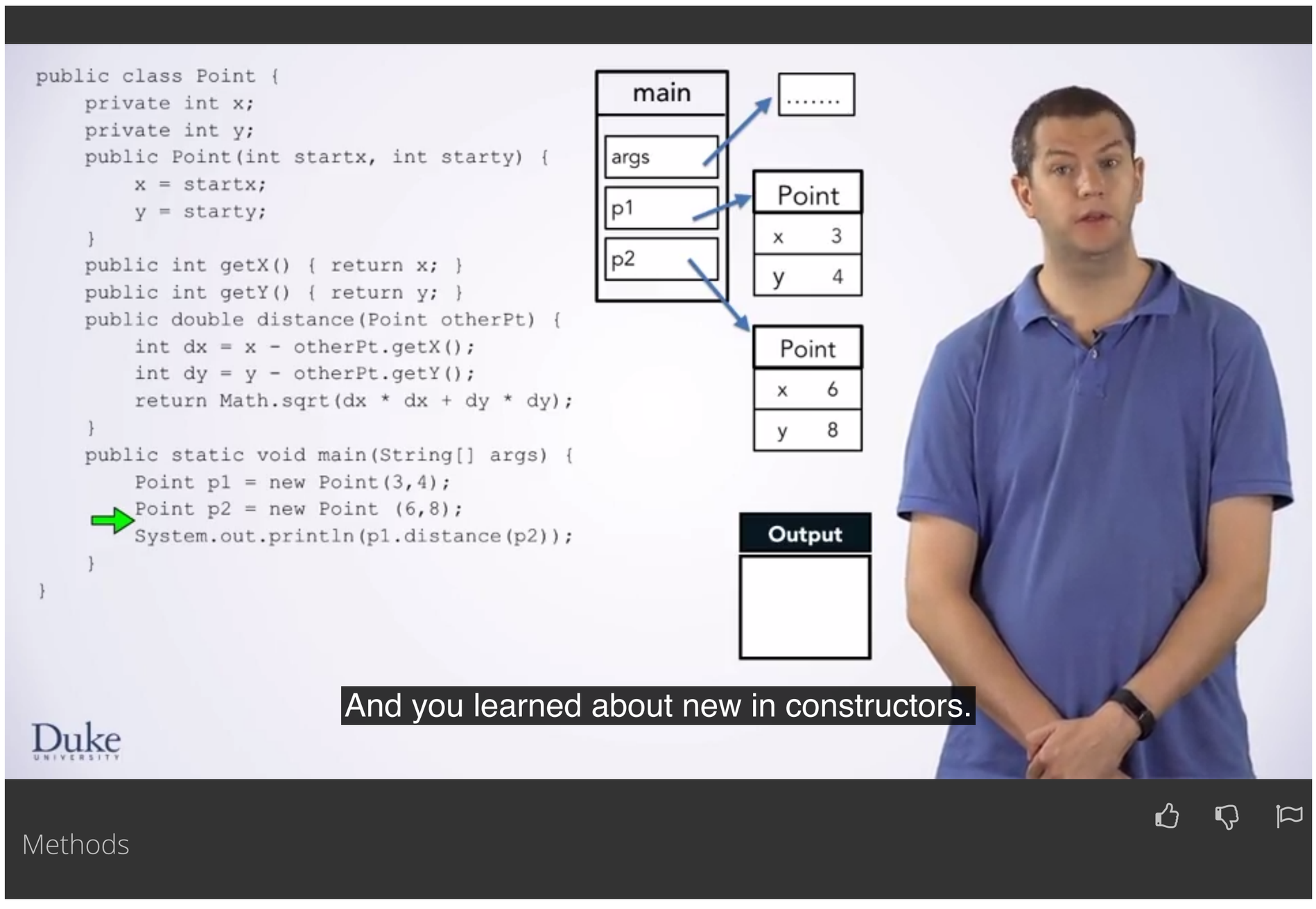

Methods

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

Interactive Transcript

Search Transcript English

0:02
In the previous video, we executed the declaration and initialization of points p1 and p2. And you learned about new in constructors. Now, it is time for you to learn about Method Calls. Method Calls work a lot like Function Calls, except that we have to pass this parameter to let the method know which object it is working on. Let us resume where we left off. This line is going to call p1.distance p2 and then print its return value. We need to set up a frame for distance which will take two parameters. The implicit this parameter which tells it which object it is acting on, and the other point parameter, which is explicitly passed to it. The this parameter has the same value as the variable before the dot. In this case, we have p1.distance, so this has the same value as p1. It is an arrow pointing at the same Ptobject. We have colored this arrow differently but that does not have special meaning. It is just to help you keep straight which arrow is which in this diagram. For the other point parameter, we just copy the value that is passed in. In this case, p2. So the value of other point is an arrow pointing at the same point object as p2. Now, we go into the distance method and begin executing code there. Executing the first line, we are declaring a variable dx, and need to initialize it to x - otherPt.getX. To evaluate that expression, we need to call otherPt.getX. So, we need to set up a frame for that method call. Again, this method takes an implicit this parameter to tell it which object to do getX on. Which object should this point at? Well, we did otherPt.getX. So, we copy the value of otherPt which is an arrow pointing at that point object. Now, we go inside of getX and begin executing code there. Here, we have return x. So, we need to evaluate the expression x, and return its value. How do we get the value of x? We follow the this pointer to find the object we are working on and then, look for the x field inside of that object. The value of that field is six. So, that is the value of the expression x here which is what will return to the color at CallSite two. Returning to CallSite number two, we want to evaluate x - 6. How do we evaluate x here? We again look at the this pointer which refers to this Ptobject, and we get the x field inside of that Ptobject. That value is three which is the value of x in this expression. So we will compute three minus six and initialize dx to minus three. Now, we are going to declare and initialize d_y using a very similar process. First, we make a box for d_y, then we call getY on other point. Notice how this, is a copy of the value of other point, an arrow pointing at our second point object. We get the value of the y field out of this object which is eight, and return it to CallSite two where we return the execution arrow to. We evaluate y by looking in this object and finding its y field which is four. Now, we are ready to finish the initialization of d_y to four minus eight which is negative four. Our next line has a little bit of math. Let's look at it in detail. We are calling math.square root of d_x squared plus d_y squared. First, we can compute the value of the parameter which is 9 plus 16, which is 25. So we need to evaluate math, that square root of 25. This looks like a method call, but where did the math object come from? It turns out that math is a class not an object, and is part of the Java library. This is a static method call. The method is being called on the class in general, not on any particular object. Here, the math class is just a handy place to put a bunch of mathematical functions together. Since we don't have the code for math.square root, we have to know what effect it has. If we did not know, we would look it up in the Java documentation. However, as you might have guessed, this method just computes the square root of its argument. So math, that square root of 25, will return 5.0. Our distance function returns that value to its caller. So the value of the call to distance in Maine, will be 5.0. We then return execution domain where we are ready to complete the print statement, which will print out 5.0. Now, we are done with Maine. So we will return from it, destroying its frame and exiting our program.

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

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