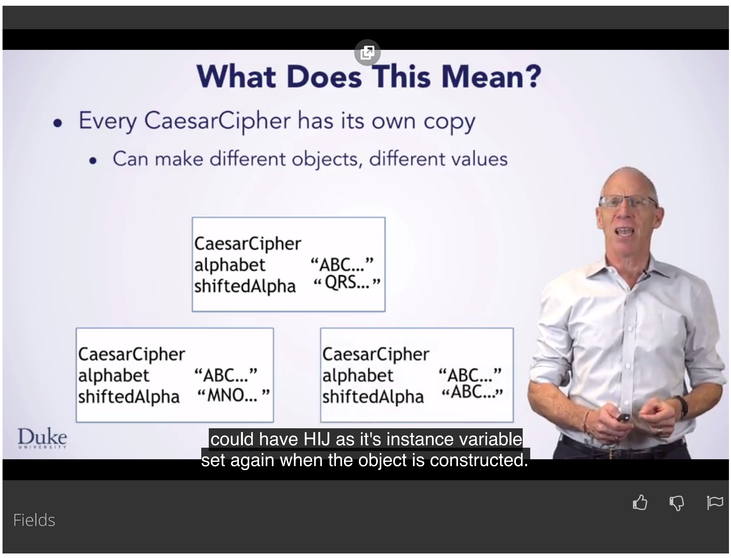

Fields

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

English

0:03
Hi. Now that you know a bit about object oriented concepts. It's time to go a bit deeper into the idea of fields, which are also called instance variables. Remember that when we redesigned our CaesarCipher to be more object oriented, we created two fields. One for the alphabet, and one for the shiftedAlphabet. These fields are declared inside of the class, but outside of any method. They belong to the object and are created when new is called to create an object. Either in code you write, or when you create an object in BlueJ that's shown on the object workbench. These fields are part of the object and exist as long as the object exists. What does all this mean? Every CaesarCipher object you create has its own alphabet and its own shifted alphabet. This is why fields are also called instance variables. They act like variables, where there's one variable per instance of the object you create.

1:01
What does this really mean? Let's look more deeply at fields and instance variables. Every CaesarCipher object has its own copy of the alphabet and the shifted alphabet. You can make different CaesarCipher objects and each will have its own inference variables. These variables are specific to the instance. If you create three CaesarCiphers, with three different keys. Each has its own copy of these fields with potentially different values.

1:31
One cipher might have a shifted alphabet of QRS for example. Based on the integer shift value passed to the constructor. While another object might have MNO as the shift value and the third object could have HIJ as it's instance variable set again when the object is constructed.

1:51
To call a method like encrypt on an object, you'll typically use a variable name like cc and you'll cc.encrypt. You can also call a method from an object on the BlueJ object workbench. These objects have names too. And when you call .encrypt for example, the method will use the values of the fields in the object you used to call the method. Like CC, if you wrote CC.encrypt. Calling encrypt on this CaesarCipher will use it's QRS shifted alphabet. So when we call .encrypt and provide the message, first legion attack east flank, as the parameter. The QRS alphabet is used to create the encrypted version that you see here.

2:36
In the same way, calling encrypt on this CaesarCipher object will use its MNO shifted alphabet. This is the principal of encapsulation. The method and the data are logically inside of the object. And the method acts on the data inside the object that it lives in.

2:53
As you can see here, calling encrypt uses this shifted alphabet and we see a different encrypted version of the same message. Because the field shifted off of it's starch, with MNO, is used here.

3:05
Finally, using this object with HIJ as the field, will result in a different encrypted message. When encrypt is called, the encrypt code uses this shifted alphabet, of this instance. And the encrypt code creates the encrypted message you see here.

3:22
Fields, or instance variables, are very important concepts in designing and using classes. Since they can be accessed by every method in the class. Like encrypt, that you see here, and the constructor as well.

3:36
As you begin to design our own classes, and think about the fields and methods to put in these classes. Here are a few design principles you should keep in mind. The first is that a class name should correspond to a noun.

3:48
Classes describe things. Each object you make for a particular class is one of that thing. Let's think about the classes you've seen so far for this course. Strings, pixels, CSV records. Each of these is a noun. It describes a thing.

4:05
A class could be a car for example. That's a noun. And then the methods and fields would correspond to things that a car can do.

4:13
Methods, on the other hand, are verbs. They're what you do to or with an object. Things like get pixel, set care at, or encrypt. Sometimes method names don't sound like verbs, such as substring or index of. But these describe actions, get a subscreen, or find the index of. The program has just shortened the name.

4:38
For the car, a method might be accelerate or brake. These are things a car can do. For example, invoking a method would make the car go faster or stop suddenly.

4:52
Fields, or instance variables, are important class concepts. Fields are also nouns. And should describe things the class has. The string class might have a field for a sequence of characters. A sequence of characters is a thing and a string has one of these things. Similarly an image might have many pixels. Fields can also be adjectives as they describe the properties of an object. For example, a pixel might have a field, or fields describing its color. That could be an adjective giving more information about the properties of the pixel. For cars, fields could include things a car has, nouns. Like an engine with a certain number of cylinders, or wheels of a certain size,

5:38
for cars adjectives might describe the color of the car. Or the kind of engine the car has or the type of wheel. As you get started making more complex classes, we will provide guidance on fields and methods you should make. But think about these designs principles as you write your code. As you gain more experience, you'll want to start designing classes on your own, based on these ideas. Happy coding acceleration.

Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?