✅ 1. Introduction to Collections Framework

🔷 Direct:

1. Write a program to demonstrate adding and printing elements from an `ArrayList`.
2. Show how to use `Collections.max()` and `Collections.min()` on a list of integers.
3. Demonstrate the use of `Collections.sort()` on a list of strings.

🔷 Scenario-Based:

4. You need to store a dynamic list of student names and display them in alphabetical order. Implement this using a suitable collection.
5. A user can input any number of integers. Your program should store them and display the sum of all elements using the Collection Framework.

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

public class ArrayListDemo {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
ArrayList<String> fruits = new ArrayList<>();

        // Adding elements
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");

        // Printing elements
        System.out.println("Fruits in the list:");
        for (String fruit : fruits) {
            System.out.println(fruit);
        }

	}

}
```

```
package CollectionsFramework;

import java.util.ArrayList;

import java.util.Collections;


public class MinMaxDemo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList<Integer> numbers = new ArrayList<>();
    numbers.add(10);
    numbers.add(5);
    numbers.add(20);
    numbers.add(3);
    numbers.add(15);

    System.out.println("Maximum value: " + Collections.max(numbers));
    System.out.println("Minimum value: " + Collections.min(numbers));

    }

}
```

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

public class SortDemo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList<String> names = new ArrayList<>();
    names.add("John");
    names.add("Alice");
    names.add("Bob");
    names.add("Eve");
```

```
        System.out.println("Before sorting: " + names);

        Collections.sort(names);

        System.out.println("After sorting: " + names);

        }


}

package CollectionsFramework;

import java.util.ArrayList;

import java.util.Collections;


public class StudentNames {


        public static void main(String[] args) {

                // TODO Auto-generated method stub

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

        students.add("Rahul");

        students.add("Priya");

        students.add("Amit");

        students.add("Neha");


        Collections.sort(students);


        System.out.println("Students in alphabetical order:");

        for (String student : students) {

            System.out.println(student);

        }


        }
```

```
}
package CollectionsFramework;

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

public class SumOfNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<Integer> numbers = new ArrayList<>();

        System.out.println("Enter integers (type 'done' to finish):");
        while (scanner.hasNextInt()) {
            numbers.add(scanner.nextInt());
        }

        int sum = 0;
        for (int num : numbers) {
            sum += num;
            scanner.close();
        }

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

✅ 2. List Interface

🔷 Direct:

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

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

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

🔷 Scenario-Based:

4. You're building a to-do list manager. Use `ArrayList` to add tasks, remove completed ones, and display pending tasks.

5. Create a simple shopping cart system where users can add/remove products using a `List`.

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

public class ArrayListOperations {

        public static void main(String[] args) {
            // TODO Auto-generated method stub
ArrayList<String> colors = new ArrayList<>();


    colors.add("Red");
    colors.add("Green");
    colors.add("Blue");


    System.out.println("First color: " + colors.get(0));


    colors.remove("Green");
    System.out.println("After removal: " + colors);

        }

}

package listintf;

import java.util.LinkedList;

public class EmployeeList {
  public static void main(String[] args) {
    LinkedList<String> employees = new LinkedList<>();

    employees.add("John Doe");
    employees.add("Jane Smith");
    employees.add("Mike Johnson");

    System.out.println("Employee Names:");
    for (String employee : employees) {
      System.out.println(employee);
```

```
        }
    }
}

package listintf;

import java.util.ArrayList;

public class ListInsertion {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(10);
        numbers.add(30);


        numbers.add(1, 20);

        System.out.println("List after insertion: " + numbers);
    }
}


package listintf;

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

public class ShoppingCart {
    public static void main(String[] args) {
        ArrayList<String> cart = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Add product\n2. Remove product\n3. View cart\n4. Exit");
            System.out.print("Choose option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter product name: ");
                    cart.add(scanner.nextLine());
                    break;
                case 2:
                    System.out.print("Enter product to remove: ");
                    cart.remove(scanner.nextLine());
                    break;
                case 3:
                    System.out.println("Cart contents: " + cart);
                    break;
                case 4:
                    return;
            }
        }
    }
```

```
}

package listintf;

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

public class TodoList {
    public static void main(String[] args) {
        ArrayList<String> tasks = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Add task\n2. Remove completed task\n3. View tasks\n4. Exit");
            System.out.print("Choose option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter task: ");
                    tasks.add(scanner.nextLine());
                    break;
                case 2:
                    System.out.print("Enter task to remove: ");
                    tasks.remove(scanner.nextLine());
                    break;
                case 3:
                    System.out.println("Pending tasks: " + tasks);
                    break;
                case 4:
                    return;


            }
        }
    }
}
```

✅ 3. Set Interface

🔷 Direct:

1. Write a program using `HashSet` to store unique student roll numbers.
2. Demonstrate how to use `TreeSet` to automatically sort elements.

