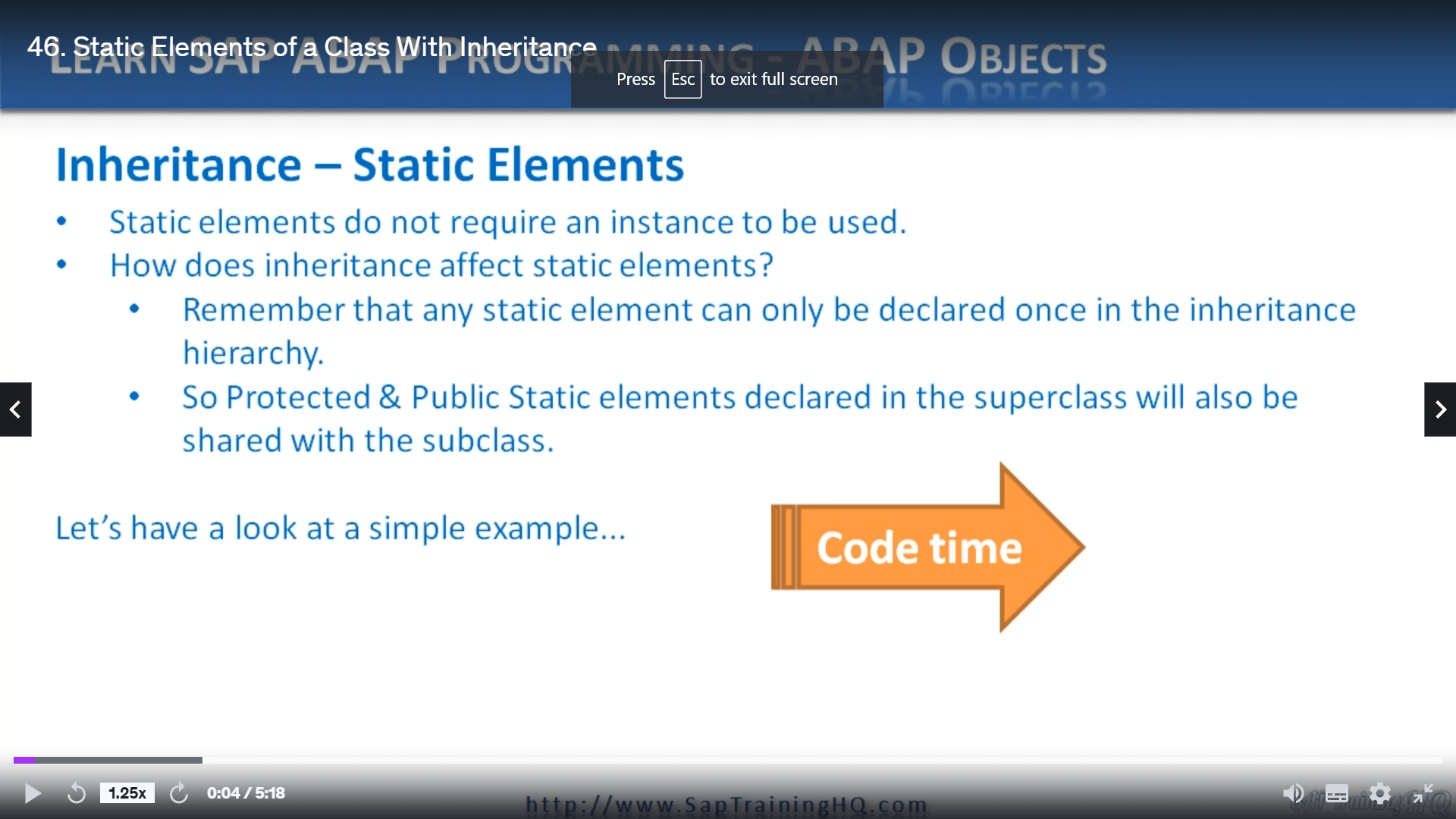
REPORT ZYNY\_CLASS\_04.  
\*----------------------------------------------------------------------\*  
\*       CLASS car1 DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS car1 DEFINITION.  
  PUBLIC SECTION.  
    CLASS-DATA: numofcars TYPE i.  
ENDCLASS.                    "car1 DEFINITION  
  
CLASS car2 DEFINITION INHERITING FROM car1.  
  
ENDCLASS.  
  
\* Our program starts here  
  
START-OF-SELECTION.  
  
car2=>numofcars = 2.  
  
WRITE car1=>numofcars.



And this lesson, we're going to look at static elements with specific reference to inheritance now,

I do want to get on to discussing constructor's with inheritance, but I first want to touch on static

attributes with inheritance because it leads onto the topic of constructor's in inheritance.

Really?

Well, you will see this as we get there.

Now, as you know, static attributes do not require you to create an instance of a class for them to

be used if you do declare an instance.

Well, it's no big deal.

All instances of a class will share the same static components that you defined in the class.

So the question comes, all right, well, what happens when we use inheritance with these static elements?

And the answer is, well, not too much.

When we define subclasses, any protected, all public static elements declared in the superclass will

also be shared with the subclass.

And it's important to remember that any static element can only be declared once in the inheritance

hierarchy.

So the elements themselves will actually be shared amongst the superclass and all its subclasses.

We can show this in a quick example by changing a static attribute of the subclass and see how it changes

the value of that static attribute in its superclass.

Let's flip over to the ABA editor and we'll see how we can do it.

So what I'm going to do is create a brand new program.

Let's stick with the normal number in here and we'll leave it at inherits scene as we're working on

inheritance.

OK, here's our program.

Let's create some blank lines, so first of all, I'm going to declare a class.

You'll be familiar by now, I love the the old classes.

I'll declare some class data.

Let's do a number of counts.

Of type integer.

So is our classic definition of just tidied up and we go.

Now, because we have no methods, there's actually no need for me to define the implementation for

Kangwon here.

So all I'm going to do is define a brand new class.

Kolkata.

And it's going to be inheriting from Cowwarr.

Nice and easy.

So now let's add some code to show how this works.

I'm going to say car to remember I'm using the equals sign because it's actually class data and I'm

referring to the class itself, not an instance of the class number of cars equals to.

Now, I'm going to do a right statement.

And say, can one last data gain the number of cars and that's it, that's all we need to code, so

let's review it.

I'm specifying some static attribute here, number of cars in the public section and then car to inherits

from Kangwon.

Then I'm using well, I'm referencing the car to class data number of cars and setting it to a value

of two.

But then when I come to do the right statements, I'm actually referencing car one.

So this right statement will prove that this static attribute number of cars is shared between the superclass

and its subclass.

So I'll save the code, do a quick check.

Activate it and test it out.

There we go, so we're writing out count number of cars came with a value of two, which was sent by

updating the car to number of cars.

Pretty easy stuff, isn't it?

Let's move on.