1.

import java.util.\*;

            public class TopKFrequentNumbers {

            private static int[] nums;

            public static void main(String[] args) {

            int[] array1 = {3, 1, 4, 4, 5, 2, 6, 1};

            int k1 = 2;

            findTopKFrequentNumbers(array1, k1);

        int[] array2 = {7, 10, 11, 5, 2, 5, 5, 7, 11, 8, 9};

        int k2 = 4;

        findTopKFrequentNumbers(array2, k2);

    }

    public static void findTopKFrequentNumbers(int[] array, int k) {

        nums = array;

        Map<Integer, Integer> frequencyMap = new HashMap<>();

        for (int num : nums) {

            frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);

        }

        PriorityQueue<Integer> minHeap = new PriorityQueue<>((n1, n2) -> frequencyMap.get(n1) - frequencyMap.get(n2));

        for (int num : frequencyMap.keySet()) {

            minHeap.add(num);

            if (minHeap.size() > k) {

                minHeap.poll();

            }

        }

        List<Integer> result = new ArrayList<>();

        while (!minHeap.isEmpty()) {

            result.add(minHeap.poll());

        }

        Collections.reverse(result);

        for (int num : result) {

            System.out.print(num + " ");

        }

        System.out.println();

    }

}

**OUTPUT:**

1 4

5 7 11 8

2.

 public class ShareTrader {

    private static int maxProfit;

    public static void main(String[] args) {

        int[] prices1 = {10, 22, 5, 75, 65, 80};

        findMaxProfit(prices1);

        int[] prices2 = {2, 30, 15, 10, 8, 25, 80};

        findMaxProfit(prices2);

    }

    public static void findMaxProfit(int[] prices) {

        int n = prices.length;

        if (n < 2) {

            maxProfit = 0;

            System.out.println("Maximum Profit: " + maxProfit);

            return;

        }

        int[] leftProfit = new int[n];

        int[] rightProfit = new int[n];

        int minPrice = prices[0];

        for (int i = 1; i < n; i++) {

            minPrice = Math.min(minPrice, prices[i]);

            leftProfit[i] = Math.max(leftProfit[i - 1], prices[i] - minPrice);

        }

        int maxPrice = prices[n - 1];

        for (int i = n - 2; i >= 0; i--) {

            maxPrice = Math.max(maxPrice, prices[i]);

            rightProfit[i] = Math.max(rightProfit[i + 1], maxPrice - prices[i]);

        }

        int maxProfit = 0;

        for (int i = 0; i < n; i++) {

            maxProfit = Math.max(maxProfit, leftProfit[i] + rightProfit[i]);

        }

        System.out.println("Maximum Profit: " + maxProfit);

    }

}

**OUTPUT:**

Maximum Profit: 87

Maximum Profit: 100