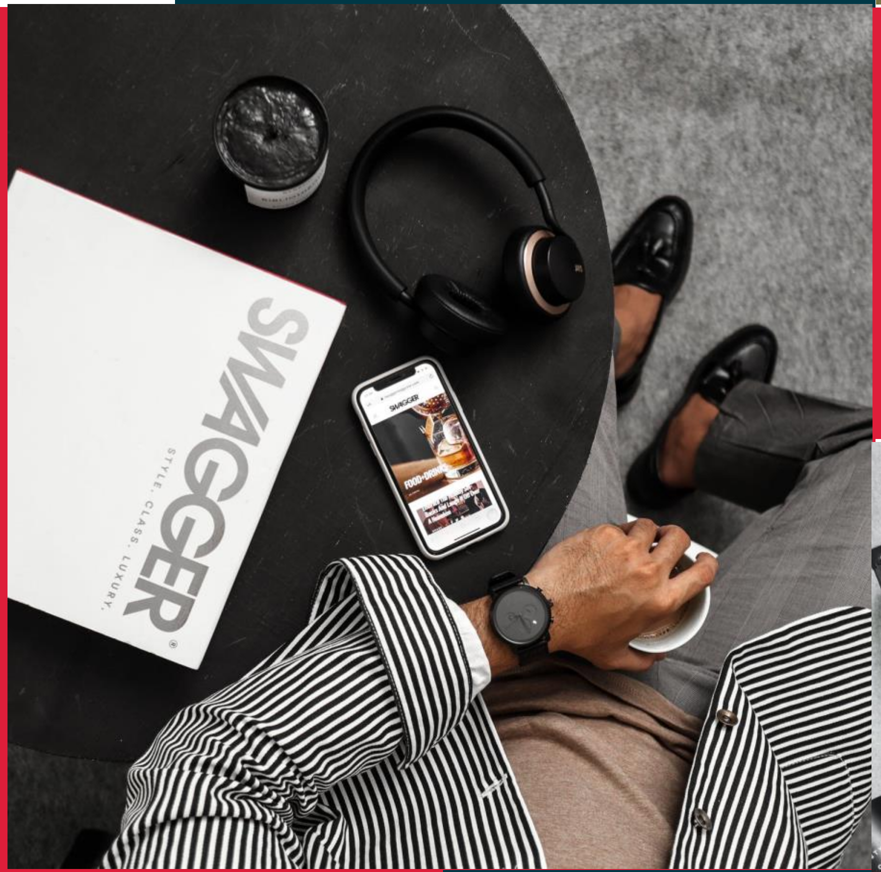
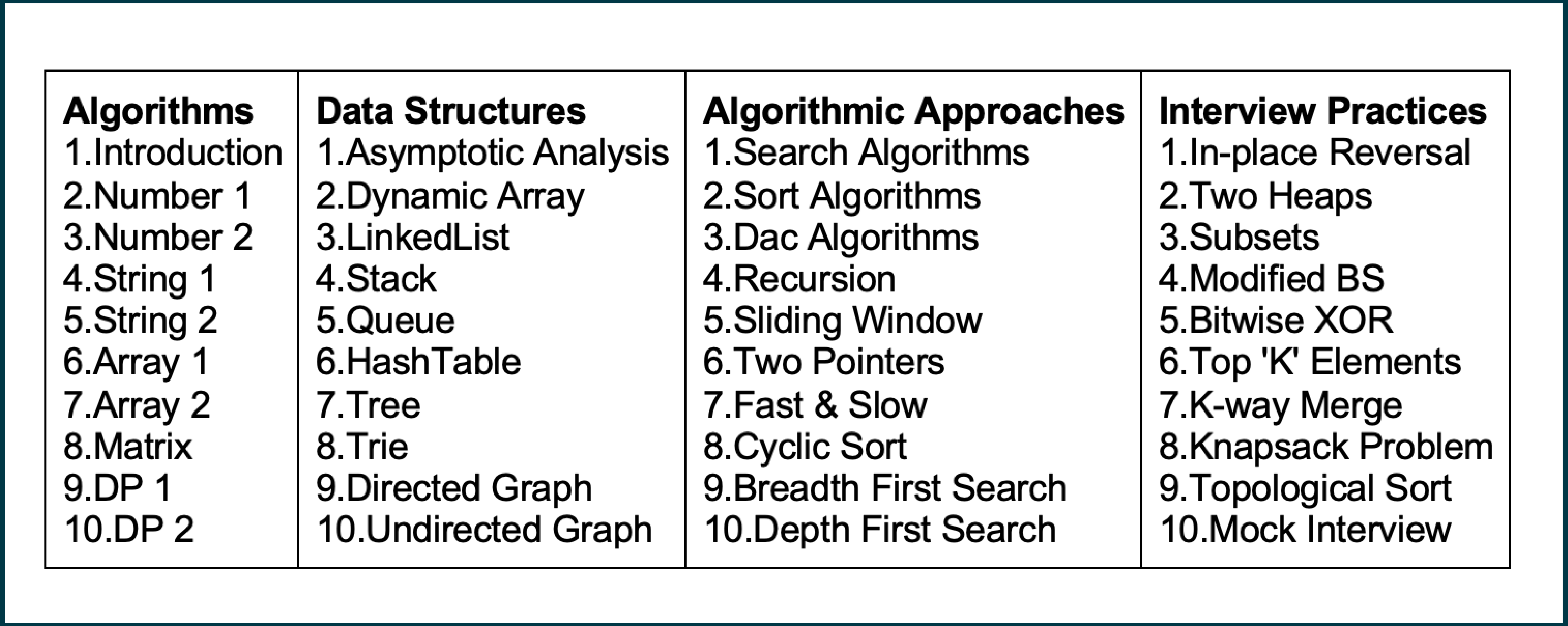
PDP
ONLINE

