Java Collections:

Part 1: Working with Lists

Task 1.1: Basic ArrayList Operations

Problem:
Create an `ArrayList` of Strings, add 5 elements, remove the 2nd element, and print all elements.

Expected Output:

```
[Java, C++, Ruby, Python]
```

Task 1.2: Sorting a List

Problem:
Sort a list of integers in:

- Ascending order
- Descending order

Task 1.3: Remove Duplicates from a List

Problem:
Given a list with duplicates: `["Java", "C", "Java", "Python"]`, write code to remove duplicates using a `Set`.

Part 2: Working with Sets

Task 2.1: Set Properties

Problem:
Create a `HashSet` of strings and add duplicate values. Print the result and show that duplicates are not allowed.

Task 2.2: Sorted Set

Problem:
Use a `TreeSet` to store a list of integers and print them in sorted order automatically.

Part 3: Working with Maps

Task 3.1: Basic Map Operations

Problem:
Create a `HashMap<Integer, String>` of student roll numbers and names. Add 5 entries, remove one, and iterate using `entrySet()`.

Task 3.2: Frequency Counter

Problem:
Count the frequency of each character in a given string using a `HashMap`.

```
Input: "hello"
Output: {h=1, e=1, l=2, o=1}
```

Task 3.3: Sort a Map by Keys

Problem:
Create a `TreeMap` from an unsorted `HashMap` to sort entries by keys.

Part 4: Iterators

Task 4.1: Use an Iterator

Problem:
Use an `Iterator` to traverse an `ArrayList` and remove elements starting with letter `A`.

Task 4.2: ListIterator for Bi-Directional Traversal

Problem:
Use `ListIterator` to traverse a list forward and backward.

Part 5: Advanced Scenarios

Task 5.1: Custom Object Sorting (Comparable)

Problem:
Create a `Student` class with `name` and `marks`. Implement `Comparable<Student>` to sort by marks in ascending order.

Task 5.2: Custom Comparator

Problem:
Sort a list of students by name (alphabetically) using a `Comparator`.

Task 5.3: Grouping Elements by Property

Problem:
Given a list of employee objects (name, department), group employees by department using `Map<String, List<Employee>>`.

Task 5.4: Convert List to Map

Problem:
Convert a `List<Student>` to a `Map<String, Student>` using student ID as the key.

Task 5.5: Top N Elements

Problem:
Find the top 3 highest scoring students using a `PriorityQueue`.

Part 6: Streams with Collections

Task 6.1: Filter Elements

Problem:
Use Stream API to filter all strings starting with "J" in a list.

Task 6.2: Map and Collect

Problem:
Given a list of numbers, square each number and return a new list.

Task 6.3: Stream Grouping

Problem:
Group a list of `Employee` objects by their department using `Collectors.groupingBy()`.

Part 7: Utility Class

Task 7.1: Reverse a List

Problem:
Use `Collections.reverse()` to reverse an `ArrayList`.

Task 7.2: Frequency of Elements

Problem:
Use `Collections.frequency()` to count how many times "Java" appears in a list.