CS 240: Using Generic Classes Transcript

Start slide description.

Jerod Wilkerson presents a recorded lecture with the aid of a slide deck. All visual content is described in the audio.

End slide description.

[00:00:00] **JEROD WILKERSON:** The main way we use generic types is to create a generic class and then instantiate it as I showed in the previous video. In fact, more often than not, the generic class already exists and you're just instantiating it.

[00:00:12] But there are a few details that you should be aware of and how we use generic types or how we create generic classes. One is when we create a class that extends a generic class.

[00:00:23] So, in the previous video, I showed you an example of creating a pair class, which is generic. And now in this example, we can see how we can extend that pair class with the non-generic class.

[00:00:34] So, here we're creating a class called "string pair" and in my string pair class and the reason I'm calling it string pair is because I want both items and the both elements in the pair to be strings.

[00:00:47] So, because I know that when I create the class, I don't need to ask the user to specify the generic types at runtime, but since it's a subclass of pair, I need the subclass to specify that the generic types are string, so I can do that in the extends class.

[00:01:01] So, if you look right here, I have string pair, extends pair and it specifies the string parameters as it specifies the actual data types for the generics in its class that it's inheriting from.

[00:01:15] And so, that makes it so when I create the string pair, I don't specify generic type, so I just create the string pair like I would create any other object and I give it strings in the constructor because I've specified that it's going to be strings right here, so that's one way that we can use generic types. And we can also specify some generic type.

[00:01:42] We can have some generic type specified in a subclass and others I'm determined to make extends class. So, if we look right here, if I have a key value pair and usually a key doesn't have to be, but it's usually a string.

[00:01:57] So, if I create a key value pair of class, I probably want the keys to be of type string, so I don't need to ask the user to specify that type. But the other element and the pair, I still don't know what that's going to be until runtime.

[00:02:11] So, here I have declared a generic type V for the key value pair. And then in my extends class, I'm still extending pairs, so I need to specify two types, so I specify a string for the first one and whatever gets passed in for V as the second one.

[00:02:27] So, if I have a generic type for the class, I can use that generic type throughout the code of the class, but I can also use it in the extends class.

[00:02:37] So, also notice right here in my key value pair in my constructor, I am using that V type right here for the second parameter to this constructor and the first one I know it's going to be string, it's not generic, so it's specified as a string.

[00:02:56] You also need to know how to use generic interfaces. So, I have a little coding example for that so we can create an interface that uses generic types.

[00:03:06] So, here I have this interface called "Function," and it has two generic types, TNR and it has one method. It has on a play method that takes something of type T and it applies something to it, apply some logic to it and it returns something of a different type, of type R. So, when I create the instance of the function, I can specify what type has passed in as a parameter and what type is returned.

[00:03:30] So, that's my interface. And now that I have the interface, I can create some number of subclasses or implementing classes of the interface. So, here I have a class called "Capitalizer" that implements the interface.

[00:03:43] And just like when I extend a generic class, I can specify the generic types, I can do the same thing when I implement an interface. So, here I'm saying that for this function, for this capitalizer function, it will take a string as a parameter and it will return a string. And then the applying method is going to change it for this capitalizer the way we want to change the string if we want to capitalize it.

[00:04:05] So, in my code I take the string parameter, I check to make sure it's not "No." And if it's not "No," I just call the strings to uppercase and I end up getting an uppercase string.

[00:04:18] So, now that I have done this, I can pass this capitalizer class or an instance of this capitalizer class into any method that expects to receive a function, so we can see that. Right here, I have a classical string manipulator.

[00:04:36] Its purpose is to change strings. And we don't know how it's going to change them because that's determined by the function. So, in my string manipulator, I pass the string to it and I pass some function that can manipulate that string.

[00:04:49] And all the code does is it calls the "manipulate" or it calls the "apply method" and whatever function I pass in.

[00:04:57] So, we want to see this in action, so I'll bring up my code editor and show you how this actually works. So, here you see the code again, here's my function, that takes the two generic types. I have one class that implements the function called "capitalizer" that will just capitalize whatever string I pass in.

[00:05:19] And I have this space remover that will remove spaces from whatever string I pass in. Now in real life, this is only going to remove a single space.

[00:05:29] So, I'd probably want to add more code and be able to remove tabs and things like that, but this is just a little example. So, it's fine that we're just removing spaces.

[00:05:37] So, now if I look at my string manipulator, this is the code that you saw in on the slide where we're passing some string that should be manipulated. We're passing the function that can manipulate it, and the code just calls the function on the string.

[00:05:53] So, I have a main method here, open that up. So, my main method first creates an instance of string manipulator, and then it calls "manipulate string" two different times and prints out the result.

[00:06:11] So, first it calls manipulate string with "this string" "my string," and it creates an instance of the capitalizer as the function that should change the string. And then it does that again, it passes the different string and it passes in the space remover as the function that should change the string.

[00:06:29] So, we run that and you can already tell what's going to happen. So, here are my string. The first time it was manipulate string was called to capitalize the string, and the second time it took the spaces out. So, that is the way you use interfaces. That's the way you use generic interfaces.