DSA – Data Structures

Queue



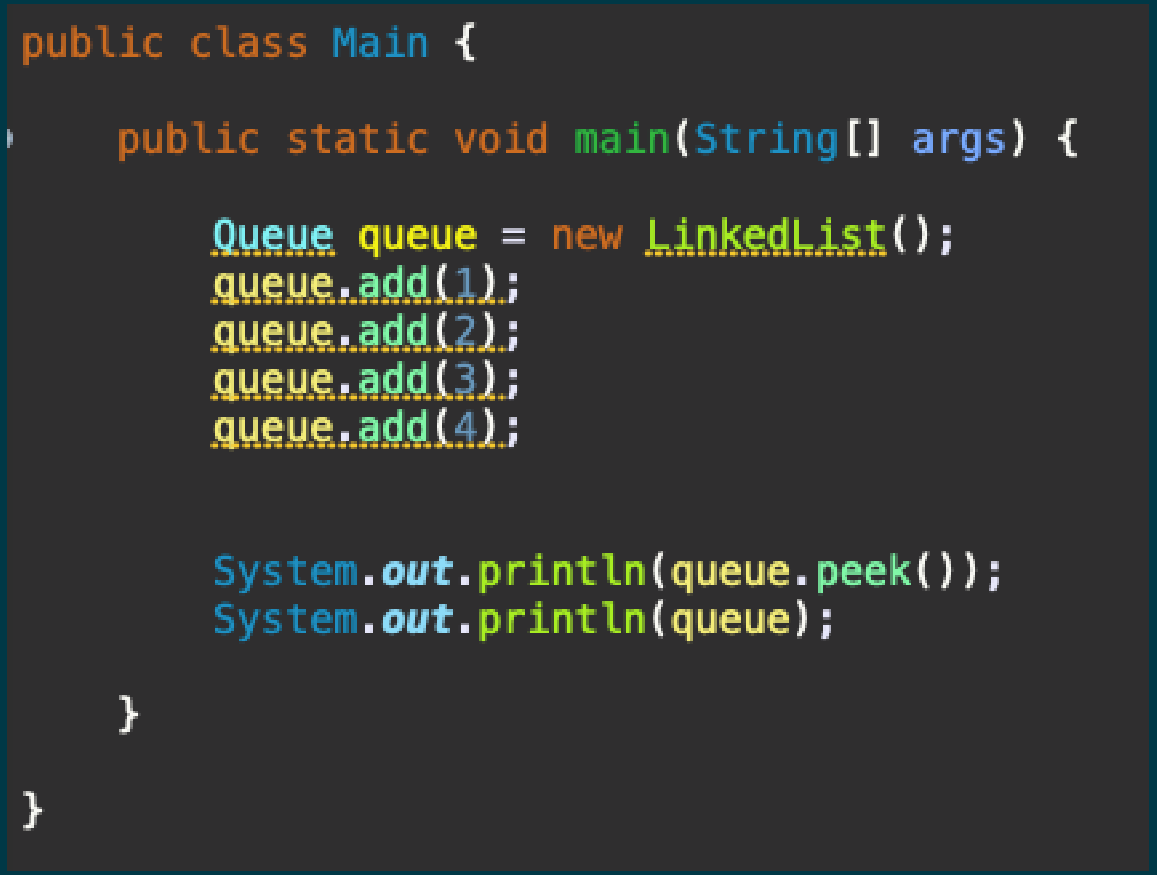
Course Planning



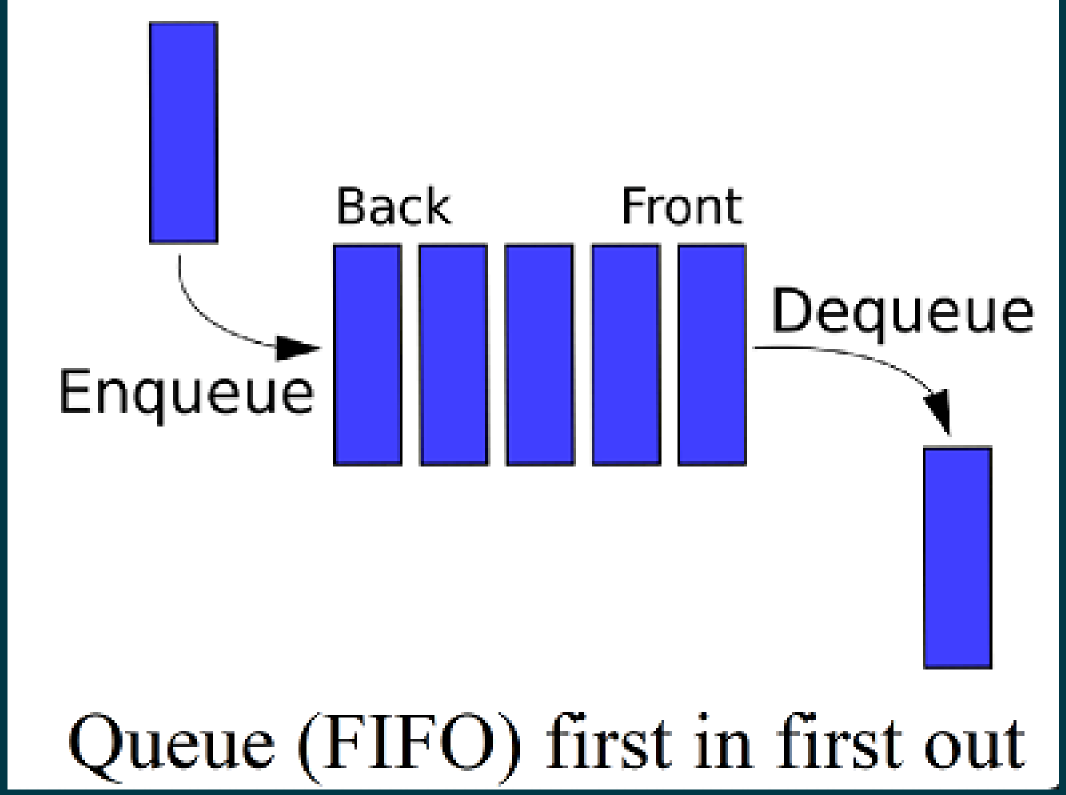
Algorithms	Data Structures	Algorithmic Approaches	Interview Practices
1.Introduction	1.Asymptotic Analysis	1.Search Algorithms	1.In-place Reversal
2.Number 1	2.Dynamic Array	2.Sort Algorithms	2.Two Heaps
3.Number 2	3.LinkedList	3.Dac Algorithms	3.Subsets
4.String 1	4.Stack	4.Recursion	4.Modified BS
5.String 2	5.Queue	5.Sliding Window	5.Bitwise XOR
6.Array 1	6.HashTable	6.Two Pointers	6.Top 'K' Elements
7.Array 2	7.Tree	7.Fast & Slow	7.K-way Merge
8.Matrix	8.Trie	8.Cyclic Sort	8.Knapsack Problem
9.DP 1	9.Directed Graph	9.Breadth First Search	9.Topological Sort
10.DP 2	10.Undirected Graph	10.Depth First Search	10.Mock Interview

