⚙️Java Collections Deep Dive Table (Core Classes & Interfaces)

Interface / Class	Implements / Extends	Ordering	Duplicates Allowed	Thread-Safe	Nulls Allowed	Backed By	Notes / Behavior
List (Interface)	Collection	Maintains insertion order	Yes	No	Yes (1 null)	Varies	Index-based access
ArrayList	List	Yes	Yes	No	Yes (1 null)	Dynamic array	Fast reads, slow inserts/ deletes
LinkedList	List, Deque	Yes	Yes	No	Yes (1 null)	Doubly-linked list	Good for frequent inserts/ removals
Vector	List	Yes	Yes	Yes	Yes (1 null)	Dynamic array	Synchronized, legacy
Stack	Vector	Yes	Yes	Yes	Yes (1 null)	Dynamic array	LIFO stack

🗓️Java Set Implementations

Interface / Class	Ordering	Duplicates Allowed	Thread-Safe	Nulls Allowed	Backed By	Behavior / Notes
Set (Interface)	No order	No	No	Yes (1 null)	Varies	Unique elements
HashSet	No	No	No	Yes (1 null)	HashMap	Fast, unordered
LinkedHashSet	Yes (insertion order)	No	No	Yes (1 null)	HashMap + LinkedList	Predictable iteration
TreeSet	Sorted (natural/ comp)	No	No	No nulls	TreeMap	Log(n) ops, sorted

🔑Java Map Implementations

Interface / Class	Ordering	Duplicate Keys	Thread-Safe	Nulls	Backed By	Behavior / Notes
Map (Interface)	No	No (keys)	No	1 null key, many values	Varies	Key-value pairs

Interface / Class	Ordering	Duplicate Keys	Thread-Safe	Nulls	Backed By	Behavior / Notes
HashMap	No	No	No	1 null key, many values	Hash table	Fast, unordered
LinkedHashMap	Yes (insertion/ access)	No	No	Same as HashMap	Hash + LinkedList	Ordered version of HashMap
TreeMap	Sorted	No	No	No null keys	Red-Black Tree	Keys sorted using Comparable/ Comparator
Hashtable	No	No	Yes	No nulls	Hash table	Legacy, synchronized
ConcurrentHashMap	No	No	Yes (partial)	No nulls	Segment-locked map	High concurrency, faster than Hashtable

⏳Java Queue Implementations

Interface / Class	Ordering	Thread-Safe	Nulls Allowed	Backed By	Behavior / Notes
PriorityQueue	Sorted	No	No	Binary heap	Min-heap by default
ArrayDeque	FIFO/ LIFO	No	No	Resizable array	Fast deque, better than Stack/LinkedList
LinkedList (as Deque)	Insertion	No	Yes	Doubly-linked list	Doubles as a stack, queue

🔍Additional Cheat Notes

- **List vs Set:** List = Ordered + Duplicates; Set = Unordered (or Ordered but Unique)
- **HashSet vs TreeSet:** HashSet = faster; TreeSet = sorted
- **HashMap vs TreeMap:** HashMap = fast access; TreeMap = sorted keys
- **Vector vs ArrayList:** Vector = thread-safe; ArrayList = faster
- **HashMap vs Hashtable vs ConcurrentHashMap:**
- HashMap = common, not thread-safe
- Hashtable = thread-safe, slow
- ConcurrentHashMap = high-concurrency