Collection Coding Challenge

1. Introduction to Collections Framework

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

```
package collection_Coding_Challenge;

import java.util.ArrayList;

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

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

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

```
package collection_Coding_Challenge;
```

```
import java.util.*;

public class MinMax {

        public static void main(String[] args) {

          List<Integer> numbers = Arrays.asList(34, 67, 12, 89, 2);


          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);

        }

      }
```

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

```
package collection_Coding_Challenge;

import java.util.*;


public class sort {

  public static void main(String[] args) {

    List<String> names = new ArrayList<>();

    names.add("Charlie");

    names.add("Alice");

    names.add("Bob");


    Collections.sort(names);
```

```
        System.out.println("Sorted Names: " + names);
    }
}
```

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

```
package collection_Coding_Challenge;
import java.util.*;

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

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

        System.out.println("Students in alphabetical order:");
        for (String student : students) {
            System.out.println(student);
        }
    }
}
```

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 collection_Coding_Challenge;
import java.util.*;

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

        System.out.println("Enter numbers (type -1 to stop):");
        while (true) {
            int num = scanner.nextInt();
            if (num == -1) break;
            numbers.add(num);
        }
        scanner.close();
        int sum = 0;
        for (int number : numbers) {
            sum += number;
        }

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

2. List Interface

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

```
package list_Interface;
import java.util.*;

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

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

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

        // Remove element
        colors.remove("Blue");

        System.out.println("Final List: " + colors);
    }
}
```

Implement a LinkedList that stores and prints employee names.

```
package list_Interface;
import java.util.*;

public class EmployeeList {
    public static void main(String[] args) {
        List<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);
        }
    }
}
```

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

```
package list_Interface;
import java.util.*;

public class InsertAtPosition {
    public static void main(String[] args) {
        List<String> languages = new ArrayList<>(Arrays.asList("Java", "Python", "C++"));
        languages.add(1, "JavaScript"); // Insert at index 1
        System.out.println("Languages: " + languages);
    }
}
```

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

```
package list_Interface;
import java.util.*;

public class ToDoList {
    public static void main(String[] args) {
        List<String> tasks = new ArrayList<>();

        try (Scanner scanner = new Scanner(System.in)) {
            while (true) {
                System.out.println("\n1. Add Task\n2. Remove Task\n3. View Tasks\n4. Exit");
                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:");
                        for (String task : tasks) {
                            System.out.println("- " + task);
                        }
                        break;
                    case 4:
                        System.out.println("Exiting To-Do List Manager.");
                        return;
                    default:
                        System.out.println("Invalid choice.");
                }
            }
        }
    }
}
```

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

```
package list_Interface;
import java.util.*;

public class ShoppingCart {
    public static void main(String[] args) {
        List<String> cart = new ArrayList<>();
```

```
try (Scanner scanner = new Scanner(System.in)) {
    while (true) {
        System.out.println("\n1. Add Product\n2. Remove Product\n3. View Cart\n4. Exit");
        int choice = scanner.nextInt();
        scanner.nextLine(); // consume newline

        switch (choice) {
            case 1:
                System.out.print("Enter product to add: ");
                cart.add(scanner.nextLine());
                break;
            case 2:
                System.out.print("Enter product to remove: ");
                cart.remove(scanner.nextLine());
                break;
            case 3:
                System.out.println("Your Cart:");
                for (String item : cart) {
                    System.out.println("- " + item);
                }
                break;
            case 4:
                System.out.println("Thank you for shopping!");
                return;
            default:
                System.out.println("Invalid option.");
        }
```

```
            }
        }
    }
}
```

3. Set Interface

Write a program using HashSet to store unique student roll numbers.

```
package SetInterface;
import java.util.*;

public class UniqueRollNumbers {
    public static void main(String[] args) {
        Set<Integer> rollNumbers = new HashSet<>();
        rollNumbers.add(101);
        rollNumbers.add(102);
        rollNumbers.add(103);
        rollNumbers.add(101); // Duplicate, will be ignored

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

Demonstrate how to use TreeSet to automatically sort elements.

```
package SetInterface;
import java.util.*;

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

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

Use LinkedHashSet to maintain insertion order and prevent duplicates

```
package SetInterface;
import java.util.*;

public class LinkedHashSetDemo {
    public static void main(String[] args) {
        Set<String> subjects = new LinkedHashSet<>();
        subjects.add("Math");
        subjects.add("Science");
        subjects.add("English");
        subjects.add("Math"); // Duplicate, will be ignored

        System.out.println("Subjects in insertion order:");
        for (String subject : subjects) {
            System.out.println(subject);
        }
    }
}
```

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

