1. Introduction to Collections Framework

1. Add and print elements from an ArrayList:

```
import java.util.ArrayList;

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

        for (String fruit : list) {
            System.out.println(fruit);
        }
    }
}
```

2. Use Collections.max() and Collections.min():

```
import java.util.*;

public class MaxMinExample {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(12, 45, 67, 2, 89);
        System.out.println("Max: " + Collections.max(numbers));
        System.out.println("Min: " + Collections.min(numbers));
    }
}
```

3. Use Collections.sort() on a list of strings:

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

```
public class SortStrings {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Zara", "Anna", "Mike");
        Collections.sort(names);
        System.out.println(names);
    }
}
```

4. Student names in alphabetical order:

```
import java.util.*;

public class StudentSort {
    public static void main(String[] args) {
        List<String> students = new ArrayList<>();
        students.add("Ravi");
        students.add("Aman");
        students.add("Bhanu");
        Collections.sort(students);
        System.out.println(students);
    }
}
```

5. Sum of all user-input integers:

```
import java.util.*;

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

```
        System.out.println("Enter numbers (-1 to stop):");
        while (true) {
            int num = sc.nextInt();
            if (num == -1) break;
            numbers.add(num);
        }

        int sum = 0;
        for (int n : numbers) sum += n;
        System.out.println("Sum: " + sum);
    }
}
```

2. **List Interface**

1. Add, remove, and access elements in ArrayList:

```
import java.util.*;

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

        list.remove("Python");
        System.out.println("Element at index 1: " + list.get(1));

        for (String lang : list) {
            System.out.println(lang);
        }
    }
```

```
}
```

2. LinkedList for employee names:

```
import java.util.*;

public class EmployeeLinkedList {
    public static void main(String[] args) {
        LinkedList<String> employees = new LinkedList<>();
        employees.add("Alice");
        employees.add("Bob");
        employees.add("Charlie");

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

3. Insert element at specific position:

```
import java.util.*;

public class InsertElement {
    public static void main(String[] args) {
        List<String> colors = new ArrayList<>(Arrays.asList("Red", "Green", "Blue"));
        colors.add(1, "Yellow");
        System.out.println(colors);
    }
}
```

Scenario-Based:

4. To-do list manager:

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

```
public class TodoList {
    public static void main(String[] args) {
        List<String> tasks = new ArrayList<>();
        tasks.add("Buy groceries");
        tasks.add("Clean room");
        tasks.add("Pay bills");

        tasks.remove("Clean room");  // Completed
        System.out.println("Pending Tasks:");
        for (String task : tasks) {
            System.out.println(task);
        }
    }
}
```

5. Shopping cart system:

```
import java.util.*;

public class ShoppingCart {
    public static void main(String[] args) {
        List<String> cart = new ArrayList<>();
        cart.add("Shirt");
        cart.add("Shoes");
        cart.remove("Shirt");
        System.out.println("Products in Cart: " + cart);
    }
}
```

3. **Set Interface**

1. HashSet for student roll numbers:

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

```
public class UniqueRolls {
    public static void main(String[] args) {
        Set<Integer> rollNumbers = new HashSet<>();
        rollNumbers.add(101);
        rollNumbers.add(102);
        rollNumbers.add(101);  // Duplicate ignored
        System.out.println(rollNumbers);
    }
}
```

2. TreeSet to sort:

```
import java.util.*;

public class SortedSet {
    public static void main(String[] args) {
        Set<String> names = new TreeSet<>();
        names.add("Zoe");
        names.add("Alex");
        names.add("Ben");
        System.out.println(names);
    }
}
```

3. LinkedHashSet to maintain order:

```
import java.util.*;

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

```
        cities.add("Mumbai");
        cities.add("Delhi");
        cities.add("Bangalore");
        cities.add("Delhi");
        System.out.println(cities);
    }
}
```

Scenario-Based:

4. Email registration system:

```
import java.util.*;

public class EmailRegister {
    public static void main(String[] args) {
        Set<String> emails = new HashSet<>();
        emails.add("user1@gmail.com");
        emails.add("user2@gmail.com");
        emails.add("user1@gmail.com");
        System.out.println(emails);
    }
}
```

5. Eliminate duplicate city names:

```
import java.util.*;

public class CityFilter {
    public static void main(String[] args) {
        List<String> inputCities = Arrays.asList("Pune", "Mumbai", "Pune", "Delhi");
        Set<String> uniqueCities = new HashSet<>(inputCities);
        System.out.println(uniqueCities);
    }
}
```

4. **Map Interface**

1. HashMap with student names and marks:

```
import java.util.*;

public class StudentMarks {
    public static void main(String[] args) {
        Map<String, Integer> marks = new HashMap<>();
        marks.put("Ravi", 90);
        marks.put("Sneha", 85);
        System.out.println(marks);
    }
}
```

2. Iterate using entrySet():

```
import java.util.*;

public class IterateMap {
    public static void main(String[] args) {
        Map<String, String> countries = Map.of("IN", "India", "US", "USA");

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

3.Update value by key:

```
import java.util.*;

public class UpdateMap {
    public static void main(String[] args) {
        Map<String, Integer> map = new HashMap<>();
        map.put("A", 100);
        map.put("A", 200);  // Overwrites old value
```

```
        System.out.println(map);
    }
}
```

Scenario-Based:

4. Phone directory:

```
import java.util.*;

public class PhoneDirectory {
    public static void main(String[] args) {
        Map<String, String> directory = new HashMap<>();
        directory.put("Alice", "1234567890");
        directory.put("Bob", "9876543210");

        System.out.println("Phone Number of Alice: " + directory.get("Alice"));
    }
}
```