https://kurbanovxurshid.medium.com/data-structures-and-algorithms-course-plan-1ab912ec1e17

java.util.Queue

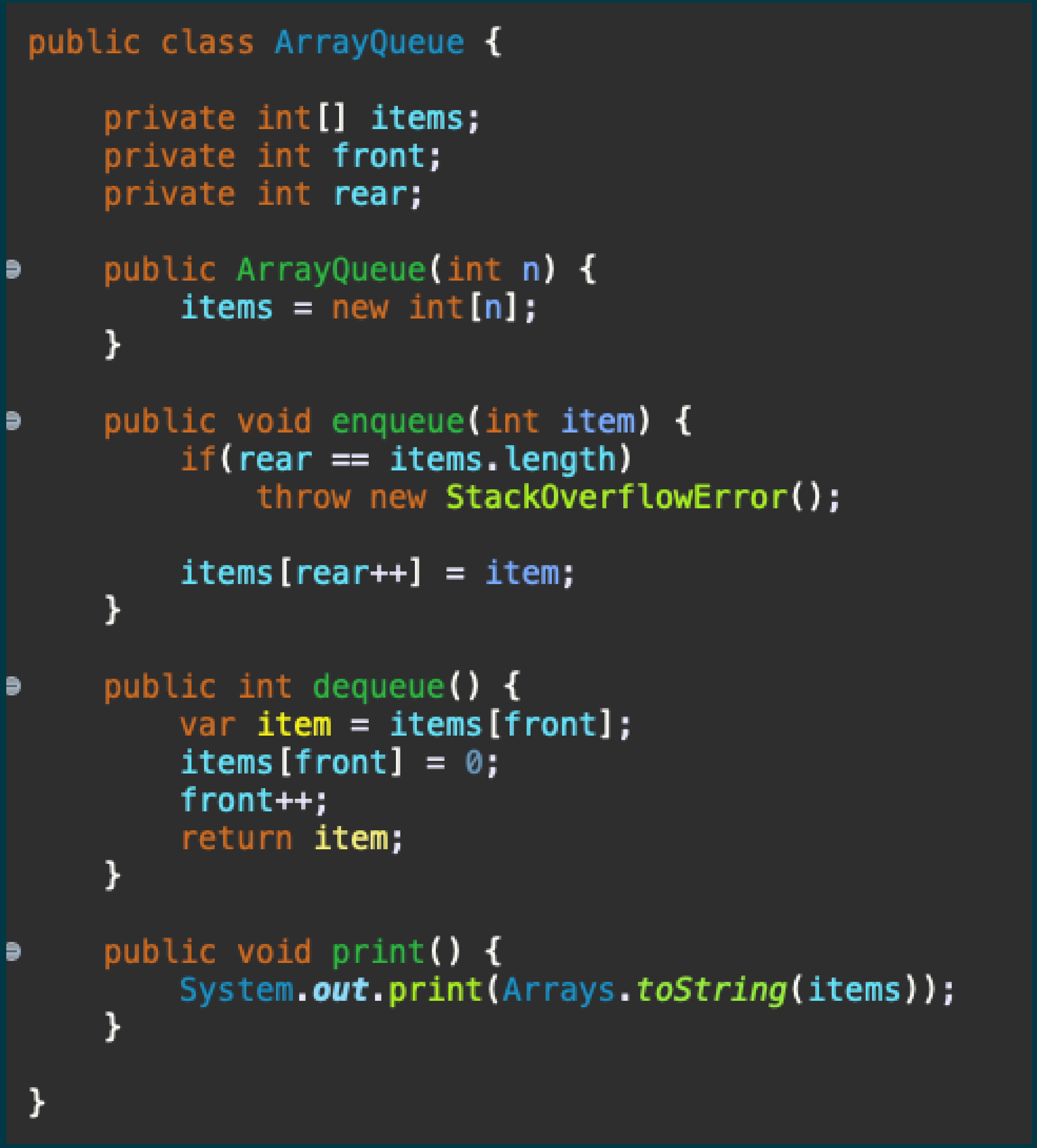


```java
public class Main {

    public static void main(String[] args) {

        Queue queue = new LinkedList();
        queue.add(1);
        queue.add(2);
        queue.add(3);
        queue.add(4);


        System.out.println(queue.peek());
        System.out.println(queue);

    }

}
```



