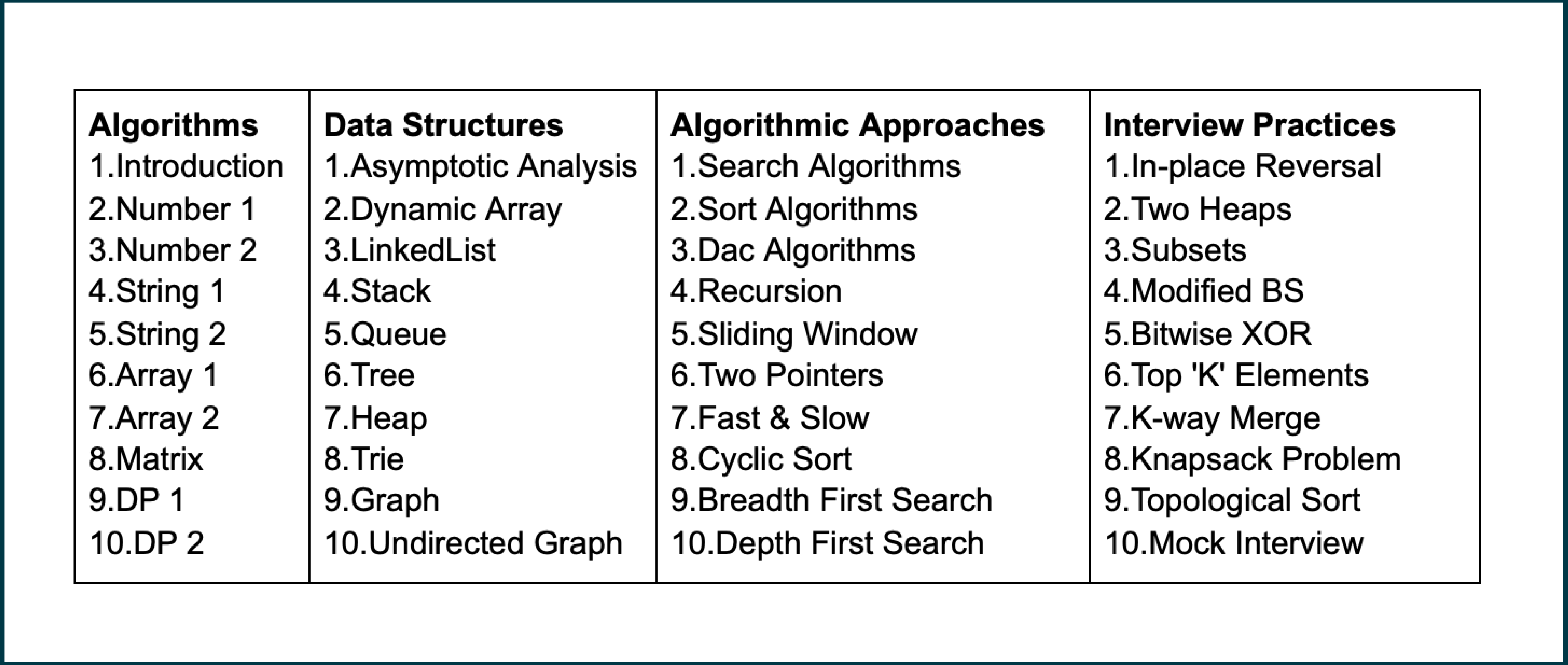
DSA – Data Structures Stack

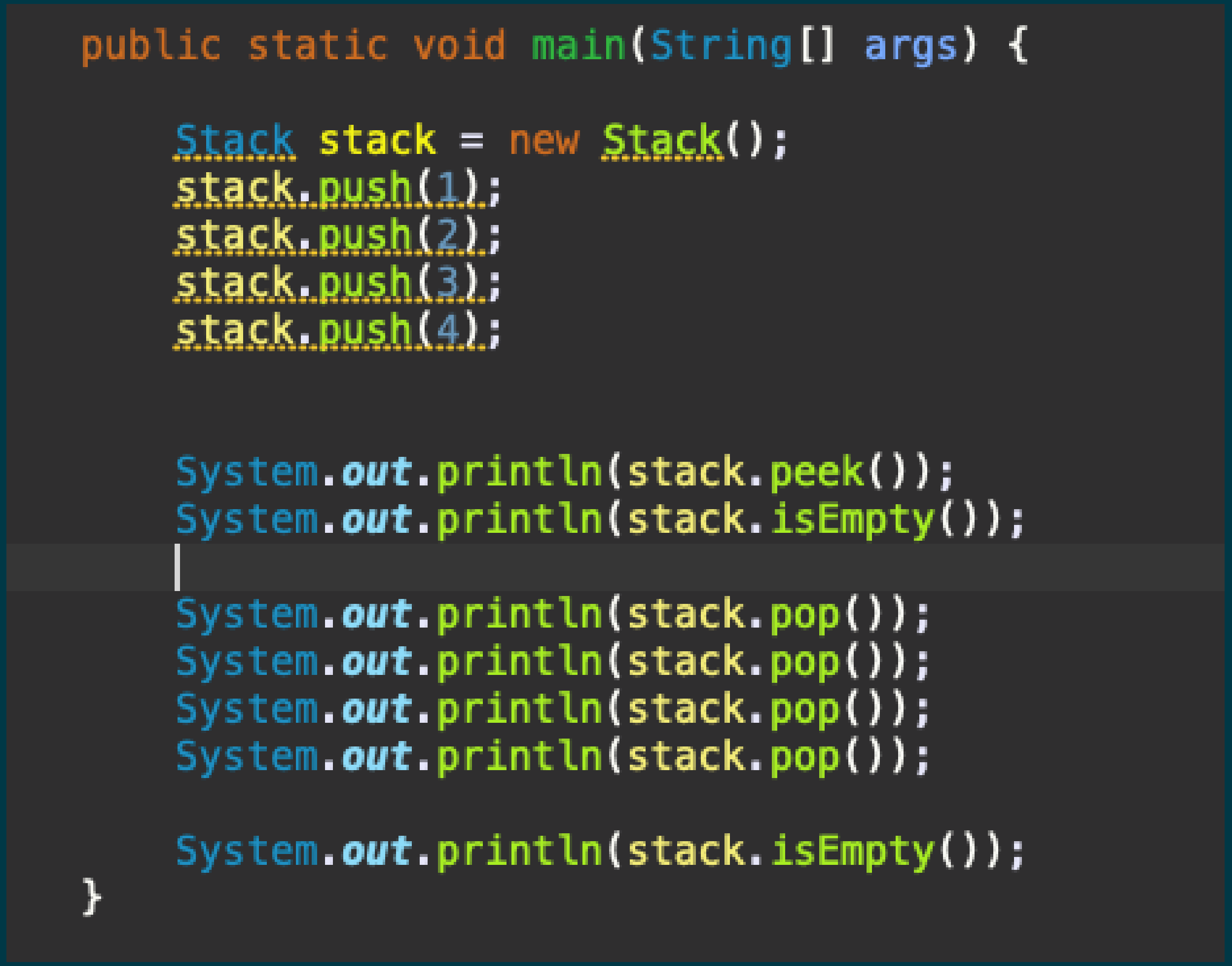
Course Planning



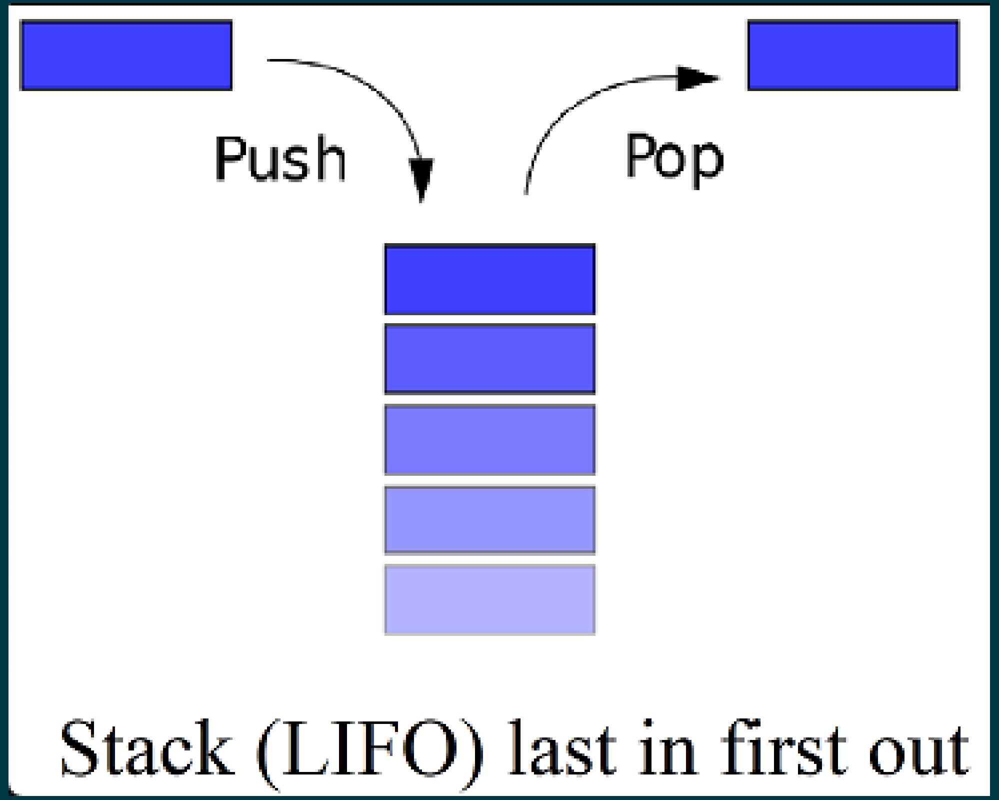
Algorithms	Data Structures	Algorithmic Approaches	Interview Practices
1.Introduction	1.Asymptotic Analysis	1.Search Algorithms	1.In-place Reversal
2.Number 1	2.Dynamic Array	2.Sort Algorithms	2.Two Heaps
3.Number 2	3.LinkedList	3.Dac Algorithms	3.Subsets
4.String 1	4.Stack	4.Recursion	4.Modified BS
5.String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

