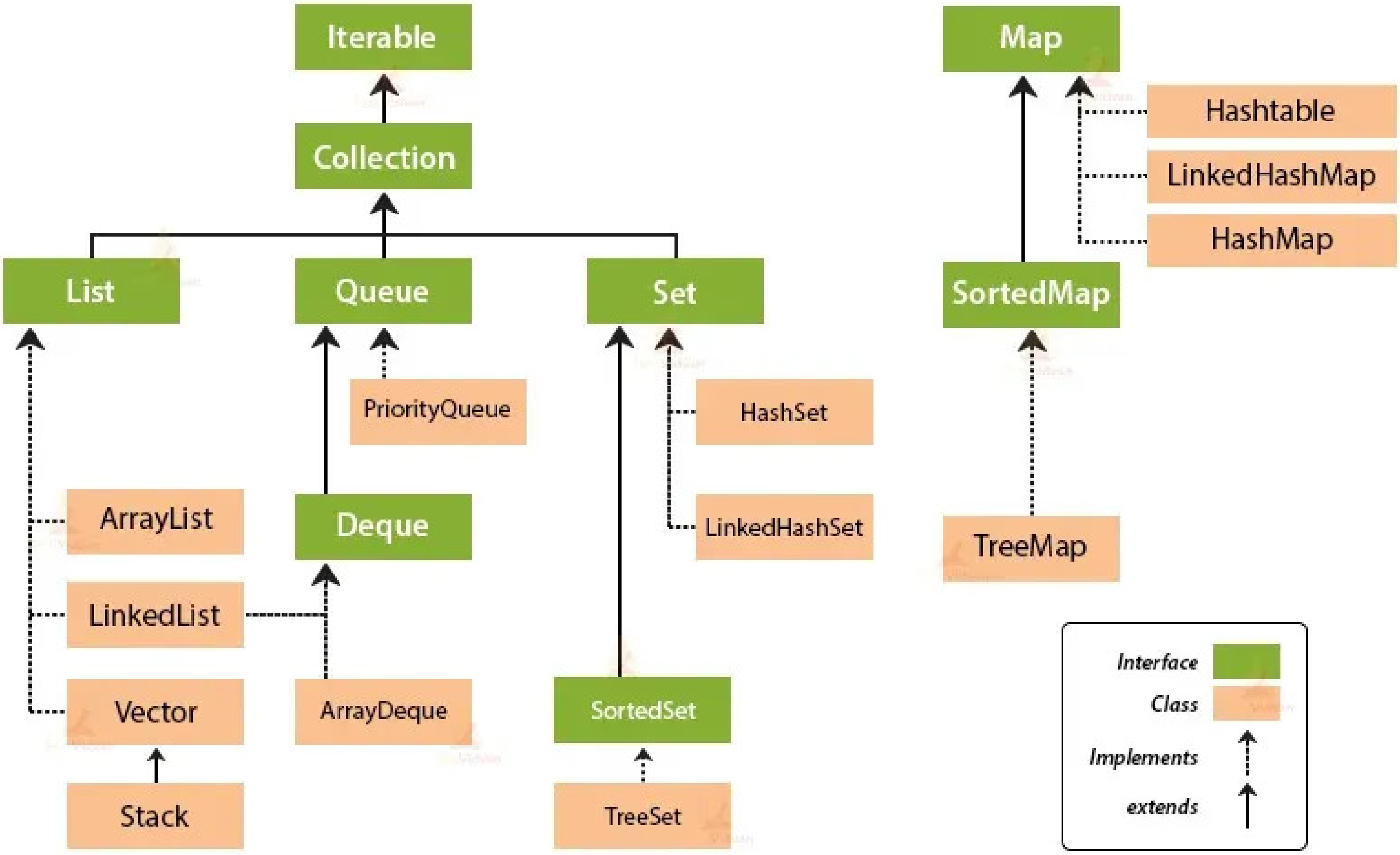
@naveenkumarqa

Collection Framework Hierarchy in Java



💹 Collections Evolution CheatSheet

🧩 Concept	🚀 Advantage over Previous
🔢 **Primitive Data Types**	Stores a **single value** (e.g., `int`, `char`)
📦 **Array**	Stores **multiple same-type values** with **fixed size**
📈 **ArrayList**	Supports **dynamic resizing** and **fast random access**
🔗 **LinkedList**	Enables **faster insert/delete** via **doubly linked nodes**
📒 **Vector**	Like ArrayList, but **thread-safe** via **synchronization**
🌀 **Stack / Queue**	Provides **LIFO/FIFO access** for **sequential processing**
⏫ **PriorityQueue**	Processes elements based on **priority (Min-Heap)**
🎯 **HashSet**	Stores **unique items** with **constant-time lookup**
🧾 **LinkedHashSet**	Keeps **insertion order** along with **uniqueness**
🌳 **TreeSet**	Stores **sorted unique elements** using **Red-Black Tree**
🗂️ **HashMap**	Stores **key-value pairs** with **fast access**
🖇️ **LinkedHashMap**	Maintains **insertion order** in key-value mappings
📚 **TreeMap**	Keeps keys **sorted** for **range-based access**
🏦 **Hashtable**	Offers **synchronized key-value access** (thread-safe)
🌐 **ConcurrentHashMap**	Provides **high-concurrency thread-safe** key-value storage

🛍️ ArrayList

➡️ **Implements**	`List` Interface
🧱 **Data Structure**	📐 Dynamic Array
🔢 **Default Size**	🔟
🧮 **Load Factor**	➕ Increases by 50% on resize
🧭 **Order**	✅ Insertion order preserved
🔐 **Sync**	🔓 Not synchronized
