

In this lesson, we're going to look at abstract classes and abstract methods.

Now, as well as redefining methods, we also have the option of creating what are called abstract classes

and abstract methods.

Let's first have a look at abstract classes.

Now, abstract classes allow users to create a class that never gets instantiated, we can never trade

objects from it.

You may ask, what's the point of this?

Well, it can be a really useful technique in object orientation.

We've already seen examples where we have used a vehicle as a set of classes where we have a vehicle

class defined at the top with the go faster and a right speed behavior, and then we have two more specialized

classes that inherit from in the car and the boats.

But did you notice that we only ever instantiated the actual car and boat objects, we never instantiated

a vehicle class object, so in this case we could consider the vehicle class to be an abstract class.

The only reason it exists is purely for the sake of being inherited.

So the vehicle class is a great candidate to become an abstract class because abstract classes are never

instantiated.

Now, this type of class is still incredibly useful as it still contains functionality in the form and

methods and attributes so that they can be inherited.

So the main goal of an abstract class is to provide shared behavior to a subclasses.

Now, the way we define a class as being abstract is really simple.

You can see on the screen here, all we do is use the key word, then the class name, and then in the

after the definition keyword, we just got the word abstract.

Let's go on to abstract methods now, abstract methods are similar to an abstract class in that they

cannot be implemented in that class.

Instead, what you have to do is implement the abstract method in a subclass which in turn means the

class where you define an abstract method also needs to be abstract as well.

When we want to define an abstract method, all we have to do is at the abstract addition to the method

statement.

As you can see here on the screen, I've got methods with the method name and then I add abstract.

Then we redefine the method in a subclass with the normal redefinition addition added to the method

statement.

Let's go back to our code and see how we can implement an abstract class and an abstract method.

So first of all, let me get rid of these breakpoints.

And then we'll go to the top of the program, so I mentioned that the vehicle class is ideal to become

an abstract class.

So to do that, all we have to do is open the.

Plus definition here, at the end of definition, we just don't the word abstract.

But that's not all we do have to change things around a bit here, don't we, because we've actually

implemented some of the methods.

But before I moved down, as we can see here, we've got the methods go faster and right speed.

Why don't we create right speed as abstract as well?

So this would be our abstract method.

So all we do is have the method statement with the method name.

Then we have the word abstract.

So that means we have to redefine right speed.

So let's scroll down and see where we have that.

So here in the car class, we're already redefining it.

So that's fine.

When we come down to boat, we've done exactly the same, so that's already implemented.

We don't need to do anything else in these parts.

Let's run down the rest of the program.

Now, here we do need to make a change because we have made right speed abstract in the vehicle class,

which is also abstract, that means we cannot implement any code for right speed in this class.

We can only have code implemented in each subclass.

So what we have to do is just comment this stuff out or delete it.

And then a comment just to say we remove this method implementation because it is now abstract.

Let's move down the rest of the code here.

We have the code implementation refueler were implemented right speed here.

So that's fine.

We don't need to do anything.

When we come down to boats, we're implementing right speed here as well, but there's one thing here

we're also calling the super right speed, OK, now, because we've actually taken the implementation

out of the superclass, that means we cannot call this because the implementation for right speed in

the superclass no longer exists.

But what I'm going to do, I'll just do a save now and.

I'm just going to do a syntax check and it should throw in Adira, but just to show you what it looks

like in case you come up against it.

Here we go.

You cannot call abstract methods using super method.

So that's why so all we need to do is take this out.

I know I have a little comment in here to that effect, we can no longer call the superclusters method

because it is an abstract method that you won't let me pop it down.

One more line.

Now we go.

So if we do a syntax check now, everything is good, we've now turned the vehicle class into an abstract

class, the right speed method is an abstract method.

And because we have implemented the right speed correctly in both subclasses, everything should work

just as it did.

So let's execute the code.

Fantastic.

Let's move on.