Collection framework

1. Adding and Printing Elements from an ArrayList

```
import java.util.ArrayList;

public class ArrayListDemo {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();

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

        // Printing elements
        System.out.println("Fruits List:");
        for (String fruit : fruits) {
            System.out.println(fruit);
        }
    }
}
```

---

2. Using Collections.max() and Collections.min()

```
import java.util.*;

public class MaxMinDemo {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(45, 12, 67, 34, 89, 23);

        int max = Collections.max(numbers);
        int min = Collections.min(numbers);

        System.out.println("Numbers: " + numbers);
        System.out.println("Maximum: " + max);
        System.out.println("Minimum: " + min);
    }
}
```

---

3. Using Collections.sort() on a List of Strings

```
import java.util.*;

public class SortStringList {
    public static void main(String[] args) {
        List<String> cities = new ArrayList<>();
        cities.add("Mumbai");
        cities.add("Delhi");
        cities.add("Chennai");
        cities.add("Kolkata");

        Collections.sort(cities);

        System.out.println("Sorted Cities:");
        for (String city : cities) {
            System.out.println(city);
        }
    }
}
```

---

4. Scenario: Store and Sort Student Names Alphabetically

```
import java.util.*;
public class StudentNameSorter {
```

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    List<String> studentNames = new ArrayList<>();

    System.out.println("Enter student names (type 'end' to stop):");
    while (true) {
      String name = scanner.nextLine();
      if (name.equalsIgnoreCase("end")) break;
      studentNames.add(name);
    }

    Collections.sort(studentNames);
    System.out.println("\nAlphabetical Student Names:");
    for (String name : studentNames) {
      System.out.println(name);
    }
  }
}
```

---

5. Scenario: Store User Input Integers & Display Sum

```
import java.util.*;

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

    System.out.println("Enter integers (type any non-integer to finish):");
    while (scanner.hasNextInt()) {
      numbers.add(scanner.nextInt());
    }
```

```
        int sum = 0;
        for (int num : numbers) {
            sum += num;
        }

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

---

list interface

1. Add, Remove, and Access Elements in an ArrayList

```
import java.util.ArrayList;

public class ArrayListOperations {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<>();

        // Add elements
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");

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

        // Remove element
        colors.remove("Green");
        // Display updated list
        System.out.println("Updated Colors: " + colors);
    }
}
```

---

2. LinkedList to Store and Print Employee Names

```
import java.util.LinkedList;

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

        employees.add("Alice");
        employees.add("Bob");
        employees.add("Charlie");

        System.out.println("Employee List:");
        for (String emp : employees) {
            System.out.println(emp);
        }
    }
}
```

---

3. Insert Element at Specific Position in a List

```
import java.util.ArrayList;

public class InsertInList {
    public static void main(String[] args) {
        ArrayList<String> languages = new ArrayList<>();
        languages.add("Java");
        languages.add("Python");
        languages.add("C++");

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

        System.out.println("Languages List: " + languages);
    }
}
```

---

4. Scenario: To-Do List Manager

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

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

        do {
            System.out.println("\n1. Add Task\n2. Remove Task\n3. Show Tasks\n4. Exit");
            choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter task: ");
                    String task = scanner.nextLine();
                    tasks.add(task);
                    break;
                case 2:
                    System.out.print("Enter task to remove: ");
                    String removeTask = scanner.nextLine();
                    tasks.remove(removeTask);
                    break;
                case 3:
                    System.out.println("Pending Tasks:");
                    for (String t : tasks) {
                        System.out.println("- " + t);
                    }
                    break;
            }
        } while (choice != 4);
    }
}
```

---

5. Scenario: Simple Shopping Cart System

```
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);
        int choice;

        do {
            System.out.println("\n1. Add Product\n2. Remove Product\n3. View Cart\n4. Exit");
            choice = scanner.nextInt();
            scanner.nextLine(); // consume newline
            switch (choice) {
                case 1:
                    System.out.print("Enter product to add: ");
                    String product = scanner.nextLine();
                    cart.add(product);
                    break;
                case 2:
                    System.out.print("Enter product to remove: ");
                    String removeProduct = scanner.nextLine();
                    cart.remove(removeProduct);
                    break;
                case 3:
                    System.out.println("Shopping Cart:");
                    for (String item : cart) {
                        System.out.println("- " + item);
                    }
                    break;
            }
        } while (choice != 4);
    }
}
```

Set interface

1. HashSet to Store Unique Student Roll Numbers

