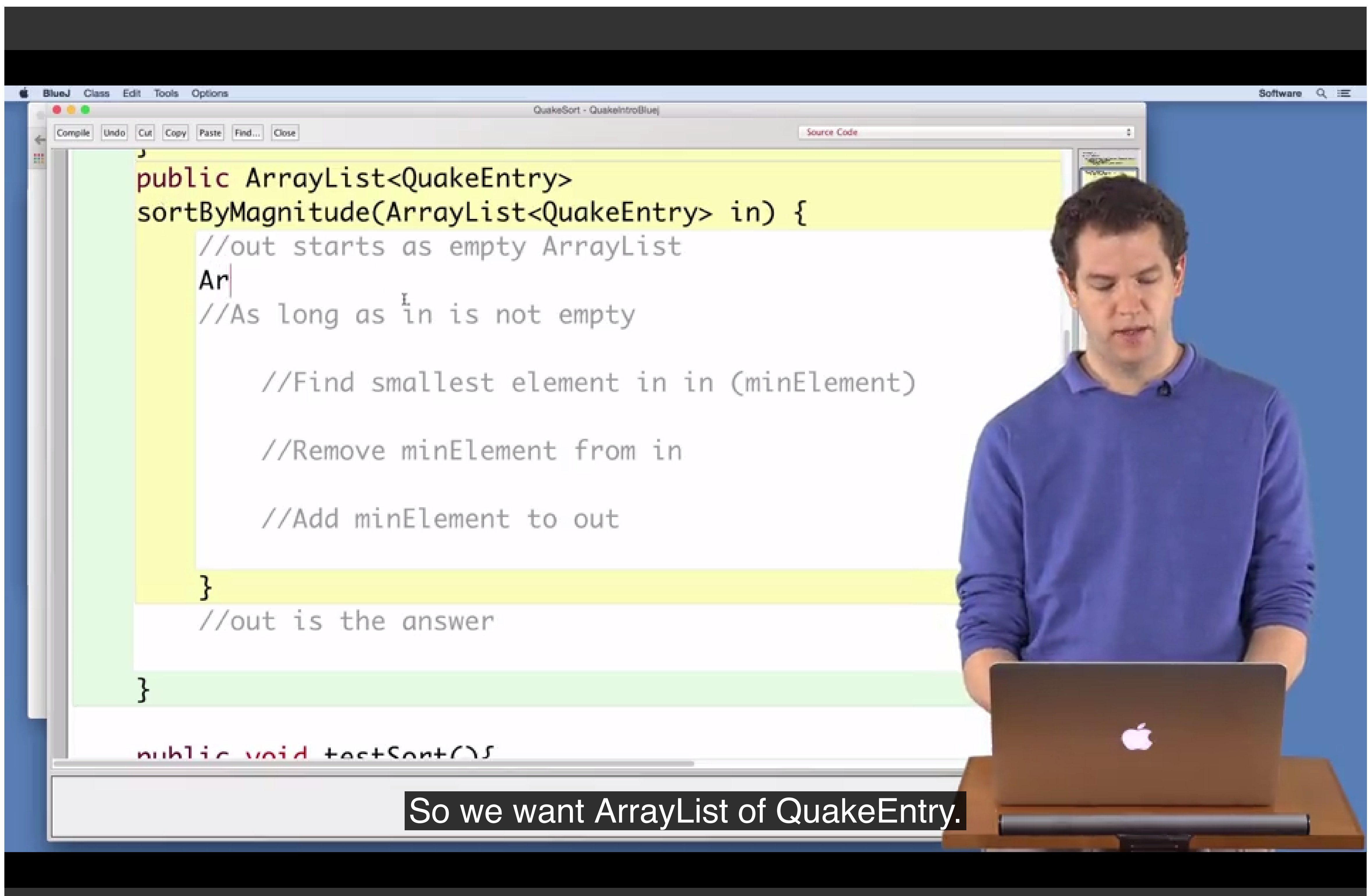

```
public ArrayList<QuakeEntry>
sortByMagnitude(ArrayList<QuakeEntry> in) {
    //out starts as empty ArrayList
    Ar
    //As long as in is not empty

        //Find smallest element in in (minElement)

        //Remove minElement from in

        //Add minElement to out

    }
    //out is the answer

}

public void testSort(){
```

So we want ArrayList of QuakeEntry.

Translating to Code

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

Interactive Transcript

Search Transcript English

0:03
Okay, so now you've developed the algorithm to sort earthquakes by magnitude, it's time to turn that into code. Here we've declared a method, sortByMagnitude which takes an ArrayList of quake entries as its input and returns an ArrayList of quake entries as its output. Having written this over two lines doesn't make a difference, it was just getting really long. So, the first thing we said to do was out starts as an empty ArrayList. So we want ArrayList of QuakeEntry. Out is going to be a new ArrayList of QuakeEntry.

0:38
And then we said as long as in is not empty. So remember, we talked about how a while loop is the most appropriate kind of loop for this type of repetition.

0:52
We wanna know if in is not empty. So we might write something like in.size() > 0, but it turns out if we look at the documentation for ArrayLists, this is just the Java documentation for ArrayLists, we'll see that it is empty which returns true if it contains no elements. So it'd be a little better to just write, as long as it's not empty.

1:18
We wanna find the smallest element and call it minElement, so minElement is going to equal something. We need to declare minElement, it's a new variable. What type of thing would it be? A QuakeEntry. And finding the minimum element seems a little bit complicated. So we should write a method to do this. We've already written one before, getSmallestMagnitude, which takes an ArrayList of QuakeEntry and returns a QuakeEntry. This should be a very familiar computation to you by now since you've found the min and max of several different types of data across these past courses. So we're just going to make use of getSmallestMagnitude, which is going to take in and return a minElement. Now we want to remove that from in so, in.remove(minElement). Again, if you weren't sure of this, you could look at the documentation and find that it has remove method in there.

2:20
And then we want to add minElement to out.add(minElement) which will put it on the back, where is exactly where we want it, and then at the end out is our answer, return out.

2:33
So we're gonna hit Compile here. There are no errors, so that's good. We've already written a method here that will test this by reading in some data, sorting by magnitude, and then printing the result. So if we go over here, to our main BlueJ window, and we make a new QuakeSort and then we do testSort(). You'll see it printing out all of these different quakes and if you look at the magnitudes of them, you'll see starting up here at the top there are lots of earthquakes. The magnitudes are all very small and as we go down they get bigger and bigger and bigger. Until we get to the biggest earthquakes at the end. So we've sorted our earthquakes by magnitude which is exactly what we wanted to do.

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?