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.

```java
package labtask;
import java.util.Vector;

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

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

        // Displaying all integers and calculating the sum
        int sum = 0;
        System.out.println("Integers in the vector:");
        for (Integer num : vector) {
            System.out.print(num + " ");
            sum += num;
        }

        System.out.println("\nSum of integers: " + sum);
```

```
run:
Integers in the vector:
1 2 3 4 5 6 7 8 9 10
Sum of integers: 55
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
package labtask;
import java.util.ArrayList;
import java.util.Scanner;
public class LabTask {
    public static void main(String[] args) {
        ArrayList<String> arrayList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        // Adding some sample strings to the ArrayList
        arrayList.add("Apple");
        arrayList.add("Banana");
        arrayList.add("Cherry");
        arrayList.add("Date");
        arrayList.add("Elderberry");

        boolean exit = false;

        // Menu-driven program
        while (!exit) {
            System.out.println("Menu:");
            System.out.println("1. Display all elements");
            System.out.println("2. Display the largest string");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine();  // Consume the newline character

            switch (choice) {
                case 1:
                    // Display all elements in the ArrayList
                    System.out.println("Elements in the ArrayList: " + arrayList);
                    break;
                case 2:
                    // Display the largest string
                    String largest = arrayList.stream().max(String::compareTo).orElse(null);
                    System.out.println("Largest string: " + largest);
                    break;
                case 3:
                    // Exit the program
                    exit = true;
                    System.out.println("Exiting the program.");
                    break;
                default:
                    // Handle invalid choice
                    System.out.println("Invalid choice! Please try again.");
            }
        }
    }
```

```
run:
Menu:
1. Display all elements
2. Display the largest string
3. Exit
Choose an option: 1
Elements in the ArrayList: [Apple, Banana, Cherry, Date, Elderberry]
Menu:
1. Display all elements
2. Display the largest string
3. Exit
Choose an option: 2
Largest string: Elderberry
Menu:
1. Display all elements
2. Display the largest string
3. Exit
Choose an option: 3
Exiting the program.
BUILD SUCCESSFUL (total time: 7 seconds)
```

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.

```java
class 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 String toString() {
        return "Employee{" +
                "empId=" + empId +
                ", empName='" + empName + '\'' +
                ", empGender='" + empGender + '\'' +
                ", yearOfJoining=" + yearOfJoining +
                '}';
    }
}
```

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
public class EmployeeSorting {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        employees.add(new Employee(1, "Alice", "F", 2024));
        employees.add(new Employee(2, "Bob", "M", 2020));
        employees.add(new Employee(3, "Charlie", "M", 2019));
        employees.add(new Employee(4, "Diana", "F", 2021));

        // Sorting using Comparable
        Collections.sort(employees, Comparator.comparingInt(emp -> emp.yearOfJoining));

        System.out.println("Employees sorted by year of joining:");
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
```

