Day 2: Find the Largest and Smallest Elements in an Array

Abhinav Yadav

"Code is like humor. When you have to explain it, it's bad."
— Cory House

---

1 Introduction

Finding the largest and smallest elements in an array is a fundamental problem in computer science and programming. It involves scanning the array and determining the maximum and minimum values efficiently. This document provides an implementation where separate functions are used to find the largest and smallest elements in the array.

2 Problem Statement

Problem: Write a program to find the largest and smallest numbers in an array of size `n` using separate functions. **Hint:** Utilize modular programming to enhance code readability and reusability.

3 Algorithm

3.1 Find Smallest Element

1. Initialize a variable `min` with the first element of the array.
2. Traverse the array:
    - If the current element is smaller than `min`, update `min`.
3. Return `min`.

3.2 Find Largest Element

1. Initialize a variable `max` with the first element of the array.
2. Traverse the array:
    - If the current element is larger than `max`, update `max`.
3. Return `max`.

4 Code

```
import java.util.Scanner;

public class SmallestLargest {

    // Function to find the smallest element in an array
    public static int findSmallest(int[] arr, int n) {
        int min = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] < min) {
                min = arr[i];
            }
        }
        return min;
    }

    // Function to find the largest element in an array
    public static int findLargest(int[] arr, int n) {
        int max = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }
        return max;
    }

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

        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        int smallest = findSmallest(arr, n);
        int largest = findLargest(arr, n);

        System.out.println("Smallest Element: " + smallest);
        System.out.println("Largest Element: " + largest);

        scanner.close();
    }
```

}

5 Complexity Analysis

5.1 Find Smallest Element

- Time Complexity: $O(n)$ (Single iteration through the array).
- Space Complexity: $O(1)$ (Constant space for the `min` variable).

5.2 Find Largest Element

- Time Complexity: $O(n)$ (Single iteration through the array).
- Space Complexity: $O(1)$ (Constant space for the `max` variable).

6 Comparison of One-Pass vs Separate Functions

Criteria	**One-Pass Algorithm**	**Separate Functions**
Time Complexity	$O(n)$	$O(2n)$
Space Complexity	$O(1)$	$O(1)$
Modularity	Moderate	High
Code Readability	Less modular	Highly modular

7 Conclusion

Using separate functions to find the smallest and largest elements increases code modularity and reusability, making the program easier to read and maintain. Although it introduces a slight increase in execution time due to two separate loops, this trade-off is acceptable for small to medium-sized datasets.

8 Output

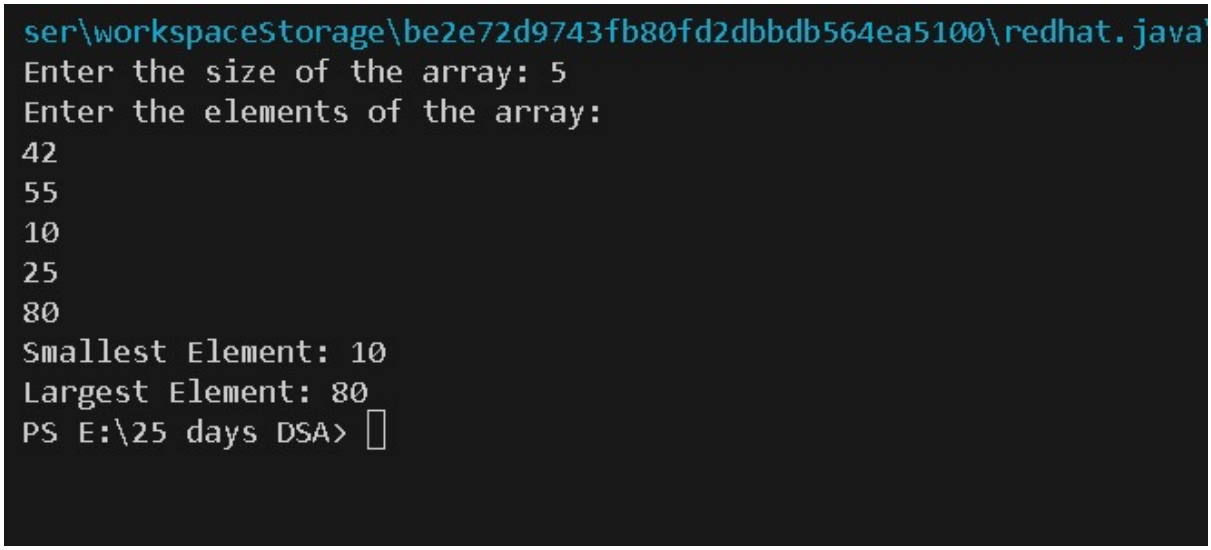

Figure 1: Output