c=='^') return 3;
        return -1;
    }

    public static String convert(String exp) {
        Stack<Character> st = new Stack<>();
        StringBuilder res = new StringBuilder();
        for (char c : exp.toCharArray()) {
            if (Character.isLetterOrDigit(c)) res.append(c);
            else if (c=='(') st.push(c);
            else if (c==')') {
                while (!st.isEmpty() && st.peek()!='(') res.append(st.pop());
                st.pop();
            } else {
                while (!st.isEmpty() && prec(c)<=prec(st.peek())) res.append(st.pop());
                st.push(c);
            }
        }
        while (!st.isEmpty()) res.append(st.pop());
        return res.toString();
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(convert(exp:"A+B*C"));
    }
}
```

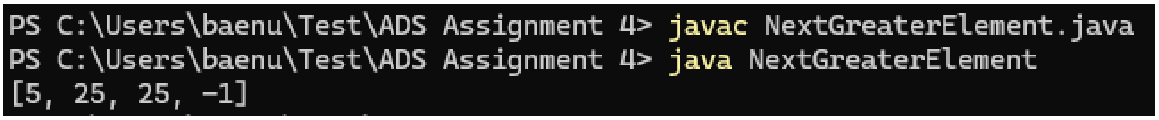


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac InfixToPostfix.java
PS C:\Users\baenu\Test\ADS Assignment 4> java InfixToPostfix
ABC*+
```

8. Find the next greater element for each element in an array using stack.
Example: [4, 5, 2, 25] → [5, 25, 25, -1].

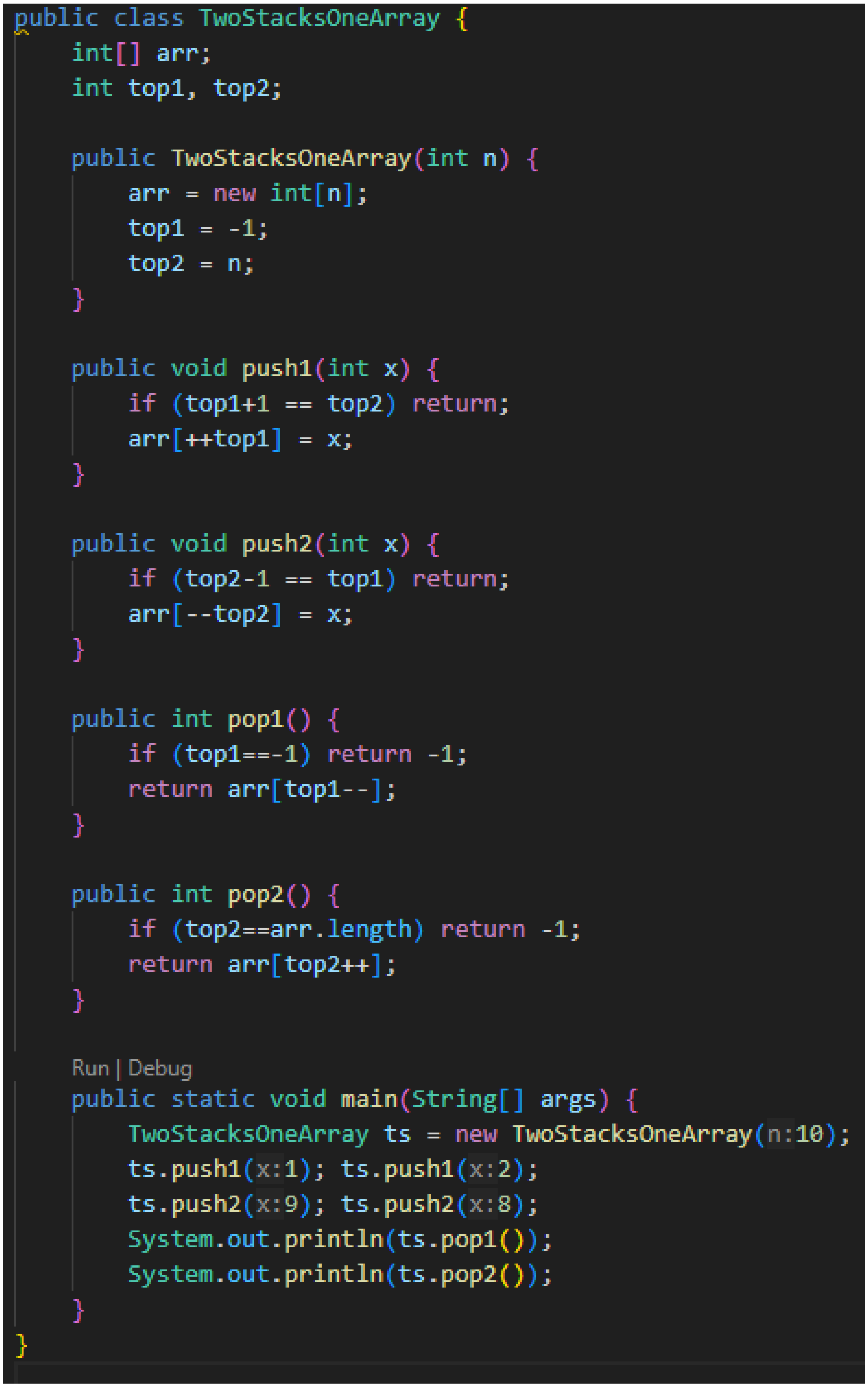


```
import java.util.*;

public class NextGreaterElement {
    public static int[] nextGreater(int[] arr) {
        int n = arr.length;
        int[] res = new int[n];
        Stack<Integer> st = new Stack<>();
        for (int i=n-1;i>=0;i--) {
            while (!st.isEmpty() && st.peek()<=arr[i]) st.pop();
            res[i] = st.isEmpty() ? -1 : st.peek();
            st.push(arr[i]);
        }
        return res;
    }

    Run | Debug
    public static void main(String[] args) {
        int[] arr = {4,5,2,25};
        System.out.println(Arrays.toString(nextGreater(arr)));
    }
}
```