```
run:
Employees sorted by year of joining:
Employee{empId=3, empName='Charlie', empGender='M', yearOfJoining=2019}
Employee{empId=2, empName='Bob', empGender='M', yearOfJoining=2020}
Employee{empId=4, empName='Diana', empGender='F', yearOfJoining=2021}
Employee{empId=1, empName='Alice', empGender='F', yearOfJoining=2024}
BUILD SUCCESSFUL (total time: 0 seconds)
```

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
package labtask;
import java.util.Collections;
import java.util.Vector;
public class LabTask {
    public static void main(String[] args) {

        Vector<Integer> vector = new Vector<>();
        for (int i = 1; i <= 10; i++) {
            vector.add(i * 10); // Adding multiples of 10
        }

        // Displaying all integers and calculating the sum
        int sum = 0;
        System.out.println("Integers in the vector:");
        for (Integer num : vector) {
            System.out.print(num + " ");
            sum += num;
        }

        System.out.println("\nSum of integers: " + sum);

        // Finding the maximum element in the vector
        Integer maxElement = Collections.max(vector);
        System.out.println("Maximum element in the vector: " + maxElement);
```

```
run:
Integers in the vector:
10 20 30 40 50 60 70 80 90 100
Sum of integers: 550
Maximum element in the vector: 100
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```java
package labtask;
import java.util.Collections;
import java.util.ArrayList;
public class LabTask {
    public static void main(String[] args) {

        ArrayList<Integer> arrayList = new ArrayList<>();
        arrayList.add(1);
        arrayList.add(3);
        arrayList.add(5);
        arrayList.add(7);
        arrayList.add(9);

        int k = 3; // For example, find the 3rd smallest element
        if (k <= arrayList.size()) {
            System.out.println(k + "-th smallest element: " + arrayList.get(k - 1));
        } else {
            System.out.println("k is larger than the size of the list.");
        }
```

```
run:
3-th smallest element: 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```java
package labtask;
import java.util.ArrayList;

public class LabTask {
    public static void main(String[] args) {
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("Apple");
        list1.add("Banana");

        ArrayList<String> list2 = new ArrayList<>();
        list2.add("Cherry");
        list2.add("Date");

        // Merging two ArrayLists without using addAll
        ArrayList<String> mergedList = new ArrayList<>(list1); // Start with elements from list1

        // Adding elements from list2 using a loop
        for (String item : list2) {
            mergedList.add(item); // Add each item from list2 to mergedList
        }

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

```
run:
Merged ArrayList: [Apple, Banana, Cherry, Date]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.Vector;
import java.util.Scanner;
import java.util.Collections;
public class HomeTaskk {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        Vector<Integer> vector = new Vector<>();

        System.out.println("Enter integers for the vector (type 'end' to finish): ");
        while (input.hasNextInt()) {
            vector.add(input.nextInt());
        }

        // Sort the vector
        Collections.sort(vector);

        // Display sorted vector
        System.out.println("Sorted Vector: " + vector);

        // Display largest and smallest numbers
        if (!vector.isEmpty()) {
            System.out.println("Largest number: " + vector.lastElement());
            System.out.println("Smallest number: " + vector.firstElement());
        } else {
            System.out.println("Vector is empty.");
        }
```

```
run:
Enter integers for the vector (type 'end' to finish):
9
3
5
1
4
end
Sorted Vector: [1, 3, 4, 5, 9]
Largest number: 9
Smallest number: 1
BUILD SUCCESSFUL (total time: 10 seconds)
```

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

```
package hometaskk;

import java.util.Scanner;

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

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

```
run:
Enter a string to generate its hash code:
Hello, World!
Hash code of the input: 1498789909
BUILD SUCCESSFUL (total time: 1 second)
```

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.Objects;


class Employee {
    private String name;
    private int id;

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

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

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

    @Override
    public int hashCode() {
        return Objects.hash(name, id);
    }

    @Override
    public String toString() {
        return "Employee{name='" + name + "', id=" + id + "}";
    }
}
```

```java
import java.util.Scanner;
import java.util.Objects;
import java.util.HashSet;
public class EmployeeManagement {
    private HashSet<Employee> employeeRecords = new HashSet<>();

    public void addEmployee(Employee employee) {
        if (employeeRecords.contains(employee)) {
            System.out.println("Employee already exists: " + employee);
        } else {
            employeeRecords.add(employee);
            System.out.println("Employee added: " + employee);
        }
    }

    public void displayAllEmployees() {
        System.out.println("All Employees:");
        for (Employee employee : employeeRecords) {
            System.out.println(employee);
        }
    }

    public boolean employeeExists(Employee employee) {
        return employeeRecords.contains(employee);
    }

    public static void main(String[] args) {
        EmployeeManagement management = new EmployeeManagement();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nOptions:\n1. Add Employee\n2. Display All Employees\n3. Check if Employee Exists\n4. Exit")
            int choice = scanner.nextInt();

            switch (choice) {
                case 1 -> {
                    System.out.print("Enter Employee Name: ");
```