⭕ **Nulls**	✅ Allowed (multiple)
‼️ **Duplicates**	✅ Allowed
💡 **Usage**	Fast random access, rare inserts/removals
⚡ **Performance**	🔍 Search – ⚡ Fast ✍️ Modify – 🐢 Slow
🌍 **Real-world**	🛒 Shopping list
🧰 **Methods**	`.add()` , `.get()` , `.set()` , `.remove()` , `.size()` , `.clear()`

🚂 LinkedList

➡️ **Implements**	`List` , `Deque` Interfaces
🧱 **Data Structure**	🔗 Doubly Linked List ***(Node)***
🔗 **Node**	`Object` data, `Node` next, `Node` prev
🔢 **Default Size**	📈 Dynamic
🧮 **Load Factor**	🚫 Not applicable
🧭 **Order**	✅ Insertion order preserved
🔐 **Sync**	🔓 Not synchronized
⭕ **Nulls**	✅ Allowed (multiple)
‼️ **Duplicates**	✅ Allowed
💡 **Usage**	Frequent inserts/deletes, Minimal random access
⚡ **Performance**	🔍 Search – 🐢 Slow ✍️ Modify – ⚡ Fast for ends
🌍 **Real-world**	🚂 Train coach
🧰 **Methods**	`.addFirst()` , `.addLast()` , `.poll()` , `.peak()`

🏥 PriorityQueue

➡️ **Implements**	`Queue` Interface
🧱 **Data Structure**	🎄 Min-Heap Binary Tree ***(FIFO)***
🔢 **Default Size**	📈 Dynamic
🧭 **Order**	⏳ Based on priority
🔐 **Sync**	🔒 Not synchronized
⭕ **Nulls**	❌ Not Allowed
‼️ **Duplicates**	✅ Allowed
💡 **Usage**	Prioritize elements before processing
⚡ **Performance**	🔍 Search – 🐢 Slow ✍️ Modify – 🐢 Slow
🌍 **Real-world**	🏥 Hospital triage
➕ **Adds to tail**	`add()` throws exception, `offer()` returns `false`
➖ **Retrieves head**	`element()` throws exception, `peek()` returns `null`
❌ **Removes head**	`remove()` throws exception, `poll()` returns `null`