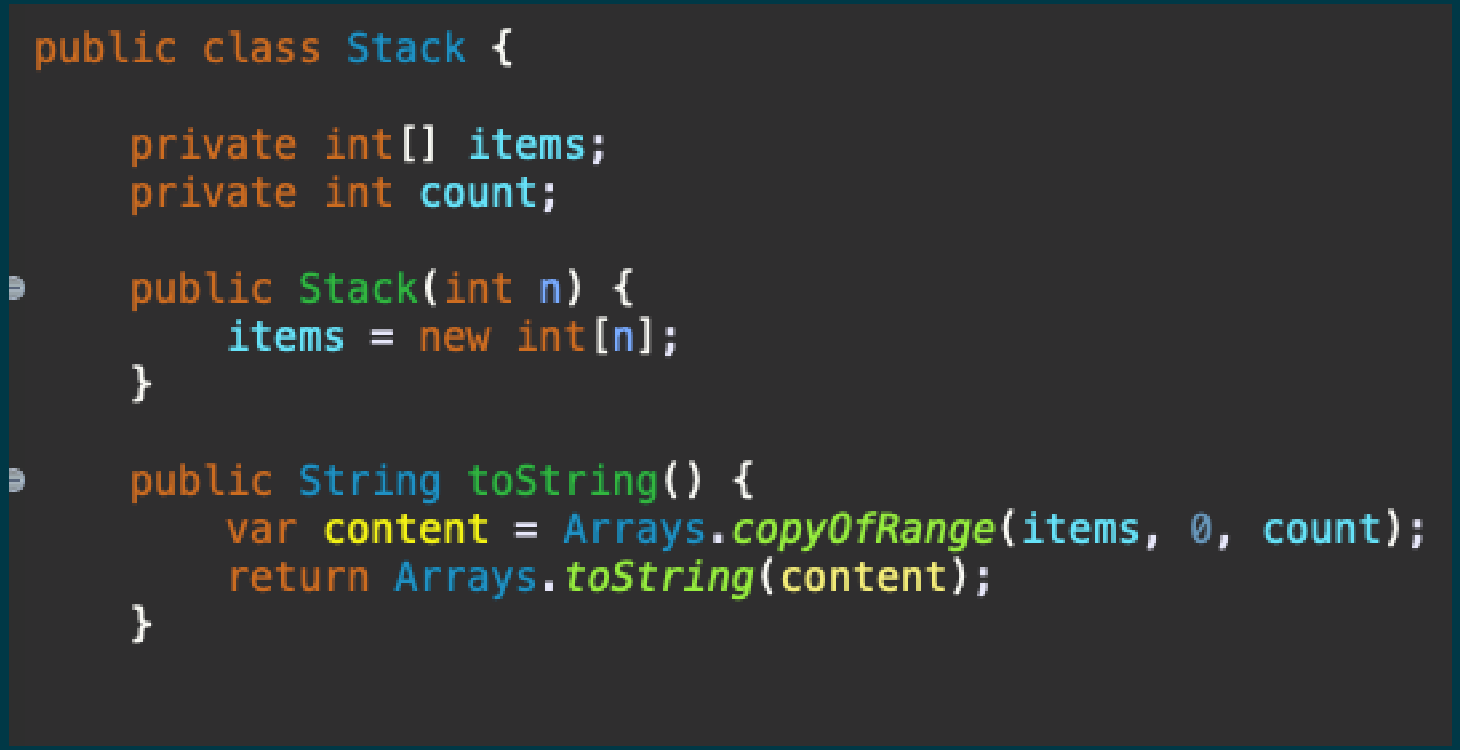
java.util.Stack



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

    Stack stack = new Stack();
    stack.push(1);
    stack.push(2);
    stack.push(3);
    stack.push(4);


    System.out.println(stack.peek());
    System.out.println(stack.isEmpty());

