The Comparable Interface

Anthony Estey

CSC 115: Fundamentals of Programming II

University of Victoria

Comparable Overview

- Learning Objectives: (You should be able to…)
  - write a class that implements the Comparable interface
  - describe when and why it is important for the objects we create to have a compareTo method

The Comparable Interface

- Java's built in Comparable interface:
  - https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html
  - From the link above: "This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's *natural ordering*, and the class's compareTo method is referred to as its *natural comparison method*."

- Why implement the Comparable interface?

1. Often we want to be able to order the items in our collection
2. If the objects in our collection implement the Comparable class, we know the class has a *compareTo* method which allows us to order any two instances of the class

Ordering elements

- Ordering elements is easy with integers…
  - We know that the value 4 is less than the value 10
  - So when determining order for a sorted list or a heap we can use < and >
  - For example:

```
int first = 4;
int second = 10;
System.out.println(first < second); // outputs true
```

Ordering elements

- But we can't use the < to compare other types, like a String:
  - For example:

```
String first = "computer";
String second = "science";
System.out.println(first < second);
```

  - Output: `error: bad operand types for binary operator '<'`

compareTo

- `compareTo` allows us to compare two objects of the same type
  - Any class that implements the **Comparable** interface must have a `compareTo`
  - The String class implements **Comparable**!

- First, let's take a look at the documentation:

  https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html#compareTo(T)

`int compareTo(T o)`

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Using compareTo

```
String first = "computer";
String second = "science";

if (first < second) // but this does not work
```

We want to determine if first should be ordered before second

We can do this with compareTo

```
if (first.compareTo(second) < 0)
```

Similarly, if we wanted to determine:

`if (first > second)` as **`if (first.compareTo(second) > 0)`**

Implementing Comparable

- We will create some objects that implement the Comparable interface
- This means we will need to include a compareTo method in the class
  - And we can define what fields to use from that object to determine how to order this object with another object of the same type
- For example:
  - In a Student class, maybe we order students by their student ID
  - … or maybe we order them by GPA
  - We can decide, and implement the compareTo method accordingly