Lec 41 Collections P6:

This session explains how Java uses interfaces like Comparable and Comparator to sort and compare objects, especially within collections such as TreeSet. It also demonstrates the utility of the Collections class for performing common operations like sorting, searching, reversing, and shuffling on data structures.

Map Implementations and Ordering

Different Map implementations in Java—HashMap, Linked HashMap, and Tree Map—are used based on whether you need random order, insertion order, or sorted order for unique keys.

Introduction to Comparable Interface

The Comparable interface in Java allows objects to define their natural ordering by implementing the compareTo() method, which is used by sorted collections like Tree Set.

Example: Tree Set and Integer Sorting

When Integer objects are added to a Tree Set, they are automatically sorted in ascending order because

Integer implements Comparable with a compareTo() method.

Custom Object Sorting: Employee Example

To store custom objects like Employee in a TreeSet, the class must implement Comparable and define the compareTo() method to specify how objects should be ordered (e.g., by ID).

Implementing Comparable in Custom Classes

Without a defined compareTo() method, inserting custom objects in sorted collections leads to runtime exceptions. Implementing and overriding compareTo() enables sorting based on chosen fields.

How Comparison Logic Determines Order

The logic in compareTo() decides the order: a positive result places the current object to the right, a negative to the left, and zero keeps the insertion order. This mechanism is used internally by sorted collections.

Default vs. Custom Sorting Orders

Default natural sorting is defined within the class (e.g., Employee by ID), while custom sorting is implemented outside the class using the Comparator interface for different criteria (e.g., by name or age).

Comparator Interface for Custom Sorting

The Comparator interface allows custom sorting logic to be written in a separate class, enabling sorting of objects based on different fields without changing the original class.

Difference Between Comparable and Comparator

Comparable is used for default sorting within a class via compareTo(), while Comparator provides custom sorting logic externally via the compare() method, allowing multiple sorting strategies.

The Collections Utility Class

The Collections class in Java provides static methods for common operations like sorting, searching, reversing, shuffling, finding min/max, and frequency counting, making collection manipulation easier.

Sorting and Manipulating ArrayLists

ArrayLists preserve insertion order and do not sort by default, but can be sorted using Collections.sort(). Collections class methods enable additional manipulations, such as reversing and shuffling.

Searching and Frequency Operations

Collections class provides methods like binarySearch() for searching and frequency() for counting occurrences, simplifying tasks that would otherwise require manual iteration.

Summary and Interview Relevance

The session concludes by emphasizing the practical importance of understanding Comparable, Comparator, and the Collections class for interviews and real-world Java programming.