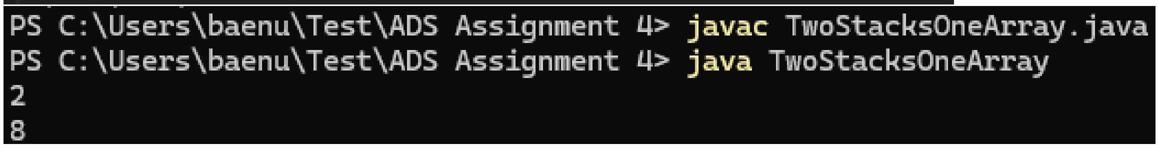


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac NextGreaterElement.java
PS C:\Users\baenu\Test\ADS Assignment 4> java NextGreaterElement
[5, 25, 25, -1]
```

9. Implement two stacks in a single array.

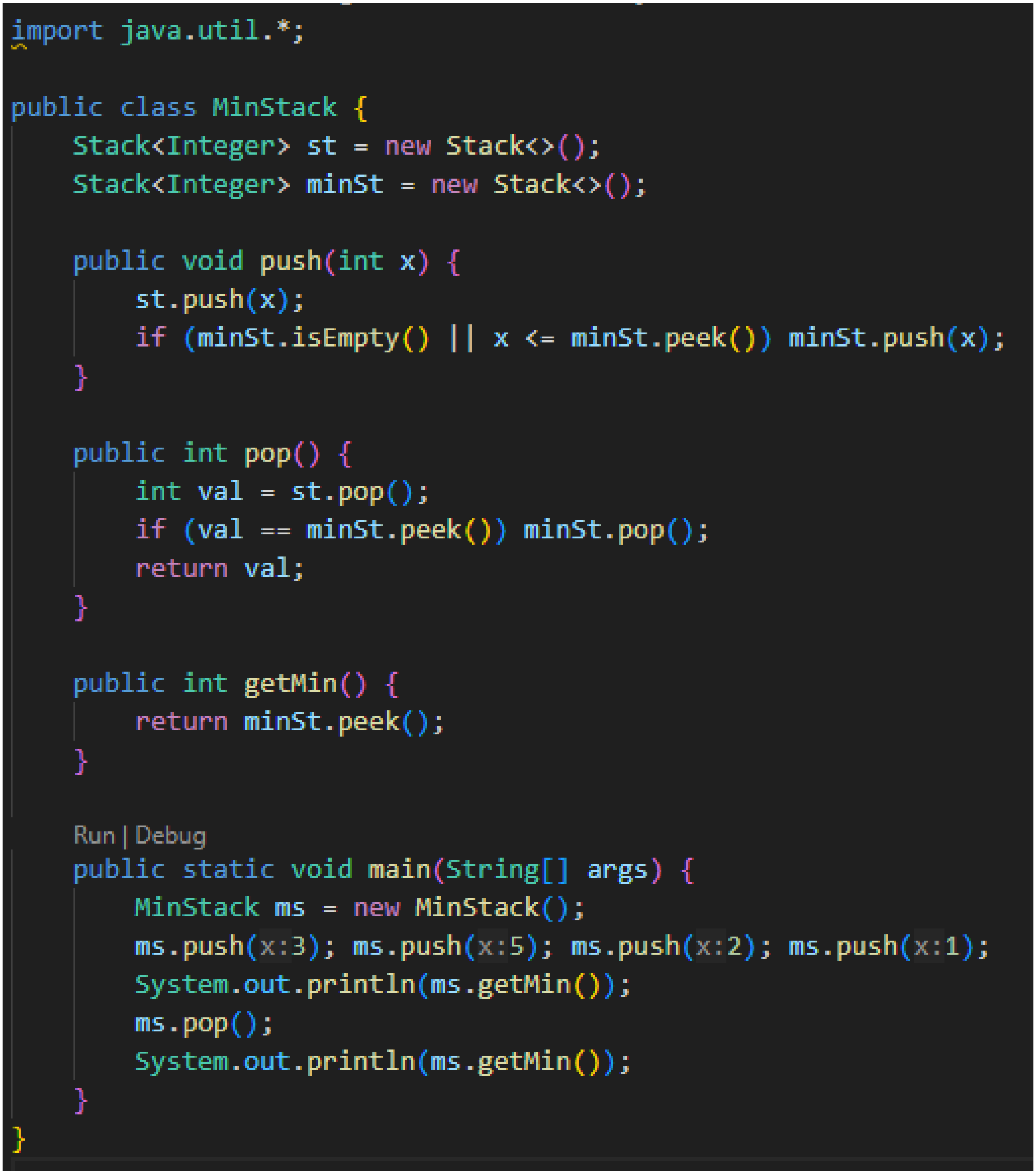


```java
public class TwoStacksOneArray {
    int[] arr;
    int top1, top2;

    public TwoStacksOneArray(int n) {
        arr = new int[n];
        top1 = -1;
        top2 = n;
    }

    public void push1(int x) {
        if (top1+1 == top2) return;
        arr[++top1] = x;
    }

    public void push2(int x) {
        if (top2-1 == top1) return;
        arr[--top2] = x;
    }

    public int pop1() {
        if (top1==-1) return -1;
        return arr[top1--];
    }

    public int pop2() {
        if (top2==arr.length) return -1;
        return arr[top2++];
    }

    Run | Debug
    public static void main(String[] args) {
        TwoStacksOneArray ts = new TwoStacksOneArray(n:10);
        ts.push1(x:1); ts.push1(x:2);
        ts.push2(x:9); ts.push2(x:8);
        System.out.println(ts.pop1());
        System.out.println(ts.pop2());
    }
}
```

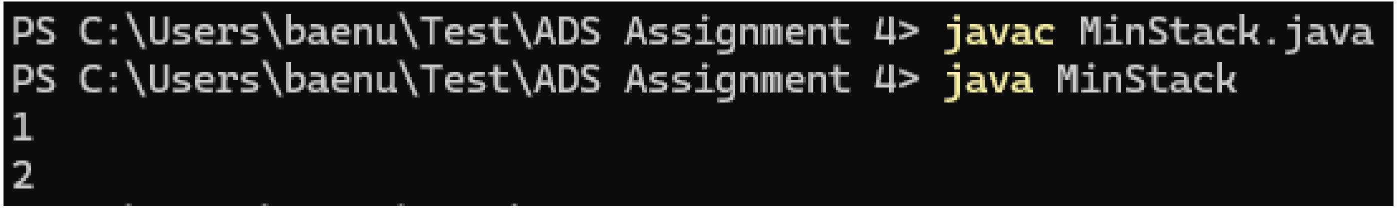


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac TwoStacksOneArray.java
PS C:\Users\baenu\Test\ADS Assignment 4> java TwoStacksOneArray
2
8
```

