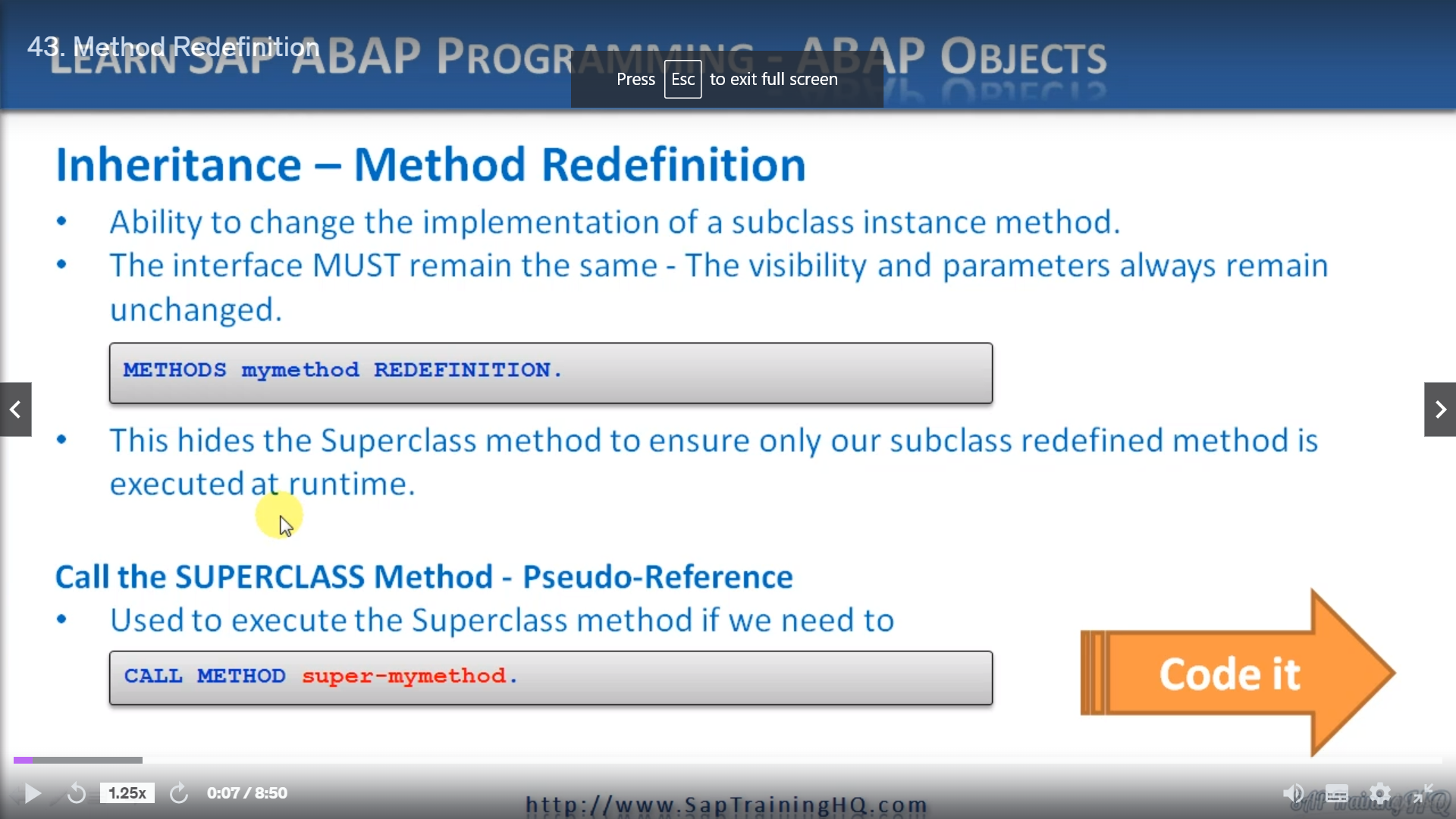
REPORT ZYNY\_CLASS\_03.  
\*----------------------------------------------------------------------\*  
\*       CLASS vehicle DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS vehicle DEFINITION.  
  PUBLIC SECTION.  
    METHODS: gofaster,  
             writespeed.  
  
  PROTECTED SECTION.  
    DATA speed TYPE i.  
  
ENDCLASS.                    "vehicle DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS car DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS car DEFINITION INHERITING FROM vehicle.  
  PUBLIC SECTION.  
    METHODS: refuel,  
             writespeed REDEFINITION.  
  
  PROTECTED SECTION.  
    DATA fuellevel TYPE i.  
ENDCLASS.                    "car DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS boat DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS boat DEFINITION INHERITING FROM vehicle.  
  PUBLIC SECTION.  
    METHODS: writespeed REDEFINITION.  
  
ENDCLASS.                    "boat DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS vehicle IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS vehicle IMPLEMENTATION.  
  METHOD gofaster.  
    speed = speed + 1.  
  ENDMETHOD.                    "gofaster  
  
  METHOD writespeed.  
    WRITE: / 'The Vehicle Speed is: ', speed LEFT-JUSTIFIED.  
  
