✅ 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.

✅ 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`.

✅ 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.

✅ 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`.

✅ 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.

✅ 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`.

✅ 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.