10. Design a stack that supports getMin() in O(1) time.



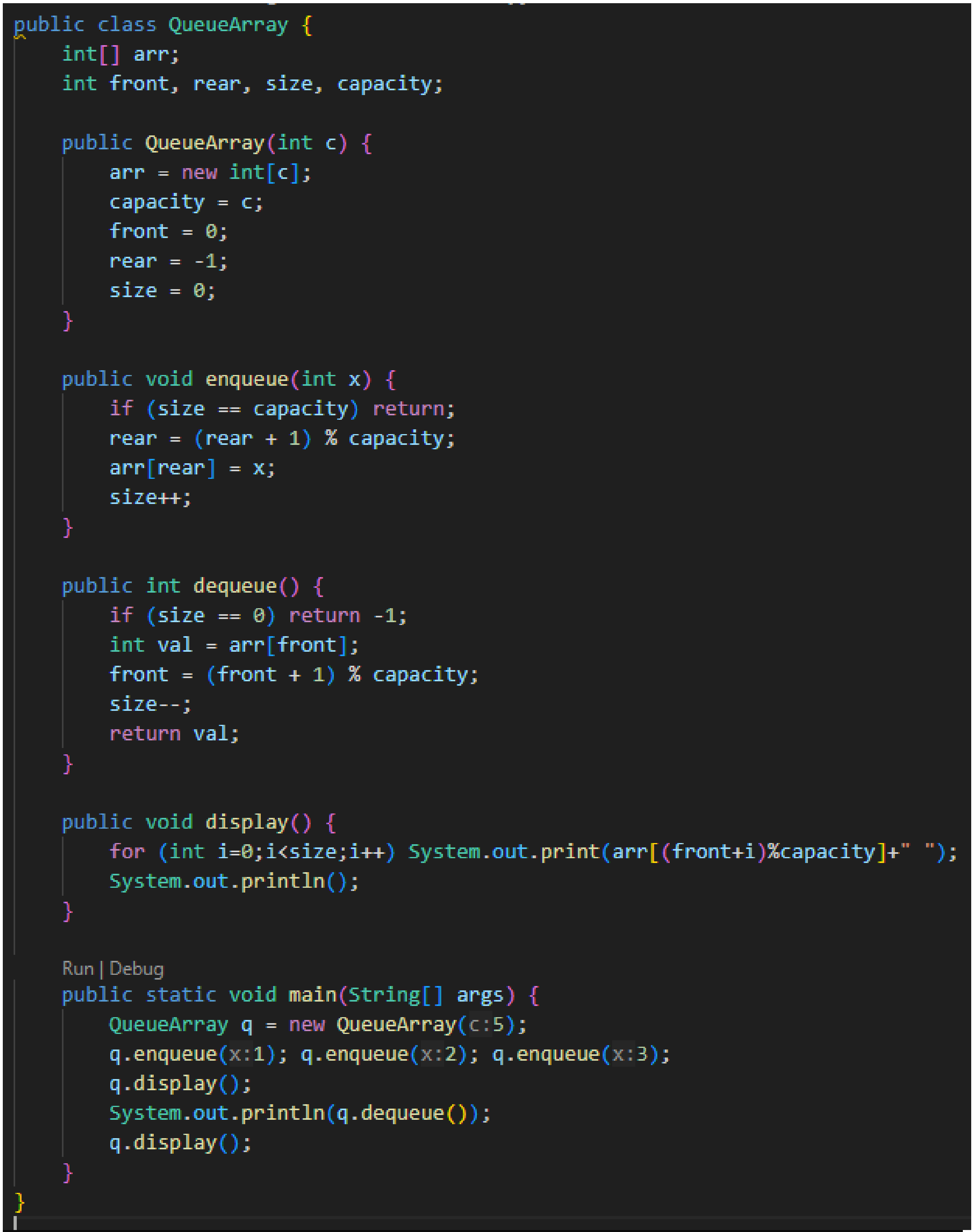
```java
import java.util.*;

public class MinStack {
    Stack<Integer> st = new Stack<>();
    Stack<Integer> minSt = new Stack<>();

    public void push(int x) {
        st.push(x);
        if (minSt.isEmpty() || x <= minSt.peek()) minSt.push(x);
    }

    public int pop() {
        int val = st.pop();
        if (val == minSt.peek()) minSt.pop();
        return val;
    }

    public int getMin() {
        return minSt.peek();
    }

    Run | Debug
    public static void main(String[] args) {
        MinStack ms = new MinStack();
        ms.push(x:3); ms.push(x:5); ms.push(x:2); ms.push(x:1);
        System.out.println(ms.getMin());
        ms.pop();
        System.out.println(ms.getMin());
    }
}
```



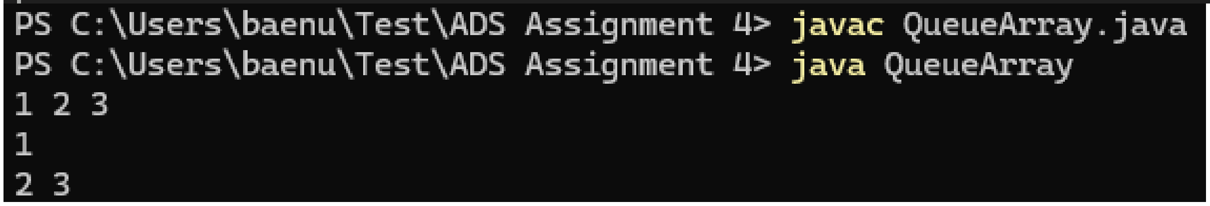
```
PS C:\Users\baenu\Test\ADS Assignment 4> javac MinStack.java
PS C:\Users\baenu\Test\ADS Assignment 4> java MinStack
1
2
```

