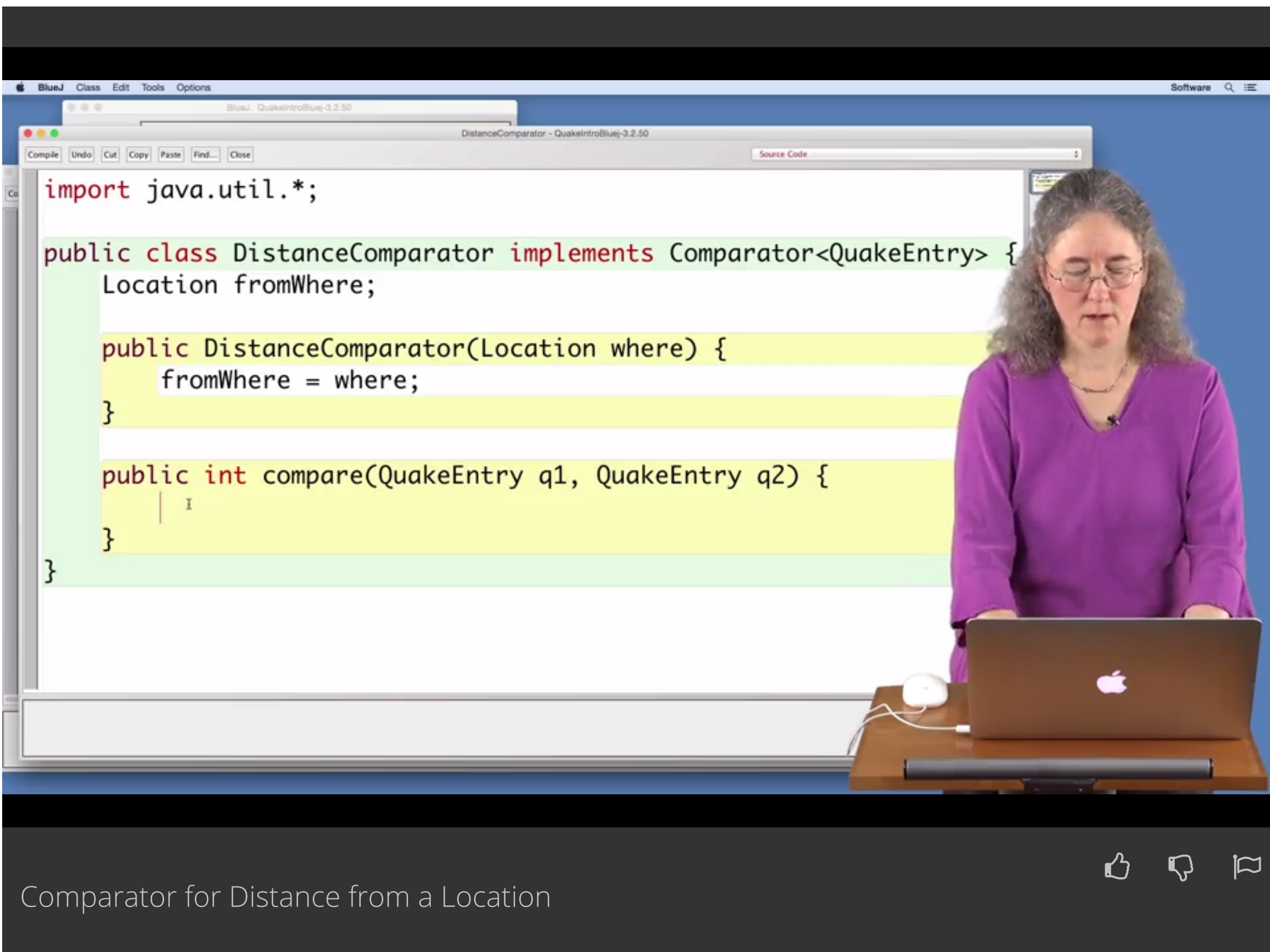


Implementing Selection Sort

Sorting at Scale

▶	Introduction	6 min
▶	Comparable	5 min
▶	Ordering Quakes by Magnitude	8 min
▶	Comparator	5 min
▶	Comparator for Distance from a Location	6 min
▶	Summary	1 min
📄	Programming Exercise: Sorting at Scale	10 min
★	Practice Quiz: Sorting at Scale	3 questions