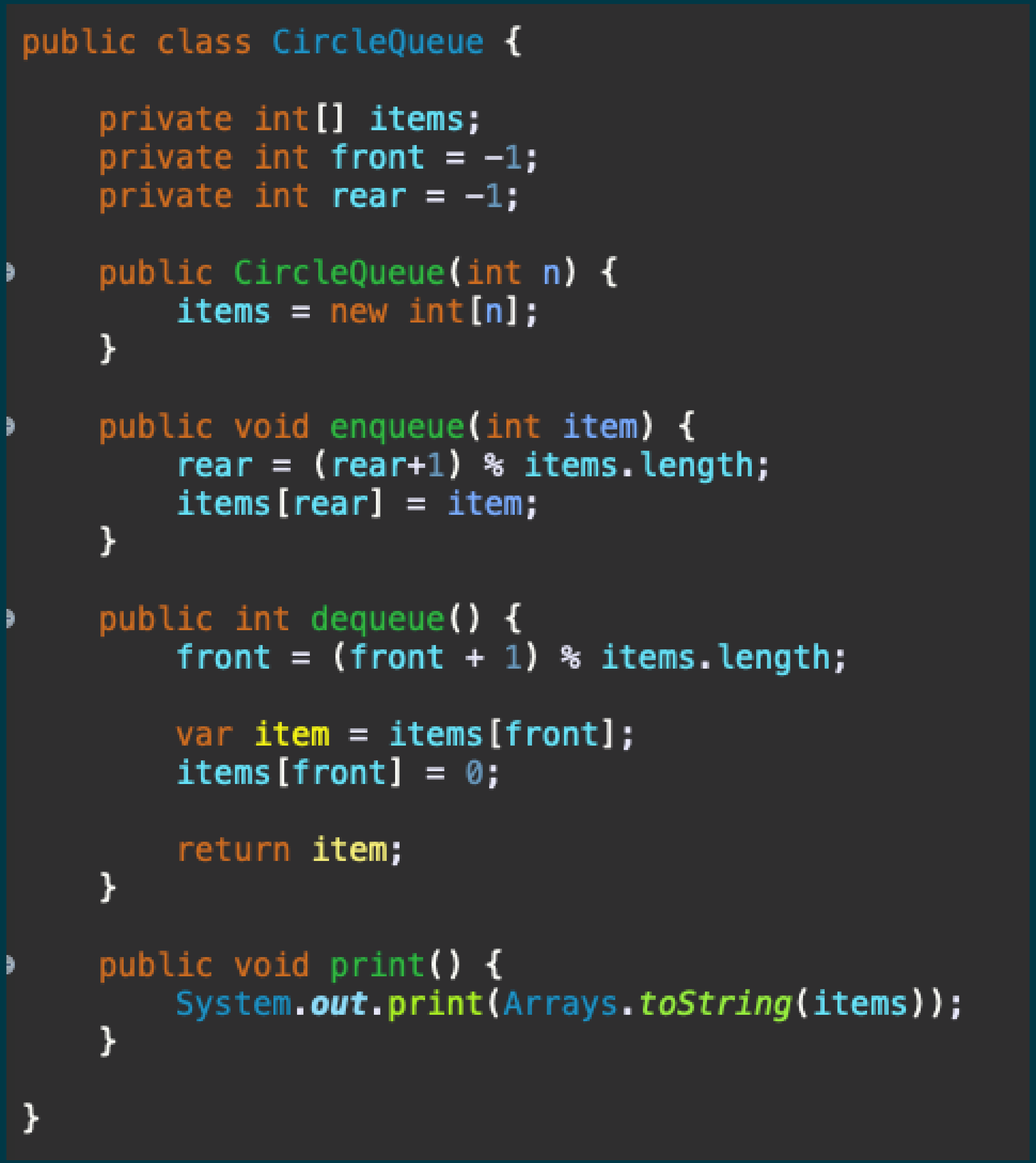
Queue (FIFO) first in first out

Queue using Array



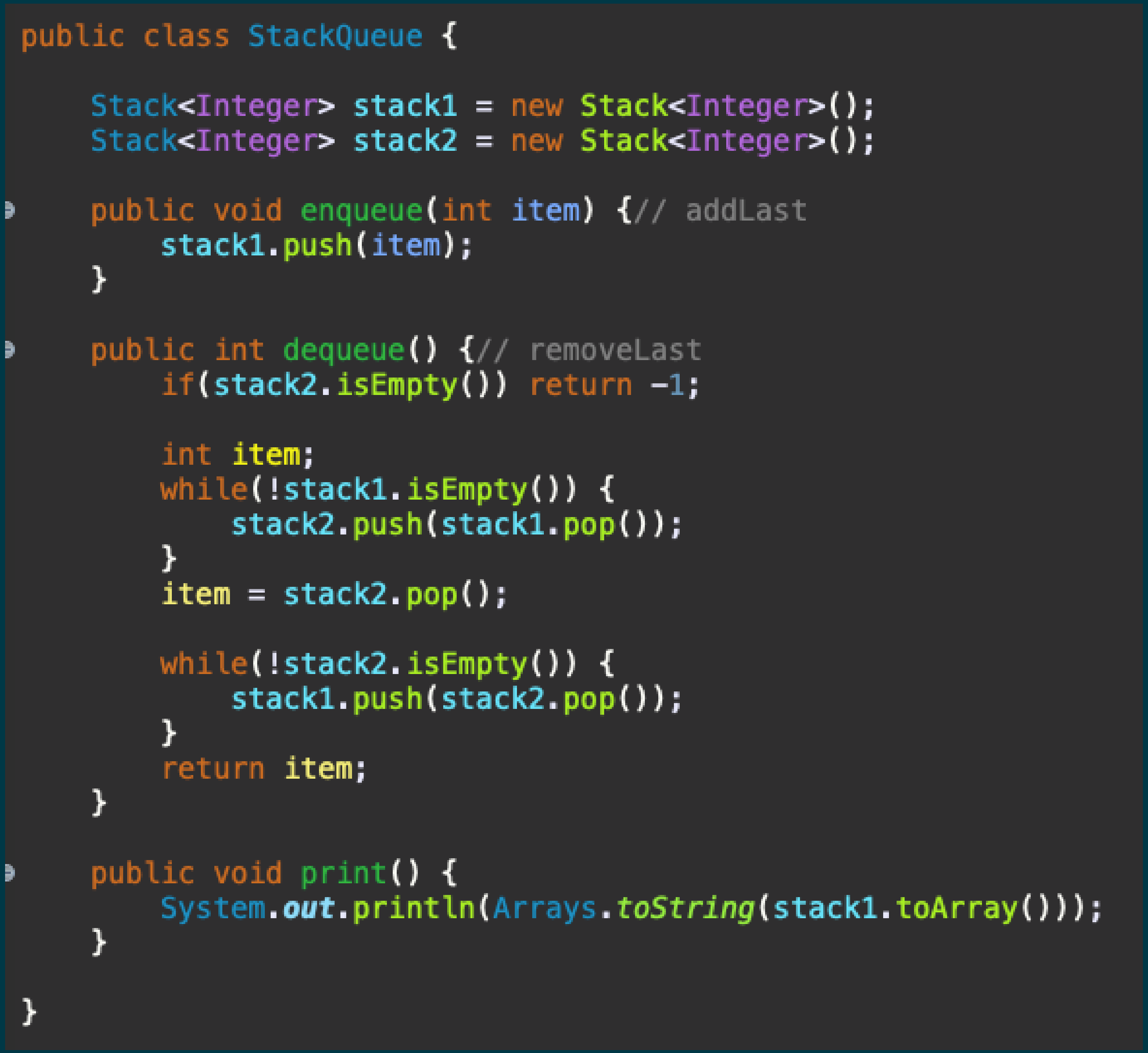
```java
public class ArrayQueue {

    private int[] items;
    private int front;
    private int rear;

    public ArrayQueue(int n) {
        items = new int[n];
    }

    public void enqueue(int item) {
        if(rear == items.length)
            throw new StackOverflowError();

        items[rear++] = item;
    }

    public int dequeue() {
        var item = items[front];
        items[front] = 0;
        front++;
        return item;
    }

    public void print() {
        System.out.print(Arrays.toString(items));
    }

}
```