    System.out.println(stack.pop());
    System.out.println(stack.pop());
    System.out.println(stack.pop());
    System.out.println(stack.pop());

    System.out.println(stack.isEmpty());
}
```

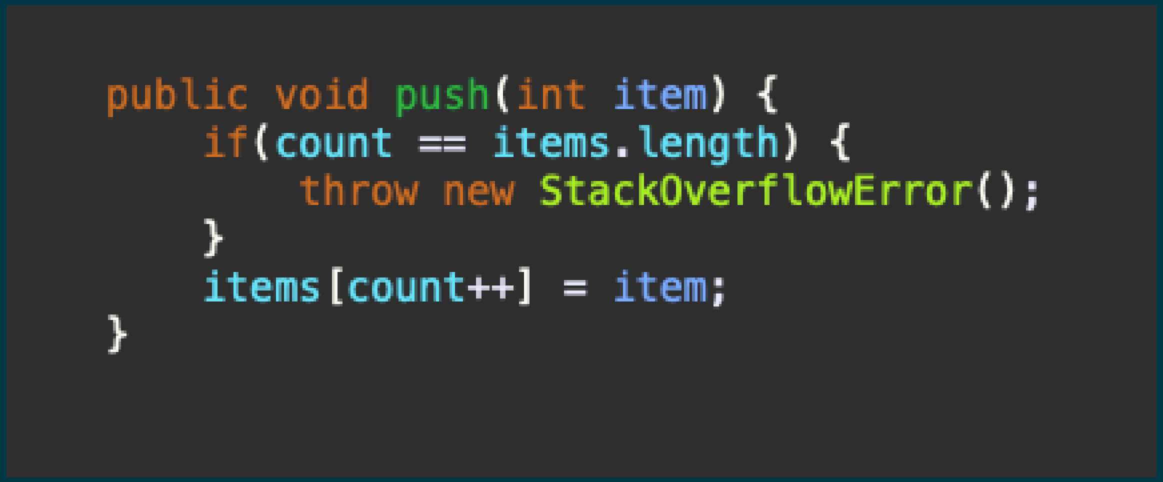


Create Stack



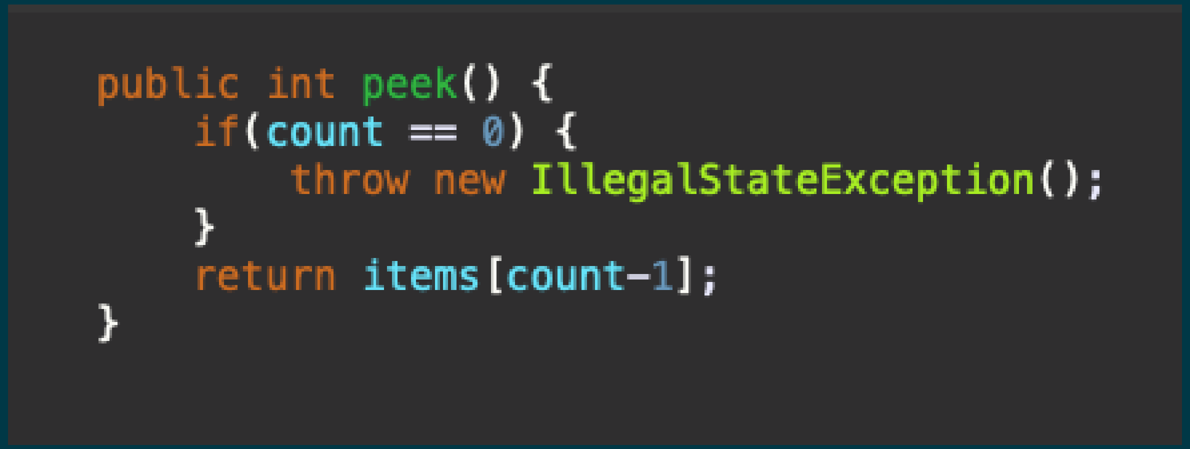
```java
public class Stack {

