NAME: ANEEQ SHAMS ROLL NO: BSE-230 SECTION: E

Lab #02

"Data Structures and Algorithms"

Exercises

Lab Tasks:

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

- Input:

```
package javaapplication2aneeq230;
import java.util.Vector;

public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {

        Vector<Integer> numbers = new Vector<>();

        // Adding 10 integers to the Vector
        for (int i = 1; i <= 10; i++) {
            numbers.add(i);
        }

        // Displaying the integers
        System.out.println("Integers in the Vector: " + numbers);

        // Calculating the sum
        int sum = 0;
        for (int num : numbers) {
            sum += num;
        }

        System.out.println("Sum of integers: " + sum);
    }
}
```

Output:

```
run:
Integers in the Vector: [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

- Input:

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

public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {
        ArrayList<String> strings = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        int choice;

        // Adding some strings to the ArrayList for demonstration
        strings.add("Apple");
        strings.add("Banana");
        strings.add("Strawberry");
        strings.add("Pineapple");
        strings.add("Orange");

        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 the ArrayList:");
                    for (String s : strings) {
                        System.out.println(s);
                    }
                    break;
                case 2:
                    String largestString = findLargestString(strings);
                    System.out.println("Largest String: " + largestString);
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 3);

        scanner.close();
    }

    // Method to find the largest string based on length
    private static String findLargestString(ArrayList<String> strings) {
        if (strings.isEmpty()) return "List is empty.";
        String largest = strings.get(0);
        for (String s : strings) {
            if (s.length() > largest.length()) {
                largest = s;
            }
        }
        return largest;
    }
}
```

Output:

```
Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 1
Elements in the ArrayList:
Apple
Banana
Strawberry
Pineapple
Orange
```

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.

- Input:

```
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;

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

    // Implement compareTo for sorting by yearOfJoining (Comparable)
    @Override
    public int compareTo(Employee other) {
        return Integer.compare(this.yearOfJoining, other.yearOfJoining);
    }
    @Override
    public String toString() {
        return "Employee ID: " + empId + ", Name: " + empName + ", Gender: " + empGender + ", Year of Joining: " + yearOfJoining;
    }
}

public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();

        // Adding Employee details
        employees.add(new Employee(101, "Alice", "Female", 2018));
        employees.add(new Employee(102, "Bob", "Male", 2020));
        employees.add(new Employee(103, "Charlie", "Male", 2017));
        employees.add(new Employee(104, "Diana", "Female", 2019));

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

```
        // Sort using Comparator (explicit order)
        employees.sort(Comparator.comparingInt(e -> e.yearOfJoining));
        System.out.println("\nEmployees sorted by joining year (Comparator):");
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
```

Output:

```
run:
Employees sorted by joining year (Comparable):
Employee ID: 103, Name: Charlie, Gender: Male, Year of Joining: 2017
Employee ID: 101, Name: Alice, Gender: Female, Year of Joining: 2018
Employee ID: 104, Name: Diana, Gender: Female, Year of Joining: 2019
Employee ID: 102, Name: Bob, Gender: Male, Year of Joining: 2020

Employees sorted by joining year (Comparator):
Employee ID: 103, Name: Charlie, Gender: Male, Year of Joining: 2017
Employee ID: 101, Name: Alice, Gender: Female, Year of Joining: 2018
Employee ID: 104, Name: Diana, Gender: Female, Year of Joining: 2019
Employee ID: 102, Name: Bob, Gender: Male, Year of Joining: 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

- Input:

```
import java.util.Vector;
import java.util.Collections;
public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {
        Vector<Integer> numbers = new Vector<>();

        // Adding 10 integers to the Vector
        for (int i = 1; i <= 10; i++) {
            numbers.add(i); // Adding integers from 1 to 10
        }

        // Display all integers
        System.out.println("Integers in the Vector: " + numbers);

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

        // Find the maximum element
        int max = Collections.max(numbers);
        System.out.println("Maximum element in the Vector: " + max);

    }
}
```

Output:

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

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

- Input:

```
import java.util.ArrayList;
import java.util.Arrays;

public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>(Arrays.asList(2, 4, 6, 8, 10, 12, 14, 16, 18));
        int k = 3; // For example, finding the 3rd smallest element

        if (k > 0 && k <= numbers.size()) {
            int kthSmallest = numbers.get(k - 1);
            System.out.println("The " + k + "-th smallest element is: " + kthSmallest);
        } else {
            System.out.println("Invalid value of k.");
        }
    }
}
```

Output:

```
run:
The 3-th smallest element is: 6
```

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

- Input:

```
import java.util.ArrayList;
import java.util.Arrays;

public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {
        ArrayList<Integer> list1 = new ArrayList<>(Arrays.asList(1, 2, 3, 4));
        ArrayList<Integer> list2 = new ArrayList<>(Arrays.asList(5, 6, 7, 8));

        // Merging list2 into list1
        list1.addAll(list2);

        System.out.println("Merged ArrayList: " + list1);
    }
}
```

