**Exercise 4: Employee Management System**

**EmployeeManagementSystem.java**

**package** mypackage;

**import** java.util.Scanner;

**class** Employee {

**private** **int** employeeId;

**private** String name;

**private** String position;

**private** **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** **int** getEmployeeId() { **return** employeeId; }

**public** String getName() { **return** name; }

**public** String getPosition() { **return** position; }

**public** **double** getSalary() { **return** salary; }

@Override

**public** String toString() {

**return** String.*format*("Employee ID: %d, Name: %s, Position: %s, Salary: ₹%,.2f",

employeeId, name, position, salary);

}

}

**public** **class** EmployeeManagementSystem {

**private** **static** **final** **int** ***MAX\_EMPLOYEES*** = 100;

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

**private** **static** **int** *count* = 0;

**private** **static** Scanner *sc* = **new** Scanner(System.***in***);

**public** **static** **void** addEmployee() {

**if** (*count* >= ***MAX\_EMPLOYEES***) {

System.***out***.println("Employee limit reached!");

**return**;

}

System.***out***.print("Enter Employee ID: ");

**int** id = *sc*.nextInt();

*sc*.nextLine();

System.***out***.print("Enter Name: ");

String name = *sc*.nextLine();

System.***out***.print("Enter Position: ");

String position = *sc*.nextLine();

System.***out***.print("Enter Salary: ");

**double** salary = *sc*.nextDouble();

*sc*.nextLine();

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

System.***out***.println("Employee added successfully!");

}

**public** **static** **void** searchEmployee() {

System.***out***.print("Enter Employee ID to search: ");

**int** id = *sc*.nextInt();

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

**if** (*employees*[i].getEmployeeId() == id) {

System.***out***.println("Employee found: " + *employees*[i]);

**return**;

}

}

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

}

**public** **static** **void** traverseEmployees() {

**if** (*count* == 0) {

System.***out***.println("No employees in the system.");

**return**;

}

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

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

System.***out***.println(*employees*[i]);

}

}

**public** **static** **void** deleteEmployee() {

System.***out***.print("Enter Employee ID to delete: ");

**int** id = *sc*.nextInt();

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

**if** (*employees*[i].getEmployeeId() == id) {

// Shift all elements after i to the left

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

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

}

*employees*[--*count*] = **null**;

System.***out***.println("Employee deleted successfully!");

**return**;

}

}

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

}

**public** **static** **void** main(String[] args) {

**int** choice;

**do** {

System.***out***.println("\nEmployee Management System");

System.***out***.println("1. Add Employee");

System.***out***.println("2. Search Employee");

System.***out***.println("3. Display All Employees");

System.***out***.println("4. Delete Employee");

System.***out***.println("5. Exit");

System.***out***.print("Enter your choice: ");

choice = *sc*.nextInt();

**switch** (choice) {

**case** 1: *addEmployee*(); **break**;

**case** 2: *searchEmployee*(); **break**;

**case** 3: *traverseEmployees*(); **break**;

**case** 4: *deleteEmployee*(); **break**;

**case** 5: System.***out***.println("Exiting..."); **break**;

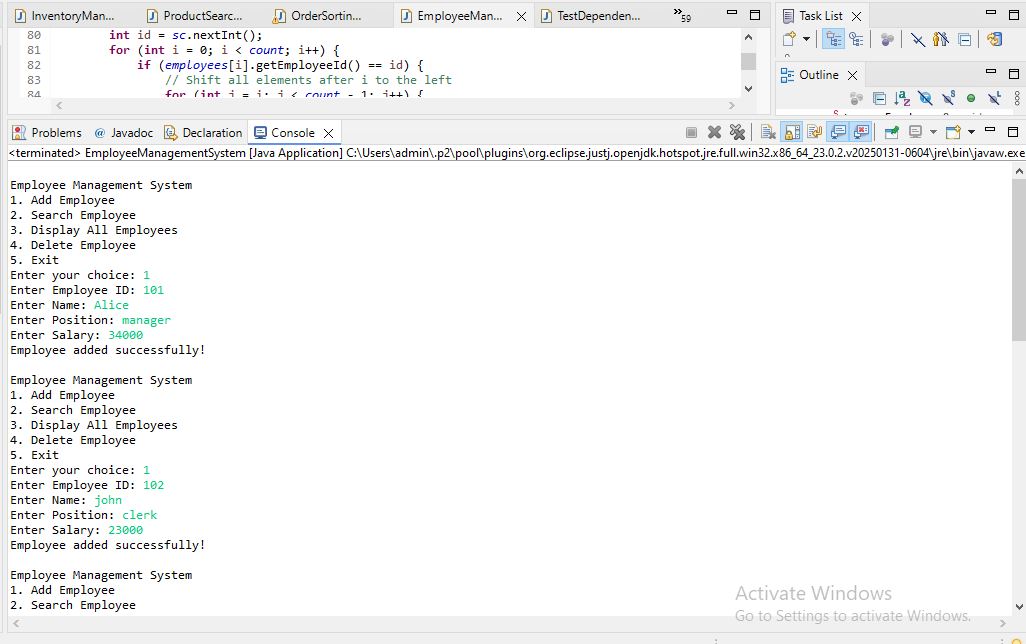
**default**: System.***out***.println("Invalid choice!");