    private int[] items;
    private int count;

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

    public String toString() {
        var content = Arrays.copyOfRange(items, 0, count);
        return Arrays.toString(content);
    }
```

push



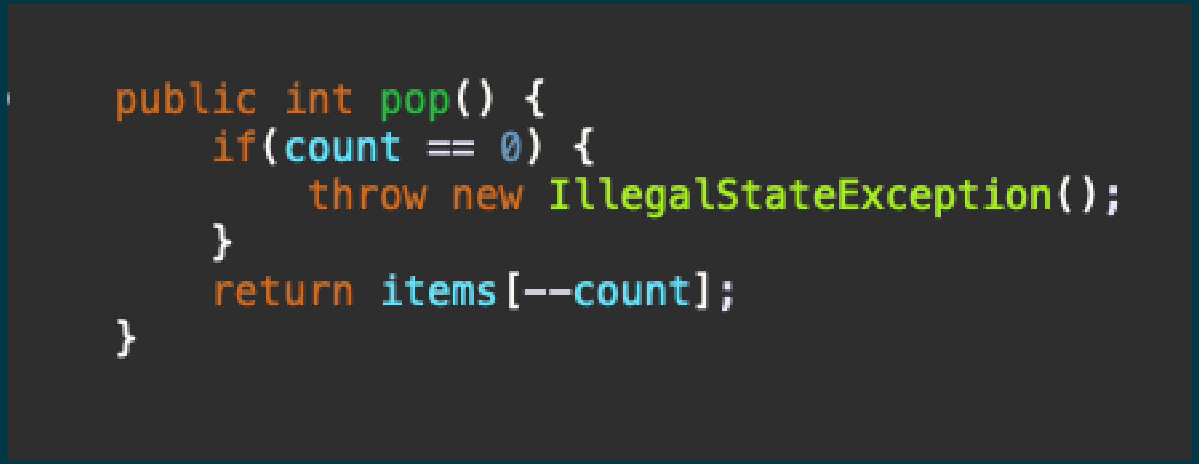
```
public void push(int item) {