3. Use `LinkedHashSet` to maintain insertion order and prevent duplicates.

🔷 Scenario-Based:

4. Design a program to store registered email IDs of users such that no duplicates are allowed.

5. Create a program where a `Set` is used to eliminate duplicate entries from a list of city names entered by users.

```java
package Setinterface;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Scanner;

public class CityNameCleaner {
    public static void main(String[] args) {
        ArrayList<String> cityList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter city names (type 'done' to finish):");
        while (true) {
            String city = scanner.nextLine();
            if (city.equalsIgnoreCase("done")) {
                break;
            }
            cityList.add(city);
        }

        HashSet<String> uniqueCities = new HashSet<>(cityList);
        System.out.println("Unique Cities: " + uniqueCities);
    }
}
```

```java
package Setinterface;


import java.util.HashSet;

import java.util.Scanner;


public class EmailRegistration {


	public static void main(String[] args) {

		// TODO Auto-generated method stub
```

```
                HashSet<String> emailIds = new HashSet<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.print("Enter email ID (or 'exit' to quit): ");
            String email = scanner.nextLine();

            if (email.equalsIgnoreCase("exit")) {
                break;
            }

            if (emailIds.add(email)) {
                System.out.println("Email registered successfully!");
            } else {
                System.out.println("Email already exists!");
            }
        }

        System.out.println("Registered Emails: " + emailIds);

        }

}
package Setinterface;
import java.util.LinkedHashSet;
```

```
public class OrderedSetDemo {
    public static void main(String[] args) {
        LinkedHashSet<String> cities = new LinkedHashSet<>();

        cities.add("Mumbai");
        cities.add("Delhi");
        cities.add("Bangalore");
        cities.add("Mumbai"); // Duplicate will be ignored

        System.out.println("Cities in insertion order: " + cities);
    }
}
package Setinterface;
import java.util.TreeSet;

public class SortedSetDemo {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
TreeSet<String> names = new TreeSet<>();

        names.add("John");
        names.add("Alice");
        names.add("Bob");
    System.out.println("Sorted Names: " + names);

    }

}
package Setinterface;
import java.util.HashSet;

public class StudentRollNumbers {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
HashSet<Integer> rollNumbers = new HashSet<>();

    rollNumbers.add(101);
    rollNumbers.add(102);
    rollNumbers.add(101); // Duplicate will be ignored

    System.out.println("Unique Roll Numbers: " + rollNumbers);

    }

}
```

✅ 4. Map Interface

🔷 Direct:

1. Write a program using `HashMap` to store student names and their marks.
2. Demonstrate how to iterate over a `Map` using `entrySet()`.
3. Show how to update the value associated with a key in a `Map`.

🔷 Scenario-Based:

4. Build a phone directory where names are keys and phone numbers are values.
5. Create a frequency counter for words in a sentence using a `Map`.

```java
package Mapinterface;

import java.util.HashMap;
import java.util.Map;

public class MapIteration {
    public static void main(String[] args) {
        HashMap<String, Integer> ages = new HashMap<>();
        ages.put("John", 25);
        ages.put("Sarah", 30);
        ages.put("Mike", 28);

        for (Map.Entry<String, Integer> entry : ages.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }
}
```

```java
package Mapinterface;

import java.util.HashMap;
import java.util.Scanner;

public class PhoneDirectory {
    public static void main(String[] args) {
        HashMap<String, String> directory = new HashMap<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Add Contact\n2. Find Number\n3. Exit");
            System.out.print("Choose option: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter phone: ");
                    String phone = scanner.nextLine();
                    directory.put(name, phone);
                    break;
                case 2:
                    System.out.print("Enter name to search: ");
                    String searchName = scanner.nextLine();
                    System.out.println("Phone: " + directory.get(searchName));
                    break;
                case 3:
                    return;
            }
        }
    }
}

package Mapinterface;

import java.util.HashMap;

public class StudentMarks {
    public static void main(String[] args) {
        HashMap<String, Integer> studentMarks = new HashMap<>();

        studentMarks.put("Alice", 85);
        studentMarks.put("Bob", 90);
        studentMarks.put("Charlie", 78);

        System.out.println("Student Marks: " + studentMarks);
    }
}

package Mapinterface;

import java.util.HashMap;

public class UpdateMap {
    public static void main(String[] args) {
        HashMap<String, String> capitals = new HashMap<>();
        capitals.put("USA", "Washington DC");
        capitals.put("France", "Paris");


        capitals.put("USA", "New York");

        System.out.println("Updated Capitals: " + capitals);
    }
}

package Mapinterface;
```

```
import java.util.HashMap;
import java.util.Map;

public class WordCounter {
    public static void main(String[] args) {
        String sentence = "hello world hello java world java programming";
        String[] words = sentence.split(" ");

        HashMap<String, Integer> frequencyMap = new HashMap<>();

        for (String word : words) {
            frequencyMap.put(word, frequencyMap.getOrDefault(word, 0) + 1);
        }

        System.out.println("Word Frequencies:");
        for (Map.Entry<String, Integer> entry : frequencyMap.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }
}
```

