Collection Framework in Java

1️⃣ What is the Collection Framework in Java?

The **Collection Framework** in Java is a **set of classes and interfaces** that provides a standardized way to **store, manipulate, and retrieve** groups of objects efficiently.

🔹 Key Features of the Collection Framework

- **Reusable and efficient** data structures.
- Supports **dynamic memory allocation** (unlike arrays).
- Includes various **data structures like List, Set, Queue, and Map**.
- Provides built-in **sorting and searching** utilities.
- Uses **Iterator** and **ListIterator** for traversal.

🔹 Collection Hierarchy

scss
Copy code
```
Collection (Interface)
   ├── List (Ordered, allows duplicates)
   │    ├── ArrayList
   │    ├── LinkedList
   │    ├── Vector
   ├── Set (Unordered, unique elements)
   │    ├── HashSet
   │    ├── TreeSet
   │    ├── LinkedHashSet
   ├── Queue (FIFO order)
   │    ├── PriorityQueue
   │    ├── Deque
```

2️⃣ Difference Between `ArrayList` and `LinkedList`

Feature	ArrayList	LinkedList
Implementation	Uses **dynamic array**	Uses **doubly linked list**
Insertion Speed	**Slow** (shifts elements)	**Fast** (only pointer changes)

Deletion Speed	**Slow** (shifts elements)	**Fast** (only pointer changes)
Random Access	**Fast** (`O(1)`)	**Slow** (`O(n)`)
Memory Usage	**Less** (stores data only)	**More** (stores data + pointers)
Best for	**Search-heavy operations**	**Insert/Delete-heavy operations**

✅ **Use `ArrayList`** when **searching** frequently.
✅ **Use `LinkedList`** when **inserting/deleting** frequently.

3️⃣ Difference Between `Iterator` and `ListIterator`

Feature	Iterator	ListIterator
Traversal	**Only forward**	**Both forward and backward**
Applicable To	**All Collections (List, Set, Queue, etc.)**	**Only Lists (ArrayList, LinkedList, etc.)**
Modification	Can remove elements	Can add, remove, and replace elements
Methods	`hasNext()`, `next()`, `remove()`	`hasNext()`, `next()`, `hasPrevious()`, `previous()`, `set()`

✅ **Use `Iterator`** when you need **simple forward traversal**.
✅ **Use `ListIterator`** when you need **both forward and backward traversal**.

4️⃣ Difference Between `Iterator` and `Enumeration`

Feature	Iterator	Enumeration
Introduced In	Java 1.2	Java 1.0
Traversal	**Only forward**	**Only forward**
Methods	`hasNext()`, `next()`, `remove()`	`hasMoreElements()`, `nextElement()`

Modification	**Can remove** elements	**Cannot remove** elements
Applicable To	**All Collections**	**Legacy Collections (Vector, Hashtable, etc.)**

✅ **Use `Iterator`** for modern collections like `ArrayList`, `HashSet`.
✅ **Use `Enumeration`** for legacy collections like `Vector`, `Hashtable`.

5️⃣ Difference Between `List` and `Set`

Feature	List	Set
Order	**Maintains insertion order**	**No guaranteed order**
Duplicates	**Allows duplicates**	**Does not allow duplicates**
Implementation Classes	`ArrayList`, `LinkedList`, `Vector`	`HashSet`, `TreeSet`, `LinkedHashSet`

✅ **Use `List`** when order matters (e.g., `ArrayList`).
✅ **Use `Set`** when uniqueness is required (e.g., `HashSet`).

6️⃣ Difference Between `HashSet` and `TreeSet`

Feature	HashSet	TreeSet
Order	**Unordered**	**Sorted (Natural Order)**
Performance	**Faster (`O(1)`)**	**Slower (`O(log n)`)**
Allows `null`?	✅ Yes (only one `null`)	❌ No
Implementation	Uses **Hash Table**	Uses **Red-Black Tree**

✅ **Use `HashSet`** for **fast lookups** (unordered).
✅ **Use `TreeSet`** when **sorting is required**.

7️⃣ Difference Between `Array` and `ArrayList`

Feature	Array	ArrayList
Size	**Fixed** at creation	**Dynamic (auto-resizes)**
Memory Efficiency	More efficient	Uses more memory
Performance	Faster for fixed-size data	Slightly slower due to resizing
Methods	No built-in methods	Many built-in methods (`add()`, `remove()`, `contains()`)
Primitive Support	Stores both **primitives** and objects	Stores only **objects**

✅ **Use `Array`** for **fixed-size** collections.
✅ **Use `ArrayList`** for **dynamic collections**.