    if(count == items.length) {
        throw new StackOverflowError();
    }
    items[count++] = item;
}
```

peek



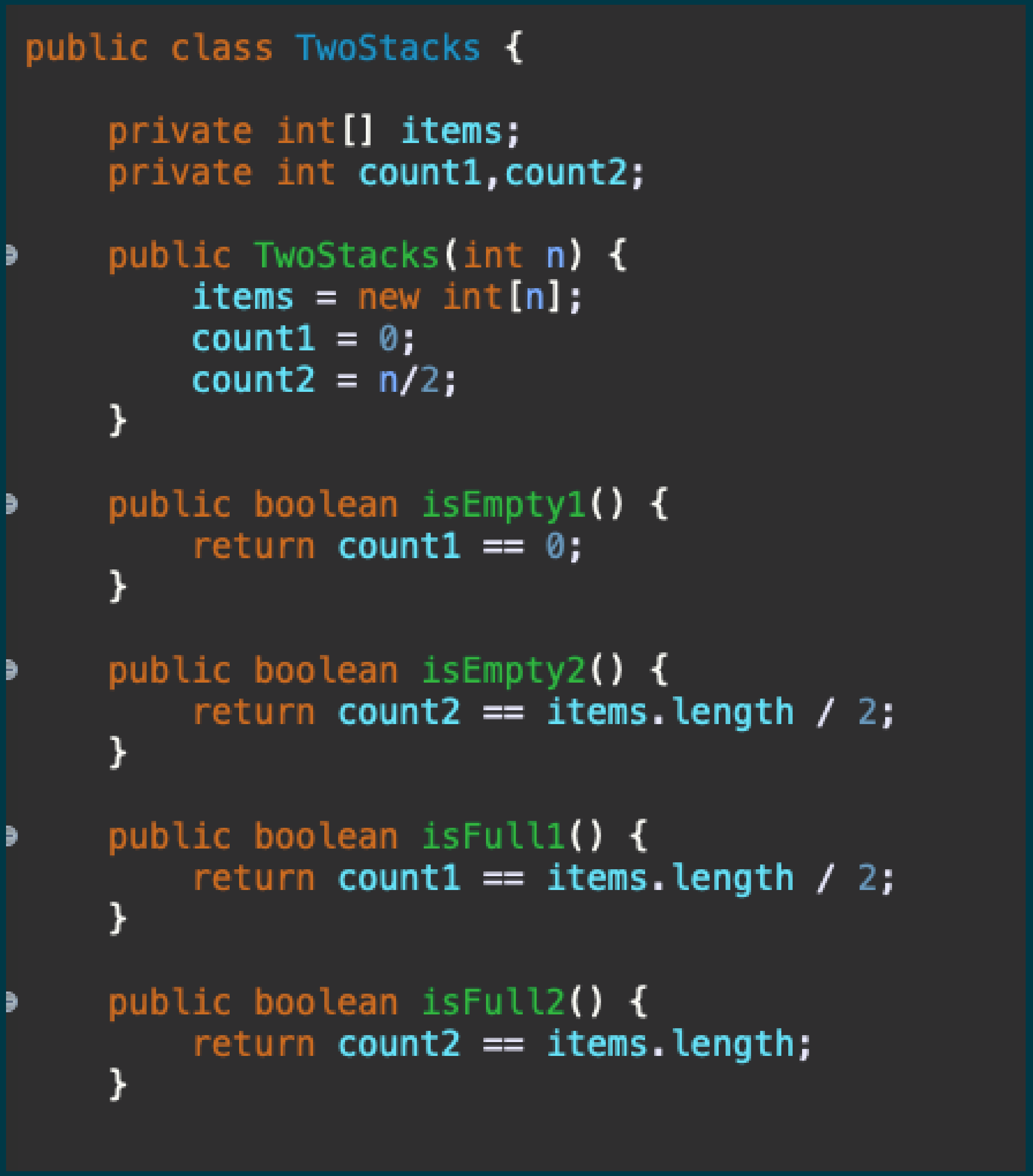
```
public int peek() {
    if(count == 0) {
        throw new IllegalStateException();
    }
    return items[count-1];
}
```

pop

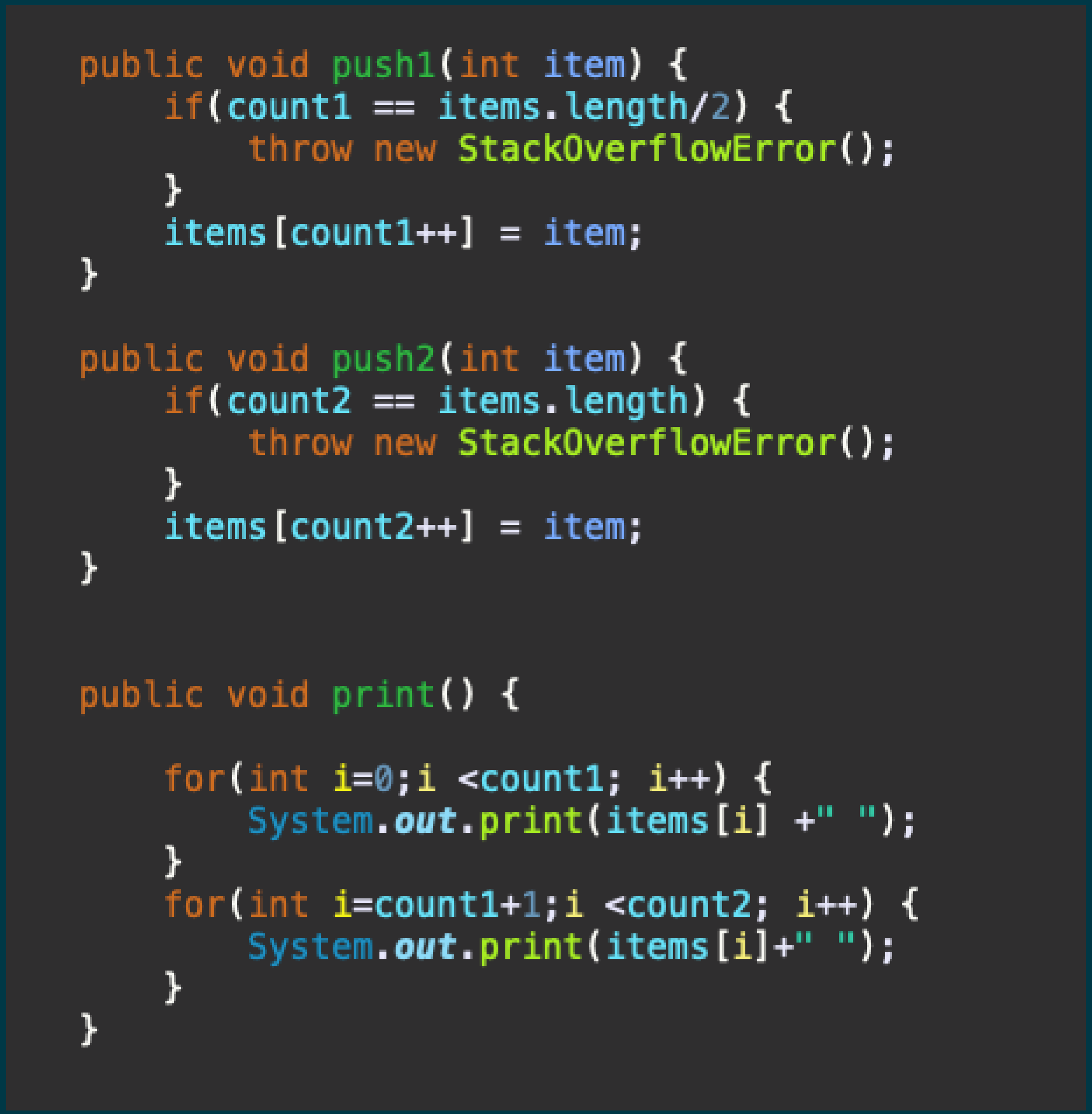


