# Java Collections

### ****Task 1: Unique Words from Text****

Write a Java program that reads a sentence and prints all the **unique words** in it.  
Use a Set to store and display the words.

### ****Task 2: Frequency Counter****

Given a word (e.g., "banana"), count how many times **each character** appears in the word.  
Use a Map<Character, Integer> for storing frequencies.

### ****Task 3: Top N Frequent Elements****

From a list of integers, find the **top 3 most frequent elements** and print them along with their counts.  
Use a Map to count and a List to sort.

### ****Task 4: Duplicate Remover****

Write a program that removes **duplicate numbers** from an ArrayList<Integer> and prints the unique values only once.

### ****Task 5: Sort Students by Marks****

Create a class Student with name and marks.  
Store 5 students in a List<Student> and **sort them in descending order** of marks.

### ****Task 6: Group Words by Length****

Given a list of words, group them into a map where the **key is word length**, and the **value is a list of words of that length**.

### ****Task 7: Preserve Insertion Order****

Take a list of words with possible duplicates.  
Store the unique words in the order they appear using a suitable Set.

### ****Task 8: Word Index Map****

From a sentence, map each word to the **first index** at which it appears.  
Use a Map<String, Integer>.

### ****Task 9: List to Sorted Set****

Convert a List<Integer> with duplicates into a **sorted set** of unique numbers.  
Use the correct Collection for sorting.

### ****Task 10: Leaderboard Ranking****

Create a Map<String, Integer> where the key is a student's name and the value is their score.  
Then display a **ranked leaderboard** sorted by score in descending order.

Absolutely! Here's another **set of 10 Java tasks** that go **deeper into Collections**, covering:

* **All core interfaces**: List, Set, Map, Queue, Deque
* **Collection utilities**: sorting, reversing, shuffling
* **Real-world logic problems**
* **Intermediate-level concepts** (custom comparator, nested maps, stack/queue simulation)

### ****Task 11: Reverse a List of Strings****

Take a List<String> of 5 elements and **print them in reverse order** using Collections API.

Hint: Use Collections.reverse() or loop from the end.

### ****Task 12: Shuffle a Deck of Cards****

Create a List<String> representing a standard 52-card deck (e.g., "2H", "3D", "KS").  
Use Collections to **shuffle the deck** and print the result.

Hint: Use nested loops or arrays to generate cards.

### ****Task 13: Find First Repeating Element****

From a list of integers, find the **first element that repeats** when traversed from left to right.

Example: [1, 2, 3, 2, 1] → Output: 2

### ****Task 14: Stack Implementation (Browser History)****

Simulate browser back/forward using two stacks.  
Add pages visited, go back, go forward.

Use Stack<String> for back/forward navigation.

### ****Task 15: Queue Simulation (Print Queue)****

Simulate a printer queue using Queue<String>.  
Each string is a print job name. Process one job at a time using .poll().

### ****Task 16: Sort Map by Values****

Given a Map<String, Integer> of product → price, sort the map by price in **ascending order** and print it.

Use List<Map.Entry<>> and a custom comparator.

### ****Task 17: Count Unique Characters Using Set****

Take a string and print how many **unique characters** it contains using a Set<Character>.

### ****Task 18: Most Frequent Word****

Given a paragraph, identify the **most frequent word** and its count using Map<String, Integer>.

### ****Task 19: Nested Map – Country and Cities****

Create a map structure where:

* Key = Country name (String)
* Value = List of Cities (List<String>)

Add 3 countries and at least 3 cities each. Print the nested structure.

### ****Task 20: Deque as Sliding Window****

Use a Deque<Integer> to implement a **sliding window** of size 3 on a list of 10 integers.  
Print the sum of each window as it slides.