Circle Queue



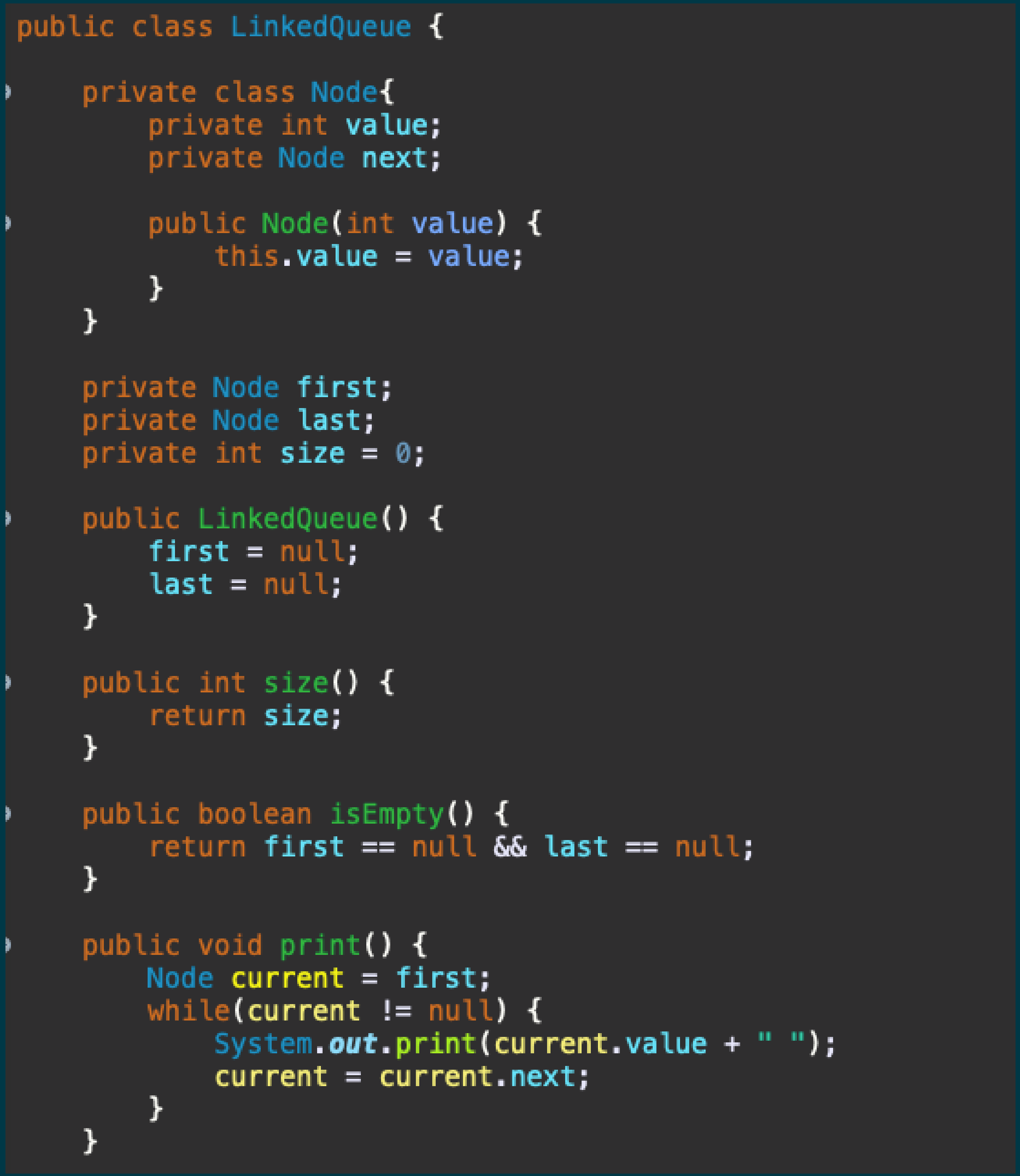
```
public class CircleQueue {

    private int[] items;
    private int front = -1;
    private int rear = -1;

    public CircleQueue(int n) {
        items = new int[n];
    }

    public void enqueue(int item) {
        rear = (rear+1) % items.length;
        items[rear] = item;
    }

    public int dequeue() {
        front = (front + 1) % items.length;

        var item = items[front];
        items[front] = 0;

        return item;
    }

    public void print() {
        System.out.print(Arrays.toString(items));
    }

}
```

