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.

```
1   package dsa.lab.pkg2;
2   import java.util.ArrayList;
3   import java.util.Vector;
4   public class DSALab2 {
5       public static void main(String[] args) {
6           // task 1
7           System.out.println("Task 1");
8           // Initialize Vector with 10 integers
9           Vector<Integer> numbers = new Vector<>();
10          for (int i = 1; i <= 10; i++) {
11              numbers.add(i);
12          }
13          // Display all integers in the Vector
14          System.out.println("The integers in the Vector are: " + numbers);
15
16          // Calculate the sum of integers
17          int sum = 0;
18          for (int num : numbers) {
19              sum += num;
20          }
21          // Display the sum of integers
22          System.out.println("The sum of these integers is: " + sum);
```

```
DSA LAB (run) ×   DSA lab 2 (run) ×
run:
Task 1
The integers in the Vector are: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The sum of these integers is: 55
```

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

```java
package dsa.lab.pkg2;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;
import java.util.Vector;
public class DSALab2 {
    public static void main(String[] args) {
        //task 2
        System.out.println("task 2");
         ArrayList<String> stringList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        // Adding some sample strings to the ArrayList
        stringList.add("apple");
        stringList.add("banana");
        stringList.add("watermelon");
        stringList.add("orange");
        stringList.add("grape");
        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:
```

```
            // Display all elements
            System.out.println("Elements in the ArrayList: " + stringList);
            break;

        case 2:
            // Display the largest string
            if (!stringList.isEmpty()) {
                String largestString = stringList.get(0);
                for (String str : stringList) {
                    if (str.length() > largestString.length()) {
                        largestString = str;
                    }
                }
                System.out.println("The largest string is: " + largestString)
            } else {
                System.out.println("The ArrayList is empty.");
            }
            break;
        case 3:
            System.out.println("Exiting program.");
            break;

        default:
            System.out.println("Invalid choice. Please try again.");
    }
} while (choice != 3);
```

DSA LAB (run) × DSA lab 2 (run) ×

```
run:
task 2

Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 1
Elements in the ArrayList: [apple, banana, watermelon, orange, grape]

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

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

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.

```
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 +
                '}';
    }
}
```

```
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

```
package dsa.lab.pkg2;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;
import java.util.Vector;
public class DSALab2 {
    public static void main(String[] args) {
                                //task 4
        System.out.println("task 4");
         // Initialize Vector with 10 integers
        Vector<Integer> numbers = new Vector<>();
        for (int i = 1; i <= 10; i++) {
            numbers.add(i);
        }
        // Display all integers in the Vector
        System.out.println("The integers in the Vector are: " + numbers);
        // Calculate the sum of integers
        int sum = 0;
        for (int num : numbers) {
            sum += num;
        }
        System.out.println("The sum of these integers is: " + sum);
        // Find the maximum element using Collections.max()
        int max = Collections.max(numbers);
        System.out.println("The maximum element in the Vector is: " + max);
```

```
DSA LAB (run) ×  DSA lab 2 (run) ×
run:
task 4
The integers in the Vector are: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The sum of these integers is: 55
The maximum element in the Vector is: 10
```

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

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;
import java.util.Vector;
public class DSALab2 {
    public static void main(String[] args) {
        // Initialize and sort ArrayList
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(10);
        numbers.add(5);
        numbers.add(30);
        numbers.add(20);
        numbers.add(15);
        Collections.sort(numbers);  // Sort the ArrayList
        System.out.println("Sorted ArrayList: " + numbers);// Display the sorted ArrayList
        // Get k value from the user
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the value of k: ");
        int k = scanner.nextInt();
        // Check if k is within the valid range
        if (k > 0 && k <= numbers.size()) {
            // k-th smallest element (1-based index)
            int kthSmallest = numbers.get(k - 1);
            System.out.println("The " + k + "-th smallest element is: " + kthSmallest);
        } else {
            System.out.println("Invalid value of k. It should be between 1 and " + numbers.size());
        }
```

```
run:
Sorted ArrayList: [5, 10, 15, 20, 30]
Enter the value of k: 4
The 4-th smallest element is: 20
BUILD SUCCESSFUL (total time: 3 seconds)
```

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

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;
import java.util.Vector;
public class DSALab2 {
    public static void main(String[] args) {
        // First ArrayList
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("Apple");
        list1.add("Banana");
        list1.add("Cherry");
        // Second ArrayList
        ArrayList<String> list2 = new ArrayList<>();
        list2.add("Date");
        list2.add("Elderberry");
        list2.add("Fig");

        // Merging both lists into a new ArrayList
        ArrayList<String> mergedList = new ArrayList<>(list1); // Start with all elements of list
        mergedList.addAll(list2); // Add all elements of list2

        // Display the merged list
        System.out.println("Merged ArrayList: " + mergedList);
```

```
DSA LAB (run) ×  DSA lab 2 (run) ×
run:
Merged ArrayList: [Apple, Banana, Cherry, Date, Elderberry, Fig]
BUILD SUCCESSFUL (total time: 1 second)
```