Output:

```
run:
Merged ArrayList: [1, 2, 3, 4, 5, 6, 7, 8]
```

Home Tasks:

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

- Input:

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

public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {
        Vector<Integer> numbers = new Vector<>();

        // Adding integers to the Vector
        numbers.add(15);
        numbers.add(3);
        numbers.add(42);
        numbers.add(7);
        numbers.add(29);

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

        // Displaying the largest and smallest numbers
        int largest = Collections.max(numbers);
        int smallest = Collections.min(numbers);

        System.out.println("Largest number: " + largest);
        System.out.println("Smallest number: " + smallest);
    }
}
```

Output:

```
run:
Sorted Vector: [3, 7, 15, 29, 42]
Largest number: 42
Smallest number: 3
```

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

- Input:

```
import java.util.Scanner;
public class JavaApplication2Aneeq230 {
    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 hashCodeValue = userInput.hashCode();

        // Displaying the hash code value
        System.out.println("Hash code of the entered string: " + hashCodeValue);

        scanner.close();
    }
}
```

Output:

```
run:
Enter a string: yes
Hash code of the entered string: 119527
```

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.

- Input:

```
import java.util.Objects;
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 int hashCode() {
        return Objects.hash(name, id);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (!(obj instanceof Employee)) return false;
        Employee other = (Employee) obj;
        return id == other.id && Objects.equals(name, other.name);
    }
    @Override
    public String toString() {
        return "Employee{" + "name='" + name + '\'' + ", id=" + id + '}';
    }
}
public class JavaApplication2Aneeq230 {
    public static void main(String[] args) {
        HashSet<Employee> employeeSet = new HashSet<>();
        Scanner scanner = new Scanner(System.in);
        String choice;

        do {
            System.out.print("Enter employee name: ");
            String name = scanner.nextLine();
            System.out.print("Enter employee ID: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            Employee employee = new Employee(name, id);

            // Add new employee if not already exists
            if (employeeSet.add(employee)) {
                System.out.println("Employee added successfully.");
            } else {
                System.out.println("Employee already exists.");
```

```
        }

        System.out.print("Do you want to add another employee? (yes/no): ");
        choice = scanner.nextLine();
    } while (choice.equalsIgnoreCase("yes"));

    // Display all employees
    System.out.println("\nEmployee Records:");
    for (Employee emp : employeeSet) {
        System.out.println(emp);
    }

    scanner.close();
  }
}
```

Output:

```
run:
Enter employee name: Aneeq
Enter employee ID: 123
Employee added successfully.
Do you want to add another employee? (yes/no): yes
Enter employee name: Ahsan
Enter employee ID: 456
Employee added successfully.
Do you want to add another employee? (yes/no): no

Employee Records:
Employee{name='Aneeq', id=123}
Employee{name='Ahsan', id=456}
```

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

- Input:

```
import java.util.Objects;

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

    // Constructor
    public Color(int red, int green, int blue) {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }
    // Getters
    public int getRed() {
        return red;
    }
    public int getGreen() {
        return green;
    }
    public int getBlue() {
        return blue;
    }
```

```
    // Override equals method
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true; // Check for reference equality
        if (!(obj instanceof Color)) return false; // Check type
        Color other = (Color) obj; // Typecast to Color
        return red == other.red && green == other.green && blue == other.blue; // Check RGB values
    }

    // Override hashCode method
    @Override
    public int hashCode() {
        return Objects.hash(red, green, blue); // Create hash code based on RGB values
    }

    // Override toString method for easy display
    @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
        Color color2 = new Color(255, 0, 0); // Red
        Color color3 = new Color(0, 255, 0); // Green

        System.out.println("Color 1: " + color1);
        System.out.println("Color 2: " + color2);
        System.out.println("Color 3: " + color3);

        // Check equality
        System.out.println("color1 equals color2: " + color1.equals(color2)); // Should be true
        System.out.println("color1 equals color3: " + color1.equals(color3)); // Should be false

        // Display hash codes
        System.out.println("Hash Code of Color 1: " + color1.hashCode());
        System.out.println("Hash Code of Color 2: " + color2.hashCode());
        System.out.println("Hash Code of Color 3: " + color3.hashCode());
    }
}
```

Output:

```
run:
Color 1: Color{red=255, green=0, blue=0}
Color 2: Color{red=255, green=0, blue=0}
Color 3: Color{red=0, green=255, blue=0}
color1 equals color2: true
color1 equals color3: false
Hash Code of Color 1: 274846
Hash Code of Color 2: 274846
Hash Code of Color 3: 37696
```