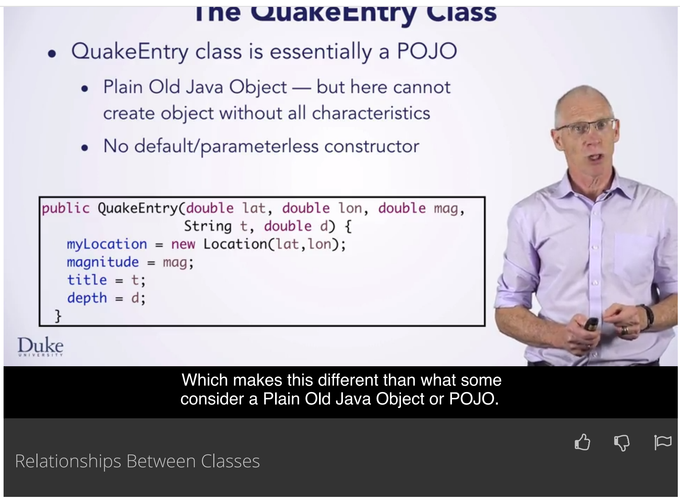
Introduction

Module Learning Outcomes / Resources 10 min
Introduction 7 min
Relationships Between Classes 4 min
Licensing and APIs 3 min
Location class documentation and license resources 10 min
Coding a Magnitude Filter 10 min
Coding the Closest Quakes 12 min
Summary 2 min
Programming Exercise: Searching Earthquake Data 10 min
Practice Quiz: Searching Earthquake Data 6 questions

Filtering Data

Review



Which makes this different than what some consider a Plain Old Java Object or POJO.

Relationships Between Classes

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

Interactive Transcript

Search Transcript | English

0:03
Hi in this lesson we're going to talk about a few concepts related to how classes are used together in object oriented programming in general, and specifically when using Java and the earthquake programs and classes you'll see. The QuakeEntry Class created in QuakeEntry.Java. Encapsulates the basic characteristics of one earthquake. Where the quake occurred, the quake's magnitude and depth, and a title for the quake. This makes QuakeEntry what's called a POJO, or Plain Old Java Object. This means a QuakeEntry object is not much more than the characteristics of the earthquake. Creating a QuakeEntry object requires providing all of these characteristics. For example, it doesn't make sense to talk about an earthquake without knowing where it occurred or how strong it is. It's not possible to creat a QuakeEntry object without supplying all the parameters of the constructor. Which makes this different than what some consider a Plain Old Java Object or POJO. As we'll see QuakeEntry objects are created when data for earthquakes is red and parched. Creating a QuakeEntry object requires supplying all the information for a quake. The latitude, the longitude, the magnitude, the tile and the depth. A QuakeEntry object represents the data for an earthquake that has occurred somewhere in the world. So it's immutable and doesn't change once constructed. Accessing the state of an QuakeEntry object is done using getter methods as you see here. Get location returns the location of a quake. Get depth returns the depth of the quake and so on. A QuakeEntry object also has a reasonable .toString method to help with printing information about an earthquake.

1:53
An earthquake occurs at a specific location, and so we'll use a location class for that. Location classes have many uses outside of the QuakeEntry class we're studying here. Smartphones use locations. Web browsers use locations and many other applications do as well. We have a choice between creating a location class that's simple that we could use and study for this course or we could use an industrial strength class that would be tested and useful in other contexts. We've adopted the location class from the Android platform. This gives us the best of several options. We can be confident that the class is well tested and the licensing of the source code allows us to use and adapt the code as we need to. We'll see more about this later.

2:39
Location objects are created from a latitude and a longitude. These specify one location on Earth. Typically these values might come from your phone which get data from GPS satellites or your web browser which can determine location based on your IP address. The Android location class has a dot distance to method to determine the distance between any two locations A and B. This will allow you to find earthquakes close to where you live.

3:06
This makes the location class more than a Plain Old Java Object or POJO. It has state, the latitude, and the longitude, but it also has behavior, a method to determine the distance between locations.

3:21
Classes in Java can have both Has-A, and Uses-A relationships. The earthquake partial class creates QuakeEntry objects when data is read and parsed. This BlueJ class diagram shows how the parser uses QuakeEntry objects by creating them. A QuakeEntry object also creates a location object as seen in the diagram. The QuakeEntry class creates a location object by passing the latitude and longitude to the location object. This creates a new location object using this data.

3:57
The location object created is stored in instance field myLocation. When one class Has-A or contains another class, we call this a Has-A relationship. The location class implements comparable, we'll discuss this at another lesson. As you can see the BlueJ diagram, the earthquake client class also uses a location object. Locations are used to determine which earthquakes are closest to where you live, or to where I live. We'll see this in the coding demo. The location .distanceTo method helps in writing this code.

4:32
A QuakeEntry object is also used by the EarthQuakeClient class. For example, this allows client programs to find all quakes of high magnitude using the getMagnitude method of the QuakeEntry class. Happy programming.

Downloads

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

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