# LAB # 02

# ArrayList and Vector in JAVA

**OBJECTIVE:** To implement ArrayList and Vector.

## Lab Tasks

1. Write a program that initializes Vector with 10 integers in it. Display all the integers and sum of these integers.

```
import java.util.Vector;

public class VectorSum {
    public static void main(String[] args) {
        Vector<Integer> vector = new Vector<>();

        // Initializing Vector with 10 integers
        for (int i = 1; i <= 10; i++) {
            vector.add(i);
        }

        // Displaying all integers
        System.out.println("Vector elements: " + vector);

        // Calculating the sum of all integers
        int sum = 0;
        for (int num : vector) {
            sum += num;
        }
        System.out.println("Sum of integers: " + sum);
    }
}
```

### Output:

```
Vector elements: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of integers: 55
```

2. Create a ArrayList of string. Write a menu driven program which:
    a. Displays all the elements
    b. Displays the largest String

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class StringListMenu {
    public static void main(String[] args) {
        ArrayList<String> stringList = new ArrayList<>();
        stringList.add("Hello");
        stringList.add("World");
        stringList.add("Java");
        stringList.add("Programming");

        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Display all elements");
            System.out.println("2. Display the largest string");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.println("Elements in ArrayList: " + stringList);
                    break;
                case 2:
                    String largest = Collections.max(stringList, (s1, s2) -> s1.length() - s2.length());
                    System.out.println("Largest String: " + largest);
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        } while (choice != 3);

        scanner.close();
    }
```

## Output:

```
Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 1
Elements in ArrayList: [Hello, World, Java, Programming]

Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 2
Largest String: Programming

Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 3
```

3. Create a Arraylist storing Employee details including Emp_id, Emp_Name, Emp_gender, Year_of_Joining (you can also add more attributes including these). Then sort the employees according to their joining year using Comparator and Comparable interfaces.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Employee implements Comparable<Employee> {
    int empId;
    String empName;
    String empGender;
    int yearOfJoining;

    public Employee(int empId, String empName, String empGender, int yearOfJoining) {
        this.empId = empId;
        this.empName = empName;
        this.empGender = empGender;
        this.yearOfJoining = yearOfJoining;
    }

    @Override
    public int compareTo(Employee other) {
        return this.yearOfJoining - other.yearOfJoining;
    }

    @Override
    public String toString() {
        return "Employee [ID=" + empId + ", Name=" + empName + ", Gender=" + empGender + ", Year=" + yearOfJoining + "]";
    }
}

public class EmployeeList {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        employees.add(new Employee(1, "John", "Male", 2018));
        employees.add(new Employee(2, "Jane", "Female", 2015));
        employees.add(new Employee(3, "Bob", "Male", 2020));
        employees.add(new Employee(4, "Alice", "Female", 2019));

        // Sorting using Comparable
        Collections.sort(employees);
        System.out.println("Employees sorted by year of joining:");
        for (Employee emp : employees) {
            System.out.println(emp);
        }

        // Sorting using Comparator (optional)
        employees.sort(Comparator.comparingInt(e -> e.yearOfJoining));
    }
```

## Output:

```
Employees sorted by year of joining:
Employee [ID=2, Name=Jane, Gender=Female, Year=2015]
Employee [ID=1, Name=John, Gender=Male, Year=2018]
Employee [ID=4, Name=Alice, Gender=Female, Year=2019]
Employee [ID=3, Name=Bob, Gender=Male, Year=2020]
```

4. Write a program that initializes Vector with 10 integers in it.

- Display all the integers
- Sum of these integers.
- Find Maximum Element in Vector

```java
import java.util.Vector;

public class VectorOperations {
    public static void main(String[] args) {
        Vector<Integer> vector = new Vector<>();

        // Initializing Vector with 10 integers
        for (int i = 1; i <= 10; i++) {
            vector.add(i);
        }

        // Displaying all integers
        System.out.println("Vector elements: " + vector);

        // Sum of the integers
        int sum = 0;
        int maxElement = Integer.MIN_VALUE;
        for (int num : vector) {
            sum += num;
            if (num > maxElement) {
                maxElement = num;
            }
        }
        System.out.println("Sum of integers: " + sum);

        // Finding the maximum element
        System.out.println("Maximum element in Vector: " + maxElement);
    }
}
```

## Output:

```
Vector elements: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of integers: 55
Maximum element in Vector: 10
```

5. Find the k-th smallest element in a sorted ArrayList.

```java
import java.util.ArrayList;
import java.util.Collections;

public class KthSmallestElement {
    public static void main(String[] args) {
        ArrayList<Integer> arrayList = new ArrayList<>();
        Collections.addAll(arrayList, 3, 1, 4, 1, 5, 9, 2, 6, 5, 3);

        Collections.sort(arrayList);
        System.out.println("Sorted ArrayList: " + arrayList);

        int k = 3; // Change k as needed
        if (k > 0 && k <= arrayList.size()) {
            System.out.println(k + "-th smallest element is: " + arrayList.get(k - 1));
        } else {
            System.out.println("Invalid k value");
        }
    }
}
```

**Output:**

```
Sorted ArrayList: [1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
3-th smallest element is: 2
```

6. Write a program to merge two ArrayLists into one.

```java
import java.util.ArrayList;

public class MergeArrayLists {
    public static void main(String[] args) {
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("A");
        list1.add("B");
        list1.add("C");

        ArrayList<String> list2 = new ArrayList<>();
        list2.add("D");
        list2.add("E");
        list2.add("F");

        // Merging lists
        ArrayList<String> mergedList = new ArrayList<>(list1);
        mergedList.addAll(list2);

        System.out.println("Merged ArrayList: " + mergedList);
    }
}
```