5. Word frequency counter:

```
import java.util.*;

public class WordCount {
    public static void main(String[] args) {
        String sentence = "this is a test this is only a test";
        String[] words = sentence.split(" ");
        Map<String, Integer> freq = new HashMap<>();

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

        System.out.println(freq);
    }
```

```
}
```

5. **Queue Interface**

1. Simple task queue using LinkedList:

```
import java.util.*;

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");

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

2. Add/remove with offer() and poll():

```
import java.util.*;

public class OfferPollExample {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<>();
        queue.offer("First");
        queue.offer("Second");

        System.out.println("Removed: " + queue.poll());
        System.out.println("Removed: " + queue.poll());
    }
```

```
}
```

3. Use PriorityQueue:

```
import java.util.*;

public class PriorityQueueExample {
    public static void main(String[] args) {
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pq.add(20);
        pq.add(10);
        pq.add(30);

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

Scenario-Based:

4. Print queue system:

```
import java.util.*;

public class PrintQueue {
    public static void main(String[] args) {
        Queue<String> printJobs = new LinkedList<>();
        printJobs.add("Document1.pdf");
        printJobs.add("Document2.docx");

        while (!printJobs.isEmpty()) {
            System.out.println("Printing: " + printJobs.poll());
        }
    }
```

```
}
```

5. Ticket booking system:

```
import java.util.*;

public class TicketQueue {
    public static void main(String[] args) {
        Queue<String> customers = new LinkedList<>();
        customers.add("Alice");
        customers.add("Bob");
        customers.add("Charlie");

        while (!customers.isEmpty()) {
            System.out.println("Serving: " + customers.poll());
        }
    }
}
```

6. **Iterator Interface**

1. Iterate through list using Iterator:

```
import java.util.*;

public class IteratorExample {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("A", "B", "C");
        Iterator<String> it = list.iterator();

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

2. Remove element while iterating:

```
import java.util.*;

public class RemoveWithIterator {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>(Arrays.asList("One", "Two", "Three"));
        Iterator<String> it = list.iterator();

        while (it.hasNext()) {
            if (it.next().equals("Two")) {
                it.remove();
            }
        }

        System.out.println(list);
    }
}
```

3. Use ListIterator in both directions:

```
import java.util.*;

public class ListIteratorExample {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("A", "B", "C");
        ListIterator<String> it = list.listIterator();

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

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

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

Scenario-Based:

4. Remove books starting with specific letter:

```
import java.util.*;

public class BookFilter {
    public static void main(String[] args) {
        List<String> books = new ArrayList<>(Arrays.asList("Harry Potter", "Alchemist", "Hunger Games"));
        Iterator<String> it = books.iterator();

        while (it.hasNext()) {
            if (it.next().startsWith("H")) {
                it.remove();
            }
        }

        System.out.println(books);
    }
}
```

5. Reverse list using ListIterator:

```
import java.util.*;

public class ReverseList {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("One", "Two", "Three");
        ListIterator<String> it = list.listIterator(list.size());
```

```
        while (it.hasPrevious()) {
            System.out.println(it.previous());
        }
    }
}
```

7. **Sorting and Searching Collections**

1. Sort integers ascending and descending:

```
import java.util.*;

public class SortIntegers {
    public static void main(String[] args) {
        List<Integer> nums = new ArrayList<>(Arrays.asList(3, 1, 4, 2));
        Collections.sort(nums);
        System.out.println("Ascending: " + nums);

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

2. Collections.binarySearch():

```
import java.util.*;

public class BinarySearchExample {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>(Arrays.asList("Apple", "Banana", "Mango"));
        Collections.sort(list);
        int index = Collections.binarySearch(list, "Mango");
        System.out.println("Index: " + index);
    }
}
```

3. Sort custom objects using Comparator:

```
import java.util.*;

class Employee {
    String name;
    Employee(String name) { this.name = name; }

    public String toString() {
        return name;
    }
}

public class SortEmployee {
    public static void main(String[] args) {
        List<Employee> list = new ArrayList<>();
        list.add(new Employee("John"));
        list.add(new Employee("Alice"));
        list.add(new Employee("Bob"));

        list.sort(Comparator.comparing(e -> e.name));
        System.out.println(list);
    }
}
```

Scenario-Based:

4. Sort products by price and search:

```
import java.util.*;

class Product {
    String name;
    int price;

    Product(String name, int price) {
```

```
        this.name = name; this.price = price;
    }

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

public class ProductSortSearch {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>();
        products.add(new Product("Laptop", 50000));
        products.add(new Product("Phone", 30000));
        products.add(new Product("Tablet", 20000));

        products.sort(Comparator.comparingInt(p -> p.price));
        for (Product p : products) {
            if (p.price >= 25000 && p.price <= 40000)
                System.out.println("In range: " + p);
        }
    }
}
```

5. Leaderboard by scores, search rank:

```
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.sort((p1, p2) -> Integer.compare(p2.score, p1.score));

        for (int i = 0; i < players.size(); i++) {
            if (players.get(i).name.equals("Arun")) {
                System.out.println("Rank of Arun: " + (i + 1));
                break;
            }
        }
    }
}
```

5. Leaderboard by scores, search rank:

```
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.sort((p1, p2) -> Integer.compare(p2.score, p1.score));

        for (int i = 0; i < players.size(); i++) {
            if (players.get(i).name.equals("Arun")) {
                System.out.println("Rank of Arun: " + (i + 1));
                break;
            }
        }
    }
}
```