1. Binary Search

```java
import java.util.Scanner;

public class BinarySearch {
    private static int binarySearch(int[] arr, int k) {
        int left = 0, right = arr.length - 1;
        while (left <= right) {
            int mid = (int) (left + right) / 2;
            if (arr[mid] == k) {
                return mid;
            } else if (arr[mid] < k) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int length = scan.nextInt();
        int[] arr = new int[length];
        for (int i = 0; i < length; i++) {
            arr[i] = scan.nextInt();
        }
        int k = scan.nextInt();
        int result = binarySearch(arr, k);
        if (result != -1) {
            System.out.println("Element is present at index " + result);
        } else {
            System.out.println("Element is not present in array");
        }
        scan.close();
    }
}
```

```
10
10 20 30 40 55 60 78 95 102 115
95
Element is present at index 7
```

2. Equilibrium Points

```java
import java.util.Scanner;

public class EquillibriumPoints {
    public static int equilibriumPoint(int arr[]) {
        if (arr.length == 1)
            return 1;
        int[] prefixSum = new int[arr.length];
        prefixSum[0] = arr[0];
        for (int i = 1; i < arr.length; i++) {
            prefixSum[i] = prefixSum[i - 1] + arr[i];
        }
        for (int i = 0; i < arr.length; i++) {
            int before = (i > 0) ? prefixSum[i - 1] : 0;
            int after = (i < arr.length - 1) ?
                        prefixSum[arr.length - 1] - prefixSum[i] : 0;
            if (before == after) {
                return (i + 1);
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int length = scan.nextInt();
        int[] arr = new int[length];
        for (int i = 0; i < length; i++) {
            arr[i] = scan.nextInt();
        }
        System.out.println(equilibriumPoint(arr));
        scan.close();
    }
}
```

```
5
1 3 5 2 2
Result: 3
```

3. Kth Smallest Element

```java
import java.util.PriorityQueue;
import java.util.Scanner;

public class KthSmallestElement {
    public static int kthSmallest(int[] arr, int k) {
        PriorityQueue<Integer> queue = new PriorityQueue<>();
        for (int elt : arr) {
            queue.add(elt);
        }
        int result = -1;
        while (k > 0) {
            result = queue.poll();
            k -= 1;
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int length = scan.nextInt();
        int[] arr = new int[length];
        for (int i = 0; i < length; i++) {
            arr[i] = scan.nextInt();
        }
        int k = scan.nextInt();
        System.out.println(kthSmallest(arr, k));
        scan.close();
    }
}
```

```
10
1 6 9 8 5 3 2 4 7 10
6
Result: 6
```

4. Minimize the Heights II

```java
import java.util.Arrays;
import java.util.Scanner;

public class MinizedDifference {
    public static int getMinDiff(int[] arr, int k) {
        Arrays.sort(arr);
        int result = arr[arr.length - 1] - arr[0];
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] - k < 0) {
                continue;
            } else {
                int minH = Math.min(arr[0] + k, arr[i] - k);
                int maxH = Math.max(arr[i-1] + k, arr[arr.length-1] - k);
                result = Math.min(result, maxH - minH);
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int length = scan.nextInt();
        int k = scan.nextInt();
        int[] arr = new int[length];
        for (int i = 0; i < length; i++) {
            arr[i] = scan.nextInt();
        }
        System.out.println(getMinDiff(arr, k));
        scan.close();
    }
}
```

```
10 2
1 1 1 10 3 2 6 9 9 8
Result: 5
```

5. Next Greater Element

```java
import java.util.Scanner;
import java.util.Stack;

public class NextGreaterElement {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        int length;
        length = scan.nextInt();

        int[] array = new int[length];
        for (int i = 0; i < length; i++) {
            array[i] = scan.nextInt();
        }

        Stack<Integer> stack = new Stack<>();
        int[] answer = new int[length];

        for (int i = length - 1; i > -1; i--) {
            while (!stack.empty()) {
                if (stack.peek() > array[i]) {
                    answer[i] = stack.peek();
                    break;
                } else {
                    stack.pop();
                }
            }
            if (stack.empty()) {
                answer[i] = -1;
            }
            stack.push(array[i]);
        }

        for (int elt : answer) {
            System.out.print(elt + " ");
        }
    }
}
```

```
4
1 8 3 25
Result: 8 25 25 -1
```

6. Parentheses Checker

```
import java.util.*;
public class ParanthesesChecker {
    static boolean isParenthesisBalanced(String s) {
        Stack<Character> stack = new Stack<>();
        for (char letter : s.toCharArray()) {
            if (stack.empty()) {
                stack.push(letter);
                continue;
            }
            switch (letter) {
                case ')':
                    if (stack.peek() == '(') {
                        stack.pop();
                    } else {
                        stack.push(letter);
                    }
                    break;
                case ']':
                    if (stack.peek() == '[') {
                        stack.pop();
                    } else {
                        stack.push(letter);
                    }
                    break;
                case '}':
                    if (stack.peek() == '{') {
                        stack.pop();
                    } else {
                        stack.push(letter);
                    }
                    break;
                default:
                    stack.push(letter);
            }
        }
        return stack.empty();
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String input = scan.next();
        System.out.println(isParenthesisBalanced(input));
    }
}
```

```
aranthesesChecker }
{[](())}
true
PS D:\Professional\Programming\13 Nov 2024> cd
$?) { java ParanthesesChecker }
{}(()()[]
false
PS D:\Professional\Programming\13 Nov 2024>
```

7. Union of Two Arrays with Duplicate Elements

```java
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class UnionOfArrays {
    public static int findUnion(int a[], int b[]) {
        Set<Integer> set = new HashSet<>();
        for (int elt : a) {
            set.add(elt);
        }
        for (int elt : b) {
            set.add(elt);
        }
        return set.size();
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int length1 = scan.nextInt();
        int length2 = scan.nextInt();
        int[] a = new int[length1];
        int[] b = new int[length2];
        for (int i = 0; i < length1; i++) {
            a[i] = scan.nextInt();
        }
        for (int i = 0; i < length2; i++) {
            b[i] = scan.nextInt();
        }
        System.out.println(findUnion(a, b));
        scan.close();
    }
}
```

```
5
3
1 2 3 4 5
1 2 3
Union of two arrays: 1 2 3 4 5
Result: 5
```