

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING
Department of Computer Science and Engineering

Academic Year	2025-2026	Estimated Time	Experiment No. 4 – 02 Hours
Course & Semester	S.E. CSE	Subject Name	Object Oriented Programming with Java Lab
Module No.	04	Chapter Title	Arrays and Vector
Experiment Type	Software Performance	Subject Code	25PCC12CS07

Name of Student		Roll No.	
Date of Performance.:		Date of Submission.:	
CO Mapping	CO1. Demonstrate Proficiency in Fundamentals of Java. CO2. Apply Object-Oriented Programming Principles to given problem.		

Objective of Experiment:

To explore and demonstrate the concepts of Arrays, Array Lists and Vectors in Java.

Pre-Requisite: Any programming language like C, C++

Tools: Java IDE

Theory:

A. Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. To declare an array, define the variable type with **square brackets**.

- `String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};`
- `int[] myNum = {10, 20, 30, 40};`

You can access an array element by referring to the index number.

- `System.out.println(cars[0]);` // Outputs Volvo

To change the value of a specific element, refer to the index number.

- `cars[0] = "Opel";`

To find out how many elements an array has, use the length property.

- `System.out.println(cars.length);` // Outputs 4

In Java, a matrix is typically implemented using a 2D array. The following is the syntax to declare and initialize a 2D array in Java.

- `int[][] matrix = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };`

Simple program to demonstrate the concept of Arrays in java

```

public class StudentScores {
    public static void main(String[] args) {
        // Declare and initialize an array of scores
        int[] scores = {85, 92, 88, 76, 95, 90};

        // Calculate the total score and average score
        int total = 0;
        for (int score : scores) {
            total += score;
        }
        double average = total / (double) scores.length;

        // Output the scores and the average
        System.out.println("Scores of the students:");
        for (int score : scores) {
            System.out.println(score);
        }
        System.out.printf("Average score: %.2f\n", average);
    }
}

```

OUTPUT:

```

Scores of the students:
85
92
88
76
95
90
Average score: 87.67

```

B. Array List

In Java, we use the ArrayList class to implement the functionality of resizable-arrays. It implements the List interface of the collections framework.



In Java, we need to declare the size of an array before we can use it. Once the size of an array is declared, it's hard to change it. To handle this issue, we can use the ArrayList class. It allows us to create resizable arrays. Unlike arrays, arraylists can automatically adjust their capacity when we add or remove elements from them. Hence, arraylists are also known as dynamic arrays.

Before using ArrayList, we need to import the java.util.ArrayList package first. Here is how we can create arraylists in Java.

- `ArrayList<Type> arrayList= new ArrayList<>();`

Here, Type indicates the type of an arraylist. For example, Integer or String.

Basic Operations on ArrayList

The ArrayList class provides various methods to perform different operations on arraylists. Some commonly used operations are:

1. Add elements: To add a single element to the arraylist, we use the `add()` method of the ArrayList class.
2. Access elements: To access an element from the arraylist, we use the `get()` method of the ArrayList class.
3. Change elements: To change elements of the arraylist, we use the `set()` method of the ArrayList class.
4. Remove elements: To remove an element from the arraylist, we can use the `remove()` method of the ArrayList class.

1. Adding elements to an ArrayList

```
import java.util.ArrayList;

class Languages{
    public static void main(String[] args){
        // create ArrayList
        ArrayList<String> languages = new ArrayList<>();

        // add() method without the index parameter
        languages.add("Java");
        languages.add("C");
        languages.add("Python");
        System.out.println("ArrayList: " + languages);
    }
}
```

OUTPUT:

```
ArrayList: [Java, C, Python]
```

2. Access ArrayList elements

```
String s0 = languages.get(0);
String s1 = languages.get(1);
String s2 = languages.get(2);
System.out.println("Element at index 0: " + s0);
System.out.println("Element at index 1: " + s1);
System.out.println("Element at index 2: " + s2);
```

OUTPUT:

```
Element at index 0: Java
Element at index 1: C
Element at index 2: Python
```

3. Change ArrayList elements

```
languages.set(1, "C++");
languages.set(2, "JavaScript");
System.out.println("Modified ArrayList: " + languages);
```

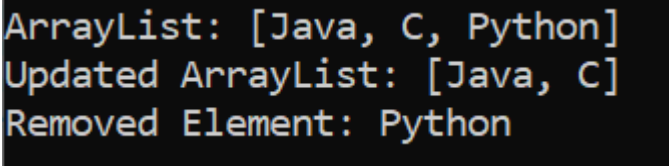
OUTPUT:

```
ArrayList: [Java, C, Python]
Modified ArrayList: [Java, C++, JavaScript]
```

4. Remove ArrayList Elements

```
String str = languages.remove(2);  
System.out.println("Updated ArrayList: " + languages);  
System.out.println("Removed Element: " + str);
```

OUTPUT:



```
ArrayList: [Java, C, Python]  
Updated ArrayList: [Java, C]  
Removed Element: Python
```

C. Vectors

The Vector class implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created. The Vector class is an implementation of the List interface that allows us to create resizable-arrays similar to the ArrayList class.