Queue using Stacks



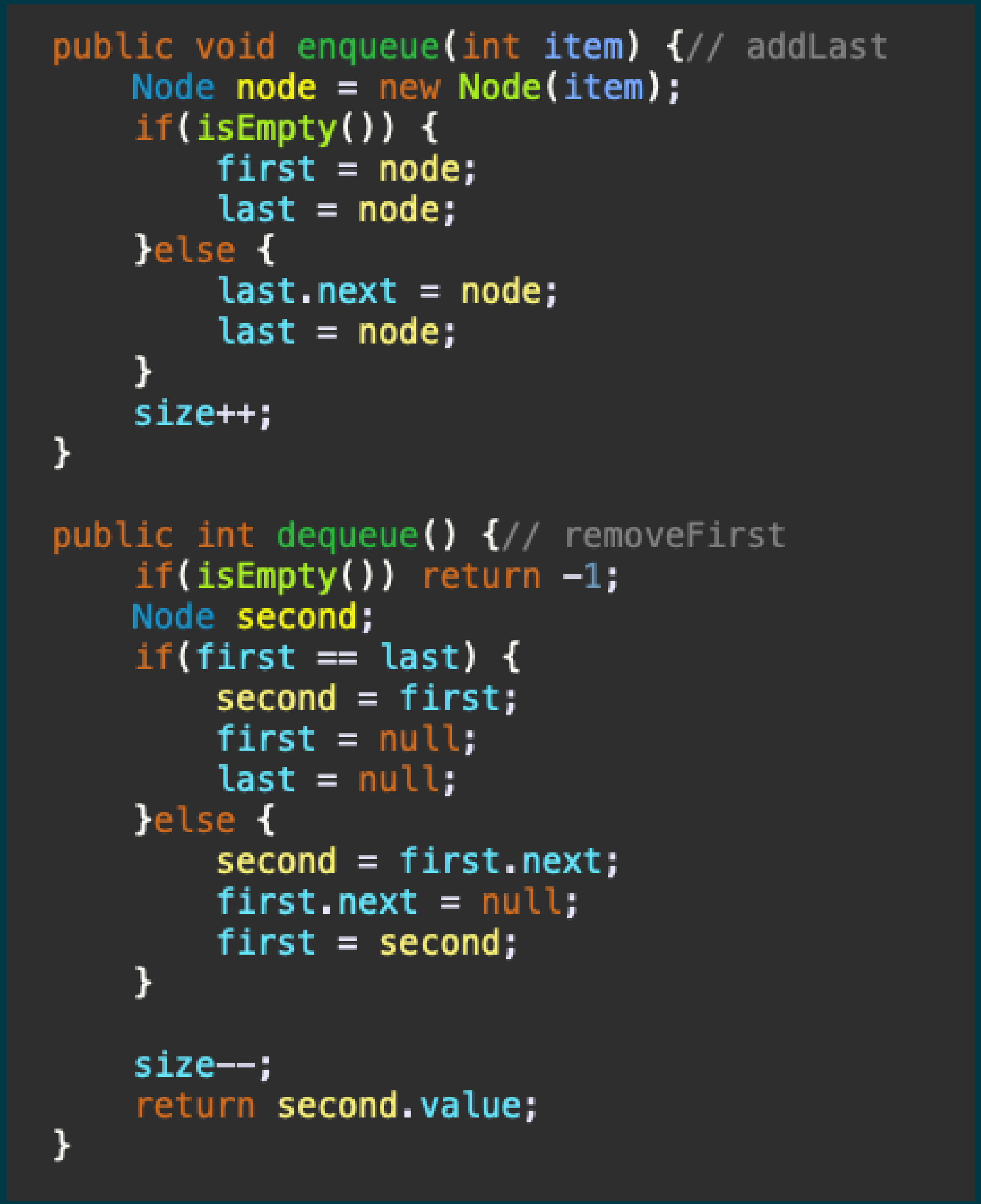
```java
public class StackQueue {

    Stack<Integer> stack1 = new Stack<Integer>();
    Stack<Integer> stack2 = new Stack<Integer>();

    public void enqueue(int item) {// addLast
        stack1.push(item);
    }

    public int dequeue() {// removeLast
        if(stack2.isEmpty()) return -1;

        int item;
        while(!stack1.isEmpty()) {
            stack2.push(stack1.pop());
        }
        item = stack2.pop();

        while(!stack2.isEmpty()) {
            stack1.push(stack2.pop());
        }
        return item;
    }

    public void print() {
        System.out.println(Arrays.toString(stack1.toArray()));
    }

}
```

