CMSC 132: OBJECT-ORIENTED PROGRAMMING II

Comparator Interface

Department of Computer Science
University of Maryland, College Park

Introduction to the Comparator Interface

- **What is Comparator?**
  - Comparator is an interface in Java used to define custom orderings of objects.

  https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Comparator.html
- Unlike the Comparable interface, which is implemented by the class itself, Comparator is a separate class or object that defines the order of objects.
- API for Comparable as seen in lecture:

  https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Comparable.html

Comparable vs Comparator

Feature	Comparable	Comparator
Purpose	Defines a natural order for objects.	Defines a custom order for objects.
Method Implemented	compareTo(T o)	compare(T o1, T o2)
Use Case	When the class has a single natural order.	When different orderings are needed.
Usage	Typically implemented in the class.	Can be created as a separate object.
Flexibility	Less flexible, only one comparison method.	More flexible, allows multiple orderings.
Key Advantage	Simple and efficient.	More flexible and reusable for different comparisons.

Key Differences Between Comparable and Comparator

- **Comparable**:
  - Modifies the class itself to define the natural ordering of its objects.
  - **compareTo** method is used for comparison.
- **Comparator**:
  - Can be used to create multiple custom orderings without modifying the class.
  - **compare** method is used for comparison.

When to Use Comparable vs Comparator

Use Comparable when:

- You want a class to have a default or natural order.
- The class only needs one way of being compared (e.g., ascending order).

Use Comparator when:

- You want to define multiple ways of comparing objects (e.g., ascending or descending order).
- You don't have access to the source code of the class (e.g., comparing objects from external libraries).

See compare package in LabWeek4 Project