🔑 HashSet

➡️ **Implements**	`Set` Interface
🧱 **Data Structure**	🧮 Hash Table
🔢 **Default Size**	16
🧮 **Load Factor**	0.75
🧭 **Order**	❌ Not preserved
🔐 **Sync**	🔓 Not synchronized
⭕ **Nulls**	✅ One null allowed
‼️ **Duplicates**	❌ Not allowed
💡 **Usage**	Ensure unique without order
⚡ **Performance**	🔍 Search – ⚡ Fast ✍️ Modify – ⚡ Fast
🌍 **Real-world**	🔑 Keychain (Each key is unique but unorderd)
🧰 **Methods**	`.clone()`

🗓️ LinkedHashSet

➡️ **Implements**	`Set` Interface
🧱 **Data Structure**	🧮 Hash Table + 🔗 Doubly Linked List
🔢 **Default Size**	16
🧮 **Load Factor**	0.75
🧭 **Order**	✅ Insertion order preserved
🔐 **Sync**	🔒 Not synchronized
⭕ **Nulls**	✅ One null allowed
‼️ **Duplicates**	❌ Not allowed
💡 **Usage**	Ensure unique with insertion order
⚡ **Performance**	🔍 Search – ⚡ Fast ✍️ Modify – ⚡ Fast
🌍 **Real-world**	🗓️ Calendar Events (chronological & unique)
🧰 **Methods**	Same as HashSet

📚 TreeSet

➡️ **Implements**	`SortedSet` & `NavigableSet` Interface
🧱 **Data Structure**	🎋 Red-Black Tree
🔢 **Default Size**	📈 Dynamic
🧭 **Order**	🔢 Sorted elements (natural/comparator)
🔐 **Sync**	🔒 Not synchronized
⭕ **Nulls**	❌ Not Allowed
‼️ **Duplicates**	❌ Not Allowed
💡 **Usage**	Maintain sorted unique elements
⚡ **Performance**	🔍 Search – 🐢 Slow ✍️ Modify – 🐢 Slow
🌍 **Real-world**	📚 Dictionary
🧰 Methods	`.first()` , `.last()` , `.lower()` , `.higher()` , `.floor()` , `.ceiling()` , `.headSet()` , `.tailSet()`

📞 HashMap

➡️ **Implements**	`Map` Interface
🧱 **Data Structure**	🧮 Hash Table
🔢 **Default Size**	16
🧮 **Load Factor**	0.75
🧭 **Order**	❌ Not preserved
🔐 **Sync**	🔓 Not synchronized
⭕ **Nulls**	✅ One null key, Multiple null values
‼️ **Duplicates**	❌ Keys must be unique, ✅ Values can repeat
💡 **Usage**	Fast key-value pair access
⚡ **Performance**	🔍 Search – ⚡ Fast ✍️ Modify – ⚡ Fast
🌍 **Real-world**	📞 Phone contacts
🧰 **Methods**	`.put()` , `.keySet()` , `.values()` , `.entrySet()` , `.getKey()` , `.getValue()`

📲 LinkedHashMap

➡️ **Implements**	`Map` Interface
🧱 **Data Structure**	🧮 Hash Table + 🔗 Doubly Linked List
🔢 **Default Size**	16
🧮 **Load Factor**	0.75
🧭 **Order**	✅ Insertion order preserved
🔐 **Sync**	🔓 Not synchronized
⭕ **Nulls**	✅ One null key, Multiple null values
‼️ **Duplicates**	❌ Keys must be unique, ✅ Values can repeat
💡 **Usage**	Maintain insertion order with fast access
⚡ **Performance**	🔍 Search – ⚡ Fast ✍️ Modify – ⚡ Fast
🌍 **Real-world**	📲 Phone’s Recent Calls (ordered in call timestamps)
🧰 **Methods**	Same as HashSet

