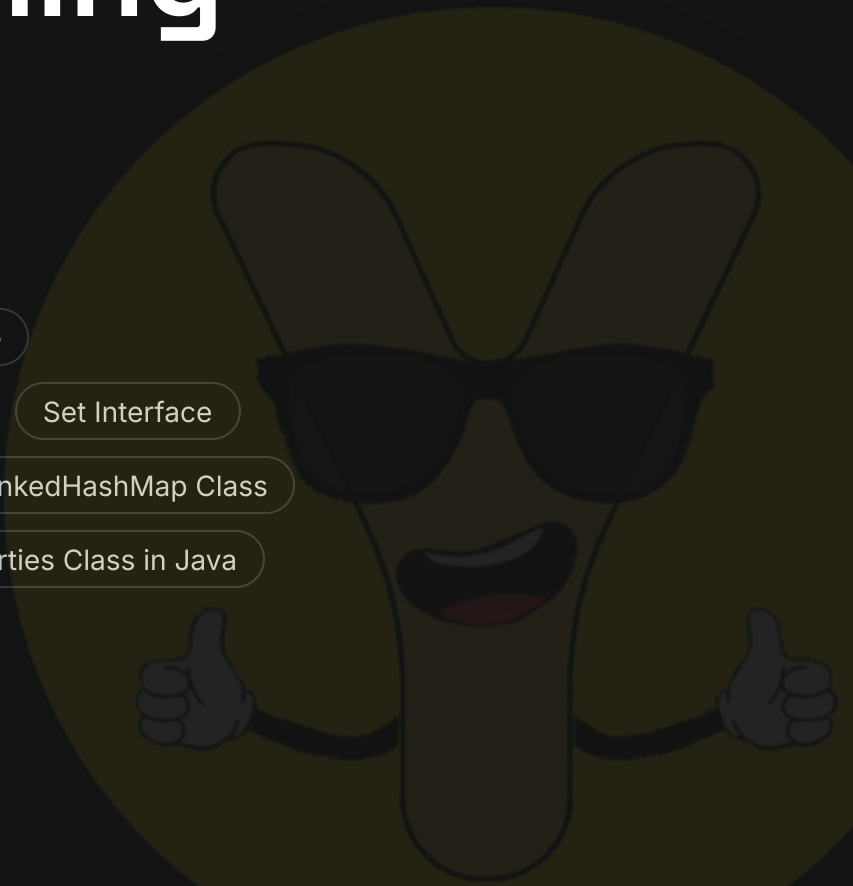
AKTU · 2ND YEAR · SEMESTER 4 · BCS403

Object Oriented Programming with Java

Unit 4 — Java Collections Framework

Collection in Java | Collection Framework in Java | Hierarchy of Collection Framework | Iterator Interface | Collection Interface | List Interface | ArrayList | LinkedList | Vector | Stack | Queue Interface | Set Interface | HashSet | LinkedHashSet | SortedSet Interface | TreeSet | Map Interface | HashMap Class | LinkedHashMap Class | TreeMap Class | Hashtable Class | Sorting | Comparable Interface | Comparator Interface | Properties Class in Java



DECODED BY

Chirag Jain

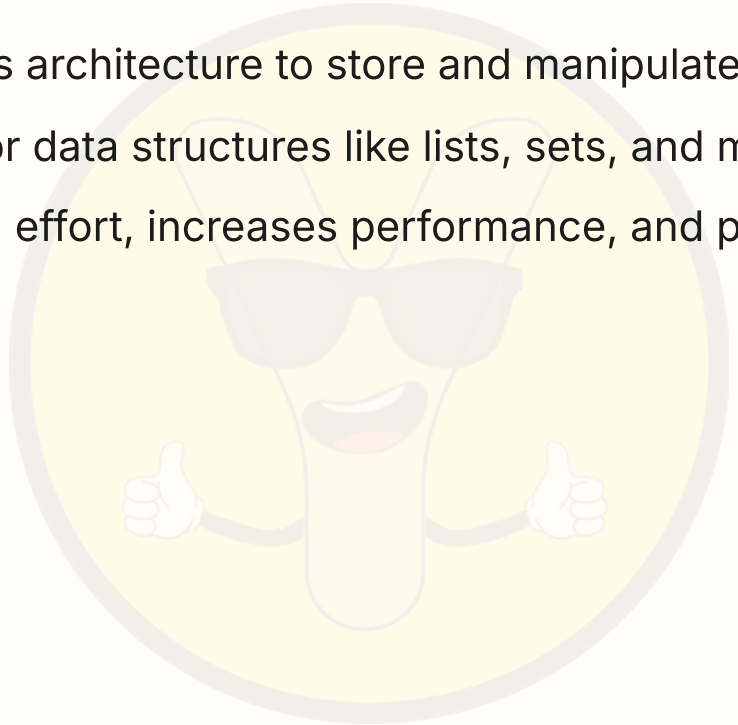
Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