```
public int pop() {
    if(count == 0) {
        throw new IllegalStateException();
    }
    return items[--count];
}
```

Two Stacks


```java
public class TwoStacks {

    private int[] items;
    private int count1,count2;

    public TwoStacks(int n) {
        items = new int[n];
        count1 = 0;
        count2 = n/2;
    }

    public boolean isEmpty1() {
        return count1 == 0;
    }

    public boolean isEmpty2() {
        return count2 == items.length / 2;
    }

    public boolean isFull1() {
        return count1 == items.length / 2;
    }

    public boolean isFull2() {
        return count2 == items.length;
    }
```

push1, push2



```java
public void push1(int item) {
    if(count1 == items.length/2) {
        throw new StackOverflowError();
    }
    items[count1++] = item;
}

public void push2(int item) {
    if(count2 == items.length) {
        throw new StackOverflowError();
    }
    items[count2++] = item;
}


public void print() {

    for(int i=0;i <count1; i++) {
        System.out.print(items[i] +" ");
    }
    for(int i=count1+1;i <count2; i++) {
        System.out.print(items[i]+" ");
    }
}
```

Task 1
Leetcode 344 – Reverse String masalasini Stack(java.util) ni ishlatib yeching va natijani ko`rsating.

https://leetcode.com/problems/reverse-string/

Task 2
Leetcode 20 – Valid Parenthesis masalasini Stack(java.util) ni ishlatib yeching va natijani ko`rsating.

https://leetcode.com/problems/valid-parentheses/

Task 3
Darsda o`tilgan Stack ning elementlaridan eng kichik qiymatini topadigan funksiya yarating hamda uning Time Complexity sini aniqlang.

```
public int min()
```

Task 4
Darsda o`tilgan Stack ning elementlaridan eng katta qiymatini topadigan funksiya yarating hamda uning Time Complexity sini aniqlang.

```
public int max()
```

Task 5
Darsda o`tilgan TwoStack uchun pop1 va pop2 funksiyalarni yarating hamda uning Time Complexity sini aniqlang.

```
public int pop1()
public int pop2()
```