# **Java Collections**

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

import java.util.\*;

public class UniqueWords {

public static void main(String[] args) {

String sentence = "Java is easy and Java is powerful";

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

Set<String> uniqueWords = new HashSet<>(Arrays.asList(words));

System.out.println("Unique words: " + uniqueWords);

}

}

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

import java.util.\*;

public class FrequencyCounter {

public static void main(String[] args) {

String input = "banana";

Map<Character, Integer> freqMap = new HashMap<>();

for (char c : input.toCharArray()) {

freqMap.put(c, freqMap.getOrDefault(c, 0) + 1);

}

System.out.println("Frequency: " + freqMap);

}

}

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

import java.util.\*;

public class TopFrequent {

public static void main(String[] args) {

List<Integer> nums = Arrays.asList(1, 2, 2, 3, 3, 3, 4);

Map<Integer, Integer> freq = new HashMap<>();

for (int n : nums) {

freq.put(n, freq.getOrDefault(n, 0) + 1);

}

List<Map.Entry<Integer, Integer>> sorted = new ArrayList<>(freq.entrySet());

sorted.sort((a, b) -> b.getValue() - a.getValue());

System.out.println("Top 3 frequent: ");

for (int i = 0; i < Math.min(3, sorted.size()); i++) {

System.out.println(sorted.get(i).getKey() + " -> " + sorted.get(i).getValue());

}

}

}

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

import java.util.\*;

public class RemoveDuplicates {

public static void main(String[] args) {

List<Integer> list = Arrays.asList(1, 2, 2, 3, 4, 4, 5);

Set<Integer> unique = new HashSet<>(list);

System.out.println("Without duplicates: " + unique);

}

}

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

import java.util.\*;

class Student {

String name;

int marks;

Student(String name, int marks) {

this.name = name;

this.marks = marks;

}

public String toString() {

return name + ": " + marks;

}

}

public class SortStudents {

public static void main(String[] args) {

List<Student> students = Arrays.asList(

new Student("Alice", 85),

new Student("Bob", 78),

new Student("Charlie", 92)

);

students.sort((a, b) -> b.marks - a.marks); // Descending

System.out.println("Sorted students: " + students);

}

}

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

import java.util.\*;

public class GroupWords {

public static void main(String[] args) {

List<String> words = Arrays.asList("hi", "world", "cat", "great", "a");

Map<Integer, List<String>> map = new HashMap<>();

for (String word : words) {

int len = word.length();

map.computeIfAbsent(len, k -> new ArrayList<>()).add(word);

}

System.out.println("Grouped words: " + map);

}

}

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

import java.util.\*;

public class PreserveOrder {

public static void main(String[] args) {

String[] data = {"apple", "banana", "apple", "cherry", "banana"};

Set<String> orderedSet = new LinkedHashSet<>(Arrays.asList(data));

System.out.println("Ordered unique elements: " + orderedSet);

}

}

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

import java.util.\*;

public class WordIndex {

public static void main(String[] args) {

String sentence = "the cat and the dog";

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

Map<String, Integer> wordIndex = new HashMap<>();

for (int i = 0; i < words.length; i++) {

wordIndex.putIfAbsent(words[i], i); // First occurrence

}

System.out.println("Word Index Map: " + wordIndex);

}

}

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

import java.util.\*;

public class ListToSortedSet {

public static void main(String[] args) {

List<Integer> list = Arrays.asList(5, 3, 1, 4, 3, 2, 1);

Set<Integer> sortedSet = new TreeSet<>(list);

System.out.println("Sorted Unique Elements: " + sortedSet);

}

}

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

import java.util.\*;

public class Leaderboard {

public static void main(String[] args) {

Map<String, Integer> scores = new HashMap<>();

scores.put("Alice", 90);

scores.put("Bob", 85);

scores.put("Charlie", 95);

TreeMap<Integer, List<String>> leaderboard = new TreeMap<>(Collections.reverseOrder());

for (Map.Entry<String, Integer> entry : scores.entrySet()) {

leaderboard.computeIfAbsent(entry.getValue(), k -> new ArrayList<>()).add(entry.getKey());

}

System.out.println("Leaderboard:");

for (Map.Entry<Integer, List<String>> entry : leaderboard.entrySet()) {

System.out.println(entry.getValue() + " -> " + entry.getKey());