theycotes.com

Introduction to Java Collections

A collection represents a group of objects as a single unit.

- Java Collection Framework provides architecture to store and manipulate a group of objects.
- It includes interfaces and classes for data structures like lists, sets, and maps.
- Advantages: Reduces programming effort, increases performance, and provides interoperability between APIs.
- It is located in the java.util package.



Hierarchy of Collection Framework

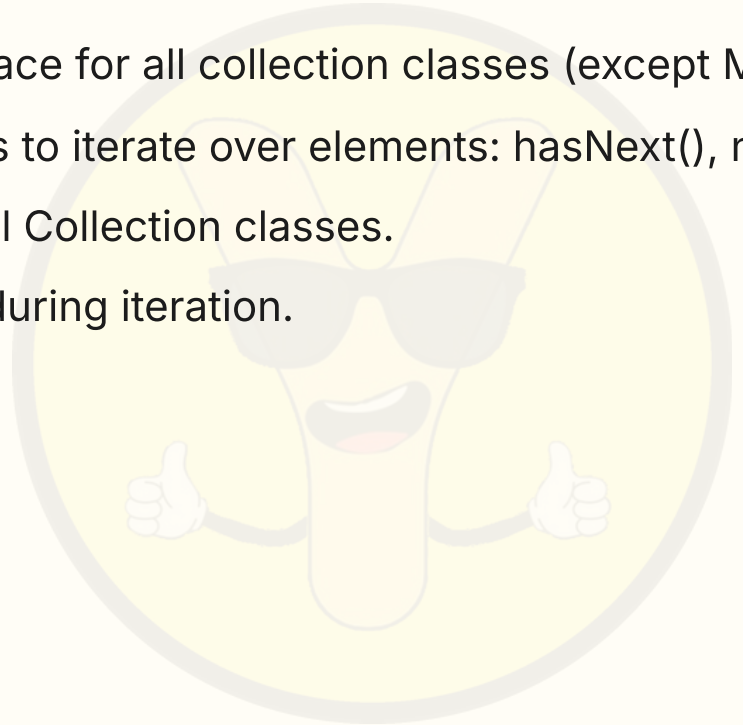


The root interface of the hierarchy is Iterable, followed by Collection.

Iterator and Collection Interfaces

Core interfaces defining the behavior of all collection classes.

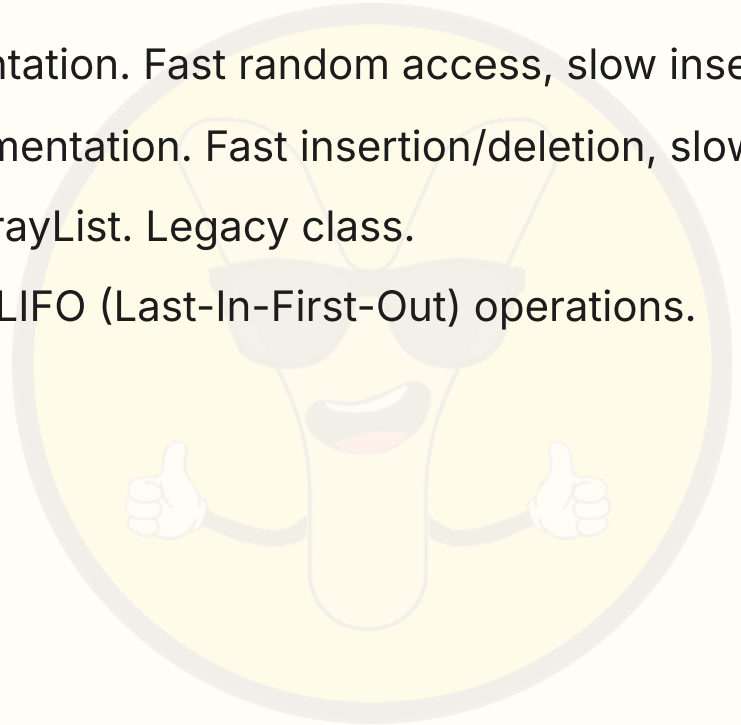
- Collection Interface: The root interface for all collection classes (except Map).
- Iterator Interface: Provides methods to iterate over elements: hasNext(), next(), and remove().
- Iterator is the universal cursor for all Collection classes.
- It allows safe removal of elements during iteration.



