Object Oriented Programming In Java

Saurabh Pandey
Associate Professor
Sitare University

Introduction to Java Collections Framework (JCF)

Definition:

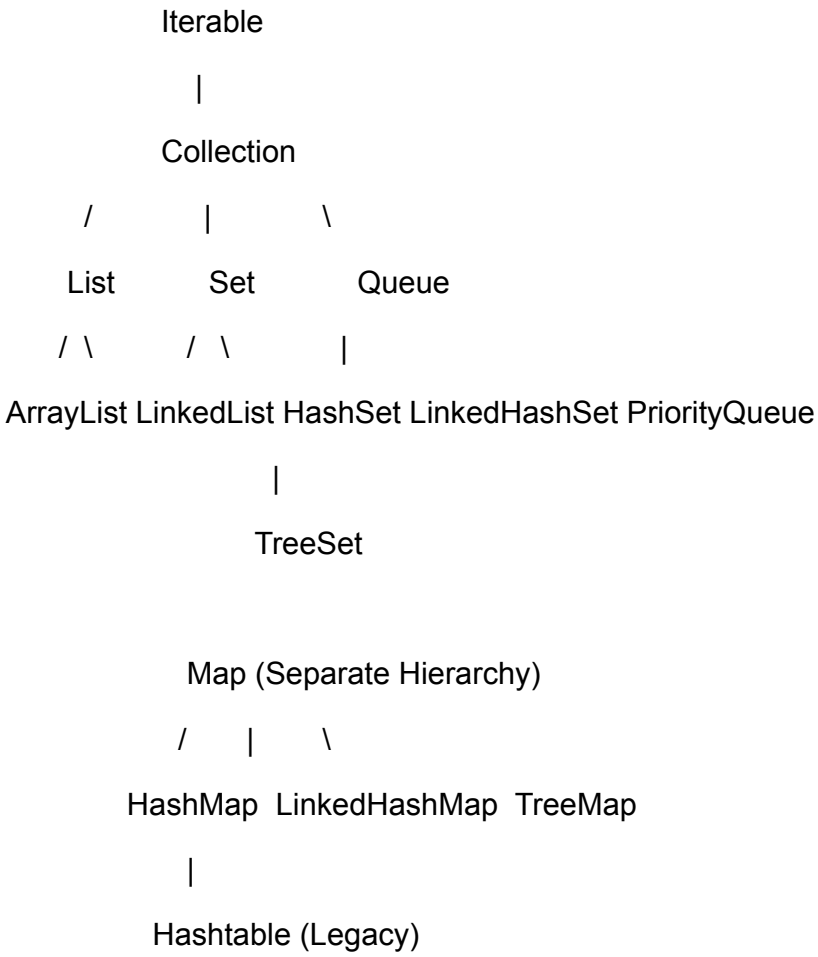
- A **unified architecture** for storing and manipulating groups of objects.
- Replaces traditional arrays, Vector, Hashtable (legacy classes).
- Provides **interfaces**, **implementations**, and **algorithms**.

Package: `java.util`

Advantages:

- Reusability
- Type-safety
- Performance
- Flexibility

Collections Framework Hierarchy



List Interface

Features:

- Ordered collection
- Allows **duplicate elements**
- Access elements by **index**

Common Classes:

- `ArrayList` – Dynamic array
- `LinkedList` – Doubly linked list
- `Vector` / `Stack` – Legacy classes

List Interface

```
List<String> list = new ArrayList<>();
list.add("Java");
list.add("C++");
list.add("Java"); // duplicate allowed
System.out.println(list);
```

Set Interface

Features:

- **Unordered**, **no duplicates**
- Null allowed once (except in TreeSet)

Implementations:

Class	Order	Allows Null	Sorted	Thread-safe
HashSet	No	Yes (1 null)	No	No
LinkedHashSet	Insertion	Yes	No	No
TreeSet	Sorted	No	Yes	No

Set Interface

```
Set<Integer> set = new HashSet<>();
set.add(10);
set.add(20);
set.add(10);
System.out.println(set); // [10, 20]
```

Map Interface

Features:

- Key–Value pairs
- Unique keys, duplicate values allowed

Implementations:

Class	Order	Null Keys	Sorted	Thread-safe
HashMap	No	1 allowed	No	No
LinkedHashMap	Insertion	1 allowed	No	No
TreeMap	Sorted by key	Not allowed	Yes	No
Hashtable	No	Not allowed	No	Yes

Map Interface

```
Map<Integer, String> map = new HashMap<>();
map.put(1, "A");
map.put(2, "B");
map.put(1, "C"); // overwrites
System.out.println(map);
```

Hashing Concept

Hashing:
A technique to convert an object into an integer **hash code** for efficient searching and insertion.

Steps in Hashing (for HashMap, HashSet):

1. Object's `hashCode()` is computed.
2. Hash code is mapped to an index in the bucket array.
3. If multiple elements share the same index → **collision** occurs.
4. Collisions are resolved using a **linked list or tree (since Java 8)**.

Important Methods:

- `hashCode()` – returns integer hash code
- `equals()` – checks equality when hash codes are same

HashMap vs Hashtable

Feature	HashMap	Hashtable
Synchronization	**Not synchronized**	**Synchronized**
Null Keys/Values	1 null key, many null values	❌ Not allowed
Performance	Faster (no sync)	Slower
Legacy	Modern (Java 1.2+)	Legacy (Java 1.0)
Fail-Fast Iterator	Yes	No
Use Case	Single-threaded apps	Multi-threaded apps (older)

Sorting in Collections: Comparable vs Comparator

Feature	Comparable	Comparator
Method	`compareTo()`	`compare()`
Package	`java.lang`	`java.util`
Used For	Natural ordering	Custom ordering
Example	`Collections.sort(list)`	`Collections.sort(list, comp)`

Example

```
class Student implements Comparable<Student>
{
    int marks;
    public int compareTo(Student s) {
        return this.marks - s.marks;
    }
}
```