1. Introduction to Collections Framework

🔻 Direct:

1. Write a program to demonstrate adding and printing elements from an `ArrayList`.

```
import java.util.ArrayList;


public class ArrayListDemo {
    public static void main(String[] args) {
        // Create an ArrayList of Strings
        ArrayList<String> courseList = new ArrayList<>();

        // Add elements to the ArrayList
        courseList.add("Java Programming");
        courseList.add("Database Systems");
        courseList.add("Operating Systems");
        courseList.add("Data Structures");

        // Print the elements using a for-each loop
        System.out.println("Courses Available:");
        for (String course : courseList) {
            System.out.println("- " + course);
        }
    }
}
```

2. Show how to use `Collections.max()` and `Collections.min()` on a list of integers.

```
public class MaxMinDemo {

    public static void main(String[] args) {

        // Create an ArrayList of Integers

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


        // Add some numbers to the list

        numbers.add(42);

        numbers.add(17);

        numbers.add(89);

        numbers.add(23);

        numbers.add(65);


        // Print the list

        System.out.println("List of Numbers: " + numbers);


        // Find and print the maximum and minimum values

        int max = Collections.max(numbers);

        int min = Collections.min(numbers);


        System.out.println("Maximum Value: " + max);

        System.out.println("Minimum Value: " + min);

    }
```

```
}
```

3.Demonstrate the use of `Collections.sort()` on a list of strings.

```
import java.util.ArrayList;

import java.util.Collections;


public class SortStringList {

    public static void main(String[] args) {

        // Create an ArrayList of Strings

        ArrayList<String> fruits = new ArrayList<>();


        // Add elements to the list

        fruits.add("Banana");

        fruits.add("Apple");

        fruits.add("Mango");

        fruits.add("Orange");

        fruits.add("Grapes");


        // Print the original list

        System.out.println("Before Sorting: " + fruits);


        // Sort the list in ascending (alphabetical) order

        Collections.sort(fruits);
```

```
        // Print the sorted list
        System.out.println("After Sorting: " + fruits);
    }
}
```

Scenario-Based:

3. You need to store a dynamic list of student names and display them in alphabetical order. Implement this using a suitable collection.

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

public class StudentNameSorter {
    public static void main(String[] args) {
        // Create a dynamic list to store student names
        ArrayList<String> studentNames = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter student names (type 'done' to finish):");

        // Input names dynamically
        while (true) {
            String name = scanner.nextLine();
```

```
            if (name.equalsIgnoreCase("done")) {
                break;
            }
            studentNames.add(name);
        }

        // Sort the list in alphabetical order
        Collections.sort(studentNames);

        // Display sorted names
        System.out.println("\nStudent Names in Alphabetical Order:");
        for (String name : studentNames) {
            System.out.println(name);
        }

        scanner.close();
    }
}
```

4. A user can input any number of integers. Your program should store them and display the sum of all elements using the Collection Framework.

```
import java.util.ArrayList;

import java.util.Scanner;
```

```
public class IntegerSumCalculator {

    public static void main(String[] args) {

        // Create an ArrayList to store integers

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

        Scanner scanner = new Scanner(System.in);



        System.out.println("Enter integers (type 'done' to finish):");



        // Input loop

        while (true) {

            String input = scanner.nextLine();

            if (input.equalsIgnoreCase("done")) {

                break;

          }

         try {
```

```
            int num = Integer.parseInt(input);

            numbers.add(num);

        } catch (NumberFormatException e) {

            System.out.println("Invalid input. Please enter an integer or 'done'.");

        }

    }



    // Calculate the sum

    int sum = 0;

    for (int num : numbers) {

        sum += num;

     // Display result

    System.out.println("\nYou entered: " + numbers);

    System.out.println("Sum of all numbers: " + sum);
```

```
        scanner.close();

    }

}
```

2. List Interface

🔻 Direct:

1. Write a Java program to add, remove, and access elements in an `ArrayList`

```
import java.util.List;
import java.util.ArrayList;


public class ListInterfaceExample {


    public static void main(String[] args) {
        // Declare a List using the ArrayList implementation
        List<String> colors = new ArrayList<>();


        // 1. Add elements to the List
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");
        System.out.println("Initial List: " + colors);
```

2.

```
        // 2. Access elements using get()
        String firstColor = colors.get(0);
        System.out.println("First element: " + firstColor);

        // 3. Remove an element using remove()
        colors.remove("Green"); // You can also use remove(index)
        System.out.println("After removing 'Green': " + colors);

        // 4. Access element at index 1
        if (colors.size() > 1) {
            System.out.println("Element at index 1: " + colors.get(1));
        }

        // 5. Print the size of the list
        System.out.println("Total elements in the list: " + colors.size());
    }
}
```

3. Implement a `LinkedList` that stores and prints employee names.

```
import java.util.LinkedList;
import java.util.List;

public class EmployeeList {
```

```
    public static void main(String[] args) {
        // Create a List of employee names using LinkedList
        List<String> employeeNames = new LinkedList<>();

        // Add employee names to the LinkedList
        employeeNames.add("Alice");
        employeeNames.add("Bob");
        employeeNames.add("Charlie");
        employeeNames.add("Diana");

        // Print employee names using enhanced for loop
        System.out.println("Employee Names:");
        for (String name : employeeNames) {
            System.out.println(name);
        }
    }
}
```

4. Demonstrate inserting an element at a specific position in a `List`.

```
    import java.util.List;
    import java.util.ArrayList;

    public class InsertUsingListInterface {
```

```
    public static void main(String[] args) {
        // Declare a List using ArrayList implementation
        List<String> languages = new ArrayList<>();

        // Add initial elements
        languages.add("Java");
        languages.add("Python");
        languages.add("C++");

        System.out.println("Original List: " + languages);

        // Insert "JavaScript" at index 1
        languages.add(1, "JavaScript");

        System.out.println("After inserting 'JavaScript' at index 1: " + languages);
    }
}
```