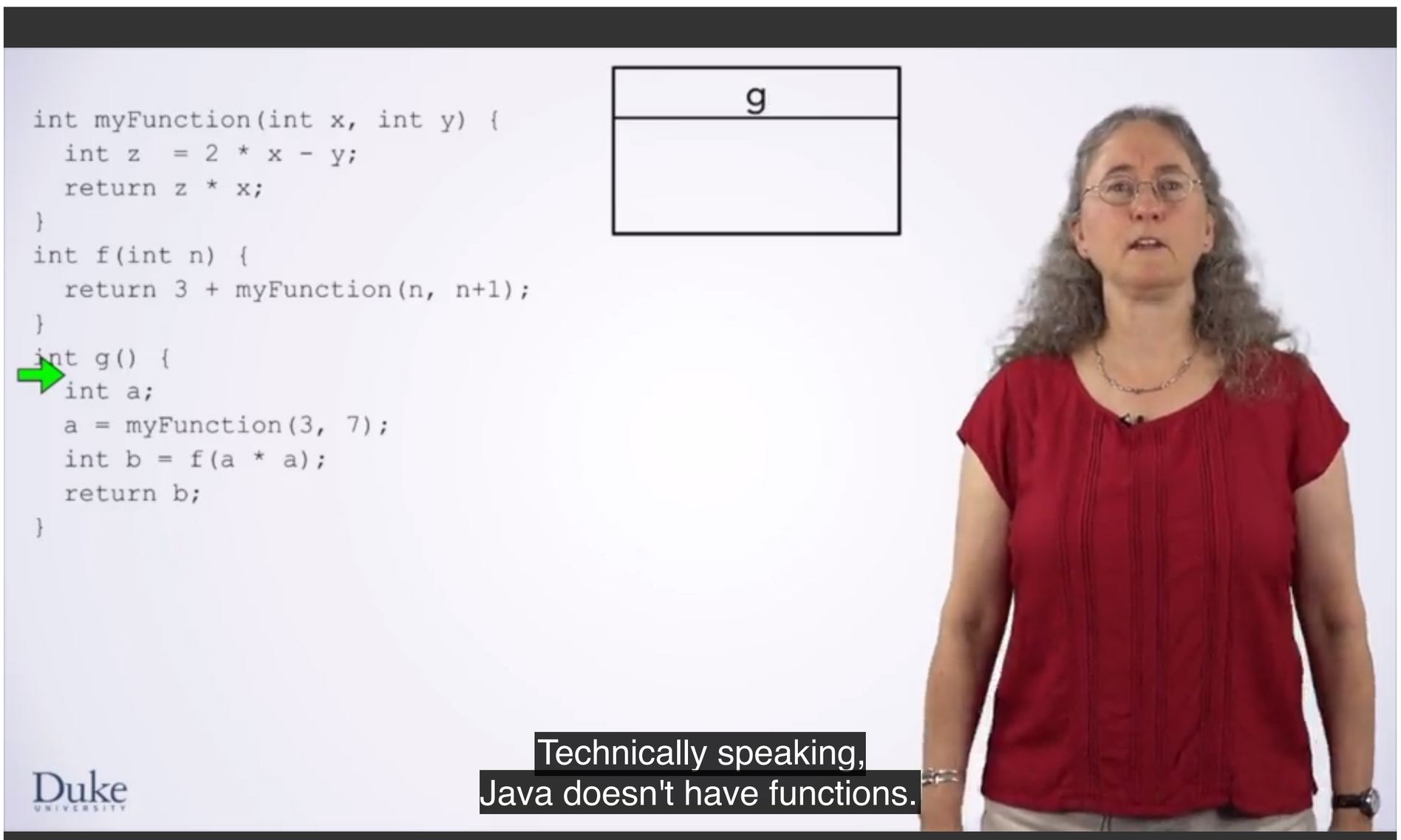

Functions

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

Interactive Transcript

Search Transcript

English

0:03
The next important idea to understand is function calls. Functions extract a computation out, giving it a name and parameters. You then can use the function to perform that computation without rewriting it. You can also think about what the function does and not how it does it. Technically speaking, Java doesn't have functions. It has methods since all code in Java is inside of objects. However, before we learn the more complex behavior of objects and methods, we'll learn the basic principles of function calls. These concepts will then lay the foundation for understanding method calls. We have three functions here. My function, f and g. Why are we starting at g? We'll assume for now that you chose to call g from the BlueJ interface. Later, you'll learn about the main method which is where programs start when you run them outside of BlueJ. We start with a frame for g and with the execution right at the start of that function. The first line declares a, so we create it's box inside of the frame for g. Next, we're going to set a equal to the value of the function call myFunction(3, 7). To evaluate this expression, we need to create a frame of the function that we're calling. In this case, myFunction. This will hold the parameters and variables of myFunction. Next, we pass the parameters to myFunction. We create a box for each parameter with the names coming from the function declaration, x and y. We initialize these by copying in the values of the expressions that were passed. Here, 3 and 7. Next, we need to note where to return when we finish executing my function. This location in the code is named the call site. The place where the function was called. We'll note it with a marker one in the code and put the same marker in the corner of the frame. Finally, we move the execution arrow into myFunction and start executing code there. Here, we declared an initial of z. Evaluating the expression 2 times x minus y. The values for x and y come from the frame from myFunctions, 3 and 7 respectively. So, z will be -1. Now, we have reached a return statement. Return statements tell us to leave the correct function, returning to the call site noted in the frame. They also tell us the value to return to the caller. The first thing we need to do is evaluate this expression to obtain the return value. Here, the expression is z times x. So we evaluate -1x3 and we get -3. Next, find where we should return. This is the call site we noted in the frame, then we copy the return value back to the call site. The function call evaluates to this return value. We move the execution arrow back to the call site and destroy the frame for the function we just returned from. Now, we're back in g. The call to my function evaluated to -3. So, this line behaves like a gets -3. We'll finish that assignment, putting -3 in its box. Our next line again has a variable declaration and a function call. We make a box for b and go through the same process to call f. You make a frame for f and past parameters. This time, there's one parameter n, whose value is a times a which is 9. We know where to return and begin executing code inside of f. Our next statement is a return statement, but the expression involves a function call. So, we have to evaluate that call before anything else. We start with a frame and pass parameters. X gets a value of n, which is 9. And y gets a value of n+1, which is 10. We know the call site will use two this time since we are already using one somewhere else and move the executioner to the start of myFunction, and start executing code there. We declare z and initialize it to a. Now, we are ready to return from my function. We evaluate z times x which is 72, then we find the call site noted in the frame and copy the return value 3. Finally, we move our execution arrow back to the call site. Destroying the frame from myFunction. Now, we pick up where we left off in f using 72 for the value of the call to myFunction. We evaluate 3 plus 72 to get 75. Since we're evaluating the return statement, this is the return value of f. So we find the call site and copy the return value there, then we return to that call site. Destroying the frame for f. Now, we can finish the initialization of b. B gets 75. Lastly, we reach the return statement for g which is where we started. When we return from the function where we started, we are done.

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

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