CS 240: Inner Class introduction Transcript

This video shows a split screen of Professor Wilkerson on the right and a web browser showing Java on Oracle on the left. Visual descriptions are not needed.

[00:00:00] **JEROD WILKERSON:** You've probably already seen so far at some point in this class, a class declared inside of another class; that's called an inner class.

[00:00:08] Now I'm going to give you a lot of details about how inner classes work and why we need them in Java.

[00:00:14] I want to introduce this with an example that will help illustrate one way in which inner classes can be useful.

[00:00:25] I want you to think about two different data structures: the list data structure and the set data structure.

[00:00:31] You would have learned about those in 235.

[00:00:34] As you know, a list is an ordered collection of elements.

[00:00:37] That's a collection of elements where order matters and you can access things by order, usually using something like a get method where you specify an index.

[00:00:46] We also have a set collection, where you have a collection of elements again, but in this case, order isn't relevant.

[00:00:55] You can't necessarily get things by order because there is no defined order for a set.

[00:01:00] With either of these collections, it's reasonable to want to be able to do something to each element in the collection one time.

[00:01:08] For example, if you have a list, you may want to process that list with a for loop and do something to every element in the list, call a method on it or do a calculation or something like that.

[00:01:17] The same thing might be true of a set.

[00:01:18] You might want to be able to see every element of the set exactly once so you can do something, but there's no order to a set.

[00:01:28] If you think about that, it'd probably be pretty easy to write a for loop that will loop through a list and let you see each element one time and do something to it.

[00:01:37] But how would you do that with a set? That's a little bit trickier.

[00:01:40] There would also be a benefit to having a common way to do that.

[00:01:44] It would be nice to have a way that seems so common to want to iterate a collection and do something to every element of the collection.

[00:01:51] That really should be a common way that we can do that, that we can see every element in a collection once so we can do something with it.

[00:01:59] There actually is, and it's built into the collection classes that come with Java, and it uses a design pattern called the iterator pattern.

[00:02:07] The iterator pattern is a pattern that provides some methods that we can call.

[00:02:16] Actually. Let me show you, first of all.

[00:02:18] I brought up the documentation for list and you can see that list has a method called Iterator.

[00:02:25] An iterator returns an instance of Iterator for a specific type.

[00:02:31] If you think about that, I'm calling up a method that's returning an object that I'm somehow going to be able to use to iterate through the elements of this list.

[00:02:43] Let's go and see what Iterator looks like.

[00:02:46] If I click on Iterator, that's an interface, and it has some methods.

[00:02:50] Let's focus on just these two methods.

[00:02:53] It has a hasNext method and a next method.

[00:02:56] I could use those two methods in a loop to be able to process everything in that list.

[00:03:02] I can have a while loop where I say while whatever my reference's iterator dot hasNext, then inside the loop, call next, and that will give me the next element.

[00:03:12] Now remember that would be easy to do without an iterator from a list, but not so easy from a set.

[00:03:18] There's a benefit to being able to iterate all collections in the same way.

[00:03:22] All collections have this ability to get an iterator.

[00:03:27] You can see it here in the list documentation, but if I switch over to the set documentation, you can see the same thing.

[00:03:37] You can see it has an Iterator.

[00:03:40] Now let's think about those iterators though.

[00:03:42] Think about how you might write them.

[00:03:44] An iterator for a list would be pretty easy to write a simple for loop, but how would you write it for a set? It probably depends on how the set is implemented.

[00:03:54] We'd have to know something about the way the set is storing its values internally in order to know how to pass them back.

[00:04:01] Every time you call next, passing in a different , one back, I'm not seeing anyone more than once.

[00:04:07] We don't really know how to write that until we know more details about how set is implemented.

[00:04:12] Since that is an interface, there could be different kind of sets, and maybe we have to implement it different ways for different sets.

[00:04:19] My point is that conceptually, you're just iterating, but physically, you're doing different things, which means there needs to be different code.

[00:04:28] Each class needs to have a different implementation of the iterator.

[00:04:33] Think about that for a minute and ask yourself.

[00:04:36] Where should that class be? Where should we define that Iterator class? Well, my answer for that is it should be defined inside of the class that needs it.

[00:04:46] Since iterators are different for different implementations, the class that needs a particular iterator is a great place to put it, and that is an example of why we might want an inner class.

[00:04:58] We can define a special iterator for each set inside of each set implementation class, just create a separate class that implements iterator.

[00:05:09] We do the same thing inside of the list.

[00:05:11] That actually is the way it's implemented.

[00:05:12] If you look inside of the source code for Java, you'll find that each of those collection classes has a class inside of it that implements Iterator.

[00:05:20] One other thing that you can see from this example is another principle.

[00:05:24] The principle is programmed to the interface.

[00:05:27] When we call Iterator, we get some object that implements the Iterator interface, but we don't even know what the data type of the object is,and we don't care.

[00:05:37] We just know that whatever it is, it implements Iterator.

[00:05:40] That's the reference type that we have, and we know that we can call those methods on it.

[00:05:46] That's a pretty useful concept.

[00:05:48] That's just one of many examples of where I would find an inner class useful.

[00:05:53] There are other examples too, and event handling in Java is a great place where we use inner classes as well.

[00:06:01] In later videos, I'm going to show you specific examples that use this concept where we have an iterator as an inner class.

[00:06:11] I will show you some different ways to do inner classes to provide something like an iterator.

[00:06:17] That will help you to understand the rules of inner classes and how they work.