Queue Coding Questions

1. Implement a Queue using arrays.

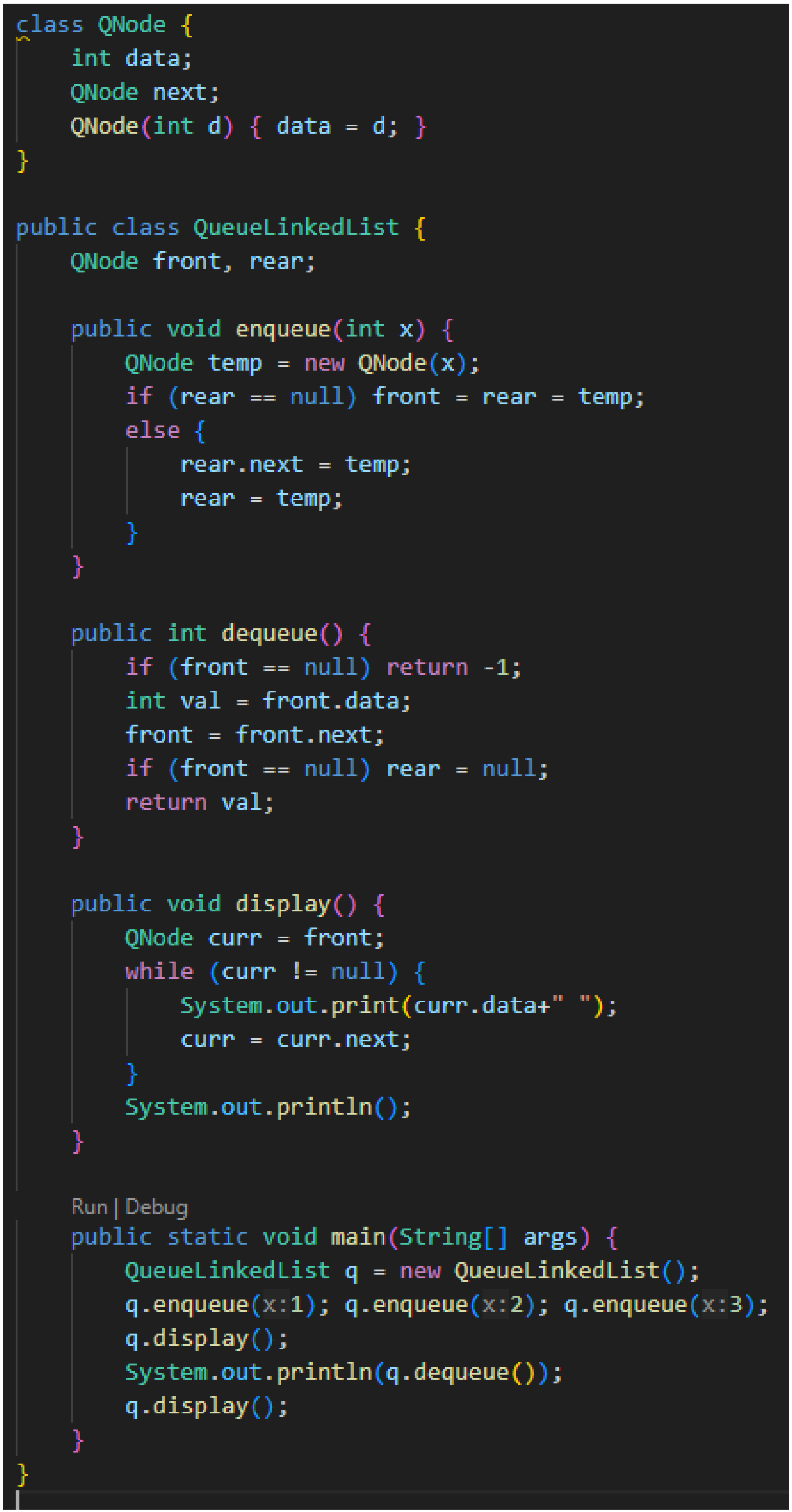


```
public class QueueArray {
    int[] arr;
    int front, rear, size, capacity;

    public QueueArray(int c) {
        arr = new int[c];
        capacity = c;
        front = 0;
        rear = -1;
        size = 0;
    }

    public void enqueue(int x) {
        if (size == capacity) return;
        rear = (rear + 1) % capacity;
        arr[rear] = x;
        size++;
    }

    public int dequeue() {
        if (size == 0) return -1;
        int val = arr[front];
        front = (front + 1) % capacity;
        size--;
        return val;
    }

    public void display() {
        for (int i=0;i<size;i++) System.out.print(arr[(front+i)%capacity]+" ");
        System.out.println();
    }

    Run | Debug
    public static void main(String[] args) {
        QueueArray q = new QueueArray(c:5);
        q.enqueue(x:1); q.enqueue(x:2); q.enqueue(x:3);
        q.display();
        System.out.println(q.dequeue());
        q.display();
    }
}
```

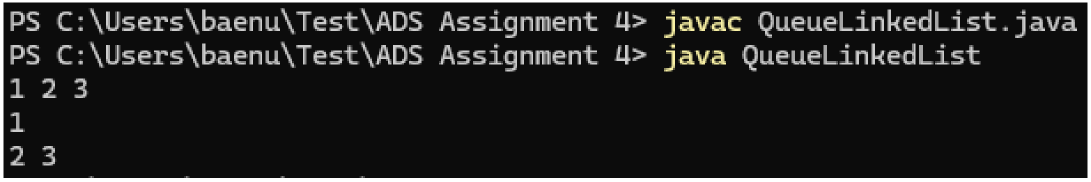


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac QueueArray.java
PS C:\Users\baenu\Test\ADS Assignment 4> java QueueArray
1 2 3
1
2 3
```

2. Implement a Queue using linked list.

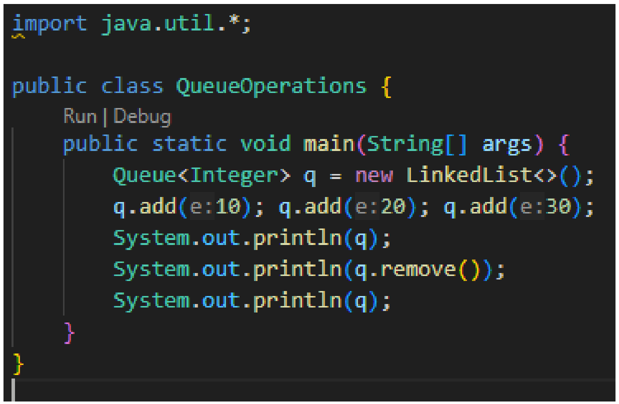


```java
class QNode {
    int data;
    QNode next;
    QNode(int d) { data = d; }
}

public class QueueLinkedList {
    QNode front, rear;

    public void enqueue(int x) {
        QNode temp = new QNode(x);
        if (rear == null) front = rear = temp;
        else {
            rear.next = temp;
            rear = temp;
        }
    }

    public int dequeue() {
        if (front == null) return -1;
        int val = front.data;
        front = front.next;
        if (front == null) rear = null;
        return val;
    }

    public void display() {
        QNode curr = front;
        while (curr != null) {
            System.out.print(curr.data+" ");
            curr = curr.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        QueueLinkedList q = new QueueLinkedList();
        q.enqueue(x:1); q.enqueue(x:2); q.enqueue(x:3);
        q.display();
        System.out.println(q.dequeue());
        q.display();
    }
}
```

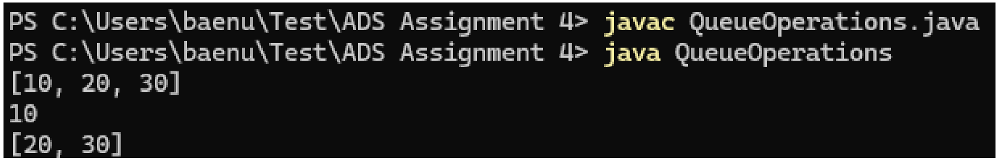


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac QueueLinkedList.java
PS C:\Users\baenu\Test\ADS Assignment 4> java QueueLinkedList
1 2 3
1
2 3
```

