

Collections

What Are Collections in Java?

- ▶ A Collection is a group of objects, known as elements.
- ▶ Java provides collection classes to store, retrieve, and manipulate data efficiently.
- ▶ Found in the `java.util` package.

Why Use Collections?

- ▶ Flexible size (unlike arrays)
- ▶ Built-in methods (add, remove, search)
- ▶ Supports sorting, filtering, iteration
- ▶ Ready-to-use data structures: lists, sets, maps, queues

Collection Interfaces Overview

- ▶ Collection (root interface)
 - ▶ List: ordered, allows duplicates
 - ▶ Set: no duplicates
 - ▶ Queue: FIFO
- ▶ Map<K, V>: key-value pairs, not a Collection but part of the framework

List Interface

```
List<String> names = new ArrayList<>();  
names.add("Alice");  
names.add("Bob");
```

- ▶ Maintains insertion order
- ▶ Allows duplicates

List Implementations

- ▶ ArrayList: resizable array, fast read, slower insert/remove
- ▶ LinkedList: good for frequent inserts/removals

Demo 1 — Using a List and For-Each Loop

- ▶ Purpose:
 - ▶ Store a sequence of items
 - ▶ Iterate using a for-each loop
- ▶ Key Concepts:
 - ▶ List interface and ArrayList implementation
 - ▶ Type-safe collection with String
 - ▶ Looping through the list

Set Interface

```
Set<String> fruits = new HashSet<>();  
fruits.add("Apple");  
fruits.add("Apple"); // Ignored
```

- ▶ No duplicates
- ▶ No guaranteed order

Demo 2 — Set and Duplicate Handling

- ▶ Purpose:
 - ▶ Demonstrate a collection with no duplicates
 - ▶ Explain Set behavior
- ▶ Key Concepts:
 - ▶ Set interface and HashSet implementation
 - ▶ No duplicates allowed
 - ▶ Automatically filters repeated entries

Map Interface

```
Map<String, Integer> scores = new HashMap<>();  
scores.put("Anna", 90);
```

- ▶ Stores key-value pairs
- ▶ Keys must be unique

Map Implementations

- ▶ HashMap: unordered, fast access
- ▶ TreeMap: sorted keys
- ▶ LinkedHashMap: preserves insertion order

Demo 3 — Storing Data in a HashMap

- ▶ Purpose:
 - ▶ Store key-value pairs
 - ▶ Retrieve values using keys
- ▶ Key Concepts:
 - ▶ Map interface and HashMap implementation
 - ▶ Unique keys
 - ▶ Accessing elements using `keySet()`

Generics Overview

```
List<String> list = new ArrayList<>();
```

- ▶ Allow code to handle any data type
- ▶ Type-safe: errors at compile-time, not runtime

Why Use Generics?

- ▶ No casting required
- ▶ Prevents `ClassCastException`
- ▶ Clearer and reusable code

Writing Generic Methods

```
public <T> void printList(List<T> list) {  
    for (T item : list) {  
        System.out.println(item);  
    }  
}
```





Demo 4 — Generic Method to Print a List

- ▶ Purpose:
 - ▶ Reuse a method with different types
 - ▶ Show how generics improve flexibility
- ▶ Key Concepts:
 - ▶ Generic methods using `<T>`
 - ▶ No need for casting
 - ▶ Works with any object type

Collections Utility Methods

- ▶ `Collections.sort(list)`
- ▶ `Collections.reverse(list)`
- ▶ `Collections.max(list)`

Summary

- ▶  Use interfaces (List, Set, Map)
- ▶  Pick implementations based on need (ArrayList, HashSet, HashMap)
- ▶  Generics make collections safer
- ▶  Collections = core part of Java and Spring apps

Programming Challenge: "Vehicle Rental System"

- ▶ You are asked to build a simple in-memory "vehicle registration system" that stores and manages Vehicle objects (like Car and Motorcycle) using Java Collections. You will use generic collections like List and Map, and create methods to display and filter vehicles.

Programming Challenge Instructions

- ▶ 1. Create a Garage class that:
 - ▶ Has a List<Vehicle> to store all vehicles.
 - ▶ Has a Map<String, Vehicle> to store vehicles by license plate.
- ▶ 2. Add methods to:
 - ▶ Add a Vehicle to both the list and the map.
 - ▶ Print all vehicles using a generic method.
 - ▶ Find a vehicle by license plate.
 - ▶ List only Car or only Motorcycle objects.
- ▶ 3. Use the Vehicle, Car, and Motorcycle classes from the previous challenge.

Suggested Steps

- ▶ 1. Reuse the Vehicle class (abstract), and Car, Motorcycle subclasses.
- ▶ 2. Create a new class Garage.
- ▶ 3. Add a List<Vehicle> and a Map<String, Vehicle>.
- ▶ 4. Add vehicle objects and test all methods.
- ▶ 5. Bonus: Add a method to count how many of each type (Car, Motorcycle) exist.