```java
            System.out.println("\nOptions:\n1. Add Employee\n2. Display All Employees\n3. Check if Employee Exists\n4. Exit")
            int choice = scanner.nextInt();

            switch (choice) {
                case 1 -> {
                    System.out.print("Enter Employee Name: ");
                    String name = scanner.next();
                    System.out.print("Enter Employee ID: ");
                    int id = scanner.nextInt();
                    Employee employee = new Employee(name, id);
                    management.addEmployee(employee);
                }
                case 2 -> management.displayAllEmployees();
                case 3 -> {
                    System.out.print("Enter Employee Name: ");
                    String name = scanner.next();
                    System.out.print("Enter Employee ID: ");
                    int id = scanner.nextInt();
                    Employee employee = new Employee(name, id);
                    if (management.employeeExists(employee)) {
                        System.out.println("Employee exists.");
                    } else {
                        System.out.println("Employee does not exist.");
                    }
                }
                case 4 -> {
                    System.out.println("Exiting...");
                    scanner.close();
                    return;
                }
                default -> System.out.println("Invalid choice. Try again.");
            }
        }
    }
}
```

```
run:

Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
4. Exit
1
Enter Employee Name: alice
Enter Employee ID: 101
Employee added: Employee{name='alice', id=101}

Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
4. Exit
1
```

```
Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
4. Exit
1
Enter Employee Name: bob
Enter Employee ID: 95
Employee added: Employee{name='bob', id=95}

Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
4. Exit
2
All Employees:
Employee{name='bob', id=95}
```

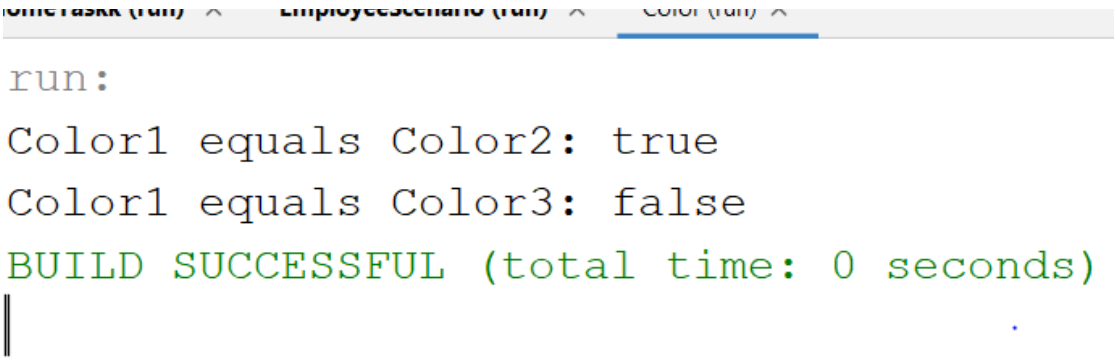
```
All Employees:
Employee{name='bob', id=95}
Employee{name='alice', id=101}

Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
4. Exit
3
Enter Employee Name: jhon
Enter Employee ID: 24
Employee does not exist.

Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
```

```
Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
4. Exit
3
Enter Employee Name: alice
Enter Employee ID: 101
Employee exists.

Options:
1. Add Employee
2. Display All Employees
3. Check if Employee Exists
4. Exit
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 o) {
        if (this == o) return true; // Check if both references point to the same object
        if (o == null || getClass() != o.getClass()) return false; // Check if the object is null or of a different class
        Color color = (Color) o; // Cast the object to Color
        return red == color.red && green == color.green && blue == color.blue; // Check RGB values
    }

    @Override
    public int hashCode() {
        return Objects.hash(red, green, blue); // Generate hash code using RGB values
    }

    @Override
    public String toString() {
        return "Color{" +
                "red=" + red +
                ", green=" + green +
                ", blue=" + blue +
                '}';
    }

    public static void main(String[] args) {
        Color color1 = new Color(255, 0, 0);
```

```
                "red=" + red +
                ", green=" + green +
                ", blue=" + blue +
                '}';
    }

    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));
        System.out.println("Color1 equals Color3: " + color1.equals(color3));
    }
}
```


run:
Color1 equals Color2: true
Color1 equals Color3: false
BUILD SUCCESSFUL (total time: 0 seconds)