```
package SetInterface;
import java.util.*;

public class EmailRegistry {
    public static void main(String[] args) {
        Set<String> emails = new HashSet<>();
        try (Scanner scanner = new Scanner(System.in)) {
            while (true) {
                System.out.print("Enter email to register (or type 'exit' to stop): ");
                String email = scanner.nextLine();
                if (email.equalsIgnoreCase("exit")) break;
                if (!emails.add(email)) {
                    System.out.println("Email already registered!");
                } else {
                    System.out.println("Email registered successfully.");
                }
            }
        }

        System.out.println("Registered Email IDs:");
        for (String email : emails) {
            System.out.println(email);
        }
    }
}
```

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

```
package SetInterface;
import java.util.*;

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

        Set<String> uniqueCities = new HashSet<>(cities);

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

4. Map Interface

Write a program using HashMap to store student names and their marks.

```
package map_Interface;

import java.util.*;

public class StudentMarks {
    public static void main(String[] args) {
        Map<String, Integer> studentMarks = new HashMap<>();
        studentMarks.put("Alice", 85);
        studentMarks.put("Bob", 92);
        studentMarks.put("Charlie", 78);

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

Demonstrate how to iterate over a Map using entrySet().

```
package map_Interface;

import java.util.*;
```

```
public class MapEntrySetIteration {
    public static void main(String[] args) {
        Map<String, String> countryCapital = new HashMap<>();
        countryCapital.put("India", "New Delhi");
        countryCapital.put("USA", "Washington D.C.");
        countryCapital.put("UK", "London");

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

Show how to update the value associated with a key in a Map.

```
package map_Interface;

import java.util.*;

public class UpdateMapValue {
    public static void main(String[] args) {
        Map<String, Integer> inventory = new HashMap<>();
        inventory.put("Apples", 50);
        inventory.put("Oranges", 30);

        // Update value
        inventory.put("Apples", inventory.get("Apples") + 20);


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

Build a phone directory where names are keys and phone numbers are values.

```
package map_Interface;


import java.util.*;


public class PhoneDirectory {
    public static void main(String[] args) {
        Map<String, String> phoneBook = new HashMap<>();
        try (Scanner scanner = new Scanner(System.in)) {
            while (true) {
                System.out.print("Enter name (or 'exit' to stop): ");
                String name = scanner.nextLine();
                if (name.equalsIgnoreCase("exit")) break;


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

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

Create a frequency counter for words in a sentence using a Map.

```
package map_Interface;

import java.util.*;

public class WordFrequencyCounter {
    public static void main(String[] args) {
        String sentence = "this is a test this is only a test";
        String[] words = sentence.split(" ");

        Map<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

Implement a simple task queue using LinkedList as a Queue.

```
package queue_Interface;

import java.util.*;

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

        taskQueue.add("Task 1 - Code");
        taskQueue.add("Task 2 - Review");
        taskQueue.add("Task 3 - Test");

        System.out.println("Task Queue:");
        while (!taskQueue.isEmpty()) {
            System.out.println("Processing: " + taskQueue.poll());
        }
    }
}
```

Demonstrate how to add and remove elements using offer() and poll().

```
package queue_Interface;


import java.util.*;


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


        queue.offer("A");
        queue.offer("B");
        queue.offer("C");


        System.out.println("Queue Elements:");
        while (!queue.isEmpty()) {
            System.out.println("Polled: " + queue.poll());
        }
    }
}
```

Use a PriorityQueue to order tasks by priority (integers).

```
package queue_Interface;


import java.util.*;


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

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

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

Simulate a print queue system where print jobs are processed in order.

```
package queue_Interface;

import java.util.*;

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

        printQueue.add("Document1.pdf");
        printQueue.add("Report.docx");
        printQueue.add("Invoice.xlsx");

        System.out.println("Starting print job...");
        while (!printQueue.isEmpty()) {
            System.out.println("Printing: " + printQueue.poll());
        }
    }
}
```

Create a ticket booking system where customer names are added to a queue and served in order.

```
package queue_Interface;

import java.util.*;

public class TicketBookingSystem {
    public static void main(String[] args) {
        Queue<String> customerQueue = new LinkedList<>();
        try (Scanner scanner = new Scanner(System.in)) {
            while (true) {
                System.out.print("Enter customer name (or 'exit' to stop): ");
                String name = scanner.nextLine();
                if (name.equalsIgnoreCase("exit")) break;
                customerQueue.offer(name);
            }
        }

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

6. Iterator Interface

Write a program to iterate through a list using Iterator.