3. Write a program to enqueue, dequeue, and display elements of a queue.

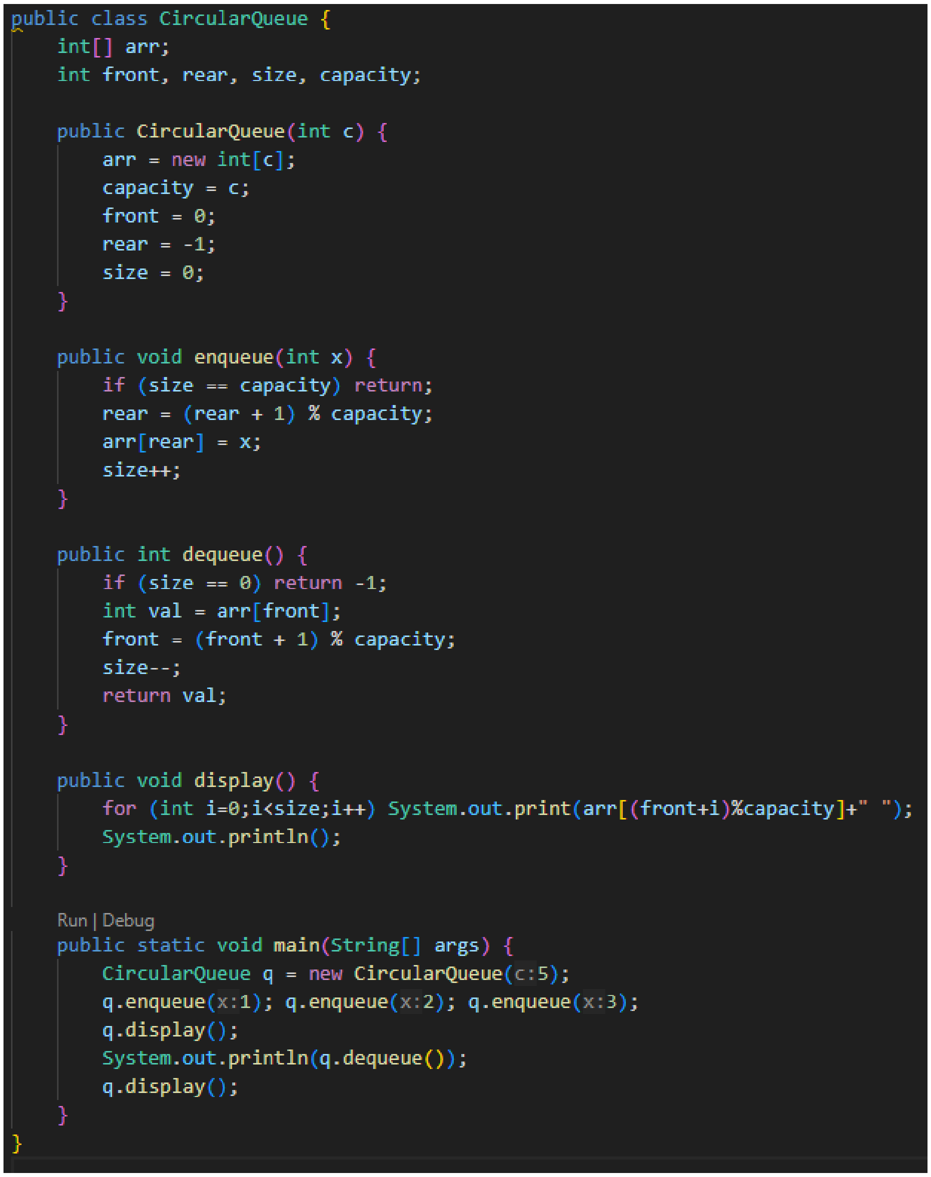


```
import java.util.*;

public class QueueOperations {
    Run | Debug
    public static void main(String[] args) {
        Queue<Integer> q = new LinkedList<>();
        q.add(e:10); q.add(e:20); q.add(e:30);
        System.out.println(q);
        System.out.println(q.remove());
        System.out.println(q);
    }
}
```

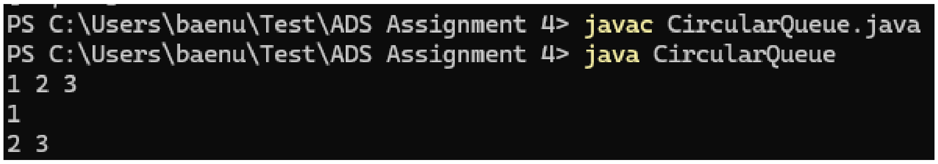


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac QueueOperations.java
PS C:\Users\baenu\Test\ADS Assignment 4> java QueueOperations
[10, 20, 30]
10
[20, 30]
```

4. Implement a Circular Queue using arrays.

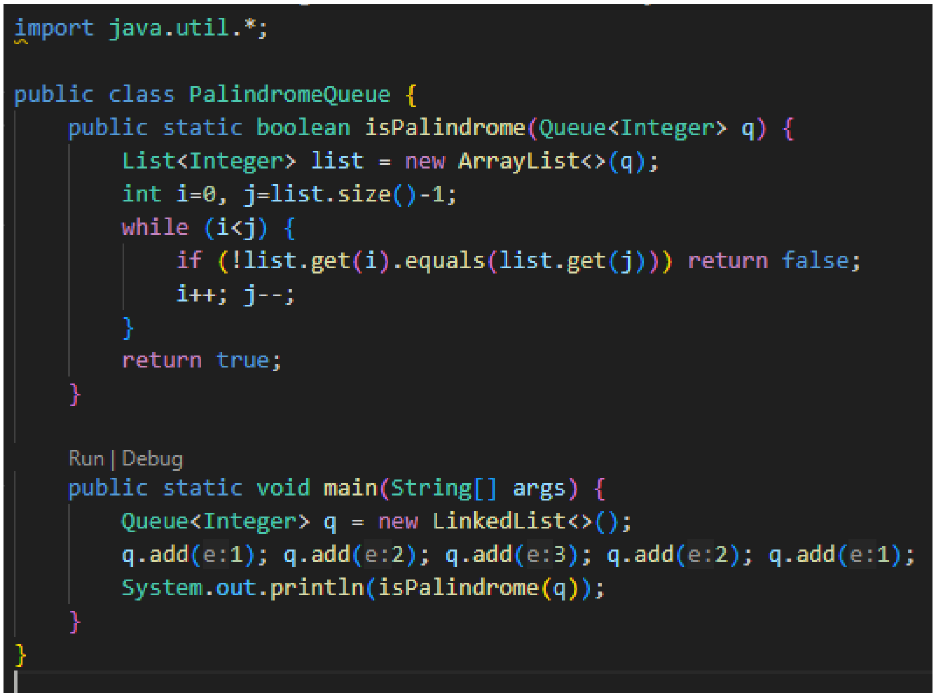