✅ 5. Queue Interface

🔷 Direct:

1. Implement a simple task queue using `LinkedList` as a `Queue`.
2. Demonstrate how to add and remove elements using `offer()` and `poll()`.
3. Use a `PriorityQueue` to order tasks by priority (integers).

🔷 Scenario-Based:

4. Simulate a print queue system where print jobs are processed in order.
5. Create a ticket booking system where customer names are added to a queue and served in order.

```
package QueueInter;

import java.util.LinkedList;
import java.util.Queue;

public class PrintQueue {
    public static void main(String[] args) {
        Queue<String> printJobs = new LinkedList<>();


        printJobs.offer("Document1.pdf");
        printJobs.offer("Report.docx");
        printJobs.offer("Image.jpg");
```

```java

        System.out.println("Processing print jobs:");
        while (!printJobs.isEmpty()) {
            String currentJob = printJobs.poll();
            System.out.println("Printing: " + currentJob);
        }
    }
}
```

```java
package QueueInter;


import java.util.PriorityQueue;
import java.util.Queue;


public class PriorityTaskQueue {
    public static void main(String[] args) {
        Queue<Integer> tasks = new PriorityQueue<>();


        tasks.add(3);
        tasks.add(1);
        tasks.add(2);


        System.out.println("Processing tasks in priority order:");
        while (!tasks.isEmpty()) {
            System.out.println("Processing task with priority: " + tasks.poll());
        }
    }
}
```

```
package QueueInter;

import java.util.LinkedList;
import java.util.Queue;

public class QueueOperations {
    public static void main(String[] args) {
        Queue<Integer> numbers = new LinkedList<>();


        numbers.offer(10);
        numbers.offer(20);
        numbers.offer(30);


        System.out.println("Processed: " + numbers.poll());
        System.out.println("Processed: " + numbers.poll());

        System.out.println("Remaining: " + numbers);
    }
}
package QueueInter;

import java.util.LinkedList;
import java.util.Queue;
```

```
public class TaskQueue {
    public static void main(String[] args) {
        Queue<String> tasks = new LinkedList<>();


        tasks.add("Task 1");
        tasks.add("Task 2");
        tasks.add("Task 3");

        System.out.println("Current tasks: " + tasks);
    }
}

package QueueInter;

import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class TicketBooking {
    public static void main(String[] args) {
        Queue<String> customers = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\n1. Add customer\n2. Serve next customer\n3. View queue\n4. Exit");
            System.out.print("Choose option: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter customer name: ");
                    customers.offer(scanner.nextLine());
                    break;
                case 2:
                    if (!customers.isEmpty()) {
                        System.out.println("Serving: " + customers.poll());
                    } else {
                        System.out.println("No customers in queue");
                    }
                    break;
                case 3:
                    System.out.println("Current queue: " + customers);
                    break;
                case 4:
                    return;
            }
        }
    }
}
```

✅ 6. Iterator Interface

🔷 Direct:

1. Write a program to iterate through a list using `Iterator`.
2. Demonstrate removing an element from a list while iterating using `Iterator`.
3. Show how to use `ListIterator` to iterate in both directions.

🔷 Scenario-Based:

4. Design a program that reads a list of book titles and removes those starting with a specific letter using an iterator.
5. Create a program that reverses the elements in a list using `ListIterator`.

```
package IteratorInte;

import java.util.ArrayList;
import java.util.Iterator;

public class BasicIterator {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<>();
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");

        Iterator<String> it = colors.iterator();
        while(it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

```
package IteratorInte;

import java.util.ArrayList;
import java.util.ListIterator;

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

        ListIterator<String> lit = fruits.listIterator();

        System.out.println("Forward iteration:");
        while(lit.hasNext()) {
            System.out.println(lit.next());
        }
        System.out.println("\nBackward iteration:");
        while(lit.hasPrevious()) {
            System.out.println(lit.previous());
        }
    }
}
```

```
package IteratorInte;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class BookFilter {
    public static void main(String[] args) {
        ArrayList<String> books = new ArrayList<>();
        books.add("Atomic Habits");
        books.add("Deep Work");
        books.add("The Alchemist");
        books.add("Digital Minimalism");