```
import java.util.HashSet;

public class StudentRollNumbers {
    public static void main(String[] args) {
        HashSet<Integer> rollNumbers = new HashSet<>();
        rollNumbers.add(101);
        rollNumbers.add(102);
        rollNumbers.add(103);
        rollNumbers.add(101); // duplicate

        System.out.println("Unique Student Roll Numbers:");
        for (int roll : rollNumbers) {
            System.out.println(roll);
        }
    }
}
```

---

2. Tree Set to Automatically Sort Elements

```
import java.util.TreeSet;

public class SortedNames {
    public static void main(String[] args) {
        TreeSet<String> names = new TreeSet<>();

        names.add("Zara");
        names.add("Amit");
        names.add("John");
        names.add("Bella");

        System.out.println("Sorted Names:");
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

---

3. LinkedHashSet to Maintain Insertion Order and Prevent Duplicates

```
import java.util.LinkedHashSet;

public class OrderedUniqueItems {
    public static void main(String[] args) {
        LinkedHashSet<String> items = new LinkedHashSet<>();

        items.add("Pen");
        items.add("Pencil");
        items.add("Eraser");
        items.add("Pen"); // duplicate

        System.out.println("Items in Insertion Order:");
        for (String item : items) {
            System.out.println(item);
        }
    }
}
```

---

4. Scenario: Store Unique Email IDs (No Duplicates)

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

public class EmailRegistry {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashSet<String> emailSet = new HashSet<>();

        System.out.println("Enter email IDs (type 'stop' to finish):");
        while (true) {
            String email = scanner.nextLine();
            if (email.equalsIgnoreCase("stop")) break;
            if (!emailSet.add(email)) {
                System.out.println("Duplicate email ignored: " + email);
            }
        }

        System.out.println("\nRegistered Email IDs:");
        for (String email : emailSet) {
            System.out.println(email);
        }
    }
}
```

---

5. Scenario: Eliminate Duplicate City Names from User Input

```
import java.util.*;

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

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

        // Eliminate duplicates using a Set
        Set<String> uniqueCities = new HashSet<>(cityList);
        System.out.println("\nUnique City Names:");
        for (String city : uniqueCities) {
            System.out.println(city);
        }
    }
}
```

Map Interface

1. HashMap to Store Student Names and Their Marks

```
import java.util.HashMap;

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

        marksMap.put("Alice", 85);
        marksMap.put("Bob", 92);
        marksMap.put("Charlie", 78);

        System.out.println("Student Marks:");
        for (String name : marksMap.keySet()) {
            System.out.println(name + ": " + marksMap.get(name));
        }
    }
}
```

---

2. Iterate over a Map using entrySet()

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

```
public class MapIterationDemo {
    public static void main(String[] args) {
        HashMap<String, String> countryCapital = new HashMap<>();

        countryCapital.put("India", "New Delhi");
        countryCapital.put("USA", "Washington D.C.");
        countryCapital.put("France", "Paris");

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

---

3. Update the Value Associated with a Key in a Map

```
import java.util.HashMap;

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

        stock.put("Apples", 50);
        stock.put("Bananas", 30);

        // Update stock of apples
        stock.put("Apples", 75);  // updated from 50 to 75

        System.out.println("Updated Stock:");
        for (String item : stock.keySet()) {
            System.out.println(item + ": " + stock.get(item));
        }
    }
}
```

---

4. Scenario: Phone Directory (Name → Phone Number)

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

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

        System.out.println("Enter name and phone number (type 'stop' to end):");
        while (true) {
            System.out.print("Name: ");
            String name = scanner.nextLine();
            if (name.equalsIgnoreCase("stop")) break;

            System.out.print("Phone Number: ");
            String number = scanner.nextLine();
            phoneBook.put(name, number);
        }

