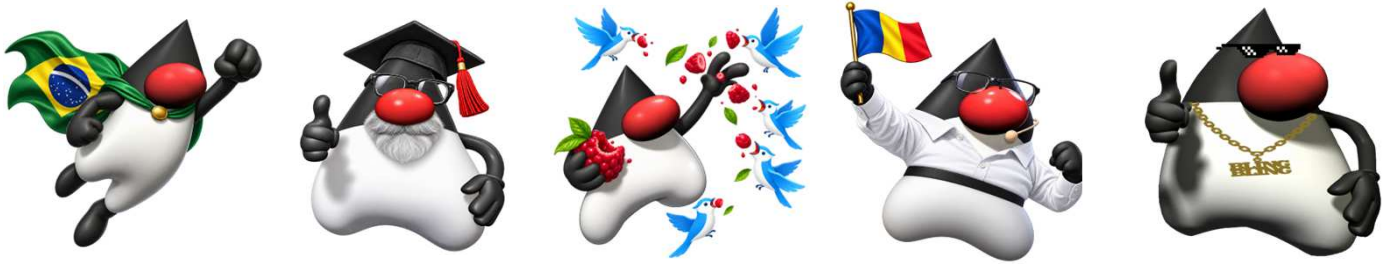
# SORTED versus ORDERED

A collection can be ordered without being sorted.

## TLDR

- Ordered = predictable encounter order.
- Sorted = elements follow a comparison rule.
- Every sorted structure has an order; not every ordered structure is sorted.



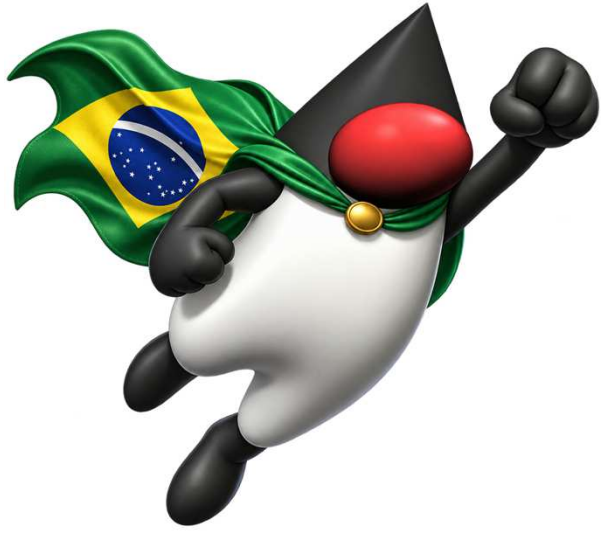

Bruno • Ken • Frank • Vlad • Bling

# 1 ORDERED: INSERTION ORDER

The sequence is preserved, even if it is not alphabetical.

```
var names = List.of(
    "Bruno", "Ken", "Frank", "Vlad"
);

System.out.println(names);
```



A List has an encounter order. Here it keeps the insertion order, but the names are not sorted alphabetically.

**Output**

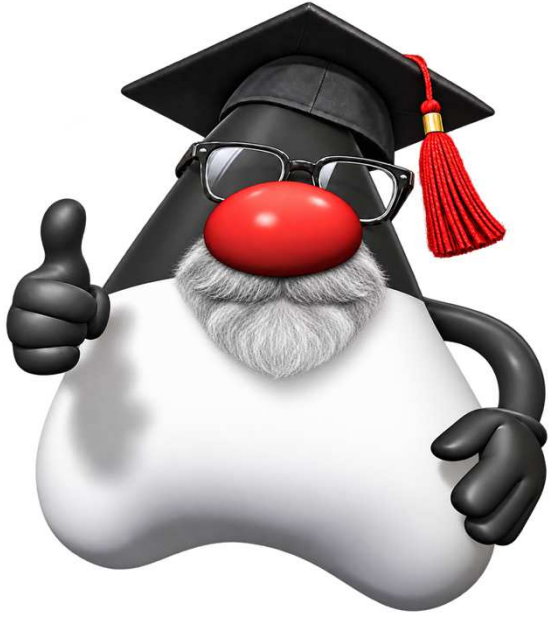
```
[Bruno, Ken, Frank, Vlad]
```

# 2 SORTED: NATURAL ORDER

Sorting changes the order using a rule.

```
var names = new ArrayList<>(
    List.of("Bruno", "Ken", "Frank", "Vlad")
);

names.sort(null);
System.out.println(names);
```



sort(null) uses natural ordering. The List remains ordered, but now its order comes from sorting.

