What line of code could you add at the end of this program to store the height of the image "chapel.png" in the variable x?

```
cImage.getHeight() = var x;
```

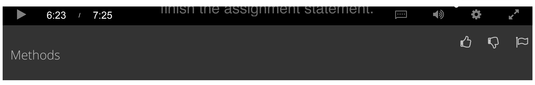
```
lImage.getWidth() = var x;
```

```
var x = lImage.getHeight();
```

```
var x = cImage.getHeight();
```

Correct
Correct!

Continue



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

Interactive Transcript

Search Transcript

English

0:03
Okay, so now you know about variables. But you also need to know how to do things like look at particular pixels and change their colors. How could you do these things? The answer is that you can call some methods that are built into SimpleImage, which already have the code to perform these operations. In general, methods let you perform some operation, possibly one that is quite complex, on an object.

0:30
Here is an example of calling two methods, getWidth and getHeight, to get the width and height of an image respectively. Let's break down the syntax. First is the name of the object we want to invoke the method on. Which SimpleImage's height do we want to get?

0:50
Whatever SimpleImage we name here.

0:53
Next is a dot, the dot operator means inside of. We want the getHeight method inside of fgImage. Next is the name of the method that we want to call. In this case, it is getHeight.

1:08
Finally, there are parentheses. Parentheses after a name indicate that it is a method, or a function, which is similar. If the method took any parameters, you would specify them inside the parentheses.

1:22
Now, that you have seen the syntax of a method call, let us see the semantics. Before we step through the behavior, we'll describe the semantics. First, execution goes into the method, you stop executing statements where you were, go inside the code for the method, and then do whatever code is there. You do that code by following all the normal rules for executing code. It has the same semantics, whether it is inside a method or not. At some point, the method will figure out what answer it wants to give back, which is called returning its answer.

1:58
The method call, which is actually an expression, evaluates to whatever value the body returned.

2:05
Then you continue executing code after the method call.

2:10
Let's see those rules in action. The first line we have seen before, it makes an image and initializes fgImage to refer to it.

2:20
Next is a call tofgImage.getWidth(). The code for getWidth() is in the Duke Learn to Program Library. We don't actually know what the code is, but that is fine as long as we know what it does.

2:33
The computer will then go into that code and start executing it. That code would figure out what the width of fgImage is, that it's 480 pixels, and decide that 480 is its answer. Since this method's answer is 480, the call fgImage.getHeight() evaluates to 480. Another way to think of that is that it's like a mathematical operation with the result 480.

3:01
Now execution comes back to where it was, which is halfway through this assignment statement. We finished the assignment statement by making a box for w and putting 480 in it. Now, a similar process happens for the call to fgImage.getHeight. The computer executes the code inside the Duke Learn to Program Library, figures out that the answer is 270, and returns that value to where the method was called. Execution then returns to where the method was called.

3:33
And the assignment statement finishes as usual.

3:37
So how do you know what a method does? Well if you have the code, you could see what it does, stepping through it line by line, following the semantics of the code.

3:48
But what if you don't have the code, as in this case? You should read the documentation, a written description of what a method does. For SimpleImage, you can find documentation about it on dukelearntoprogram.com.

4:04
If you were to go to the dukelearntoprogram.com website, you would see a link for documentation. If you clicked on it and looked at the left, you would find a list of the topics that are documented. One of these is SimpleImage.

4:21
If you click on that, you would get to a page that lists the various methods and SimpleImage, including getHeight.

4:30
If you were to look at this entry, you would see that it describes the behavior of the method. It gives you the height of the image in pixels.

4:39
We have said that methods are invoked on an object, but what exactly does that mean?

4:46
Well, if you are asking for the height of an image, which image's height do you want?

4:52
The answer is, whichever image you invoke the method on. To see this in action, let us look at this example, which has two images.

5:01
Our first line of code makes one image called fgImage, and our second line of code makes the other image called dImage. Now, we are about to do fgImage.getWidth().

5:16
Since we are invoking the getWidth method on fgImage, it's going to work with the imagery referred to by fgImage and give us its height.

5:26
Execution jumps onto the getWidth method in the dukelearntoprogram.com library and does whatever the code there says to.

5:34
Notice that the method looked at fgImage and found its width, coming up with an answer of 480. The method call evaluates to 480, and the assignment executes as you have learned.

5:49
Now the next line is dImage.getWidth. Since this method is being invoked on dImage, the method will operate on the image that it refers to. Again, the execution jumps into the dukelearntoprogram.com library's code and does whatever the code there says to do. However, this time, it is working with a different image. So it comes up with a different answer, 140, which is what this method call evaluates to.

6:19
Finally, we return to the method call and finish the assignment statement.

6:24
Some methods have parameters. For example, if you wanted to call getPixel on an image, which gives you a particular pixel, it might look like this.

6:34
Notice that we have 0, 0 in the parentheses. What do these mean?

6:41
These are the parameters to the method, they give the method more specific information about what it's supposed to do. In a particular case of getPixel, the parameters specify which pixel it should get from the image by giving the desired pixel's x and y coordinates.

6:59
The particular meaning of the parameters is specific to each method, but should be described in the documentation.

7:09
Methods combine together multiple potentially complex steps that operate on a specific object.

7:17
There's a similar concept, which does not work on an object called a function, which we will discuss next. Thank you.

Downloads

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

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