List Interface and Implementations

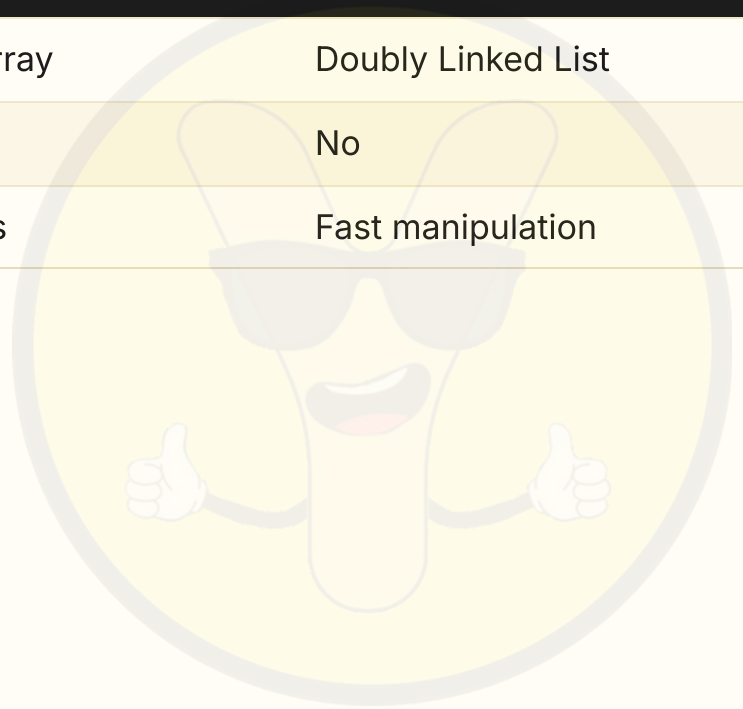
List is an ordered collection that allows duplicate elements.

- ArrayList: Resizable array implementation. Fast random access, slow insertion/deletion.
- LinkedList: Doubly-linked list implementation. Fast insertion/deletion, slow random access.
- Vector: Synchronized version of ArrayList. Legacy class.
- Stack: Extends Vector, implements LIFO (Last-In-First-Out) operations.



Comparison: List Implementations

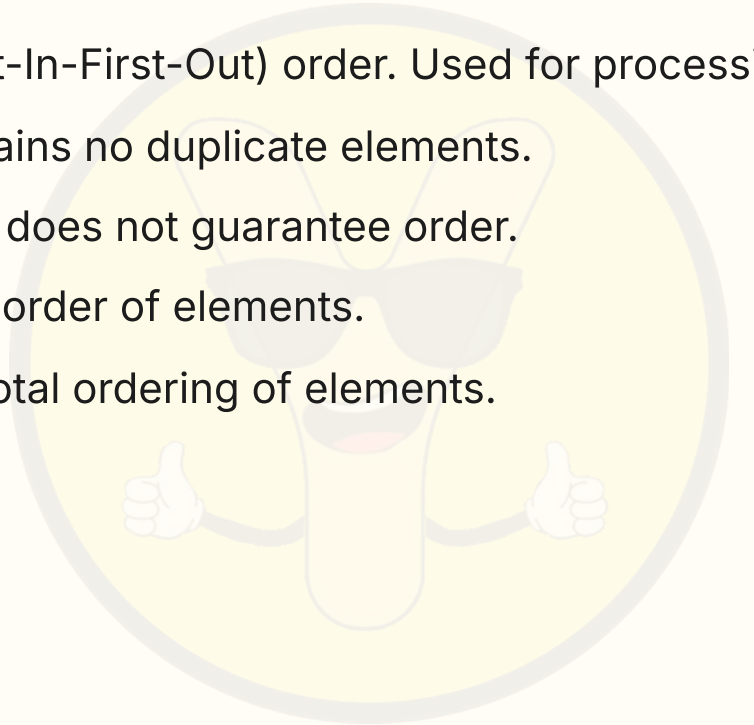
Feature	ArrayList	LinkedList	Vector
Underlying DS	Dynamic Array	Doubly Linked List	Dynamic Array
Synchronization	No	No	Yes
Performance	Fast access	Fast manipulation	Slow (Thread-safe)



