**Data Structures Algorithms**

**EXERCISE 4: Employee Management System**

**Source Code**

// Main.java

import java.util.Scanner;

// Step 2: Employee class

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: Management System using Array

class EmployeeManagementSystem {

private Employee[] employees;

private int count;

public EmployeeManagementSystem(int capacity) {

employees = new Employee[capacity];

count = 0;

}

// Add employee

public void addEmployee(Employee emp) {

if (count >= employees.length) {

System.out.println("Array full. Cannot add more employees.");

return;

}

employees[count++] = emp;

System.out.println("Employee added: " + emp.name);

}

// Search by ID

public void searchEmployee(int id) {

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

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

System.out.println("Employee Found:");

employees[i].display();

return;

}

}

System.out.println("Employee with ID " + id + " not found.");

}

// Traverse all

public void displayAllEmployees() {

if (count == 0) {

System.out.println("No employees to display.");

return;

}

System.out.println("Employee List:");

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

employees[i].display();

}

}

// Delete by ID

public 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 with ID " + id + " deleted.");

return;

}

}

System.out.println("Employee with ID " + id + " not found.");

}

}

// Step 4: Test and Analysis

public class Main {

public static void main(String[] args) {

EmployeeManagementSystem system = new EmployeeManagementSystem(10);

// Add Employees

system.addEmployee(new Employee(101, "Hima", "Developer", 90000));

system.addEmployee(new Employee(102, "Rishi", "Manager", 80000));

system.addEmployee(new Employee(103, "Prani", "Analyst", 50000));

System.out.println("\n-- All Employees --");

system.displayAllEmployees();

System.out.println("\n-- Search for Employee with ID 102 --");

system.searchEmployee(102);

System.out.println("\n-- Delete Employee with ID 101 --");

system.deleteEmployee(101);

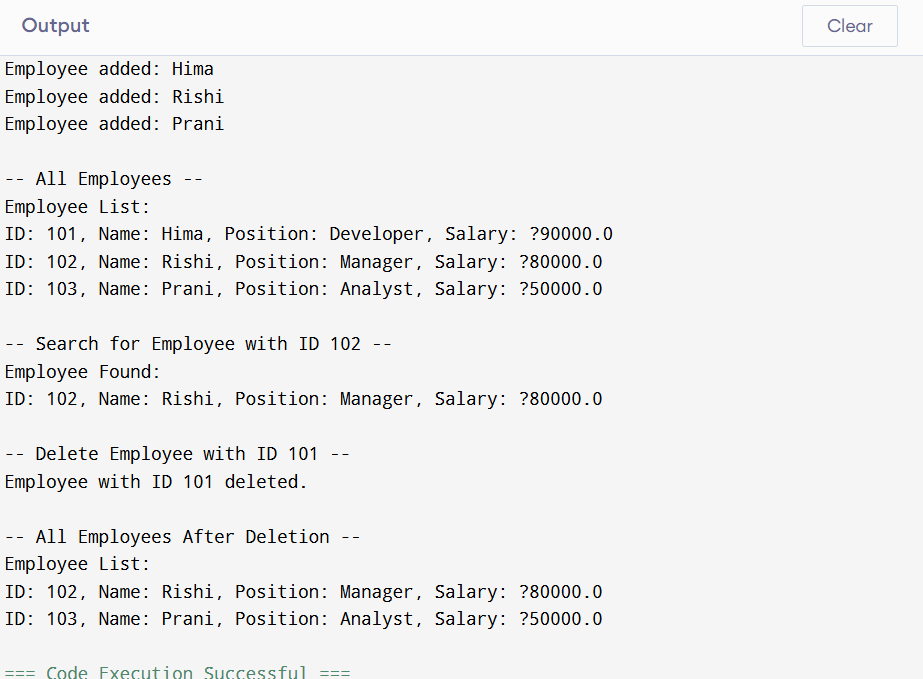
System.out.println("\n-- All Employees After Deletion --");

system.displayAllEmployees();

}

}

**OUTPUT**

****

🧠 **Array Representation & AnalysisMemory:** Arrays are stored in contiguous memory, which allows constant-time access by index.**Advantages:** Fast traversal, low overhead, and simplicity.**Time Complexities:Add:** O(1) if space available (insert at count).**Search:** O(n) linear scan.**Traverse:** O(n).**Delete:** O(n) (due to shifting).

**Limitations:**1.Fixed size (not dynamically resizable).2.Insertion and deletion (except at the end) can be expensive.3.Inefficient if sparsely populated—memory is wasted.