**Employee Management System**

**Step 1: Understand Array Representation**

**Arrays:**

* An array is a continuous block of memory where each element is stored sequentially.
* Each element is accessed by an index. The indexing starts from 0.
* Arrays are fixed in size.

**Advantages of arrays:**

* Fast access using index: O(1) time to access any element.
* Simple and easy to use when the number of elements is known in advance.

**Step 2: Setup**

class Employee {

int employeeId;

String name;

String position;

double salary;

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

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

public void display() {

System.out.println("ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: " + salary);

}

}

**Step 3: Implementation**

import java.util.Scanner;

public class EmployeeManagement {

    static final int MAX\_EMPLOYEES = 100;

    static Employee[] employees = new Employee[MAX\_EMPLOYEES];

    static int count = 0;

    public static void addEmployee(Employee employee) {

        if (count < MAX\_EMPLOYEES) {

            employees[count++] = employee;

        } else {

            System.out.println("Employees cannot be added further.");

        }

    }

    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 is not found.");

    }

    public static void displayAllEmployees() {

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

            employees[i].display();

        }

    }

    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];

                }

                employees[--count] = null;

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

                return;

            }

        }

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

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        addEmployee(new Employee(1, "Elsa", "Developer", 50000));

        addEmployee(new Employee(2, "Anna", "Developer", 40000));

        addEmployee(new Employee(3, "Cindrella", "Tester", 35000));

        displayAllEmployees();

        searchEmployee(2);

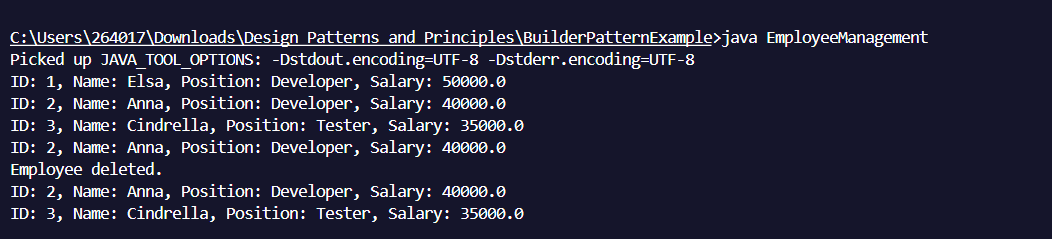
        deleteEmployee(1);

        displayAllEmployees();

    }

}

**Output:**

****

**Step 4: Analysis**

| **Operation** | **Time** | **Complexity Explanation** |
| --- | --- | --- |
| **Add** | **O(1)** | **Insertion at the end of an array.** |
| **Search** | **O(n)** | **Linear search, need to check all elements.** |
| **Traverse** | **O(n)** | **Displays all the elements one by one.** |
| **Delete** | **O(n)** | **Display remaining elements left.** |

**Limitations of Arrays**

* Fixed size: You must know the maximum number of employees in advance.
* No dynamic resizing: To increase the size you must create a new array and copy elements.
* Slow deletions: Need to shift elements after removing one.

**When to Use Arrays**

* When the number of items is known and fixed.
* When fast access by index is needed and to increase memory efficiency.