Java Collections Quick-Reference Cheat Sheet

ArrayList, LinkedList, HashMap & HashSet — method comparisons and common defect patterns for QA, BA, and IT professionals.

1. Which Collection Am I Looking At?

Type	Ordered?	Duplicates?	Best For	Common Use Case
ArrayList	Yes	Yes	Fast index access	API response records, page element lists, test steps
LinkedList	Yes	Yes	Frequent mid-list insert/remove	Queue-style message processing
HashMap	No	Keys: No	Fast key lookup	ID-to-record mapping, config parsing
LinkedHashMap	Yes	Keys: No	Key lookup + order	Fixes 'field order is wrong' defects
HashSet	No	No	Uniqueness checks	Deduplicating a list before comparison

2. Core ArrayList Methods

Method	Does	Watch For
`add(element)`	Appends to the end	Insertion order may matter for sequence assertions
`get(index)`	Returns element at position	Wrong index = #1 cause of ArrayIndexOutOfBoundsException
`remove(int)` vs `remove(Object)`	Removes by position vs. by value	Overload confusion is a frequent logic-error source
`size()`	Returns element count	Used constantly in record-count assertions
`contains(element)`	Boolean presence check	Common in existence-based assertions
`isEmpty()`	Checks for zero elements	Check before get(0) to avoid a runtime exception
`clear()`	Removes all elements	Missing calls cause test pollution between runs
`sort()`	Orders elements	Changes index positions — breaks scripts assuming original order

3. Five Defect Patterns to Recognize on Sight

Fixed-index reads on variable-length data
Asserting against list.get(4) on data that doesn't always have 5+ elements. Common in HL7/FHIR repeating fields. Fix: loop and validate size separately.

Modifying a list while iterating
Calling remove() inside a for-each loop throws ConcurrentModificationException. Fix belongs in loop structure, not the data.

Unsynchronized access under load
ArrayList is not thread-safe. Two threads writing simultaneously can corrupt state or drop elements — a frequent cause of 'only fails in production' defects.

Duplicate entries inflating mismatch counts
Retried API calls or unfiltered joins add duplicates. A reconciliation job comparing lists without deduplication can report false discrepancies.

Null elements causing downstream NPEs
ArrayList permits null values. A NullPointerException several calls away often traces back to an unvalidated list load.

Edge case: a LinkedHashMap exists because teams got tired of HashMap's unordered output. It behaves like a HashMap but preserves insertion order — if you see it in code, someone likely fixed an order-related defect at some point.