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.Scanner;
import java.util.Vector;
import java.util.Collections;
public class hometask {
    public static void main(String[] args) {
        Vector<Integer> numbers = new Vector<>();
        Scanner scanner = new Scanner(System.in);
        // Taking input from user
        System.out.println("Enter 5 integers to add to the Vector:");
        for (int i = 0; i < 5; i++) {
            System.out.print("Enter integer " + (i + 1) + ": ");
            int number = scanner.nextInt();
            numbers.add(number);
        }
        // Sorting the Vector
        Collections.sort(numbers);
        // Display sorted Vector
        System.out.println("Sorted Vector: " + numbers);
        // Display the largest number
        int largest = Collections.max(numbers);
        System.out.println("Largest number in the Vector: " + largest);
        // Display the smallest number
        int smallest = Collections.min(numbers);
        System.out.println("Smallest number in the Vector: " + smallest);
    }
}
```

```
run:
Enter 5 integers to add to the Vector:
Enter integer 1: 15
Enter integer 2: 30
Enter integer 3: 50
Enter integer 4: 70
Enter integer 5: 20
Sorted Vector: [15, 20, 30, 50, 70]
Largest number in the Vector: 70
Smallest number in the Vector: 15
BUILD SUCCESSFUL (total time: 19 seconds)
```

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

```java
package dsa.lab.pkg2;
import java.util.Scanner;
public class LAB2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String userInput = scanner.nextLine();
        // Getting the hash code of the user input
        int hashCode = userInput.hashCode();
        System.out.println("The hash code of the input string \"" + userInput + "\" is: " + hashCode);
    }
}
```

```
run:
Enter a string: hello world
The hash code of the input string "hello world" is: 1794106052
BUILD SUCCESSFUL (total time: 8 seconds)
```

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
public 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 hashCode and equals to ensure uniqueness based on name and id
    @Override
    public int hashCode() {
        return name.hashCode() + Integer.hashCode(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 String toString() {
        return "Employee Name: " + name + ", ID: " + id;
    }
```

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

public class EmployeeManagement {
    private HashSet<Employee> employees;

    public EmployeeManagement() {
        employees = new HashSet<>();
    }

    // Add a new employee
    public boolean addEmployee(String name, int id) {
        Employee newEmployee = new Employee(name, id);
        if (employees.contains(newEmployee)) {
            System.out.println("Employee already exists in the record
            return false;
        } else {
            employees.add(newEmployee);
            System.out.println("Employee added successfully.");
            return true;
        }
    }

    // Check if an employee already exists
    public boolean checkEmployee(String name, int id) {
```

```
        Employee employeeToCheck = new Employee(name, id);
        return employees.contains(employeeToCheck);
    }

    // Display all employees
    public void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            System.out.println("Employee Records:");
            for (Employee employee : employees) {
                System.out.println(employee);
            }
        }
    }

    // Main method for testing
    public static void main(String[] args) {
        EmployeeManagement employeeManagement = new EmployeeManagement();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nOptions: 1) Add Employee  2) Check Employee  3) Display
            System.out.print("Choose an option: ");
            int 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();
        employeeManagement.addEmployee(name, id);
        break;
    case 2:
        System.out.print("Enter Employee Name to Check: ");
        name = scanner.nextLine();
        System.out.print("Enter Employee ID to Check: ");
        id = scanner.nextInt();
        if (employeeManagement.checkEmployee(name, id)) {
            System.out.println("Employee exists in the records.");
        } else {
            System.out.println("Employee does not exist in the records.");
        }
        break;
    case 3:
        employeeManagement.displayEmployees();
        break;
    case 4:
        System.out.println("Exiting...");
        scanner.close();
        return;
```

Copy code

```
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

```
Options: 1) Add Employee  2) Check Employee  3) Display Employees  4) Exit
Choose an option: 1
Enter Employee Name: Alice
Enter Employee ID: 101
Employee added successfully.

Options: 1) Add Employee  2) Check Employee  3) Display Employees  4) Exit
Choose an option: 1
Enter Employee Name: Bob
Enter Employee ID: 102
Employee added successfully.

Options: 1) Add Employee  2) Check Employee  3) Display Employees  4) Exit
Choose an option: 1
Enter Employee Name: Alice
Enter Employee ID: 101
Employee already exists in the records.

Options: 1) Add Employee  2) Check Employee  3) Display Employees  4) Exit
Choose an option: 2
Enter Employee Name to Check: Alice
Enter Employee ID to Check: 101
Employee exists in the records.

Options: 1) Add Employee  2) Check Employee  3) Display Employees  4) Exit
Choose an option: 2
Enter Employee Name to Check: Charlie
Enter Employee ID to Check: 103
Employee does not exist in the records.

Options: 1) Add Employee  2) Check Employee  3) Display Employees  4) Exit
Choose an option: 3
Employee Records:
Employee Name: Alice, ID: 101
Employee Name: Bob, ID: 102

Options: 1) Add Employee  2) Check Employee  3) Display Employees  4) Exit
Choose an option: 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

```java
public 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;
    }

    public int getRed() {
        return red;
    }

    public int getGreen() {
        return green;
    }

    public int getBlue() {
        return blue;
    }

    // Override hashCode to generate a unique code based on RGB values
    @Override
    public int hashCode() {
        return Integer.hashCode(red) + Integer.hashCode(green) + Integer.hashCode(blue);
    }

    // Override equals to consider two colors equal if their RGB values are the same
    @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 String toString() {
        return "Color [R=" + red + ", G=" + green + ", B=" + blue + "]";
    }
}
```

```
Color1: Color(R=255, G=0, B=0)
Color2: Color(R=255, G=0, B=0)
Color3: Color(R=0, G=255, B=0)
Color1 is equal to Color2: true
Color1 is equal to Color3: false
```