Review



```
import java.util.*;

public class DistanceComparator implements Comparator<QuakeEntry> {
    Location fromWhere;

    public DistanceComparator(Location where) {
        fromWhere = where;
    }

    public int compare(QuakeEntry q1, QuakeEntry q2) {
        // ...
    }
}
```

Comparator for Distance from a Location

Have a question? Discuss this lecture in the week forums. >

Interactive Transcript

Search Transcript

English ▾

0:03
Hi. We're gonna write a distance comparator. We wanna sort the earthquakes based on their location from a specific location, so I've already started this class here called Distance Comparator. It's gonna implement a comparator of quake entries. You can see that. We're gonna have a variable location called from where. And we've already got the constructor that we're going to pass in the variable where and set where to where. And now we just have to write the compare method. So that's what we're going to do here. We're going to take two quick entries and we want to compare them.

0:47
So, what we're gonna do is, first of all, we're gonna get the distance of the location [of the first earthquake to our variable from where, our parameters from where?](#)

1:01
So, we'll create a variable called distance 1.

1:08
And we'll look at QuakeEntry q1. We'll need to get its location.

1:15
So we'll call the getLocation method. And then from there, we need to calculate the distanceTo fromWhere.

1:32
And then we're gonna do the same thing for the other quake, so we'll create another variable. This one will be called dist2. And then we'll do pretty much the same thing,

1:45
we'll need to look at q2's location.

1:49
Use the getLocation method and the distanceTo.

2:02
And so now we know for each of the two earthquakes q1 and q2 how far they are from our specified location. That we're gonna compare all the earthquakes to. And then we just need to compare them now. So what we're gonna do is we're gonna call the compare method. Double.compare. With these two distances. [NOISE]. Hm. [NOISE]. So we'll compare distance one and distance two.

2:42
And we'll return that value. So that would either return that they're equal by returning zero or we return a positive number or negative number depending on which one is bigger.

2:55
And we'll compile this, and it looks like it compiled and now we want to use the DistanceComparator so we can sort our earthquakes based on. Their location from a specific place. So I have a second method here, a second class here, called DistanceSorter which I've already started and again what we're gonna doing here cuz we're gonna read a bunch of earthquakes in. I have a specific location in there but you could change it. I've got the location set to Durham, North Carolina and then what I'm doing right now is just printing out all that earthquakes. But I would like to sort them first based on their distance to Durham, North Carolina. So I just need to add in one line here so that we can use the distance computer we've just wrote.

3:42
So I'm gonna add in, we're gonna use Collections.sort and

3:50
we're gonna sort a list of quick entries which is called list. So we'll pass that in and then we're gonna specify that we want to use our computer we just wrote. So we need to create a new distance computer.

4:10
And whenever you create the distance comparator you need to pass in one parameter, which is some location you want them all sorted based on their difference to that location. And so we're just gonna pass in where, which happens to be in this case Durham, North Carolina.

4:26
And so that should sort all the earthquakes based on their distance and location and then we'll print them out and we should see that. So let me just see if this compiles.

4:37
Okay so we did good no syntax errors. We can then run our program. We'll go to Bluejay.

4:44
And we'll just create a new distance order.

4:50
And we'll run the sort by distance method we just modified and now we're getting all kinds of data. Let's see what we got. So, if we look at this [COUGH] you can see that they're all sorted, cuz they're close to, these are all in Indonesia. We're gonna scroll up, these are in Japan, now we're gonna slowly get closer to Durham, North Carolina. Now we're looking at Greece, we're looking at Alaska, we're looking at Hawaii. [SOUND] Keep going. And let's see what we get. A lot of earthquakes in Alaska.

5:25
Now we get California, lots of earthquakes in there. And then finally Oregon. We're going all the way up.

5:33
And then finally we see there was one earthquake in North Carolina, one in South Carolina, one in Georgia. But you can see the first ones are the one that are closest to Durham, North Carolina. So it looks like our earthquakes are sorted by their location from a specific space. All right, happy coding.

Downloads

Lecture Video	mp4
Subtitles (English)	WebVTT
Transcript (English)	txt

Would you like to [help us translate](#) the transcript and subtitles into additional languages?