**Output**

```
[Bruno, Frank, Ken, Vlad]
```

# 3 ORDERED SET ≠ SORTED SET

A predictable Set order is not always a sorted order.

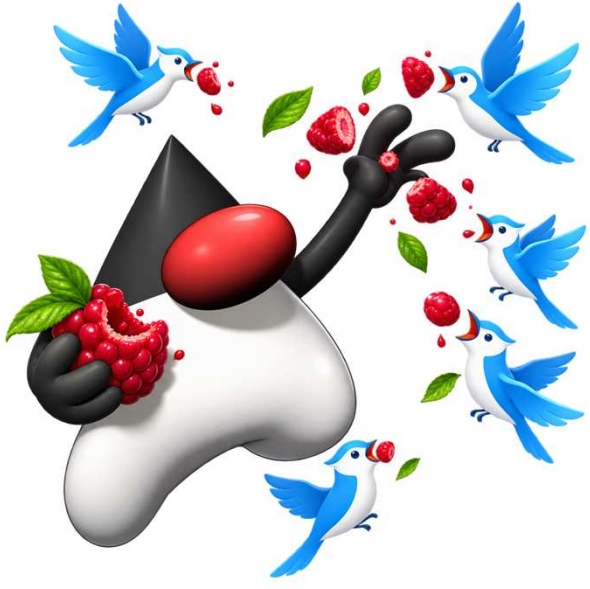
```
Set<String> names = new LinkedHashSet<>();

names.add("Bruno");
names.add("Ken");
names.add("Frank");
names.add("Vlad");
```

LinkedHashSet preserves insertion order. It is ordered, but it does not automatically sort the elements.

**Output**

```
Bruno, Ken, Frank, Vlad
```



# 4 SORTED SET

A sorted structure maintains elements by comparison.

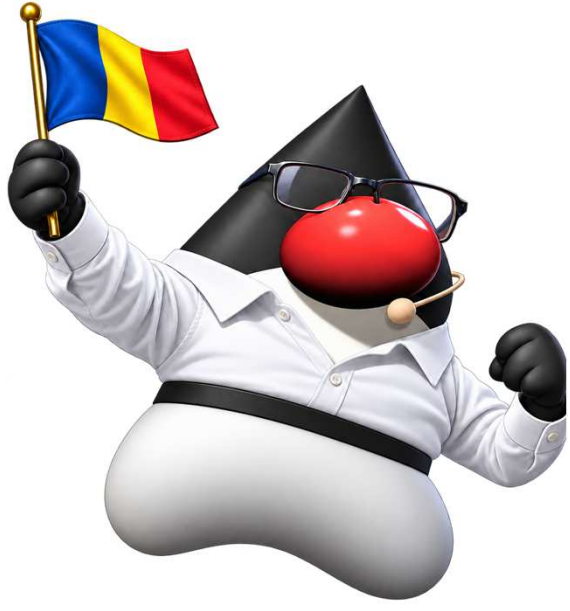
```
Set<String> names = new TreeSet<>();

names.add("Bruno");
names.add("Ken");
names.add("Frank");
names.add("Vlad");
```

TreeSet maintains elements in natural order, or by a supplied Comparator.