Queue and Set Interfaces

Specialized collections for ordering and uniqueness.

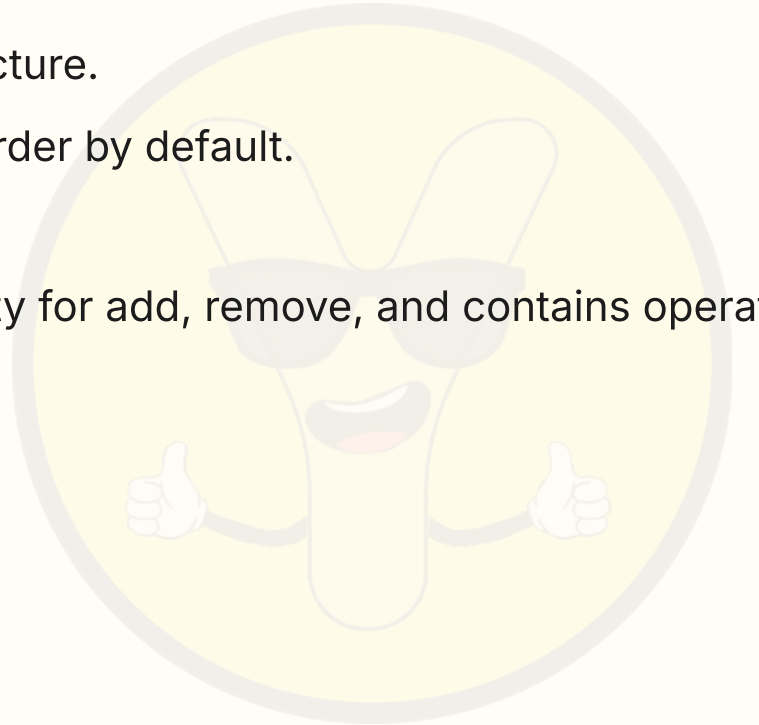
- Queue Interface: Follows FIFO (First-In-First-Out) order. Used for processing elements in order.
- Set Interface: A collection that contains no duplicate elements.
- HashSet: Uses hashing for storage; does not guarantee order.
- LinkedHashSet: Maintains insertion order of elements.
- SortedSet: Interface that provides total ordering of elements.



TreeSet and SortedSet

TreeSet implements SortedSet to keep elements in natural order.

