✅ CopyOnWriteArrayList Summary

🔷 What is It?

A **thread-safe**, **ArrayList-like** collection designed for **read-heavy** concurrent scenarios.
It works by **copying the entire array on every write** operation.

🔷 1. Internal Working

- **Reads:** Fast, lock-free, operate on a snapshot.
- **Writes (add, set, remove):** Creates a **new copy** of the array; synchronized internally.
- **Iterator:** Snapshot iterator – doesn't reflect modifications after its creation.

✅ **No `ConcurrentModificationException`**
❌ **Slower writes** and **higher memory usage**

🔷 2. Thread-Safety & Performance Comparison

Feature	CopyOnWriteArrayList	Vector	ArrayList
Thread Safety	✅ Lock-free reads	✅ Fully synchronized	❌ Not thread-safe
Write Performance	❌ Slower (copying)	❌ Slower (locking)	✅ Fast
Read Performance	✅ Fast (no locks)	❌ Slower (locks)	✅ Fast
Iterator Behavior	✅ Snapshot	❌ Fail-fast	❌ Fail-fast
Memory Overhead	❌ High	✅ Low	✅ Low

🔷 3. Best Use Cases

- **Read-heavy** systems
- **Event listeners** in GUIs
- **Caching systems**
- **Configuration stores** where updates are rare

🔷 4. Limitations

- ❌ **High memory usage** during frequent writes
- ❌ **Slower write performance**
- ❌ **May serve stale data** to readers

🔷 5. Key Methods

Method	Description
`add(E e)`	Adds element to end
`add(int index, E e)`	Inserts at index
`remove(int index)`	Removes element at index
`remove(Object o)`	Removes first occurrence
`set(int index, E e)`	Replaces element at index
`get(int index)`	Gets element at index
`size()`	Gets number of elements
`isEmpty()`	Checks if empty
`contains(Object o)`	Checks if element exists
`iterator()`	Returns a **snapshot iterator**
`listIterator()`	Returns a **snapshot list iterator**
`addAll(Collection)`	Adds all elements from another collection
`clear()`	Clears the list

Let me know if you want the **complete runnable Java code** that demonstrates all methods and multi-threaded access.