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

5. Scenario: Word Frequency Counter Using a Map

```
import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;


public class WordFrequencyCounter {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a sentence:");


        String sentence = scanner.nextLine().toLowerCase();

        String[] words = sentence.split("\\s+");


        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());

        }

    }

}
```

Queue Interface

1. Task Queue using LinkedList as a Queue

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

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

        tasks.add("Email check");
        tasks.add("Team meeting");
        tasks.add("Code review");

        System.out.println("Tasks in Queue:");
        for (String task : tasks) {
            System.out.println(task);
        }
    }
}
```

---

2. Add and Remove Elements Using offer() and poll()

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

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

        queue.offer("Task 1");
        queue.offer("Task 2");
        queue.offer("Task 3");

        System.out.println("Queue before poll: " + queue);
        String removed = queue.poll();  // removes the head
        System.out.println("Removed: " + removed);
        System.out.println("Queue after poll: " + queue);
    }
}
```

---

3. PriorityQueue to Order Tasks by Priority (Lower number = higher priority)

```
import java.util.PriorityQueue;

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

        taskQueue.add(5); // Low priority
        taskQueue.add(1); // High priority
        taskQueue.add(3);

        System.out.println("Processing Tasks by Priority:");
        while (!taskQueue.isEmpty()) {
            System.out.println("Processing task with priority: " + taskQueue.poll());
        }
    }
}
```

---

4. Scenario: Simulate a Print Queue System

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

public class PrintQueueSimulator {
    public static void main(String[] args) {
        Queue<String> printQueue = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter print jobs (type 'done' to stop):");
        while (true) {
            String job = scanner.nextLine();
            if (job.equalsIgnoreCase("done")) break;
            printQueue.offer(job);
        }

        System.out.println("\nProcessing Print Jobs:");
        while (!printQueue.isEmpty()) {
            System.out.println("Printing: " + printQueue.poll());
        }
    }
}
```

---

5. Scenario: Ticket Booking System (Customer Queue)

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

public class TicketBookingSystem {
    public static void main(String[] args) {
        Queue<String> customerQueue = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter customer names (type 'end' to finish):");
        while (true) {
            String name = scanner.nextLine();
            if (name.equalsIgnoreCase("end")) break;
            customerQueue.offer(name);
        }

        System.out.println("\nServing Customers:");
        while (!customerQueue.isEmpty()) {
            System.out.println("Serving: " + customerQueue.poll());
        }
    }
}
```

Iterator Interface

1. Iterate Through a List Using Iterator

```
import java.util.*;

public class IteratorExample {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Alice", "Bob", "Charlie");

        Iterator<String> iterator = names.iterator();
        System.out.println("Iterating using Iterator:");
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

---

2. Remove an Element While Iterating Using Iterator

```
import java.util.*;
```

```
public class RemoveWhileIterating {
    public static void main(String[] args) {
        List<String> items = new ArrayList<>(Arrays.asList("Apple", "Banana", "Mango", "Banana"));

        Iterator<String> iterator = items.iterator();
        while (iterator.hasNext()) {
            if (iterator.next().equals("Banana")) {
                iterator.remove(); // Safe removal
            }
        }

        System.out.println("After removing 'Banana': " + items);
    }
}
```

---

3. Use ListIterator to Iterate in Both Directions

```
import java.util.*;

public class ListIteratorDemo {
    public static void main(String[] args) {
        List<String> languages = Arrays.asList("Java", "Python", "C++");

        ListIterator<String> listIterator = languages.listIterator();

        System.out.println("Forward Iteration:");
        while (listIterator.hasNext()) {
            System.out.println(listIterator.next());
        }

        System.out.println("Backward Iteration:");
        while (listIterator.hasPrevious()) {
```

```
        System.out.println(listIterator.previous());
      }
    }
}
```

---

4. Scenario: Remove Book Titles Starting With a Specific Letter