📖 TreeMap

➡️ **Implements**	`SortedMap` & `NavigableMap` Interface
🧱 **Data Structure**	🎋 Red-Black Tree
🔢 **Default Size**	📈 Dynamic
🧭 **Order**	🔢 Sorted by keys (natural/comparator)
🔐 **Sync**	🔒 Not synchronized
⭕ **Nulls**	❌ Null keys not allowed, ✅ Null values allowed
‼️ **Duplicates**	❌ Duplicate keys not allowed
💡 **Usage**	Maintain sorted key-value pairs
⚡ **Performance**	🔍 Search – 🐢 Slow ✍️ Modify – 🐢 Slow
🌍 **Real-world**	📖 Encyclopedia (alphabetical order)
🧰 Methods	`.firstKey()` , `.lastKey()` , `.lowerKey()` , `.higherKey()` , `.subMap()` , `.headMap()` , `.tailMap()`

🏦 HashTable

➡️ **Implements**	`Map` Interface
🧱 **Data Structure**	🧮 Hash Table
🔢 **Default Size**	11
🧮 **Load Factor**	0.75
🧭 **Order**	❌ Not preserved
🔐 **Sync**	✅ Thread-safe
⭕ **Nulls**	❌ Not allowed
‼️ **Duplicates**	❌ Keys must be unique, ✅ Values can repeat
💡 **Usage**	Thread-safe key-value storage (legacy)
⚡ **Performance**	🔍 Search – 🐌 Moderate ✍️ Modify – 🐌 Moderate
🌍 **Real-world**	🏦 Bank Locker system with one person at a time
🧰 **Methods**	`.keys()` , `.elements()` , `.clone()` , `.rehash()`

🌏 Real World Analogy

Collection	Real-World Analogy
ArrayList	🛒 **Shopping list** — items added in order, fast access by index
LinkedList	🚂 **Train Coach** — easy to attach/detach (nodes) from either end
Vector	📽️ **Film Projector** — reel-to-reel one frame at a time
HashSet	🔑 **Jumbled Keychain** — each key (element) is unique
LinkedHashSet	🏛️ **Museum artifacts** — unique items maintained in order of arrival
TreeSet	📖 **Dictionary** — sorted words without duplicates
PriorityQueue	🏥 **Hospital triage** — most urgent patient (smallest element) treated first
ArrayDeque	🛣️ **Toll booth line —** cars (elements) enter/exit from both ends
Queue (LinkedList)	🎟️ **Movie ticket queue —** maintains insertion order
HashMap	📞 **Contact list** — names (keys) linked to phone numbers (values)
LinkedHashMap	📝 **Recipe steps** — ordered key-value pairs, preserving insertion order
TreeMap	📚 **Encyclopedia** — sorted topics with their explanations
ConcurrentHashMap	🌐 **Wikipedia Edits** — allows safe, parallel access to users
Hashtable	🏦 **Bank vault** — synchronized and thread-safe, sorted dates

💻 Tech Analogy

Collection	Tech Analogy
ArrayList	📸 **Photo gallery app** — fast to view any photo by index, good for browsing
LinkedList	🎧 **Music playlist** — songs linked in order, easy to insert/remove anywhere
Vector	🧾 **Shared Google Sheet** — multiple people can safely edit (thread-safe ArrayList)
HashSet	🔐 **Password manager** — stores only unique passwords, fast to check existence
LinkedHashSet	📆 **Event Calendar** — events added chronologically no duplicates
TreeSet	📠 **Autocomplete Suggestions** — results are sorted and unique
PriorityQueue	🔄 **Task Scheduler** — OS scheduler executes high-priority tasks first
ArrayDeque	⏪ **Undo/Redo stack in Editor** — quick undo or redo efficiently.
Queue (LinkedList)	📧 **Messaging app** — FIFO Outgoing message queue
HashMap	📱 **Contacts app** — store name-number pairs for fast lookup
LinkedHashMap	🖼️ **Instagram Story queue** — key-value pairs shown in order
TreeMap	📁 **Sorted folder names** — keys auto-sorted, like alphabetical files
ConcurrentHashMap	📊 **JIRA dashboard** — multiple users reading/writing data safely
Hashtable	🔏 **Shared Google Sheet** — but only one person can edit at a time