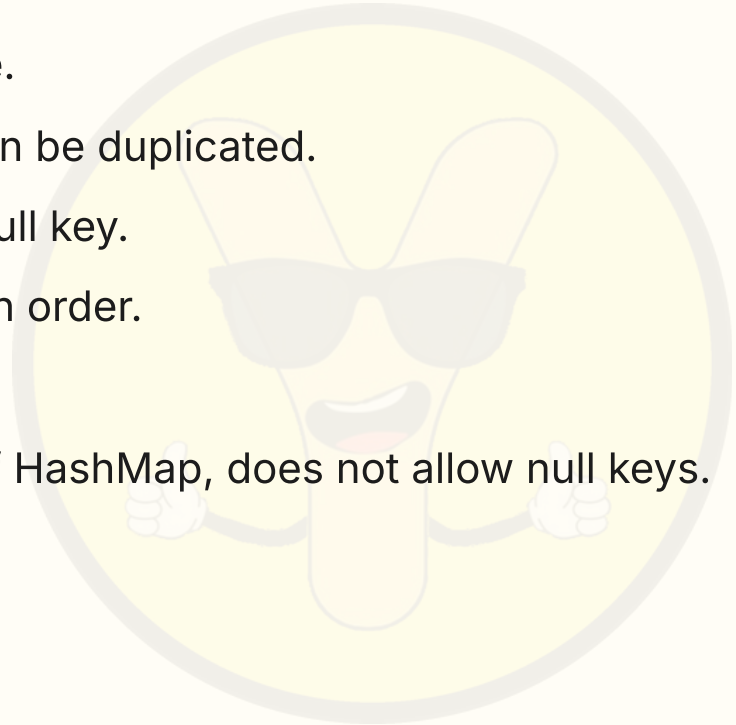
- TreeSet uses a Red-Black tree structure.
- Elements are stored in ascending order by default.
- Does not allow null elements.
- Provides logarithmic time complexity for add, remove, and contains operations.



Map Interface

Map stores data in Key-Value pairs.

- Each key maps to exactly one value.
- Keys must be unique, but values can be duplicated.
- HashMap: Unordered, allows one null key.
- LinkedHashMap: Maintains insertion order.
- TreeMap: Sorted order of keys.
- Hashtable: Synchronized version of HashMap, does not allow null keys.



Map Implementations Comparison

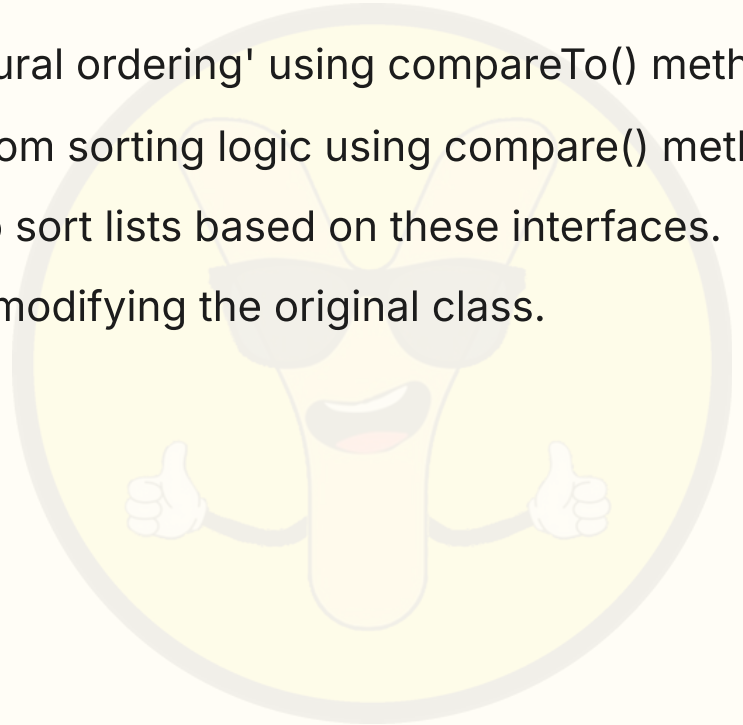
Class	Ordering	Null Key/Value	Thread-safe
HashMap	None	Yes/Yes	No
LinkedHashMap	Insertion	Yes/Yes	No
TreeMap	Natural	No/Yes	No
Hashtable	None	No/No	Yes



