**Sequenced Collection in Java 21-2025**

**What is a Sequenced Collection?:** In Java 21, a new interface hierarchy was added to handle **ordered collections** more consistently:

* **SequencedCollection**
* **SequencedSet**
* **SequencedMap**

**Objective**: To provide a **standardized way** to access the *first* and *last* elements of a collection, as well as to traverse it *forwards* or *backwards*, **regardless of the specific implementation** (like ArrayList, LinkedList, TreeSet, etc.).

Why Introduced?

Before Java 21:

* You could get the first element of a List using list.get(0), but not from a Set.
* You could iterate a LinkedHashMap in order, but there was no consistent way to access the *last entry*.
* Different collections had inconsistent APIs for ordered access.

**Advantages**

* Direct access to first and last elements (getFirst(), getLast())
* Insertion/removal at both ends (addFirst(), addLast(), removeFirst(), removeLast())
* Reversed view of the collection (reversed())

**SequencedCollection<E>**

* Extends Collection<E>, adds methods:
* E getFirst();
* E getLast();
* E removeFirst();
* E removeLast();
* void addFirst(E e);
* void addLast(E e);
* SequencedCollection<E> reversed();

**SequencedSet<E>**

Extends both SequencedCollection<E> and Set<E> — for ordered sets.

**SequencedMap<K,V>**

Extends Map<K,V>, adds:

Map.Entry<K,V> firstEntry();

Map.Entry<K,V> lastEntry();

Map.Entry<K,V> pollFirstEntry();

Map.Entry<K,V> pollLastEntry();

SequencedMap<K,V> reversed();

Examples

**Using SequencedCollection**

SequencedCollection<String> names = new ArrayList<>();

names.add("Alice");

names.add("Bob");

names.add("Charlie");

System.out.println("First: " + **names.getFirst()**); // Alice

System.out.println("Last: " + **names.getLast()**); // Charlie

**names.addFirst("Zara");**

**names.addLast("David");**

System.out.println("After adding at ends: " + names);

**names.removeFirst(); // removes Zara**

**names.removeLast(); // removes David**

Example: **SequencedMap**

SequencedMap<Integer, String> map = new LinkedHashMap<>();

map.put(1, "One");

map.put(2, "Two");

map.put(3, "Three");

**System.out.println("First entry: " + map.firstEntry());**

**System.out.println("Last entry: " + map.lastEntry());**

**map.putFirst(0, "Zero"); // new in Java 21**

**map.putLast(4, "Four");**

**Sequenced Sets**

SequencedSet<String> sequencedSet = new LinkedHashSet<>();  
**sequencedSet.addFirst("Apple");**  
 sequencedSet.add("Banana");  
 **sequencedSet.addLast("Cherry");**  
  
SequencedSet<String> sequencedSet = new LinkedHashSet<>();  
sequencedSet.addFirst("Apple");  
sequencedSet.add("Banana");  
sequencedSet.addLast("Cherry");

**Benefits**

✅ Unified access to first/last elements across Lists, Sets, and Maps.  
✅ Cleaner, more readable code.  
✅ reversed() gives a **live view** — not a copied collection.  
✅ Backward compatible — existing classes (ArrayList, LinkedHashSet, LinkedHashMap) automatically implement these interfaces.