```
package iterator_interface;

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();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Demonstrate removing an element from a list while iterating using Iterator.

```
package iterator_interface;

import java.util.*;

public class RemoveBooksByLetter {
    public static void main(String[] args) {
        List<String> books = new ArrayList<>(Arrays.asList("Harry Potter", "Alice in Wonderland", "Hamlet", "Great Gatsby"));

        char letter = 'H';
        Iterator<String> iterator = books.iterator();
        while (iterator.hasNext()) {
            String title = iterator.next();
            if (title.startsWith(String.valueOf(letter))) {
                iterator.remove();
            }
        }

        System.out.println("Books after removal: " + books);
    }
}
```

Show how to use ListIterator to iterate in both directions.

```
package iterator_interface;

import java.util.*;

public class ListIteratorDemo {
    public static void main(String[] args) {
        List<String> animals = Arrays.asList("Cat", "Dog", "Elephant");

        ListIterator<String> listIterator = animals.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());
        }
    }
}
```

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

```
package iterator_interface;
import java.util.*;

public class IteratorInterfaceScenario {
    public static void main(String[] args) {
        List<String> books = new ArrayList<>(Arrays.asList(
            "Harry Potter", "Alice in Wonderland", "Hamlet", "Great Gatsby", "Hobbit"
        ));

        char startingLetter = 'H';
        Iterator<String> iterator = books.iterator();
        while (iterator.hasNext()) {
            String title = iterator.next();
            if (title.startsWith(String.valueOf(startingLetter))) {
                iterator.remove();
            }
        }

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

        ListIterator<String> listIterator = books.listIterator(books.size());
        System.out.println("\nBooks in reverse order:");
        while (listIterator.hasPrevious()) {
            System.out.println(listIterator.previous());
        }
    }
}
```

7. Sorting and Searching Collections

Sort an ArrayList of integers in ascending and descending order.

```
package sortingSearchingCollections;

import java.util.*;

public class SortIntegers {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(42, 15, 8, 23, 4);
        // Ascending
        List<Integer> ascending = new ArrayList<>(numbers);
        Collections.sort(ascending);
        System.out.println("Ascending: " + ascending);

        // Descending
        List<Integer> descending = new ArrayList<>(numbers);
        descending.sort(Collections.reverseOrder());
        System.out.println("Descending: " + descending);
    }
}
```

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

```
package sortingSearchingCollections;

import java.util.*;

public class BinarySearch {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>(Arrays.asList("Alice", "Bob", "Charlie", "David"));
        Collections.sort(names); // Binary search requires sorted list
        System.out.println("Sorted Names: " + names);

        int index = Collections.binarySearch(names, "Charlie");
        if (index >= 0) {
            System.out.println("Charlie found at index: " + index);
        } else {
            System.out.println("Charlie not found.");
        }
    }
}
```

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

```
package sortingSearchingCollections;

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("Ravi", 102));
        employees.add(new Employee("Anita", 101));
        employees.add(new Employee("Kiran", 103));

        // Sort by name
        employees.sort(Comparator.comparing(e -> e.name));
        System.out.println("Employees sorted by name:");
        for (Employee e : employees) {
            System.out.println(e);
        }
    }
}
```

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

```
package sortingSearchingCollections;

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;
```

```java
    }
}

public class ProductSorter {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>();
        products.add(new Product("Laptop", 55000));
        products.add(new Product("Phone", 20000));
        products.add(new Product("Monitor", 15000));
        products.add(new Product("Tablet", 30000));

        // Sort by price (ascending)
        products.sort(Comparator.comparingDouble(p -> p.price));

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

        // Search for products in a specific price range
        double min = 15000, max = 30000;
        System.out.println("\nProducts in price range ₹" + min + " - ₹" + max + ":");
        for (Product p : products) {
            if (p.price >= min && p.price <= max) {
                System.out.println(p);
            }
        }
    }
```

```
}
```

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

```
package sortingSearchingCollections;


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("Alice", 120));
        players.add(new Player("Bob", 150));
        players.add(new Player("Charlie", 100));
        players.add(new Player("David", 180));

        // Sort players by score -descending
        players.sort((p1, p2) -> p2.score - p1.score);

        System.out.println("Leaderboard (Highest Score First):");
        int rank = 1;
        for (Player p : players) {
            System.out.println("Rank " + rank++ + ": " + p);
        }

        // Search for a player's rank
        String searchName = "Bob";
        for (int i = 0; i < players.size(); i++) {
            if (players.get(i).name.equalsIgnoreCase(searchName)) {
                System.out.println("\n" + searchName + "'s Rank: " + (i + 1));
                break;
            }
        }
    }
}
```