```
public class CircularQueue {
    int[] arr;
    int front, rear, size, capacity;

    public CircularQueue(int c) {
        arr = new int[c];
        capacity = c;
        front = 0;
        rear = -1;
        size = 0;
    }

    public void enqueue(int x) {
        if (size == capacity) return;
        rear = (rear + 1) % capacity;
        arr[rear] = x;
        size++;
    }

    public int dequeue() {
        if (size == 0) return -1;
        int val = arr[front];
        front = (front + 1) % capacity;
        size--;
        return val;
    }

    public void display() {
        for (int i=0;i<size;i++) System.out.print(arr[(front+i)%capacity]+" ");
        System.out.println();
    }

    Run | Debug
    public static void main(String[] args) {
        CircularQueue q = new CircularQueue(c:5);
        q.enqueue(x:1); q.enqueue(x:2); q.enqueue(x:3);
        q.display();
        System.out.println(q.dequeue());
        q.display();
    }
}
```

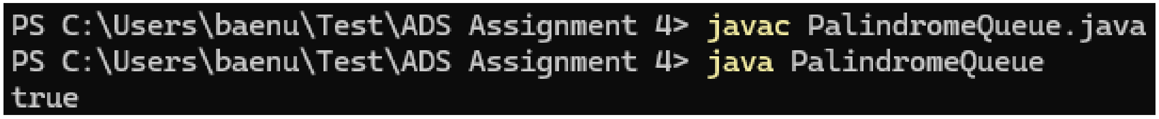


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac CircularQueue.java
PS C:\Users\baenu\Test\ADS Assignment 4> java CircularQueue
1 2 3
1
2 3
```

5. Check if a queue is palindrome (using stack or two-pointer approach).
Example: 1 → 2 → 3 → 2 → 1 → Palindrome.

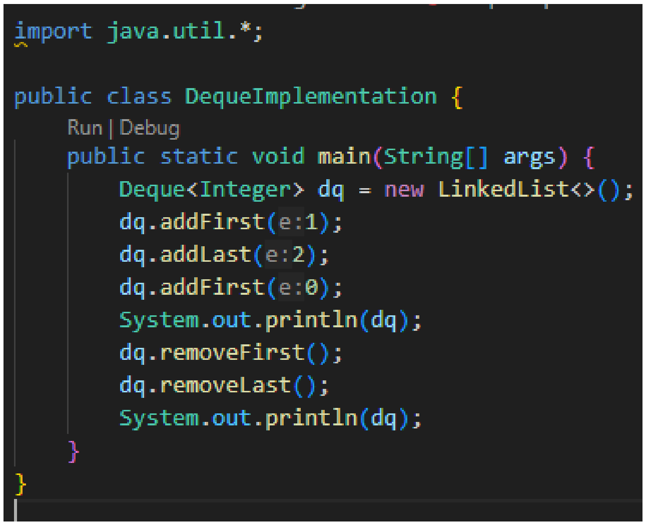


```
import java.util.*;

public class PalindromeQueue {
    public static boolean isPalindrome(Queue<Integer> q) {
        List<Integer> list = new ArrayList<>(q);
        int i=0, j=list.size()-1;
        while (i<j) {
            if (!list.get(i).equals(list.get(j))) return false;
            i++; j--;
        }
        return true;
    }

    Run | Debug
    public static void main(String[] args) {
        Queue<Integer> q = new LinkedList<>();
        q.add(e:1); q.add(e:2); q.add(e:3); q.add(e:2); q.add(e:1);
        System.out.println(isPalindrome(q));
    }
}
```

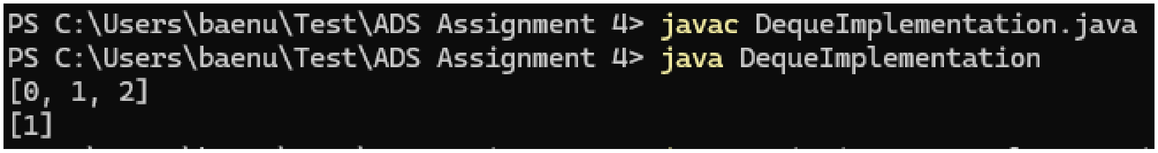


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac PalindromeQueue.java
PS C:\Users\baenu\Test\ADS Assignment 4> java PalindromeQueue
true
```

6. Implement a Double Ended Queue (Deque).

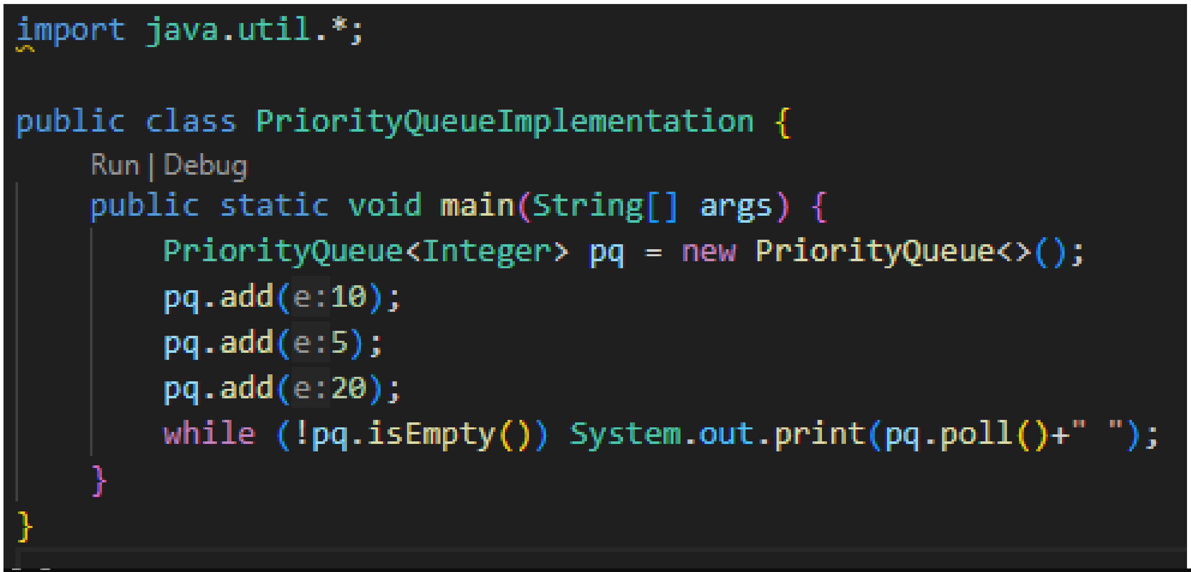