Linked Queue



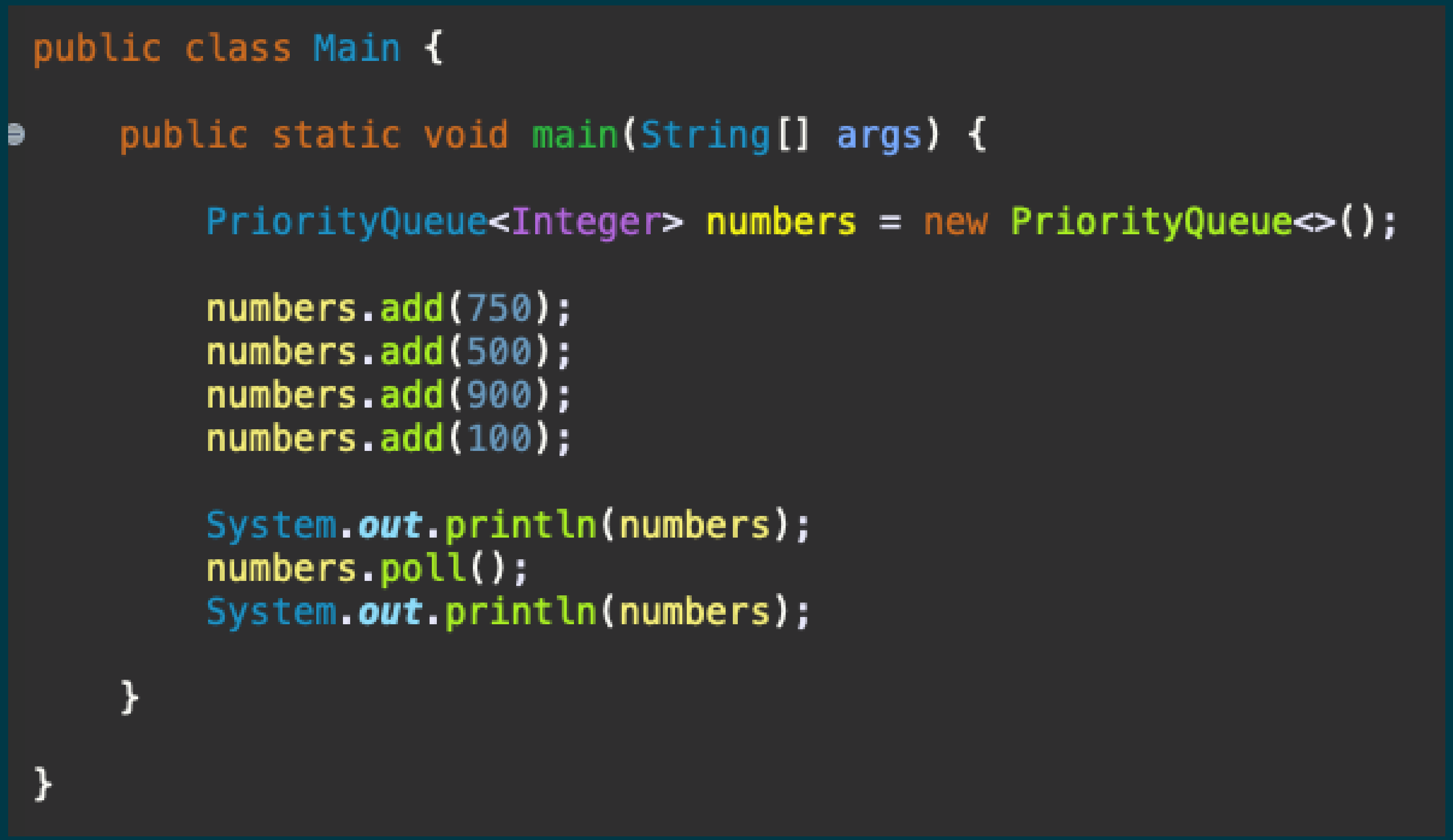
```java
public class LinkedQueue {

    private class Node{
        private int value;
        private Node next;

        public Node(int value) {
            this.value = value;
        }
    }

    private Node first;
    private Node last;
    private int size = 0;

    public LinkedQueue() {
        first = null;
        last = null;
    }

    public int size() {
        return size;
    }

    public boolean isEmpty() {
        return first == null && last == null;
    }

    public void print() {
        Node current = first;
        while(current != null) {
            System.out.print(current.value + " ");
            current = current.next;
        }
    }
```

enqueue, dequeue



```java
public void enqueue(int item) {// addLast
    Node node = new Node(item);
    if(isEmpty()) {
        first = node;
        last = node;
    }else {
        last.next = node;
        last = node;
    }
    size++;
}

public int dequeue() {// removeFirst
    if(isEmpty()) return -1;
    Node second;
    if(first == last) {
        second = first;
        first = null;
        last = null;
    }else {
        second = first.next;
        first.next = null;
        first = second;
    }

    size--;
    return second.value;
}
```

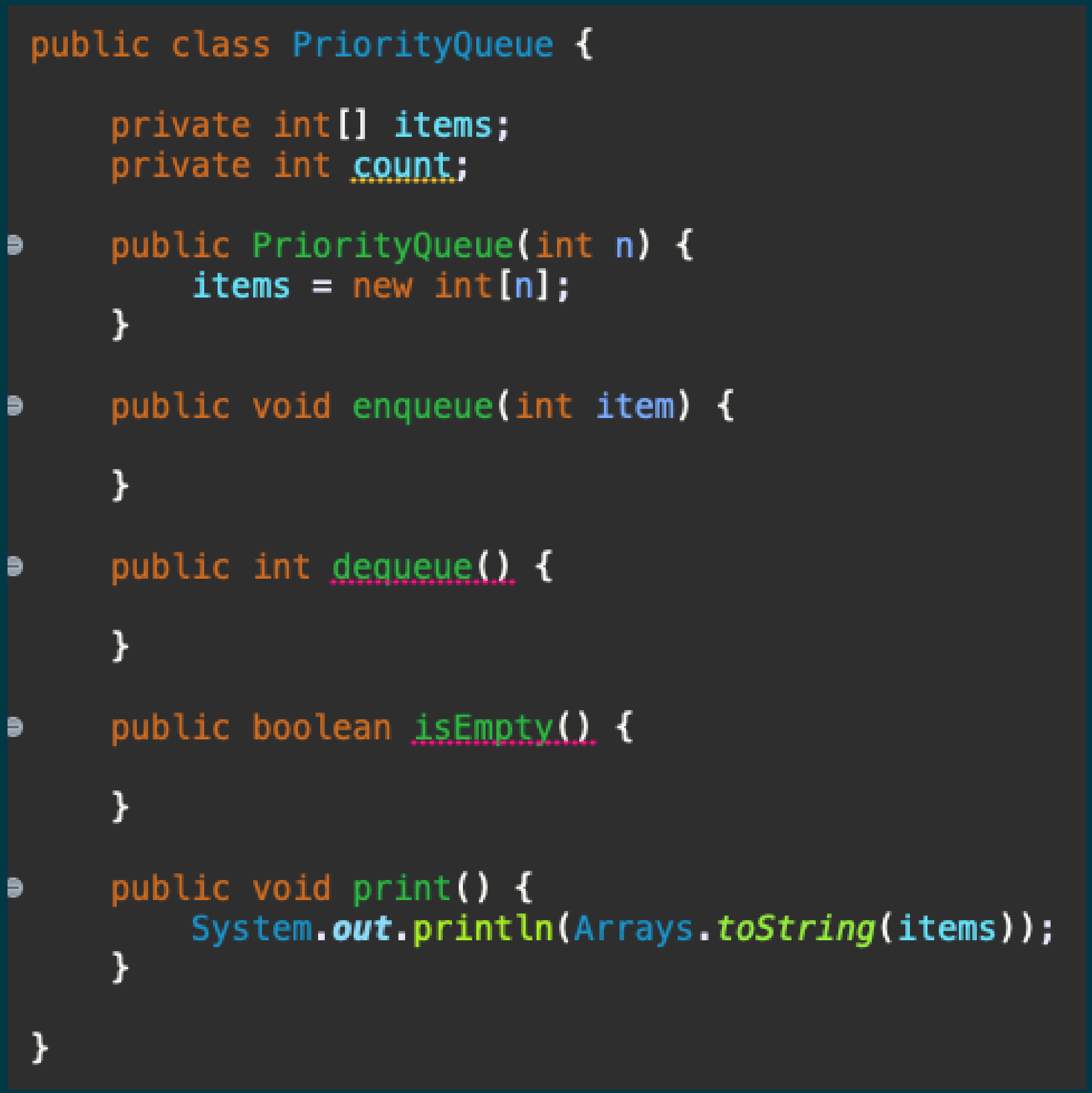
java.util.PriorityQueue



```
public class Main {

    public static void main(String[] args) {

        PriorityQueue<Integer> numbers = new PriorityQueue<>();

        numbers.add(750);
        numbers.add(500);
        numbers.add(900);
        numbers.add(100);

        System.out.println(numbers);
        numbers.poll();
        System.out.println(numbers);

    }

}
```

