

We are now coming onto a topic that often takes a little bit of time to sink in.

I know that when I first learned about it, I had to go over it quite a few times before I really got to grips with it.

So if you feel the same, watch the video over and over again until it sort of makes sense.

We will have some examples coming up.

Now, what is it that I'm talking about?

Well, narrowing casts and this is all to do with the type of reference variables we use in our programs.

Up to now, we have created objects and our reference variables have all been of a certain type, they've

been known as a static type response variable.

When we instantiate an object, we have simply created our object we code similar to what we have here on the top of the slide.

As you can see, we use the data statement, then the object name time reference to and then the class name.

Then we instantiate the object with the create object statement.

Pretty simple stuff, isn't it?

Well, we have the ability to make a reference variable based on a superclass point to a subclass object, and this ability changes how our object reference variables can be described.

Before knowing this, we could say something like a reference variable is a pointer to an object of a class.

But now it can be termed a little bit differently, we could say a reference variable is a pointer to an object of a class or one of its subclasses.

This means we have the ability to use a narrowing cast with our objects.

And the term itself is a reference to making our object reference variables see a narrower scope of object elements.

To describe this a little bit better, let's use our car analogy a little bit more.

Let's say we have a car object containing one attribute and one method.

And we also have a subclass object called Mercedes's that inherits from car the Mercedes, so class

also has additional attributes and methods, which in turn means the Mercedes object also has its own

additional attributes and methods.

When we create an object reference variable Photoshop class and the superclass, we can assign the object

reference of the subclass to the superclass.

This results in our Supercluster reference variable car pointing to our subclass object, Mercedes's.

But there is an important restriction to understand when we do this.

Because our cast Supercluster Reference forgettable, only knows about the elements defined within the

superclass, it can only directly access these elements.

It cannot reference any of the additional elements that were part of the Mercedes class.

This is what a not knowing cast is.

The car reference variable has a narrower, less specific view of the object it points to.

This means our reference variables can now point to different objects at runtime, and if we create

code to do this, our reference variables can now be referred to as dynamic type reference variables

instead of the static type reference variables mentioned earlier.

Take a look at the second example I have here on the slide here, you can see an example of a narrowcast

I've declared two objects.

We have the car, a Mercedes object.

The car is a superclass in the Mercedes.

Is this, of course.

But then this statement down here, we assign the Mercedes object to our car reference variable so I

can reference that Apple is pointing to the Mercedes subclass.

But because this is narrowcasting, our car reference variable can only perform methods on the subclass object that were first defined in the superclass.

So the car reference variable now has a narrower scope.

It can only perform those methods common to both the superclass in the subclass and when it comes to attributes, only access those attributes that were part of the superclass to begin with or defined within the superclass that the Mercedes subclass as inherited.

Now, let's focus on this example at the bottom when making the assignment for the reference variable.

Of the superclass to the subclass, you have the option of using the time position of the create object statement to perform an implicit narrowcast at the time the object is instantiated.

So as you can see, we have the create object statement, but because of how our reference variables are defined, the car pointing to Supercross and the Mercedes pointing to the subclass outrate object statement is implicitly carrying out a narrowcast by creating the object car, but pointing it to the Mercedes subclause object.

Now, what do you think of that topic?

Me, when I first came to grips with it, it was pretty complicated.

If you get it fantastic, you're doing a lot better than I did, if not, try and go over it, but alsowatch the coming videos because we will create some examples to try and demonstrate how it can be used.