**Output:**

```
Merged ArrayList: [A, B, C, D, E, F]
```

## Home Tasks

1. Create a Vector storing integer objects as an input.
    a. Sort the vector
    b. Display largest number
    c. Display smallest number

```java
import java.util.Collections;
import java.util.Vector;

public class VectorOperations {
    public static void main(String[] args) {
        Vector<Integer> vector = new Vector<>();

        // Adding sample integer objects
        Collections.addAll(vector, 34, 12, 45, 7, 23, 89, 2, 56, 90, 15);

        // Sorting the vector
        Collections.sort(vector);
        System.out.println("Sorted Vector: " + vector);

        // Displaying largest number
        int largest = Collections.max(vector);
        System.out.println("Largest number: " + largest);

        // Displaying smallest number
        int smallest = Collections.min(vector);
        System.out.println("Smallest number: " + smallest);
    }
}
```

### Output:

```
Sorted Vector: [2, 7, 12, 15, 23, 34, 45, 56, 89, 90]
Largest number: 90
Smallest number: 2
```

2. Write a java program which takes user input and gives hashcode value of those inputs using hashCode () method.

```java
import java.util.Scanner;

public class HashCodeGenerator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string to get its hash code: ");
        String input = scanner.nextLine();

        int hashCode = input.hashCode();
        System.out.println("Hash code of the input: " + hashCode);

        scanner.close();
    }
}
```

## Output:

```
Enter a string to get its hash code: HelloWorld
Hash code of the input: 1794106052
```

### 3. Scenario based

Create a java project, suppose you work for a company that needs to manage a list of employees. Each employee has a unique combination of a name and an ID. Your goal is to ensure that you can track employees effectively and avoid duplicate entries in your system.

Requirements

a. Employee Class: You need to create an Employee class that includes:

- name: The employee's name (String).
- id: The employee's unique identifier (int).
- Override the hashCode() and equals() methods to ensure that two employees are considered equal if they have the same name and id.

b. Employee Management: You will use a HashSet to store employee records. This will help you avoid duplicate entries.

c. Operations: Implement operations to:

- Add new employees to the record.
- Check if an employee already exists in the records.
- Display all employees.

```java
import java.util.HashSet;
import java.util.Scanner;

class Employee {
    private String name;
    private int id;

    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Employee employee = (Employee) obj;
        return id == employee.id && name.equals(employee.name);
    }

    @Override
    public int hashCode() {
        return name.hashCode() + Integer.hashCode(id);
    }

    @Override
    public String toString() {
        return "Employee [Name=" + name + ", ID=" + id + "]";
    }
}

public class EmployeeManagement {
    public static void main(String[] args) {
        HashSet<Employee> employees = new HashSet<>();
        Scanner scanner = new Scanner(System.in);
        int choice;
```

```
        do {
            System.out.println("\n1. Add new employee");
            System.out.println("2. Check if an employee exists");
            System.out.println("3. Display all employees");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter employee name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter employee ID: ");
                    int id = scanner.nextInt();
                    Employee newEmployee = new Employee(name, id);

                    if (employees.add(newEmployee)) {
                        System.out.println("Employee added successfully.");
                    } else {
                        System.out.println("Employee already exists.");
                    }
                    break;
                case 2:
                    System.out.print("Enter employee name to check: ");
                    String checkName = scanner.nextLine();
                    System.out.print("Enter employee ID to check: ");
                    int checkId = scanner.nextInt();
                    Employee checkEmployee = new Employee(checkName, checkId);

                    if (employees.contains(checkEmployee)) {
                        System.out.println("Employee exists in the records.");
                    } else {
                        System.out.println("Employee does not exist.");
                    }
                    break;
                    case 3:
                        System.out.println("All employees:");
                        for (Employee emp : employees) {
                            System.out.println(emp);
                        }
                        break;
                    case 4:
                        System.out.println("Exiting...");
                        break;
                    default:
                        System.out.println("Invalid choice. Try again.");
                }
            } while (choice != 4);

            scanner.close();
        }
```

**Output:**

```
1. Add new employee
2. Check if an employee exists
3. Display all employees
4. Exit
Enter your choice: 1
Enter employee name: John Doe
Enter employee ID: 101
Employee added successfully.

Enter your choice: 3
All employees:
Employee [Name=John Doe, ID=101]

Enter your choice: 2
Enter employee name to check: John Doe
Enter employee ID to check: 101
Employee exists in the records.

Enter your choice: 4
Exiting...
```

4.Create a Color class that has red, green, and blue values. Two colors are considered equal if their RGB values are the same.

```
import java.util.Objects;

class Color {
    private int red;
    private int green;
    private int blue;

    public Color(int red, int green, int blue) {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Color color = (Color) obj;
        return red == color.red && green == color.green && blue == color.blue;
    }

    @Override
    public int hashCode() {
        return Objects.hash(red, green, blue);
    }

    @Override
    public String toString() {
        return "Color [R=" + red + ", G=" + green + ", B=" + blue + "]";
    }
}

public class ColorComparison {
    public static void main(String[] args) {
        Color color1 = new Color(255, 0, 0);
        Color color2 = new Color(255, 0, 0);
        Color color3 = new Color(0, 255, 0);

        System.out.println("Color1 equals Color2: " + color1.equals(color2)); // true
        System.out.println("Color1 equals Color3: " + color1.equals(color3)); // false
    }
}
```

**Output:**

```
Color1 equals Color2: true
Color1 equals Color3: false
```