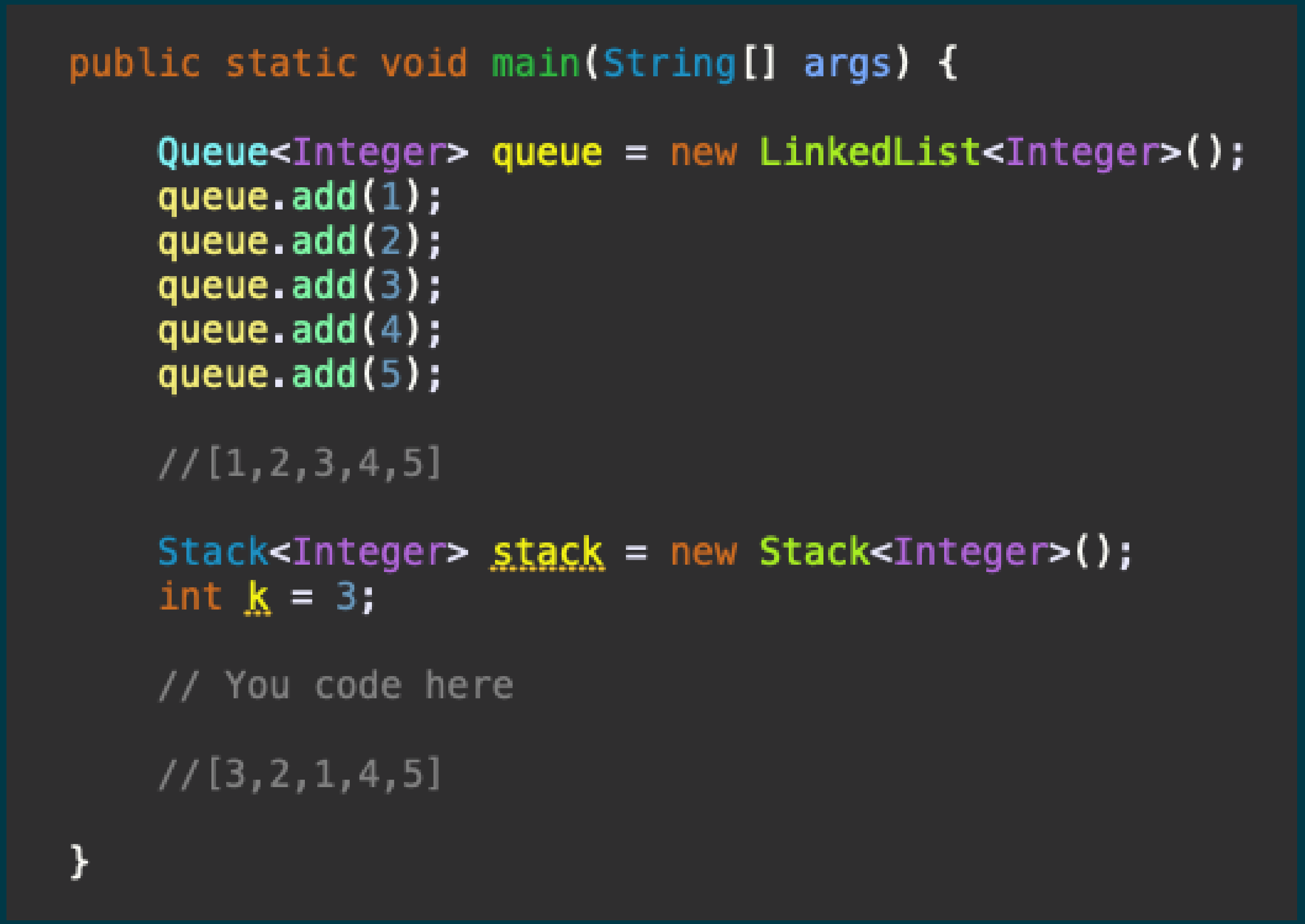
Task 1

Priority Queue ni java.util dagi holatini darsda o`rgandik. Va shu Priority Queue  ni Array dan foydalanib o`zingiz yaratib ko`rsating.



```
public class PriorityQueue {

    private int[] items;
    private int count;

    public PriorityQueue(int n) {
        items = new int[n];
    }

    public void enqueue(int item) {

    }

    public int dequeue() {

    }

    public boolean isEmpty() {

    }

    public void print() {
        System.out.println(Arrays.toString(items));
    }

}
```

Task 2
Quyidagicha berilgan Queue(java.util) berilgan bo`lsa, birinchi K elementini teskarisiga almashtiradigan qilib dastur yozing va bunda albatta Stack(java.util) dan foydalaning.



```java
public static void main(String[] args) {

    Queue<Integer> queue = new LinkedList<Integer>();
    queue.add(1);
    queue.add(2);
    queue.add(3);
    queue.add(4);
    queue.add(5);

    //[1,2,3,4,5]

    Stack<Integer> stack = new Stack<Integer>();
    int k = 3;

    // You code here

    //[3,2,1,4,5]

}
```