```
import java.util.*;

public class DequeImplementation {
    Run | Debug
    public static void main(String[] args) {
        Deque<Integer> dq = new LinkedList<>();
        dq.addFirst(e:1);
        dq.addLast(e:2);
        dq.addFirst(e:0);
        System.out.println(dq);
        dq.removeFirst();
        dq.removeLast();
        System.out.println(dq);
    }
}
```

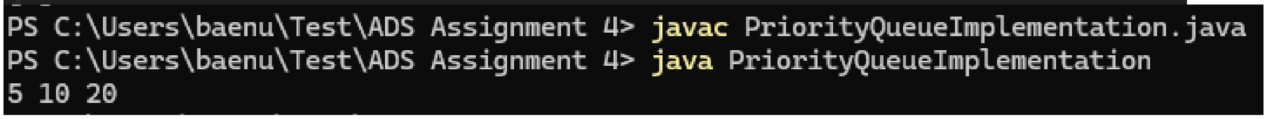


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac DequeImplementation.java
PS C:\Users\baenu\Test\ADS Assignment 4> java DequeImplementation
[0, 1, 2]
[1]
```

7. Implement a Priority Queue (using array or heap).

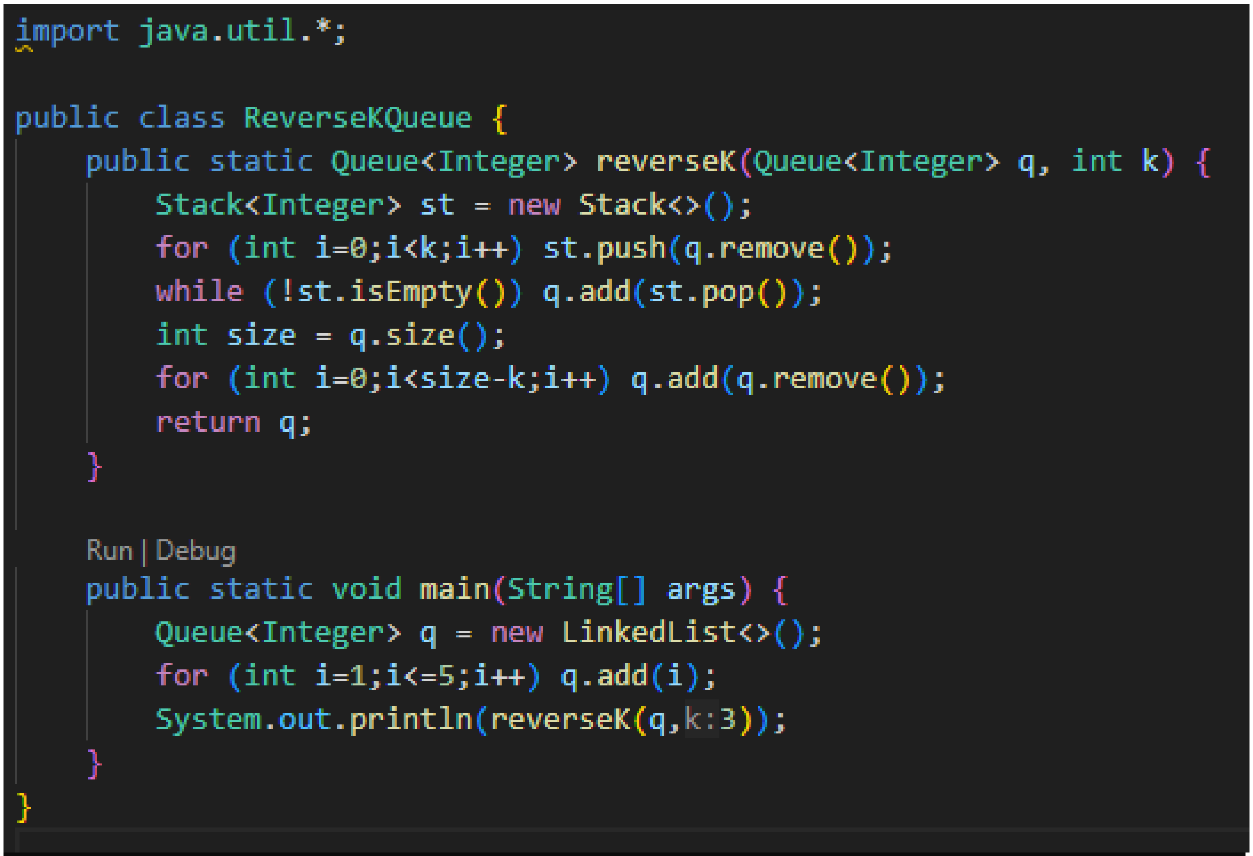


```java
import java.util.*;

public class PriorityQueueImplementation {
    Run | Debug
    public static void main(String[] args) {
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pq.add(e:10);
        pq.add(e:5);
        pq.add(e:20);
        while (!pq.isEmpty()) System.out.print(pq.poll()+" ");
    }
}
```