- Syntax to create vectors in Java: `Vector<Type> vector = new Vector<>();` Type indicates the type of a linked list. For example: Integer, String.

ArrayList is not synchronized whereas Vector is synchronized. So, ArrayList is faster than Vector. ArrayList prefers the iterator interface to traverse the components whereas Vector prefers Enumeration or Iterator interface to traverse the elements.

Some common methods when working with vectors:

1. Add Elements to Vector.
 - o `add(element)` - adds an element to vectors
 - o `add(index, element)` - adds an element to the specified position
 - o `addAll(vector)` - adds all elements of a vector to another vector

```
import java.util.Vector;

class V1 {
    public static void main(String[] args) {
        Vector<String> birds = new Vector<>();

        // Using the add() method
        birds.add("Sparrow");
        birds.add("Eagle");

        // Using index number
        birds.add(2, "Owl");
        System.out.println("Vector: " + birds);

        // Using addAll()
        Vector<String> animals = new Vector<>();
        animals.add("Lion");

        animals.addAll(birds);
        System.out.println("New Vector: " + animals);
    }
}
```

OUTPUT:

```
Vector: [Sparrow, Eagle, Owl]
New Vector: [Lion, Sparrow, Eagle, Owl]
```

2. Access vector Elements

- o `get(index)` - returns an element specified by the index
- o `iterator()` - returns an iterator object to sequentially access vector elements

```

import java.util.Iterator;
import java.util.Vector;

class V2{
    public static void main(String[] args) {
        Vector<String> animals = new Vector<>();
        animals.add("Dog");
        animals.add("Horse");
        animals.add("Cat");

        // Using get()
        String element = animals.get(2);
        System.out.println("Element at index 2: " + element);

        // Using iterator()
        Iterator<String> iterate = animals.iterator();
        System.out.print("Vector: ");
        while(iterate.hasNext()) {
            System.out.print(iterate.next());
            System.out.print(", ");
        }
    }
}

```

OUTPUT:

```

Element at index 2: Cat
Vector: Dog, Horse, Cat,

```

3. Remove Vector Elements

- o remove(index) - removes an element from specified position
- o removeAll() - removes all the elements
- o clear() - removes all elements. It is more efficient than removeAll()

```

import java.util.Vector;

class V3{
    public static void main(String[] args) {
        Vector<String> fruits = new Vector<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");

        System.out.println("Initial Vector: " + fruits);

        // Using remove()
        String element = fruits.remove(1);
        System.out.println("Removed Element: " + element); // Banana
        System.out.println("New Vector: " + fruits);        // [Apple, Cherry]

        // Using clear()
        fruits.clear();
        System.out.println("Vector after clear(): " + fruits); // []
    }
}

```

OUTPUT:

```
Initial Vector: [Apple, Banana, Cherry]
Removed Element: Banana
New Vector: [Apple, Cherry]
Vector after clear(): []
```

Problem Statement:**Contact Management App:**

Design and implement **Java application** to demonstrate the use of **Array, ArrayList, and Vector** with real-life scenarios. The program should provide the following functionalities:

1. **Array (Product Database Management):**
 - Store Salary of 5 products using an **array**.
 - Find the **highest, lowest, and average price**.
2. **ArrayList (Restaurant Database Management):**
 - Use an **ArrayList** to maintain a list of food items name in a store.
 - Allow the user to **add, remove, change, and display** food items dynamically.
3. **Vector (Mobile Store Management System):**
 - Use a **Vector** to store Mobile details (Model_name, cost, color).
 - Allow the user to **add new mobile, search by model_name, and display all mobiles information**.

Post Lab Question:

Add all programs implemented during practical session.

On time Completion and Submission (2)	Knowledge of the topic (4)	Implementation and Output (4)	Total (10)

References: Study Materials https://www.w3schools.com/java/ https://www.geeksforgeeks.org/java/ https://www.codecademy.com/learn/learn-java https://www.programiz.com/java-programming/arraylist	Video Channels: https://www.youtube.com/user/programmingwithmosh https://www.youtube.com/c/TheNetNinja https://www.youtube.com/c/Freecodecamp https://www.youtube.com/user/Simplilearn
Study Materials used for Demo <Add links here>	

Note:-students are expected to paste screenshot of the program with output