Lecture 21: Ordered Structures

CS 62
Fall 2018
Alexandra Papoutsaki & William Devanny
Comparing Objects

• To compare references
  \( o1==o2 \) or \( o1!=o2 \):
    • Compare to see if reference is \texttt{null}
    • Compare to see if pointing to same object

• To compare object equality
  \( o1.equals(o2) \)
    • Automatically inherited from all classes
    • Already implemented in standard Java classes
    • If not overridden, same as ==
    • Has to be overridden to perform intelligent comparisons for your own classes
Overriding `equals()`

```java
public int compareTo(Ratio that) {
    return this.getNumerator() * that.getDenominator()
            - that.getNumerator() * this.getDenominator();
}
```

```java
public boolean equals(Object that) {
    return compareTo((Ratio)that) == 0;
}
```

Notice that need to cast to `Ratio`, as `equals` requires an `Object`. Need to also implement `hashCode()` (later)
Comparable interface

- Functional interface that imposes *natural ordering* of the objects of each class that implements it

- **`class T implements Comparable<T>`**
  - Must implement method `public int compareTo(T other) {...}
  - Referred to as *natural comparison method*

- `compareTo(T other)` returns:
  - negative if `this < other`
  - 0 if equal
  - Positive if `this > other`

- Should be consistent with `equals`
- `e.compareTo(null)` throws NullPointerException
- `e.equals(null)` returns false
**Sorting Collections**

Collections class contains:

- `public static <T extends Comparable<? super T>> void sort(List<T> list)`
  - Generic methods introduce their own type parameters
  - use `extends` with generics, even if the type parameter implements an interface.
  - the class `T` itself or one of its ancestors implements `Comparable`

- `Collections.sort(list)`
  - Implemented as optimized mergesort
  - If list’s elements do not implement `Comparable`, throw `ClassCastException`
Example: How can we sort associations?

- **public class** Association<K, V>
  - **protected** K theKey; // key of the key-value pair
  - **protected** V theValue; // value of key-value pair

- We want associations where we can order by key
Example: ComparableAssociation

```java
public class ComparableAssociation<K extends Comparable<K>, V>
    extends Association<K, V> implements Comparable<ComparableAssociation<K, V>>{
    public ComparableAssociation(K key, V value) {
        super(key, value);
    }
    public int compareTo(ComparableAssociation<K, V> that) {
        return this.getKey().compareTo(that.getKey());
    }
    ...
}
```

Now we can use Sort!
Comparator interface

- Used when we want to sort some objects in an order other than their natural ordering or if we want to sort some objects that don't implement Comparable.

- `public interface Comparator {
  int compare(T o1, T o2);
}

- Returns:
  - negative if o1 < o2
  - 0 if o1 equal to o2
  - Positive if o1 > o2
Example: how to compare strings

• When comparing strings, leading and trailing whitespaces count
  • “Pomona rocks!”, “Pomona rocks!” and “Pomona rocks!” are all different

• public class TrimComparator implements Comparator<String> {
  /**
   * pre: s1 and s2 are strings
   * post: returns negative, zero, or positive depending on relation between trimmed parameters.
   */
  
  public int compare(String s1, String s2) {
    String s1trim = s1.trim();
    String s2trim = s2.trim();
    return s1trim.compareTo(s2trim);
  }
}
Comparing

Classes supporting `sort` or other operations using comparisons generally have two versions:

- **From Collections class:**
  - `static <T extends Comparable<? super T>> void sort(List<T> list)`
  - `static <T> void sort(List<T> list, Comparator<? super T> c)`
  - `Collections.sort(data, new TrimComparator());`
  - If you try to sort a collection whose elements do not implement `Comparable` or cannot be compared with the `Comparator`, it will throw a `ClassCastException`
Using Lambda expressions

• In Java 8, can use lambda expression rather than Comparator method:
• Collections.sort(data,
    (s1,s2) -> {
        String s1trim = s1.trim();
        String s2trim = s2.trim();
        return s1trim.compareTo(s2trim);
    });
• See TestComparator.java
Ordered Structures in `structure5`

- See `OrderedArrayList.java`, especially `locate` method which does binary search
- Also `OrderedList.java` with singly-linked list implementation
- See text for discussion of operations on ordered structures
  - E.g., find, add, etc.