  ENDMETHOD.                    "writespeed  
ENDCLASS.                    "vehicle IMPLEMENTATION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS car IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS car IMPLEMENTATION.  
  METHOD refuel.  
    fuellevel = 60.  
    WRITE: / 'You have just filled up your fuel tank.'.  
  ENDMETHOD.                    "refuel  
  
  METHOD writespeed.  
      WRITE: / 'The CAR Speed is: ', speed LEFT-JUSTIFIED.  
  
  ENDMETHOD.  
  
ENDCLASS.                    "car IMPLEMENTATION  
  
CLASS boat IMPLEMENTATION.  
  METHOD writespeed.  
      WRITE: / 'The BOAT Speed is: ', speed LEFT-JUSTIFIED.  
      call method super->writespeed.  
  
  ENDMETHOD.  
  
ENDCLASS.                    "car IMPLEMENTATION  
  
START-OF-SELECTION.  
  
  DATA: car1 TYPE REF TO car.  
  DATA: boat1 TYPE REF TO boat.  
  
  CREATE OBJECT: car1,  
                 boat1.  
  
  car1->gofaster( ).  
  car1->writespeed( ).  
\*  car1->refuel( ).  
  
skip 2.  
  
  boat1->gofaster( ).  
  boat1->writespeed( ).



This lesson is going to focus on method redefinition.

Now, as long as the interface of a method stays the same, we have the ability to change the implementation however we like, we can therefore redefine instant's methods in subclauses.

To do this, you must specify the method again with the method statement in the same visibility section as was defined by the superclass in the declaration of the subclass is the syntax.

We use the methods, he would then use the method name and then we just use the special keyword redefinition.

That's all there is to it.

Now, by doing this, what we effectively do with this statement is to reimplement the method in the implementation part of our subclause.

We don't need to specify the parameter interface again for this method because we're not allowed to change the interface.

It remains exactly the same as defined in the superclass.

Now, once the method has been redefined, it effectively hides the method in the superclass to ensure our object can only execute our redefined method.

This applies to any object reference that we have created that points to the subclass and to any other subclasses which inherit from this subclass.

Now there is a way that we can still execute the original method of the superclass, and that's by using a pseudo reference which has the effect of getting around the fact that we have actually hidden the superclass implementation of the method.

So as you can see here, the syntax we have call Method Super Dasch, then the method name.

This can be useful if you first want to execute the functionality of the superclass and then extend it with the functionality you have redefined in your SOCOs.

Let's switch over to the abuelita and we'll see how we can use this in the inheritance program that we are playing around with that has vehicle class, a car class and a boat class.

So here we are back in the ed and what I'm going to do, I'm going to go to the top of the program and then I want to find a car definition which inherits from vehicle.

So at the moment, we can see we have a refuel method.

Which only exists for the costs, of course, so if we want to redefine one of the existing vehicle methods, either go faster or right speed, all we do is come down to the public section again here.

And underneath, we can redefine the right speed method.

By tapping right speed.

Redefinitions.

The left one just too pretty to have got something to apologize.

Let me stick.

A comma in the.

Try it again, that's better so you can see it's very easy.

Oh, I have to do is key in the method name with redefinitions named.

If I want to do the same with the boat class.

Well, we have no methods anyway or need to do is come down here to create the public section and then

once again make an entry for the right speeds method.

So that's the classic definition we now need to go down to the implementation section and do the same

sort of thing, but instead of using the redefinition keyword and so on.

Let me just locate the cursor properly.

You go go to car through some new lines.

All we need to do is specify the method as normal.

And clean our code.

So if I just move up and copy this joint statement.

Here would go and then I'll edit it just to show.

To ease the redefinition that gets executed, we'll say the car speed is Anna Nicole speed, and if

we go down to our boat, of course, we don't have a boat.

So what we must do is actually create one.

Tell you what, let's use the old copy and paste.

We are both implementation, obviously, we don't need to refuel, we just need the right speed.

And that's it.

So in the definition area of our subconscious, let me find it, here we go.

We just have the method name and the keyword redefinition.

And then in the implementation for that class, we just write our method as normal.

Let's save the program and check it.

And let's run it and see how it executes, tell you what, let's run in debug mode instead.

So then we can see stepping in to either the parent class, the superclass or the child class.

So a great point on the Redefined.

So activate a breakpoint on the redefine method and I'll put one that as well.

And then we can see what executes.

So we're straight, we're going to call the right speed for Kawan.

And as you can see, it actually comes to the subclause car and performs this right speed instead.

Onto the boats, exactly the same thing goes to the subclause and performs that statement.

Or method, should I say, now we go the count speed is one and the boat speed is one.

Now just before we move on, what I want to do is just show you how we can call the method of the superclass.

And we can do this by if we go to the boat implementation as an example.

We can create a new line and then we can use the syntax call method and we use the special word super.

And then the method name, which in our case is right, speed again.

So what's going to happen is when this subclause method is called right speed, it's going to output

this and then this method itself is going to call the right speed method of its superclass, which is

vehicle.

So we should see an output saying the vehicle speed is as well.

So if I save this.

Let's activate it, give it a test.

So we're going to go faster four, count one, we don't care about that one.

So we'll zoom through it.

We will execute that go faster for about one, and then when we come to BO1 right speed, I step into

it.

Here we are within the subclause right speed.

It will output the first line, then it will call the super right speed.

And as you can see, it's gone into the superclass vehicle and it will output this.

I'll zoom through now we see the output, so it called the boat right speed, then it's super and then

the program finished.

Let's move on.