REPORT ZYNY\_CLASS\_00.  
  
CLASS CAR DEFINITION.  
  PUBLIC SECTION.  
    CLASS-DATA: NUMOFCARS TYPE I.           "Static Arrtibute  
  
    CLASS-METHODS CLASS\_CONSTRUCTOR.        "Static-Constructor  
  
    METHODS CONSTRUCTOR                     "Instance Constructor  
      IMPORTING  
        MAKE TYPE C  
        MODEL TYPE C  
        NUMSEATS TYPE I  
        MAXSPEED TYPE I.  
  
    METHODS VIEWCAR.  
  
    METHODS SETNUMSEATS  
      IMPORTING  
        NEWSEATNUM TYPE I.  
\*     EXPORTING                             "Placed all 4 on screen to show the options we have. We selected IMPORTING  
\*     CHANGING  
\*     RETURNING  
  
    METHODS GOFASTER  
      IMPORTING  
        INCREMENT TYPE I  
      EXPORTING  
        RESULT TYPE I.  
  
    METHODS GOSLOWER                        "Just as an example we have defined this method using RETURNING.  
      IMPORTING                             "It can be exactly the same as the GOFASTER method - Importing & Exporting  
        INCREMENT TYPE I  
      RETURNING  
        VALUE(RESULT) TYPE I.  
  
  
  
  PRIVATE SECTION.  
    DATA: MAKE TYPE C LENGTH 20,            "Have to use a METHOD to read or change it. It could go into the PUBLIC section  
                                            "and be defined as: data: make type c LENGTH 20 READ-ONLY. This would allow other  
                                            "objects to read it but not change it due to the READ-ONLY addition.  
          MODEL TYPE C LENGTH 20,  
          NUMSEATS TYPE I,  
          SPEED TYPE I,  
          MAXSPEED TYPE I.  
  
  CLASS-DATA: carlog TYPE c LENGTH 40.    "Used by the CLASS\_CONSTRUCTOR  
  
ENDCLASS.                    "car DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS car IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS CAR IMPLEMENTATION.  
  
  METHOD CLASS\_CONSTRUCTOR.  
    CARLOG = 'Car class has been used class constructor'.  
    WRITE: / CARLOG.  
  ENDMETHOD.                    "CLASS\_CONSTRUCTOR  
  
   METHOD CONSTRUCTOR.                       "Instance Constructor  
     write : / 'normal constructor'.  
\*    make = make.                           "Run a syntax check and notice the error. We have an ATTRIBUTE & PARAMETER with the same name  
\*    model = model.                         "This is where we can use the SELF-Reference notation  me->attribute\_name  
\*    numseats = numseats.  
\*    maxspeed = maxspeed.  
    ME->MAKE = MAKE.                        "Add the ME-> after the slides  
    ME->MODEL = MODEL.  
    ME->NUMSEATS = NUMSEATS.  
    ME->MAXSPEED = MAXSPEED.  
    NUMOFCARS = NUMOFCARS + 1.              "Increment the car counter Static Variable  
  ENDMETHOD.                    "constructor  
  
  METHOD VIEWCAR.                           " Add this as a check and DEBUG the code.  
    WRITE: / 'Make = ', 19 MAKE.  
    WRITE: / 'Model = ', 19 MODEL .  
    WRITE: / 'Number of Seats = ', 19 NUMSEATS LEFT-JUSTIFIED.  
    WRITE: / 'Max Speed = ', 19 MAXSPEED LEFT-JUSTIFIED.  
    WRITE: / 'Speed = ', 19 SPEED LEFT-JUSTIFIED.  
  ENDMETHOD.                    "viewcar  
  
  METHOD SETNUMSEATS.  
    NUMSEATS = NEWSEATNUM.  
  ENDMETHOD.                    "setnumseats  
  
  METHOD GOFASTER.  
    DATA  TMPSPEED TYPE I.  
    TMPSPEED = SPEED + INCREMENT.  
    IF TMPSPEED <= MAXSPEED.  
      SPEED = SPEED + INCREMENT.  
    ENDIF.  
    RESULT = SPEED.  
  ENDMETHOD.                    "gofaster  
  
  METHOD GOSLOWER.  
    DATA  TMPSPEED TYPE I.  
    TMPSPEED = SPEED - INCREMENT.  
    IF TMPSPEED >= 0.  
      SPEED = SPEED - INCREMENT.  
    ENDIF.  
    RESULT = SPEED.  
  ENDMETHOD.                    "goslower  
  
ENDCLASS.                    "car IMPLEMENTATION  
  
\* Our program starts here.  
  
START-OF-SELECTION.             "We need to tell the system where to start our program..  
  
  DATA CAR1 TYPE REF TO CAR.    "Define an object reference variable. Notice how we have to use TYPE REF TO  
  CREATE OBJECT CAR1  
    EXPORTING  
      MAKE     = 'AUDI'  
      MODEL    = 'A4'  
      NUMSEATS = 5  
      MAXSPEED = 120.  
  
