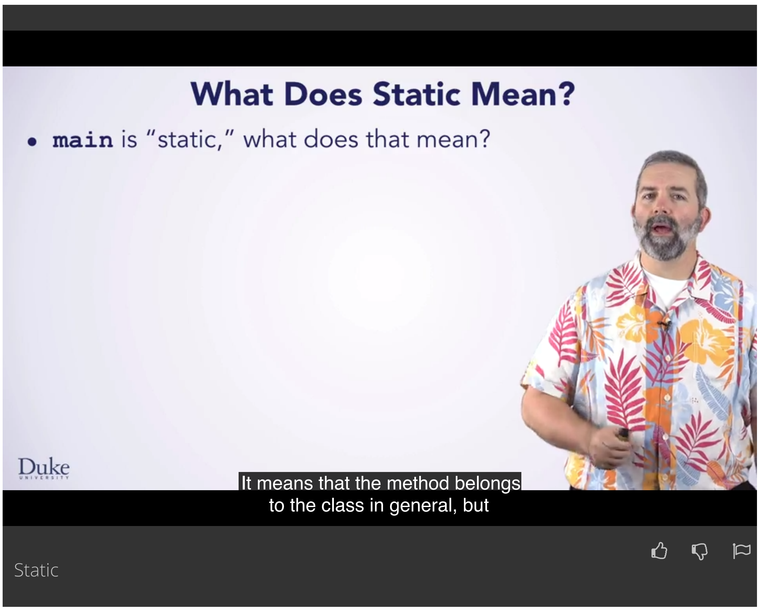
Using Java Beyond BlueJ

Mainstream Java

Review



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

Interactive Transcript

Search Transcript

English

0:03
You just learned about the main method, which is static. But what does that mean? It means that the method belongs to the class in general, but not any particular instance. To see what this means, let us first look at non-static fields for which there is one copy per instance of the class. As an example, suppose you were writing some banking software. You decide to make a class for a bank account and declare fields, the account balance and account number. These are great fields for a bank account class because each instance, that is each different bank account, has its own account number and its own balance.

0:41
If you made three instances of this class to store the data for three accounts at your bank, they might look like this. We can see here how each instance has its own account number and balance. These fields are not static, they exist in each instance.

0:57
Now suppose that as you write this software, you want to keep track of the next account number to assign.

1:04
So that when you create a new account you know what number to give it. Creating an instance variable for this next account number, as shown here, would not work very well. Declaring this field makes each bank account have its own next account number, which is not really what you want.

1:23
Lets see the problem this creates by looking at a constructor you might want to write. Which uses this next account number to initialize the account number of the bank account being constructed, and then increments that next account number as you see here.

1:40
When you create the first bank account object, all the fields will start with 0 values. Then you will execute the code in the constructor. The first line will initialize this object's acctNum from this object's nextAcctNum, setting it to 0. Then, the next line will increment this object's nextAcctNum to 1. So far this behavior is fine, but you will see the problem when we create a second bank account object.

2:08
This newly created object has its own copy of each field. Now when you execute the first line of the constructor, it uses the object's next account number to initialize this object's account number.

2:21
Then you increment the account number inside the object to be 1. Oh no, you've ended up with two accounts with the same number! That will cause problems with your banking software. In fact, every account you create will be numbered 0. The problem here is the next account number is not a property of each different bank account, but rather something shared by all bank accounts.

2:47
This is exactly what static is for. When something does not belong in each object of a particular type, but rather is shared by all objects of that type. Here you can see that we declared nextAcctNum to be static.

3:03
Now that the nextAcctNum data is not stored in each bank account, but rather there is one copy that is shared by all of them.

3:12
When you want to refer to a static field or method from outside of its class, you can do so by naming the class before the dot, since it does not belong inside of any particular object. For this field you can write BankAccount.nextAcctNum. So now you know a bit about static. For static fields there is only one copy, not one copy for each object instance. These tend to be much less common than instance fields. Often, you will want to describe objects in terms of properties that each instance of them have. However, there are times when static is appropriate, so it is good to know about.

3:52
You can also declare static methods. With regular methods, you can think of them as being inside of each instance. But with static methods, as with static fields, you can think of them as being only one shared by all instances of the class. Static methods can only access static fields and other static methods of the class they belong to.

4:13
However, they cannot directly access non-static fields or methods. If you wanted to access a non-static field or method from a static method, you would need to specify which object instance's field or method you would want to operate on with that object.TheFIeld or object.TheMethod.

Downloads

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

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