**Output**

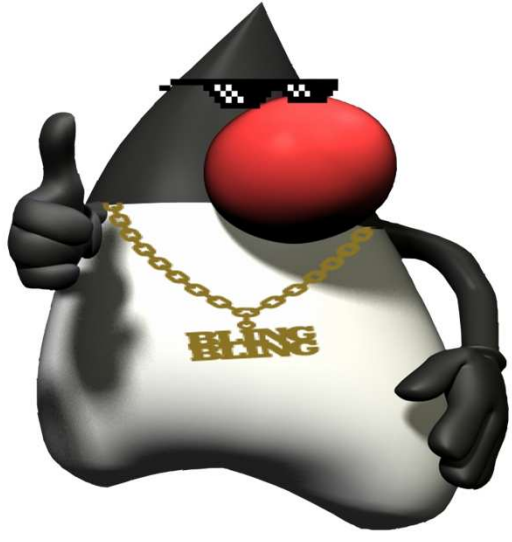
```
Bruno, Frank, Ken, Vlad
```



# 5 STREAM ORDER AND SORTING

Streams can have encounter order, then sorted() can impose a new one.

```
List.of("Bruno", "Ken", "Frank", "Vlad")
    .stream()
    .sorted()
    .forEach(System.out::println);
```




Stream.sorted() explicitly imposes a sorted encounter order. Without it, the stream may keep the source order.

Output

```
Bruno
Frank
Ken
Vlad
```

# TAKEAWAYS

Same word family. Very different contract.

**Ordered →** where elements appear relative to each other.

**Sorted →** why they appear in that order.

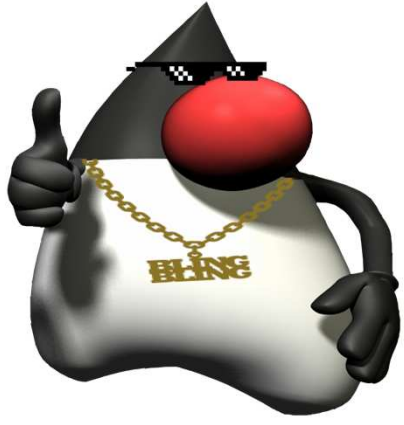
**List →** ordered by index.

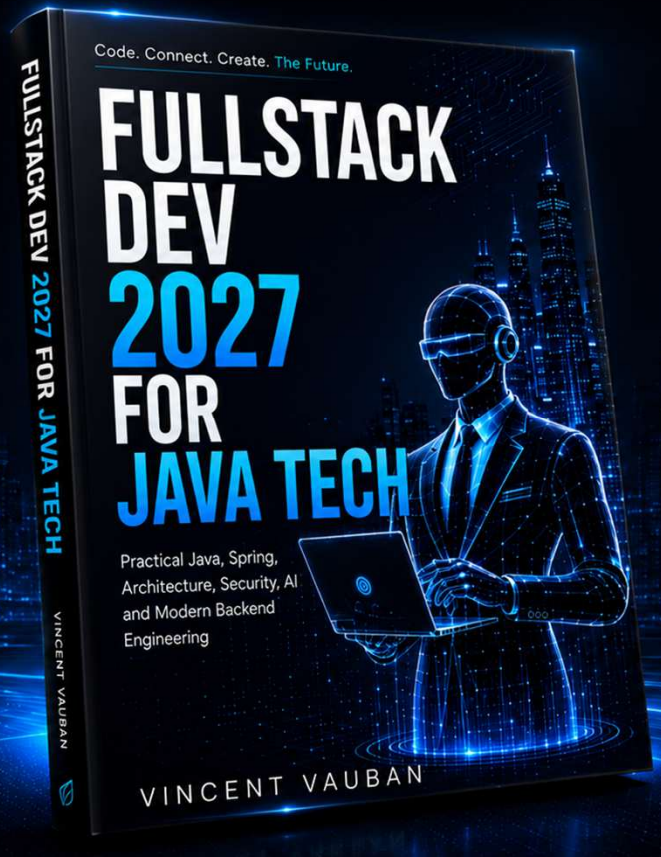
**LinkedHashSet →** insertion-ordered.

**TreeSet →** sorted.

**HashSet →** no defined encounter order.

**Stream.sorted() →** explicitly sorts the stream.






# BOOK RELEASE

Fullstack Dev 2027 for Java Tech is out now

https://bit.ly/jfullstack27