  DATA CAR2 TYPE REF TO CAR.    "Define an object reference variable. Notice how we have to use TYPE REF TO  
  CREATE OBJECT CAR2  
    EXPORTING  
      MAKE     = 'AUDI'  
      MODEL    = 'A4'  
      NUMSEATS = 5  
      MAXSPEED = 120.  
  
  CAR1->VIEWCAR( ).  
    
  car1->viewcar( ).                     "Call the instance method  
  uline.  
  
  car1->setnumseats( 4 ).               "Call the method with the param inside ( ) only  
  car1->viewcar( ).  
  uline.  
  
  car1->setnumseats( newseatnum = 3 ).  "Exactly the same result as above  
  car1->viewcar( ).                     "Call again to output the new values  
  uline.  
  
  car1->gofaster( EXPORTING increment = 25 IMPORTING result = theresult ).  
  car1->viewcar( ).                     "Call again to output the new values  
  WRITE: / 'The Result of GOFASTER is: ', theresult LEFT-JUSTIFIED.  
  uline.  
  
  car1->goslower( EXPORTING increment = 15 RECEIVING result = theresult ).  
  car1->viewcar( ).                     "Call again to output the new values  
  WRITE: / 'The Result of GOSLOWER is: ', theresult LEFT-JUSTIFIED.  
  uline.  
  
  WRITE: / 'The number of cars created is: ', car=>numofcars LEFT-JUSTIFIED. "Reference the static-data  
  
\* A better way of calling a functional method is this...  
  theresult = car1->goslower( 5 ).  
  car1->viewcar( ).                     "Call again to output the new values  
  WRITE: / 'The Result of GOSLOWER (functional Method) is: ', theresult LEFT-JUSTIFIED.  
  uline.

In this lesson, we're going to go through and actually apply code to our existing Karplus, and when

we're creating the object, we're going to start calling methods and referencing static attributes.

So let's switch over to the ABA Ed, and then we will see the code in action.

So the first thing I'm going to do, I'm just going to collapse a few bits and pieces, a few areas

of our class down, so it's a bit easier to reference.

It would go expand the definition.

OK, that will do that will make it a bit easier to reference.

So I'll go to the bottom and I'll get rid of this new line.

So what I'm going to do, I'm going to show how we call the method that we just created.

When I find it here would go the Vuko method.

And all we do is reference one object, and then because it's an instance method, we use the dash greater

than sine and then the method name.

So Vukani.

And remember, when we refer back to the slides here, we must have brackets, that is the previous

one.

Here we go.

Bracket says no space bracket.

Space bracket.

So that's how it sits and full stop, and that's it, that's how we implement this mythical to order

pop that there I will activate the program.

And now we will debug our code.

So let me set a break point.

We will test it out, you've seen this before.

We.

Trade the object it right, Santa Clause constructor.

Information then executes the constructor itself.

And then we get down to calling our view commented.

Goes to -- and they all write out the information that we have stored in our attributes of our object.

Just execute all along and then we have it right now, the information for our car.

So this first line was when the class constructor was called and then this line was went out.

Vukani method was executed.

Now, to show you that I'm aware at the moment, all we've done is create one object, so to create

another object, it's really, really easy.

All we need to do is more or less duplicate this.

So if I come down here and do this, then I can say I want to create a brand new object.

Let's call this car two.

And instead of Audi, let's pass in BMW three.

Twenty number of seats, five, maximum speed, let's say 130 miles per hour.

And then all I need to do is change our method call instead of using or referencing our object.

Count one, we reference object car to simple as that.

The only thing I need to do I guess is say here car one time reference to car.

Oops.

And also to put space in to a Syntex check.

Everything is good, so we'll activate it and let's test it out.

There you go.

So now we have our first object.

Well, I'll tell you what, what what we can cover here, we can see this is the static method that

was called gets written out once the notice.

This isn't written out again when we instantiate our second object because it only gets executed the

first time the class is touched.

OK, so our first car, our Audi, is written out and then we create our second object, BMW, and that

information gets written out as well.

So that shows you very quickly how we can create a number of objects, all based on one class.

And I tell you what we'll do, we'll just under and on the line here to separate them out.

That's a bit easier to see, isn't it?

Now, I want to turn this up a bit.

Let me get rid of this second object here.

Yes, so we're going to keep it nice and clean going forward and we go now.

Fine with me on the line now, the next thing we're going to see is we've seen how to call this method

with no parameters being passed.

Let's call a method that has one important parameter.

Tell you what I'm going to do.

I'm going to get rid of that.

I'm going to go to the top.

And if we scroll down, we can see that our method set seats has one important parameter.

So if we refer back to our slide and we can just scoot along, you would go.

When a method has only one parameter, we can place that parameter inside parentheses, there's no need

for us to actually type out exporting or importing and so on when it's just one parameter.

We can miss this off.

So back to our code, we go to the bottom.

Then we can call our method, just like we did with the View car, we can say car one dash greater than

and it was I think it was set in some seats.

And we go and then remember, no space open bracket.

