




## Java collections

 Type	 What It Is	 When to Use It
List	Ordered collection, allows duplicates	Use when order matters — like ArrayList, LinkedList
Set	Unordered, no duplicates allowed	Great for storing unique items — HashSet, TreeSet
Map	Key-value pairs, keys are unique	Use when you want to map data — like a phonebook (HashMap)
Queue	Follows FIFO (First In First Out)	Perfect for task scheduling — LinkedList, PriorityQueue
Deque	Double-ended queue — add/remove from both ends	Use when you need both stack and queue features
Stack	Follows LIFO (Last In First Out)	Undo functionality, parsing — classic use case
PriorityQueue	Orders elements by priority	Useful for jobs with urgency — like a printer queue
LinkedList	Doubly-linked list	Great when you need frequent insert/delete in middle
ArrayList	Dynamic array implementation of List	Best for fast reads and simple usage

HashSet	No duplicates, no order	Fast membership checks — ideal for filtering values
TreeSet	Sorted Set	When you need items in sorted order
HashMap	Unordered key-value storage	Fast lookup with unique keys — super common in apps
LinkedHashMap	Maintains insertion order	When you want Map + predictable order
TreeMap	Sorted Map	Keeps keys sorted — great for leaderboard-style tasks
Hashtable	Thread-safe Map (legacy)	Use only if you need synchronized Map in old codebases
EnumSet	Set specifically for enums	Super fast when dealing with enums
ConcurrentHashMap	Thread-safe version of HashMap	Go-to for Maps in multithreaded apps
Vector	Thread-safe List (legacy)	Rarely used today, replaced by ArrayList or CopyOnWriteArrayList