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter letter to filter by: ");
        char letter = scanner.next().charAt(0);

        Iterator<String> it = books.iterator();
        while(it.hasNext()) {
            String title = it.next();
            if(title.charAt(0) == Character.toUpperCase(letter)) {
                it.remove();
            }
        }

        System.out.println("Filtered book list: " + books);
    }
}
```

```
package IteratorInte;

import java.util.ArrayList;
import java.util.ListIterator;

public class ListReverser {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");
        names.add("Diana");

        System.out.println("Original list: " + names);
        ListIterator<String> fwd = names.listIterator();
        ListIterator<String> rev = names.listIterator(names.size());

        for(int i=0; i<names.size()/2; i++) {
            String temp = fwd.next();
            fwd.set(rev.previous());
            rev.set(temp);
        }

        System.out.println("Reversed list: " + names);
    }
}
```

```
package IteratorInte;

import java.util.ArrayList;
import java.util.Iterator;

public class SafeRemoval {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(10);
        numbers.add(20);
        numbers.add(30);
        numbers.add(40);

        Iterator<Integer> it = numbers.iterator();
        while(it.hasNext()) {
            int num = it.next();
            if(num > 25) {
                it.remove();
            }
        }

        System.out.println("Numbers after removal: " + numbers);
    }
}
```

✅ 7. Sorting and Searching Collections

🔷 Direct:

1. Sort an `ArrayList` of integers in ascending and descending order.

2. Use `Collections.binarySearch()` to find an element in a sorted list.

3. Sort a list of custom objects like Employees by name using `Comparator`.

🔷 Scenario-Based:

4. You have a list of products with prices. Sort them by price and then search for a product

within a specific price range.

5. Build a leaderboard system that keeps players sorted by scores (highest first). Allow searching for a specific player's rank.

```
package Search_Sort;


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


public class BinarySearchDemo {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");
        names.add("Diana");



        Collections.sort(names);


        int index = Collections.binarySearch(names, "Charlie");
        System.out.println("'Charlie' found at index: " + index);
    }
}
```

```
package Search_Sort;


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

```
class Employee {
    String name;
    int id;

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

    @Override
    public String toString() {
        return name + " (ID: " + id + ")";
    }
}

public class EmployeeSorting {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        employees.add(new Employee("John", 101));
        employees.add(new Employee("Alice", 103));
        employees.add(new Employee("Bob", 102));



        Collections.sort(employees, Comparator.comparing(e -> e.name));
        System.out.println("Sorted by name: " + employees);
    }
}
package Search_Sort;

import java.util.ArrayList;
```

```java
import java.util.Collections;

public class IntegerSorting {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(5);
        numbers.add(2);
        numbers.add(8);
        numbers.add(1);


        Collections.sort(numbers);
        System.out.println("Ascending order: " + numbers);


        Collections.sort(numbers, Collections.reverseOrder());
        System.out.println("Descending order: " + numbers);
    }
}
package Search_Sort;

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

class Player {
    String name;
    int score;

    public Player(String name, int score) {
        this.name = name;
        this.score = score;
    }
}


public class Leaderboard {
    public static void main(String[] args) {
        ArrayList<Player> players = new ArrayList<>();
        players.add(new Player("Alice", 1500));
        players.add(new Player("Bob", 2200));
        players.add(new Player("Charlie", 1800));



        Collections.sort(players, Comparator.comparingInt(p -> -p.score));


        System.out.println("Leaderboard:");
        for (int i = 0; i < players.size(); i++) {
            System.out.println((i+1) + ". " + players.get(i).name + " - " +
players.get(i).score);
        }



        String searchName = "Charlie";
        for (int i = 0; i < players.size(); i++) {
            if (players.get(i).name.equals(searchName)) {
                System.out.println("\n" + searchName + "'s rank: " + (i+1));
                break;
            }
        }
    }
```

```
}
package Search_Sort;

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

class Product {
    String name;
    double price;

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return name + " ($" + price + ")";
    }
}

public class ProductManager {
    public static void main(String[] args) {
        ArrayList<Product> products = new ArrayList<>();
        products.add(new Product("Laptop", 999.99));
        products.add(new Product("Phone", 699.99));
        products.add(new Product("Tablet", 349.99));



        Collections.sort(products, Comparator.comparingDouble(p -> p.price));
        System.out.println("Products sorted by price: " + products);


        double minPrice = 500.0;
        double maxPrice = 1000.0;
        System.out.println("\nProducts in price range $" + minPrice + "-$" + maxPrice + ":");
        for (Product p : products) {
            if (p.price >= minPrice && p.price <= maxPrice) {
                System.out.println(p);
            }
        }
    }
}
```