**Module 2 - Data Structures and Algorithms**

**Exercise 4: Employee Management System**

**1). Array Representation in Memory**

* Arrays are stored in **contiguous memory locations**.
* Direct access via index is **O(1)**.
* Arrays are fixed in size and type.

**2,3). Setup and Implementation:**

**Code:**

class Employee {

    int employeeId;

    String name;

    String position;

    double salary;

    Employee(int employeeId, String name, String position, double salary) {

        this.employeeId = employeeId;

        this.name = name;

        this.position = position;

        this.salary = salary;

    }

    void display() {

        System.out.println(employeeId + " " + name + " " + position + " " + salary);

    }

}

public class EmployeeManagement {

    static Employee[] employees = new Employee[100];

    static int count = 0;

    public static void addEmployee(int id, String name, String position, double salary) {

        employees[count++] = new Employee(id, name, position, salary);

    }

    public static void searchEmployee(int id) {

        for (int i = 0; i < count; i++) {

            if (employees[i].employeeId == id) {

                employees[i].display();

                return;

            }

        }

        System.out.println("Employee not found");

    }

    public static void deleteEmployee(int id) {

        for (int i = 0; i < count; i++) {

            if (employees[i].employeeId == id) {

                for (int j = i; j < count - 1; j++) {

                    employees[j] = employees[j + 1];

                }

                count--;

                System.out.println("Employee deleted");

                return;

            }

        }

        System.out.println("Employee not found");

    }

    public static void traverseEmployees() {

        for (int i = 0; i < count; i++) {

            employees[i].display();

        }

    }

    public static void main(String[] args) {

        addEmployee(1, "Alice", "Manager", 60000);

        addEmployee(2, "Bob", "Developer", 50000);

        addEmployee(3, "Charlie", "Tester", 40000);

        System.out.println("All Employees:");

        traverseEmployees();

        System.out.println("Search Employee ID 2:");

        searchEmployee(2);

        System.out.println("Delete Employee ID 2:");

        deleteEmployee(2);

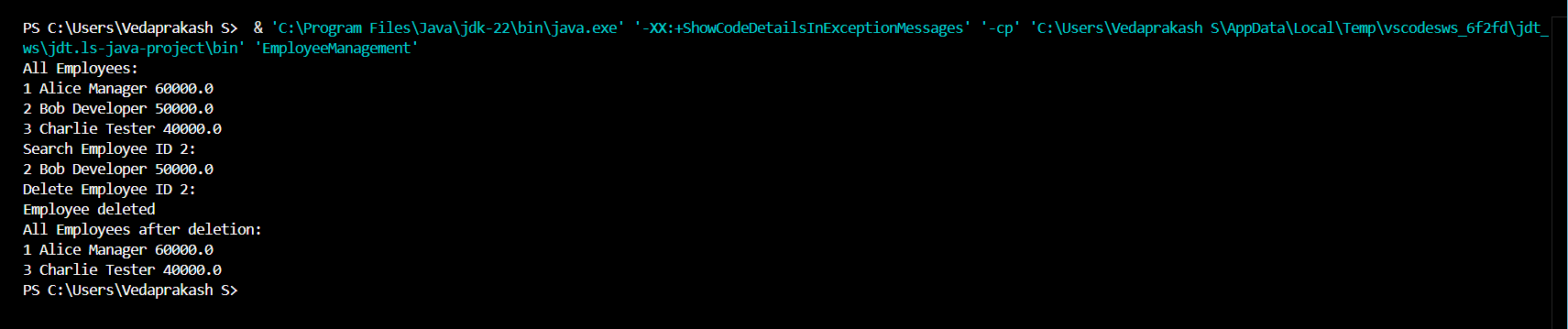
        System.out.println("All Employees after deletion:");

        traverseEmployees();

    }

}

**Output:**



**4). Analysis**

* **Add**: O(1) if space is available.
* **Search**: O(n) linear search.
* **Traverse**: O(n)
* **Delete**: O(n) (shift elements)

**Limitations**

* Fixed size – can’t grow dynamically.
* Insertion/deletion in middle is expensive.
* Not ideal for frequent inserts/removals.