Comparing Objects

Learning Objectives

- To be able to implement the `Comparable` interface
- To be able to compare objects

Common Java object methods

- Four methods underline many of Java's built-in functionality
  - `equals` – you should be familiar with this one at this point
  - `hashCode` – you will learn about it in your next CIS course
  - `compare` and `compareTo` – we'll talk about today

- Many built-in Java objects (like `String`) define these
  - For your own objects, you must define them

Compare /CompareTo

- We have already seen `compareTo` for String values
  - `firstString.compareTo(secondString);`

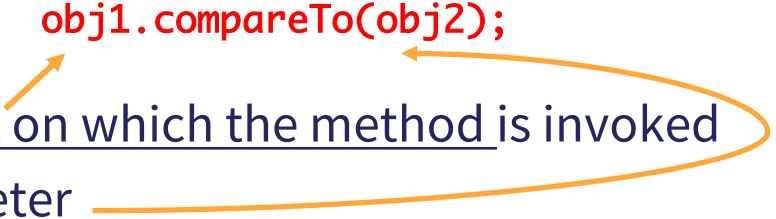
    Will return:
    - **0** if they are equal
    - A **negative value** if firstString is LESS THAN secondString
    - A **positive value** if firstString is GREATHER THAN secondString
- Example:
  - `"hello".compareTo("hello");` ➔ 0
  - `"hello".compareTo( "world");` ➔ a negative value (-15)
  - `"world".compareTo("hello");` ➔ a positive value (15)

The `Comparable` ADT

- Built-in Java interface
- Defines a comparison method: `compareTo`
- A class that implements `Comparable` must provide an implementation of `compareTo`
- The objects of a class that implement `Comparable` are “sortable”

The `Comparable` ADT

- `compareTo`:
  - Compares two objects for order
  - Returns a **negative integer** if the object on which the method is invoked is less than the object passed as parameter
  - Returns zero if the object on which the method is invoked is equal to the object passed as parameter
  - positive integer if the object on which the method is invoked is greater than the object passed as parameter



Making an object sortable

- Simply implement `Comparable`
  - `Comparable` says *"this object can be sorted"*
  - `Comparable` is generically typed, so you have to specify the type

```
public class Student implements Comparable<Student> {
    String name;
    int score;

    public Student (String name, int score) {
        this.name = name;
        this.score = score;
    }

    public String toString() {
        return name + " - " + score;
    }

    @Override
    public int compareTo(Student o) {
        // TODO Auto-generated method stub
        return 0;
    }
```

Implementing `Comparable`

```java
@Override
public int compareTo(Student o) {
    // TODO Auto-generated method stub
    return 0;
}
```

- Only one required method
- Example, `first.compareTo(second)`
- Return 0 if `first.equals(second) == true`
- Return negative if `first “<“ second`
- Return positive if `second “<“ first`
- **The exact value is irrelevant, only the sign matters**

Implementing `Comparable`

- Example we want to compare two students by their last names
- If they have the same last name, then we compare them by their first names
- If they have the same last names and first names, then they are equals

Implementing Comparable

```
public int compareTo(Student s) {
  if (this.lastName.compareTo(s.lastName) < 0) {
    return -1;
  } else if (this.lastName.compareTo(s.lastName) > 0){
    return 1;
  }
  else{
    if (this.firstName.compareTo(s.firstName) < 0) {
      return -1;
    } else if (this.firstName.compareTo(s.firstName) > 0){
      return 1;
    }
    else{
      return 0;
    }
  }
}
```

```
public int compareTo(Student s) {
        if (this.lastName.equals(s.lastName)) {
            return this.firstName.compareTo(s.firstName);
        } else {
            return this.lastName.compareTo(s.lastName);
        }
    }
```

Same logic written differently