Sorting and Comparators

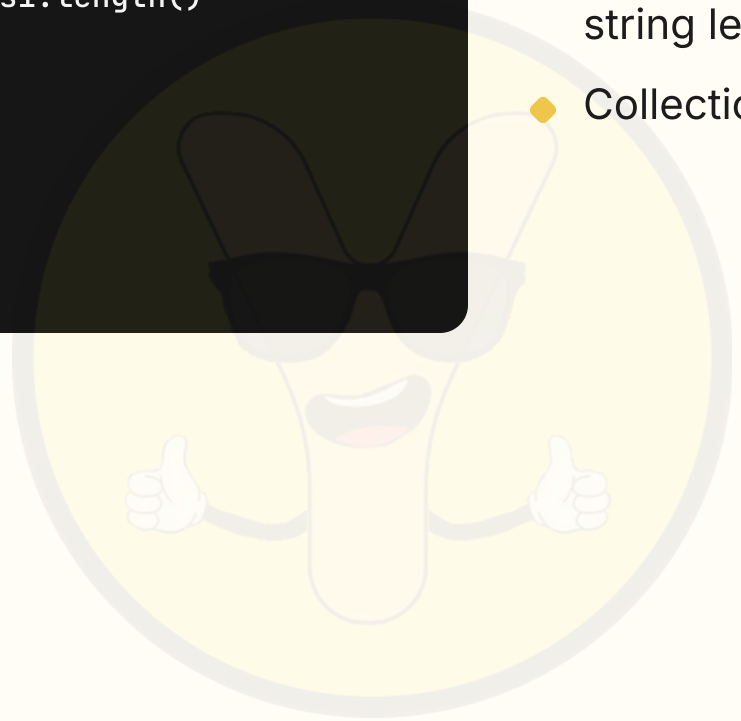
Mechanisms to define custom sorting logic.

- Comparable Interface: Defines 'natural ordering' using compareTo() method.
- Comparator Interface: Defines custom sorting logic using compare() method.
- Collections.sort() method is used to sort lists based on these interfaces.
- Comparator allows sorting without modifying the original class.



Example: Using Comparator

```
Comparator<String> comp = (s1, s2) → s1.length() -
s2.length();
List<String> list = new ArrayList<>();
list.add("Apple");
list.add("Kiwi");
Collections.sort(list, comp);
// Result: [Kiwi, Apple]
```

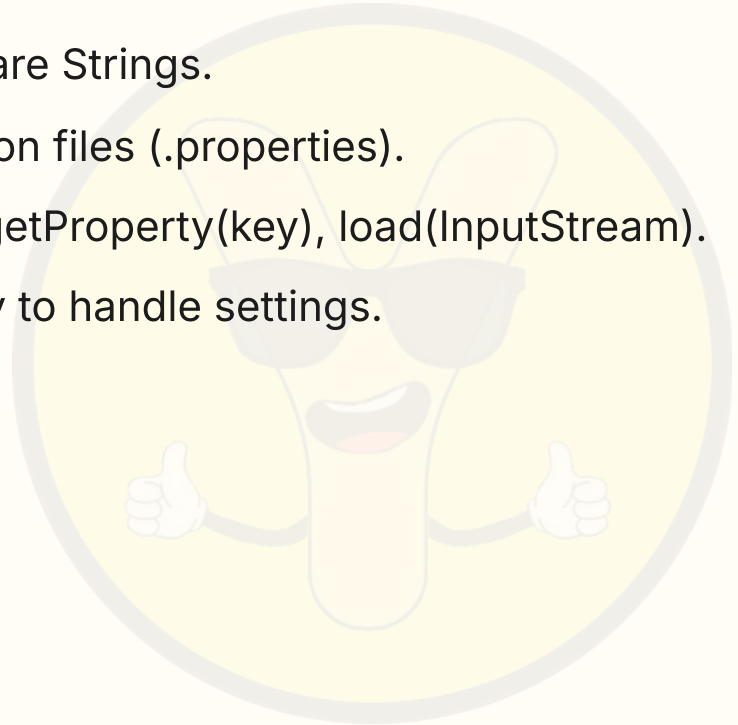


- Uses a lambda expression to define sorting by string length.
- Collections.sort() applies the custom logic.

Properties Class

A subclass of Hashtable used for configuration management.

- Stores key-value pairs where both are Strings.
- Commonly used to load configuration files (.properties).
- Methods: setProperty(key, value), getProperty(key), load(InputStream).
- Provides platform-independent way to handle settings.



Exam corner

Q1 Differentiate between ArrayList and LinkedList with time complexity.

Q2 Explain the hierarchy of the Java Collection Framework.

Q3 What is the difference between Comparable and Comparator interfaces?

Q4 Explain the working of HashMap and how it handles collisions.

Q5 Differentiate between Set and List interfaces.

Q6 What is the role of the Iterator interface in Java?

Q7 Explain the Properties class and its use in Java applications.

Unit 4 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 5 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

