**Exercise 4: Employee Management System**

**Scenario:**

You are developing an employee management system for a company. Efficiently managing employee records is crucial.

**Steps:**

1. **Understand Array Representation:**
   * Explain how arrays are represented in memory and their advantages.

Array Representation in Memory:

•Arrays in Java are objects that store multiple elements of the same type in a contiguous block of memory.

•When you create an array, Java allocates a continuous block of memory to hold the array elements.

•Each element in the array is accessed by its index, which starts from 0.

•The memory address of the first element (index 0) is known, and you can access other elements by adding the index offset to this base address.

Advantages of Arrays:

1.Fixed Size: Once declared, the size of the array cannot be changed, which makes it simple and efficient.

2.Efficient Access: Accessing an element by index is very fast (O(1) time complexity).

3.Contiguous Memory: Arrays store elements in contiguous memory locations, which can be cache-friendly and enhance performance.

1. **Analysis:**
   * Analyze the time complexity of each operation (add, search, traverse, delete).

Time Complexity Analysis:

•Add Employee: O(1) – Adding an employee to the end of the array is a constant-time operation.

•Search Employee: O(n) – In the worst case, you may need to check all employees.

•Traverse Employees: O(n) – You need to visit each element once.

•Delete Employee: O(n) – In the worst case, you need to shift all elements after the deleted employee.

* + Discuss the limitations of arrays and when to use them.

Limitations of Arrays:

1.Fixed Size: The size of the array is fixed at the time of creation and cannot be changed.

2.Inefficient Insertions and Deletions: Inserting or deleting elements in the middle of the array requires shifting elements, which is time-consuming (O(n) complexity).

3.Memory Usage: If the array size is much larger than the number of elements, it can lead to wasted memory.

When to Use Arrays:

•When you know the exact number of elements in advance.

•When you need fast access to elements using indices.

•When memory needs to be contiguous and predictable.