15: Comparators and Iterators
Lecture 15: Comparators and Iterators

- Interface Comparable
- Interface Comparator
- Sorting
- Iterators

Some slides adopted from Algorithms 4th Edition or COS226
Comparable

- Interface with a single method that we need to implement:
  `public int compareTo(T that)`

- Implement it so that `v.compareTo(w)`:
  - Returns $>0$ if $v$ is greater than $w$.
  - Returns $<0$ if $v$ is smaller than $w$.
  - Returns $0$ if $v$ is equal to $w$.

- Corresponds to natural ordering.
How to make your class T comparable?

1. Implement Comparable<T> interface.

2. Implement compareTo(T that) method to compare this T object to that based on natural ordering.
Lecture 15: Comparators and Iterators

- Interface Comparable
- Interface Comparator
- Sorting
- Iterators
Comparator

- Sometimes the natural ordering is not the type of ordering we want.

- Comparator is an interface which allows us to dictate what kind of ordering we want by implementing the method: `public int compare(T this, T that)`

- Implement it so that `compare(v, w)`:
  - Returns >0 if v is greater than w.
  - Returns <0 if v is smaller than w.
  - Returns 0 if v is equal to w.
How to define an alternative ordering for your class T?

1. Make a new class that implements Comparator\(<T>\) interface.

2. Implement \(\text{compare}(T \ t1, \ T \ t2)\) method to compare \(t1\) object to \(t2\) based on an alternative ordering.

3. Alternatively, implement an anonymous inner class:

```java
public static Comparator\(<T>\) nameOfComparator = new Comparator\(<T>\)()
{
    @Override
    public int compare(T t1, T t2) {
        //return something;
    }
};
```
Lecture 15: Comparators and Iterators

- Interface Comparable
- Interface Comparator
- Sorting
- Iterators
The Java Collections Framework

Collections

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, that is timsort.
  - If list’s elements do not implement `Comparable`, throw `ClassCastException`.
Alternative sorting of Collections

- Collections class contains:
  - `static <T> void sort(List<T> list, Comparator<? super T> c)`
  - `Collections.sort(list, someComparator);`
  - `Collections.sort(list, new ExternalComparatorClass()); or:
  - `Collections.sort(list, T.InnerAnonymousClass);`
  - If list’s elements do not implement Comparable or cannot be compared with Comparator, throw ClassCastException.
Example: Natural and alternative sorting for Employees

https://github.com/pomonacs622020sp/LectureCode/blob/master/ComparatorsIterators/Employee.java

https://stackoverflow.com/questions/2266827/when-to-use-comparable-and-comparator
Lecture 15: Comparators and Iterators

- Interface Comparable
- Interface Comparator
- Sorting
- Iterators
Iterator Interface

- Interface that allows us to traverse a collection one element at a time.

```java
public interface Iterator<E> {
    // returns true if the iteration has more elements
    // that is if next() would return an element instead of throwing an exception
    boolean hasNext();

    // returns the next element in the iteration
    // pre: hasNext has been called
    // post: advances the iterator to the next value
    E next();

    // removes the last element that was returned by next
    default void remove(); // optional, better avoid it altogether
}
```

https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html
Iterator Example for java.util.ArrayList

List<String> myList = new ArrayList<String>();
//… operations on myList

Iterator listIterator = myList.iterator();

while(listIterator.hasNext()){
    String elt = listIterator.next();
    System.out.println(elt);
}
Java8 introduced lambda expressions

- Iterator interface now contains a new method.

```java
default void forEachRemaining(Consumer<? super E> action)
```

- Performs the given action for each remaining element until all elements have been processed or the action throws an exception.

```java
listIterator.forEachRemaining(System.out::println);
```
Iterables Interface

- Interface that allows an object to be the target of a for-each loop:

```java
for(String elt: myList){
    System.out.println(elt);
}
```

```java
interface Iterable<E>{
    //returns an iterator over elements of type E
    Iterator<E> iterator();

    //Performs the given action for each element of the Iterable until all elements
    //have been processed or the action throws an exception.
    default void forEach(Consumer<? super E> action);
}
```

```java
myList.forEach(elt-> {System.out.println(elt)});
myList.forEach(System.out::println);
```

https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html
How to make your data structures of E elements iterable?

1. Implement `Iterable<E>` interface.
2. Make a private inner class that implements the `Iterator<E>` interface.
3. Override `iterator()` method to return an instance of the private inner class.
Example: making our own ArrayList iterable and traversing it

https://github.com/pomonacs622020sp/LectureCode/blob/master/ComparatorsIterators/ArrayList.java
Lecture 15: Comparators and Iterators

- Interface Comparable
- Interface Comparator
- Sorting
- Iterators
Readings:

- **Textbook:**
  - Chapter 2.1 (Page 247), Chapter 2.5 (Pages 338-339)

- **Code:**

- **Oracle Documentation:**
  - Collections: [https://docs.oracle.com/javase/tutorial/collections/intro/index.html](https://docs.oracle.com/javase/tutorial/collections/intro/index.html)
  - Comparable: [https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html)
  - Comparator: [https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html](https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html)
  - Iterator: [https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html](https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html)
  - Iterable: [https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html](https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html)