# Programming Exercise: Sorting at Scale

For the following assignments, you will start with the files provided, using most of the classes, and modifying only a few of them.

First there are several classes provided from the previous lesson that are unchanged:

- The class **Location**, from the Android platform and revised for this course, a data class representing a geographic location. One of the constructors has parameters latitude and longitude, and one of the public methods is distanceTo.
- The class **QuakeEntry**, from the lesson, which has a constructor that requires latitude, longitude, magnitude, title, and depth. It has several get methods and a toString method. It also has a **compareTo** method to sort earthquakes by magnitude (and commented out code that previously sorted earthquakes by distance to a location). You will be modifying the **compareTo** method in the first assignment.
- The class **EarthQuakeParser**, from the lesson, which has a read method with one String parameter that represents an XML earthquake data file and returns an ArrayList of QuakeEntry objects.

There are several new classes

- The **DifferentSorters** class includes several methods that are similar to code shown in the videos to test several of the classes/methods in this assignment. You'll be adding additional tester methods to this class.
- The **MagnitudeComparator** class implements Comparator to allow one to sort QuakeEntry's by magnitude from smallest to largest magnitude. This method was mentioned in a video.
- The **DistanceComparator** class implements Comparator to allow one to sort QuakeEntry's by their distance to a specified location that is passed in as a parameter. This method was also mentioned in a video.

## Assignment 1: compareTo Method

In this assignment, you will modify the **compareTo** method to sort quakes in a different way. Specifically, for this assignment, you will:

- Modify the **compareTo** method in the **QuakeEntry** class. (You may want to comment out the current code first). The **compareTo** method should now sort quake by magnitude first, from smallest magnitude to largest magnitude, and then break ties (use == operator to determine whether magnitudes are equal) by depth, from smallest (most negative) depth to largest depth.
- Test the **compareTo** method by running the **sortWithCompareTo** method in the **DifferentSorters** class with any data file. The sort used is **Collections.sort**. You should be able to see that the earthquakes are sorted by magnitude, and those with the same magnitude are sorted by depth. Modify this method to print out the QuakeEntry in the ArrayList in position 10 (which is actually the 11th element in the ArrayList) by adding the following code at the end of this method, after sorting and printing out all the elements.

```
int quakeNumber = 10;
System.out.println("Print quake entry in position " + quakeNumber);
System.out.println(list.get(quakeNumber));
```

When you run your method on the file **nov20quakedata.atom**, the element in position 10 that is printed should be:

```
(36.75, -116.15), mag = -0.20, depth = -4200.00, title = 57km ESE of Beatty,
Nevada
```

*This output has been modified to reflect the updated EarthQuakeParser class, 1/12/16.

## Assignment 2: Title Comparator

In this assignment, you will write a Comparator to sort earthquakes by title first and break ties by depth.

Specifically, for this assignment, you will:

- Write the **TitleAndDepthComparator** class that implements a Comparator of type QuakeEntry. In this class you should write the **compare** method that has two parameters, a QuakeEntry named **q1** and a QuakeEntry named **q2**. This method should compare the title of **q1** and **q2**. If **q1**'s title comes before **q2**'s title in alphabetical order, then this method should return a negative integer. If **q1**'s title comes after **q2**'s title, then this method should return a positive integer. If **q1**'s title is the same as **q2**'s title, then this method should compare the depth of the two earthquakes. If **q1**'s depth is less than **q2**'s depth, then this method should return a negative number. If **q1**'s depth is greater than **q2**'s depth, then this method should return a positive integer. Otherwise, this method should return 0.
- Write the void method **sortByTitleAndDepth** in the **DifferentSorters** class. This method should create an EarthQuakeParser, read data from a file on earthquakes and create an ArrayList of QuakeEntry's. Then this method should call **Collections.sort** on this ArrayList and use the **TitleAndDepthComparator** to sort the earthquakes. You should be able to see that the earthquakes are sorted by title first, and those with the same title are sorted by depth. Modify this method to print out the QuakeEntry in the ArrayList in position 10 (which is actually the 11th element in the ArrayList) after sorting and printing out all the elements.

When you run your method on the file **nov20quakedata.atom**, the element in position 10 that is printed should be:

```
(49.76, 155.83), mag = 4.70, depth = -58380.00, title = 104km SSW of
Severo-Kuril'sk, Russia
```

## Assignment 3: Last Word in Title Comparator

In this assignment, you will write a Comparator to sort earthquakes by the last word in their title first and break ties by magnitude.

Specifically, for this assignment, you will:

- Write the **TitleLastAndMagnitudeComparator** class that implements a Comparator of type QuakeEntry. In this class you should write the **compare** method that has two parameters, a QuakeEntry named **q1** and a QuakeEntry named **q2**. This method should compare the last word in the title of **q1** and **q2**. If **q1**'s last word comes before **q2**'s last word in alphabetical order, then this method should return a negative integer. If **q1**'s last word comes after **q2**'s last word, then this method should return a positive integer. If **q1**'s last word is the same as **q2**'s last word, then this method should compare the magnitude of the two earthquakes. If **q1**'s magnitude is less than **q2**'s magnitude, then this method should return a negative number. If **q1**'s magnitude is greater than **q2**'s magnitude, then this method should return a positive integer. Otherwise, this method should return 0.
- Write the void method **sortByLastWordInTitleThenByMagnitude** in the **DifferentSorters** class. This method should create an EarthQuakeParser, read data from a file on earthquakes and create an ArrayList of QuakeEntry's. Then this method should call **Collections.sort** on this ArrayList and use the **TitleLastAndMagnitudeComparator** to sort the earthquakes. You should be able to see that the earthquakes are sorted by the last word in their title, and those with the same last word are sorted by magnitude. Modify this method to print out the QuakeEntry in the ArrayList in position 10 (which is actually the 11th element in the ArrayList) after sorting and printing out all the elements.

When you run your method on the file **nov20quakedata.atom**, the element in position 10 that is printed should be:

```
(64.47, -149.48), mag = 0.40, depth = -16300.00, title = 21km WSW of North
Nenana, Alaska
```