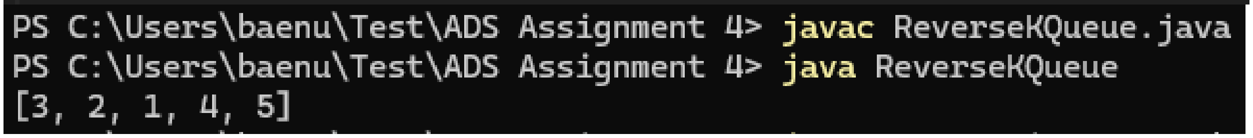


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac PriorityQueueImplementation.java
PS C:\Users\baenu\Test\ADS Assignment 4> java PriorityQueueImplementation
5 10 20
```

8. Reverse the first K elements of a queue.
Example: Queue = [1,2,3,4,5], K=3 → [3,2,1,4,5].

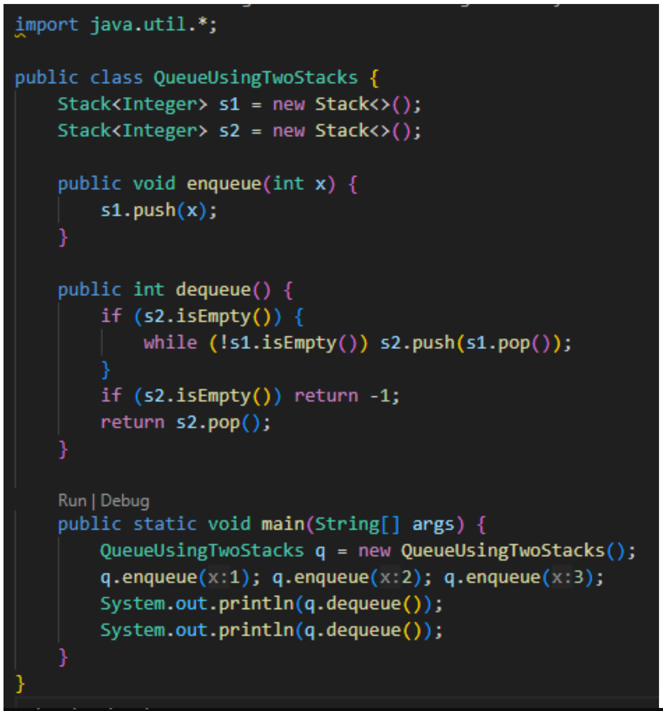


```java
import java.util.*;

public class ReverseKQueue {
    public static Queue<Integer> reverseK(Queue<Integer> q, int k) {
        Stack<Integer> st = new Stack<>();
        for (int i=0;i<k;i++) st.push(q.remove());
        while (!st.isEmpty()) q.add(st.pop());
        int size = q.size();
        for (int i=0;i<size-k;i++) q.add(q.remove());
        return q;
    }

    Run | Debug
    public static void main(String[] args) {
        Queue<Integer> q = new LinkedList<>();
        for (int i=1;i<=5;i++) q.add(i);
        System.out.println(reverseK(q,k:3));
    }
}
```

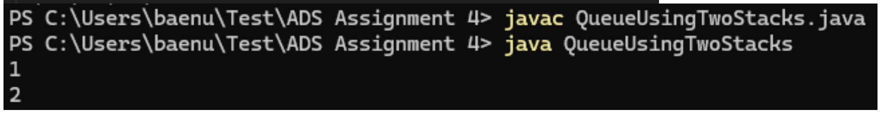


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac ReverseKQueue.java
PS C:\Users\baenu\Test\ADS Assignment 4> java ReverseKQueue
[3, 2, 1, 4, 5]
```

9. Implement a queue using two stacks.

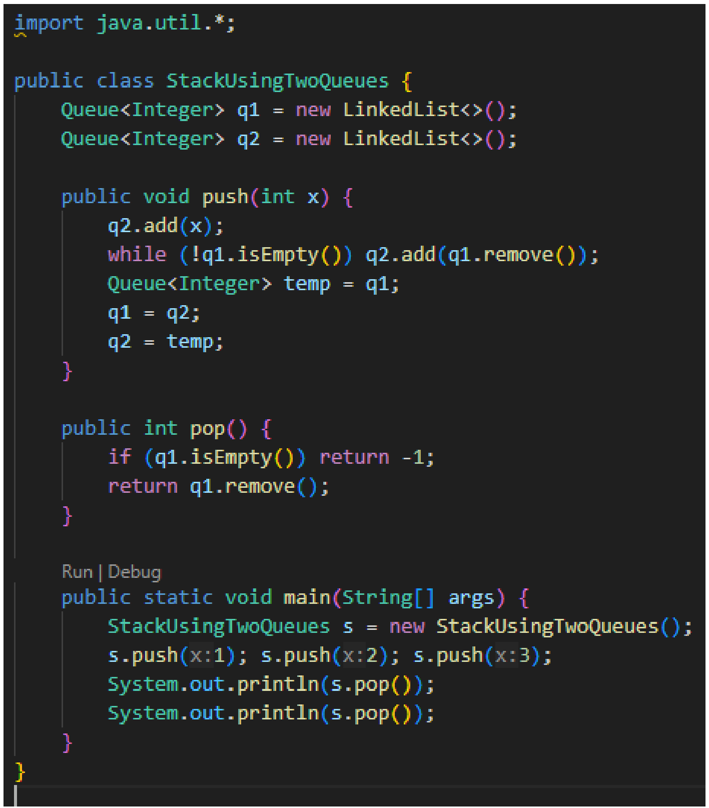


```
import java.util.*;

public class QueueUsingTwoStacks {
    Stack<Integer> s1 = new Stack<>();
    Stack<Integer> s2 = new Stack<>();

    public void enqueue(int x) {
        s1.push(x);
    }

    public int dequeue() {
        if (s2.isEmpty()) {
            while (!s1.isEmpty()) s2.push(s1.pop());
        }
        if (s2.isEmpty()) return -1;
        return s2.pop();
    }

    Run | Debug
    public static void main(String[] args) {
        QueueUsingTwoStacks q = new QueueUsingTwoStacks();
        q.enqueue(x:1); q.enqueue(x:2); q.enqueue(x:3);
        System.out.println(q.dequeue());
        System.out.println(q.dequeue());
    }
}
```

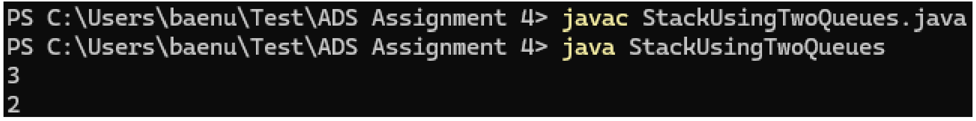


```
PS C:\Users\baenu\Test\ADS Assignment 4> javac QueueUsingTwoStacks.java
PS C:\Users\baenu\Test\ADS Assignment 4> java QueueUsingTwoStacks
1
2
```

10. Implement a stack using two queues.



```
import java.util.*;

public class StackUsingTwoQueues {
    Queue<Integer> q1 = new LinkedList<>();
    Queue<Integer> q2 = new LinkedList<>();

    public void push(int x) {
        q2.add(x);
        while (!q1.isEmpty()) q2.add(q1.remove());
        Queue<Integer> temp = q1;
        q1 = q2;
        q2 = temp;
    }

    public int pop() {
        if (q1.isEmpty()) return -1;
        return q1.remove();
    }

    Run | Debug
    public static void main(String[] args) {
        StackUsingTwoQueues s = new StackUsingTwoQueues();
        s.push(x:1); s.push(x:2); s.push(x:3);
        System.out.println(s.pop());
        System.out.println(s.pop());
    }
}
```



```
PS C:\Users\baenu\Test\ADS Assignment 4> javac StackUsingTwoQueues.java
PS C:\Users\baenu\Test\ADS Assignment 4> java StackUsingTwoQueues
3
2
```