}

}

}

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

import java.util.\*;

public class ReverseList {

public static void main(String[] args) {

List<String> list = new ArrayList<>(Arrays.asList("Java", "Python", "C++", "Go", "Rust"));

Collections.reverse(list);

System.out.println("Reversed list: " + list);

}

}

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

import java.util.\*;

public class ShuffleDeck {

public static void main(String[] args) {

String[] suits = {"H", "D", "C", "S"};

String[] values = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"};

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

for (String suit : suits) {

for (String value : values) {

deck.add(value + suit);

}

}

Collections.shuffle(deck);

System.out.println("Shuffled Deck: " + deck);

}

}

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

import java.util.\*;

public class FirstRepeating {

public static void main(String[] args) {

List<Integer> list = Arrays.asList(1, 2, 3, 2, 1);

Set<Integer> seen = new HashSet<>();

for (int num : list) {

if (!seen.add(num)) {

System.out.println("First repeating: " + num);

return;

}

}

System.out.println("No repeats found.");

}

}

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

import java.util.\*;

public class BrowserHistory {

public static void main(String[] args) {

Stack<String> back = new Stack<>();

Stack<String> forward = new Stack<>();

back.push("Home");

back.push("About");

back.push("Contact");

System.out.println("Current: " + back.peek());

forward.push(back.pop()); // Going back

System.out.println("After going back, current: " + back.peek());

back.push(forward.pop()); // Going forward

System.out.println("After going forward, current: " + back.peek());

}

}

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

import java.util.\*;

public class PrintQueue {

public static void main(String[] args) {

Queue<String> printJobs = new LinkedList<>();

printJobs.add("Job1");

printJobs.add("Job2");

printJobs.add("Job3");

while (!printJobs.isEmpty()) {

System.out.println("Printing: " + printJobs.poll());

}

}

}

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

import java.util.\*;

public class SortMapByValue {

public static void main(String[] args) {

Map<String, Integer> map = new HashMap<>();

map.put("Pen", 10);

map.put("Notebook", 40);

map.put("Eraser", 5);

List<Map.Entry<String, Integer>> entries = new ArrayList<>(map.entrySet());

entries.sort(Map.Entry.comparingByValue());

for (Map.Entry<String, Integer> entry : entries) {

System.out.println(entry.getKey() + " -> " + entry.getValue());

}

}

}

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

import java.util.\*;

public class UniqueChars {

public static void main(String[] args) {

String input = "collection";

Set<Character> unique = new HashSet<>();

for (char c : input.toCharArray()) {

unique.add(c);

}

System.out.println("Unique characters: " + unique);

System.out.println("Total unique: " + unique.size());

}

}

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

import java.util.\*;

public class FrequentWord {

public static void main(String[] args) {

String paragraph = "apple banana apple orange banana apple";

String[] words = paragraph.split(" ");

Map<String, Integer> freq = new HashMap<>();

for (String word : words) {

freq.put(word, freq.getOrDefault(word, 0) + 1);

}

String mostFreq = "";

int max = 0;

for (Map.Entry<String, Integer> entry : freq.entrySet()) {

if (entry.getValue() > max) {

max = entry.getValue();

mostFreq = entry.getKey();

}

}

System.out.println("Most frequent word: " + mostFreq + " -> " + max);

}

}

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

import java.util.\*;

public class CountryCities {

public static void main(String[] args) {

Map<String, List<String>> countries = new HashMap<>();

countries.put("India", Arrays.asList("Delhi", "Mumbai", "Chennai"));

countries.put("USA", Arrays.asList("New York", "LA", "Chicago"));

countries.put("UK", Arrays.asList("London", "Manchester", "Bristol"));

for (Map.Entry<String, List<String>> entry : countries.entrySet()) {

System.out.println(entry.getKey() + ": " + entry.getValue());

}

}

}

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

import java.util.\*;

public class SlidingWindow {

public static void main(String[] args) {

List<Integer> nums = Arrays.asList(1, 3, 5, 7, 9, 2, 4, 6, 8, 10);

Deque<Integer> window = new LinkedList<>();

int windowSize = 3;

for (int i = 0; i < nums.size(); i++) {

window.addLast(nums.get(i));

if (window.size() > windowSize) {

window.removeFirst();

}

if (window.size() == windowSize) {

int sum = 0;

for (int num : window) sum += num;

System.out.println("Window sum: " + sum);

}

}

}

}