CO225-Apr2025 : Software Construction

Lab 09 : Collection Framework Part III

Question 01

Mid Marks Management System with TreeSet

Assume that mid exam marks have been released and you want to order them based on E number as well as the highest mark to lowest marks. Implement a program to manage student marks using `TreeSet` with both natural ordering, meaning based on E number (with `Comparable<T>`) and custom ordering, meaning based on the marks (with `Comparator<T>`).

Use the given skeleton code for implementations.

Task 1: Implement Comparable`<Student>`

- Create a `Student` class with the following attributes:
    - `String name`
    - `int Enumber`
    - `double grade`
- Make `Student` implement `Comparable<Student>` to define natural ordering by `Enumber` in ascending order.
    - Override `compareTo()` to compare students based on `Enumber`.
    - Override `equals()` and `hashCode()` consistently with `compareTo()`.

Task 2: Use TreeSet with Natural Ordering

- In a `main` method:
    - Create a `TreeSet<Student>` and add `Student` objects with given Enumbers/grades. This is already done in the given Skeleton code.
    - Print the set to show ordering by `Enumber`.

Task 3: Custom Comparator for Grade-Based Ordering

- Create a `GradeComparator` class that implements `Comparator<Student>` to sort by `grade` in descending order (highest grade first).
    - *Handle ties by comparing `names` alphabetically.*
- In the `main` method:
    - Create a new `TreeSet<Student>` using `GradeComparator`.
    - Add the 10 students and print the set to show grade-based ordering.