```
import java.util.*;

public class RemoveBooksByLetter {
    public static void main(String[] args) {
      List<String> books = new ArrayList<>(Arrays.asList(
        "Harry Potter", "Hobbit", "C Programming", "Head First Java", "Atomic Habits"));

      char letter = 'H';
      Iterator<String> iterator = books.iterator();

      while (iterator.hasNext()) {
        String book = iterator.next();
        if (book.startsWith(String.valueOf(letter))) {
          iterator.remove();
        }
      }

      System.out.println("Books after removing titles starting with '" + letter + "':");
      for (String book : books) {
        System.out.println(book);
      }
    }
}
```

---

5. Scenario: Reverse a List Using ListIterator

```
import java.util.*;

public class ReverseListUsingListIterator {
    public static void main(String[] args) {
        List<String> cities = new ArrayList<>(Arrays.asList("Delhi", "Mumbai", "Chennai", "Kolkata"));

        ListIterator<String> iterator = cities.listIterator(cities.size());
        List<String> reversed = new ArrayList<>();

        while (iterator.hasPrevious()) {
            reversed.add(iterator.previous());
        }

        System.out.println("Original List: " + cities);
        System.out.println("Reversed List: " + reversed);
    }
}
```

Sorting and Searching Collections

1. Sort an ArrayList of Integers (Ascending & Descending)

```
import java.util.*;

public class SortIntegers {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>(Arrays.asList(40, 10, 70, 30, 50));
        Collections.sort(numbers); // Ascending
        System.out.println("Ascending: " + numbers);

        Collections.sort(numbers, Collections.reverseOrder()); // Descending
        System.out.println("Descending: " + numbers);
    }
}
```

---

2. Use Collections.binarySearch() in a Sorted List

```
import java.util.*;

public class BinarySearchExample {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>(Arrays.asList("Alice", "Bob", "Charlie", "David"));
        Collections.sort(names); // Required for binarySearch

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

---

3. Sort a List of Employees by Name Using Comparator

```
import java.util.*;

class Employee {
    String name;
    int id;

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

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

public class SortEmployees {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("John", 101));
        employees.add(new Employee("Alice", 105));
        employees.add(new Employee("Bob", 102));

        employees.sort(Comparator.comparing(emp -> emp.name));

        System.out.println("Employees Sorted by Name:");
        for (Employee e : employees) {
            System.out.println(e);
        }
    }
}
```

Scenario-Based Use Cases

---

4. Sort Products by Price and Search in Price Range

```
import java.util.*;

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

    public String toString() {
        return name + " - ₹" + price;
    }
}

public class ProductSearch {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>();
        products.add(new Product("Pen", 10));
        products.add(new Product("Notebook", 50));
        products.add(new Product("Pencil", 5));
        products.add(new Product("Eraser", 7));

        products.sort(Comparator.comparingDouble(p -> p.price));

        System.out.println("Sorted Products by Price:");
        for (Product p : products) {
            System.out.println(p);
        }

        System.out.println("\nProducts within ₹6 to ₹20:");
        for (Product p : products) {
            if (p.price >= 6 && p.price <= 20) {
                System.out.println(p);
            }
        }
    }
}
```

5. Leaderboard System Sorted by Scores (Highest First) + Rank Search

```
import java.util.*;

class Player {
    String name;
    int score;

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

    public String toString() {
        return name + " - " + score;
    }
}

public class Leaderboard {
    public static void main(String[] args) {
        List<Player> players = new ArrayList<>();
        players.add(new Player("Ravi", 150));
        players.add(new Player("Sneha", 200));
        players.add(new Player("Arun", 180));
        players.add(new Player("Meena", 170));

        // Sort by score descending
        players.sort((p1, p2) -> p2.score - p1.score);
        System.out.println("Leaderboard:");
        for (int i = 0; i < players.size(); i++) {
            System.out.println((i + 1) + ". " + players.get(i));
        }

        // Search for player rank
        String searchName = "Meena";
        int rank = -1;
        for (int i = 0; i < players.size(); i++) {
            if (players.get(i).name.equalsIgnoreCase(searchName)) {
                rank = i + 1;
                break;
            }
        }

        if (rank != -1) {
            System.out.println("\n" + searchName + "'s Rank: " + rank);
        } else {
            System.out.println("\nPlayer not found.");
        }
    }
}
```