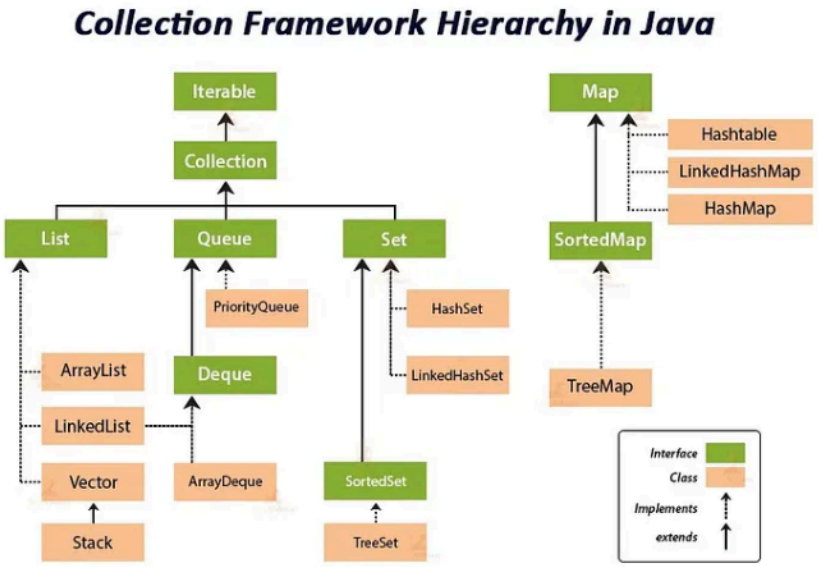
Set

Hierarchy of sets in Java



- Set(I) is a interface extended by collection interface.
- Set interface is implemented by HashSet and LinkedHashSet.

Properties of set :

- Sets do not store duplicate elements.

HashSet :

- Stores only unique element as it implements Set.
- ***Does not maintain specific order of set.***
- Can store null value, but only one.
- not thread safe, external synchronization is required.
- performance is fast for add, remove and contains.
- internal structure → Hashtable.

```
import java.util.HashSet;

public class HashSetExample {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<>();

```

```
        set.add("Java");
        set.add("Spring");
        set.add("Boot");
        set.add("Java"); // duplicate
        set.add(null);   // allowed

        System.out.println("Set contents: " + set);
        System.out.println("Size: " + set.size());
        System.out.println("Contains 'Spring'? " + set.contains("Spring"));

        set.remove("Boot");
        System.out.println("After removal: " + set);
    }
}

```

LinkedHashSet :

- Stores only unique element as it implements Set.
- Duplicates not allowed
- ***Maintains insertion order.***
- Can store null value, but only one.
- not thread safe, external synchronization is required.
- average performance, slower than HashSet.
- Internal structure → Hashtable + Linkedlist.

⚙️ Internal Working of `LinkedHashSet`

- Inherits from `HashSet`, which uses a **hash table**.
- Also maintains a **doubly-linked list** across all entries to remember the **order of insertion**.
- When an element is added:
  - Its hash is computed (`hashCode()`), and it's placed in a hash bucket.
  - Its position is also linked to the previous element (maintains order).
- Duplicate entries are **ignored** using `equals()` check.

```
import java.util.LinkedHashSet;

public class LinkedHashSetExample {
    public static void main(String[] args) {
        LinkedHashSet<String> set = new LinkedHashSet<>();

        set.add("Java");
        set.add("Spring");
        set.add("Boot");
        set.add("Java"); // duplicate
```

```
        set.add(null);   // allowed

        System.out.println("Set contents: " + set);
        System.out.println("Size: " + set.size());
        System.out.println("Contains 'Spring'? " + set.contains("Spring"));

        set.remove("Boot");
        System.out.println("After removal: " + set);
    }
}

```

TreeSet :

- Stores **unique elements** in **sorted (ascending) order** by default.
- **Does not allow** `null` (for non-empty sets).
- not thread safe, external synchronization is required.

```java
import java.util.TreeSet;

public class TreeSetExample {
    public static void main(String[] args) {
        TreeSet<String> set = new TreeSet<>();

        set.add("Java");
        set.add("Spring");
        set.add("Boot");
        set.add("Java"); // duplicate
        // set.add(null); // ❌ Throws NullPointerException

        System.out.println("Set contents: " + set); // Sorted order
        System.out.println("First element: " + set.first());
        System.out.println("Last element: " + set.last());

        set.remove("Boot");
        System.out.println("After removal: " + set);
    }
}

```

HashSet, LinkedHashset and TreeSet :

Feature	***HashSet***	***LinkedHashSet***	***TreeSet***
Duplicates	❌	❌	❌
Order	❌	✅ insertion order	✅ sorted order
Null allowed	✅ one	✅ one	❌

performance	✅ Fastest	Average(Slightly slow)	Slow
Backed by	Hash Table	Hash Table + Linked list	Red-Black Tree

Document by ***Suyash*** 😇