Then we must have a space and then we're going to pass in a value.

So we're passing the number of seats.

So when we create this object, we're saying the number of seats is five.

But when we call this method from our calling program, we can change that value.

So let's say, no, it's only got four and we'll put a space after it period at the end.

And that should be good.

Let's check in with code and give it a test.

Well, I guess we're not going to see anything different, so let's come back out.

Here we go.

So what we actually need to do is called the Vukani method again.

So once we change the number of seats, we will say, show us this kind of information again so we can

see that number, the number of seats has changed.

Here you go.

So the object first gets created, the number of seats is set to five, and then when we call our method

set number of seats, assigning a value of four, he changes.

So that's the shortened version when we only have one important parameter, what we can do just for

completeness, again, let's see how it works using the long form or just longer, it's not really massive.

We can see number of seats.

Oops, and then we can explicitly state the parameter name, so new seat number, so we can say New

Sweetenham and this time let's set it to three.

So if I screw up, I'll just show you in the method our set Satnam seats is new scene, and that's the

important parameter.

OK, and then we're going to set the attribute of the object.

Which is number seats, so you see is the parameter name, and that's what we're specifying here, the

parameter name, and then we're setting equal to either the date or item, the variable we have in our

local program or in this case, a specific litoral, a numeric value of three quickly pop over to the

slide just to indicate that here we have the parameter name and we're making equal to what we have in

our local program, our data item or the literal.

So back here, we will activate the code tested again.

And now we can see it has changed to number three.

And I'll just demonstrate that if we don't have spaces and all that sort of stuff, we are going to

get syntax errors.

Process Adira statement cowboy is not defying, check your spelling.

OK, so these spaces before and after the parentheses and things like that are very important, go on.

This is not expected and so on.

OK, so make sure you get this right.

I'm sure they'll slap you up because it slips me up all the time.

OK, so the next method call we want to take a look at.

We had some blank lines here.

So instead of Seaton's now let's call a method that has importing and exporting parameters.

So if Push Roll-Up.

Here we go.

So our go faster method and go slower method of importing parameters and exporting for go faster and

on the go slower, it's importing and returning.

Now, remember, we change things around.

So when it's importing in the method, we have to specify exporting in our calling program.

Same thing with exporting in the method we specify importing when it's returning.

We say receiving the back to the code.

So we're going to say the method here is go faster open bracket than we specify exporting, which is

going to be referring to the importing parameter of our method.

And our method parameter is called increment.

And let's set this to 25, then we want to import back out of the method, the methods parameter called

result, and we're going to store that in our calling programs, the results.

Which we need to define, so I'll put a full stop there to save just scroll up.

Here we go.

Tell you what, let's keep that separate and we'll just pop it up here just for clarity.

It was Syntex check.

Everything is good.

So go faster will increment the speed by 25 importing data into our local variable, the result and

we're going to write out the details of the car and tell you what, why don't we write down the value

of the results as well?

Let's save it and do a syntax check.

Everything is good, let's give it a go.

Excellent.

So here we can see the speed is set at zero, and then when we finally call our method, I would go

faster to increment by 25, speed to 25.

And then we write out the result of the go faster method is twenty five.

Fantastic.

So we can see their data is being passed in and it's been returned to us.

Let's have one more example.

And this one is going to be they go slower.

And if you remember, the go slower was defined.

It go as returning, so back over to the slide, you can see a returning in the method needs to be receiving

in our culling program.

So back to the bottom, no need to type all this, we are exporting an increment again, so we've increased

it to 25.

Let's slow it down by 15.

And then we want receiving results.

And we're going to still result in our the results, again, view account information and we'll say

the results of go slower is the result.

Syntax check, activate the program, give it a test out and scroll down.

So here's the go faster, here's the go slower.

So you can see we incremented go faster by 25, given as a result of twenty five.

And then when we subtracted 15 by executing the go slow method, our speed came down to ten telling

us that the final speed is ten.

And that's it, that's how we call the methods and our objects.

Now, I guess one thing we haven't seen here is how we reference specific static attributes and things

like that.

So let's add one more line of code or a few.

And what we're going to do will pop up here if we want to.

Let's do a few things.

Let's say we want to access.

And there's our concern.

Let's go to the in our public section, we have a number of cars.

So if we want to access how many cars are being instantiated at any given time, we can reference this

data.

Just by coming down here.

And we can say object name or even better, because it's cost data, we can say the actual class name

canwe so we can say car.

And then because it's static, we use the equal sign greater than sign, and then we would say number

of cars and what we have to do is assign it to something.

So let's say the result is equal to the static attribute of number of cars or even better.

Let's not overwrite that.

All I need to do is I'll just write it out.

So the number of cars created is.

The static attribute left just defined.

Activate it, give it a test.

Scroll to the bottom, here we go.

The number of cars created is one fantastic, and that's because we have only instantiated one object.

Is there anything else that we need to have a look at, let's have a look, we've access to class data.

We've called our methods, we called to construct.

I think that's about it, isn't it?