|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Interface / Class | Maintains Order | Allows Duplicates | Sorted | Time Complexity (Lookup) | Key Traits |
| **List** | ArrayList | ✅ Yes  (by index) | ✅ Yes | ❌ No | O(1) | Indexed dynamic array |
| **List** | LinkedList | ✅ Yes | ✅ Yes | ❌ No | O(n) | Doubly linked list |
| **Queue** | ArrayDeque | ✅ Yes  (FIFO/LIFO) | ✅ Yes | ❌ No | O(1) | Double-ended queue |
| **Queue** | PriorityQueue | ❌ No  (priority order) | ✅ Yes | ✅ Yes | O(log n) | Sorted by priority |
| **Set** | HashSet | ❌ No | ❌ No | ❌ No | O(1) | Fast, unordered uniqueness |
| **Set** | LinkedHashSet | ✅ Yes (insertion order) | ❌ No | ❌ No | O(1) | Ordered uniqueness |
| **Set** | TreeSet | ✅ Yes  (sorted) | ❌ No | ✅ Yes | O(log n) | Sorted uniqueness |
| **Map** | HashMap | ❌ No | ❌ (keys) | ❌ No | O(1) | Fast key-value storage |
| **Map** | LinkedHashMap | ✅ Yes  (insertion order) | ❌ (keys) | ❌ No | O(1) | Ordered key-value storage |
| **Map** | TreeMap | ✅ Yes  (sorted by key) | ❌ (keys) | ✅ Yes | O(log n) | Sorted key-value pairs |

📊 Java Collections Overview

🧠 Notes & Tips

* List is best when **order matters** and duplicates are allowed.
* Set ensures **uniqueness**, with variants for speed or ordering.
* Map is for **key-value storage**, not designed to hold duplicate keys.
* Queue and its implementations (ArrayDeque, PriorityQueue) manage elements with **specific access policies** (FIFO, priority).
* TreeSet and TreeMap require natural ordering or Comparator.
* None of these are **thread-safe** by default—use Collections.synchronized\*() or concurrent versions if needed.