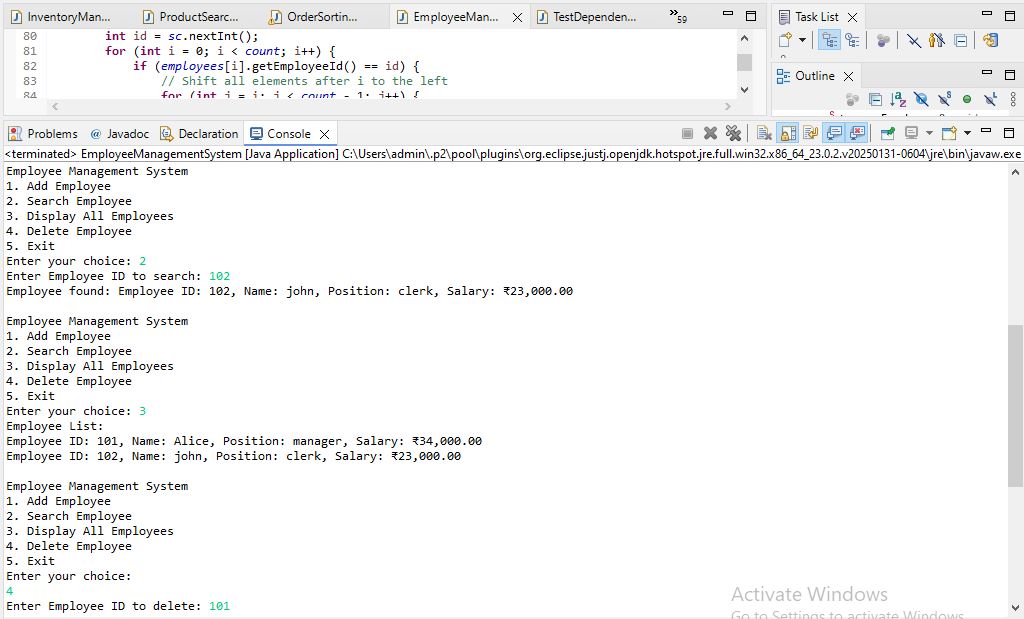
}

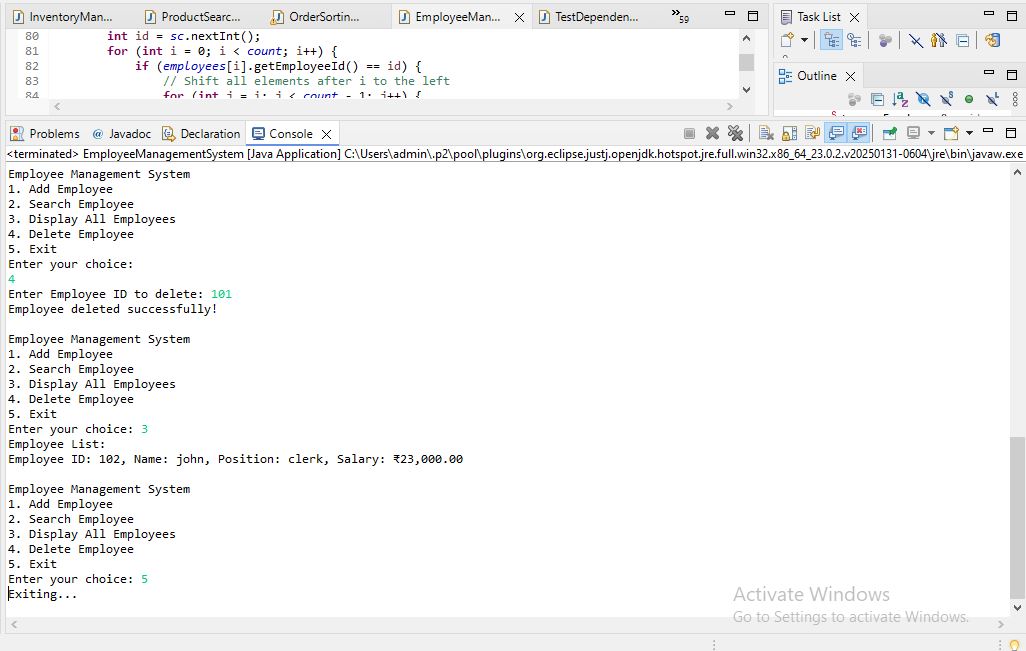
} **while** (choice != 5);

}

}







**1. Understand Array Representation**

**➔ How arrays are represented in memory:**

* Arrays store elements in **contiguous memory locations**.
* Each element can be directly accessed using its index.
* Memory address for element i = base address + (i \* size\_of\_element)
* Efficient for random access.

**➔ Advantages of arrays:**

* Fast access (O(1)) using index.
* Simple structure, easy to use.
* Efficient use of memory for fixed-size data.

**2. Setup**

* We create a class Employee with:
  + employeeId (int)
  + name (String)
  + position (String)
  + salary (double)
* An array Employee[] is used to store employee records.

**3. Implementation**

* Use a fixed-size array Employee[] employees = new Employee[100].
* Operations implemented:
  + **Add Employee** → adds a new record.
  + **Search Employee** → searches by employeeId.
  + **Traverse Employees** → lists all employees.
  + **Delete Employee** → removes an employee by shifting elements.

**4. Analysis**

**➔ Time Complexity**

| **Operation** | **Time Complexity** | **Explanation** |
| --- | --- | --- |
| **Add** | O(1) | Inserting at end (using count index) |
| **Search** | O(n) | Linear search |
| **Traverse** | O(n) | Loop through all employees |
| **Delete** | O(n) | Search + shift elements |

**➔ Limitations of arrays:**

* Fixed size — difficult to resize dynamically.
* Insertion and deletion (except at end) require shifting elements (O(n)).
* Waste of memory if many empty slots.
* No built-in dynamic growth like ArrayList or LinkedList.

**➔ When to use arrays:**

* When maximum size is known beforehand.
* When frequent random access (using index) is needed.
* For small to moderate datasets where dynamic resizing is not critical.