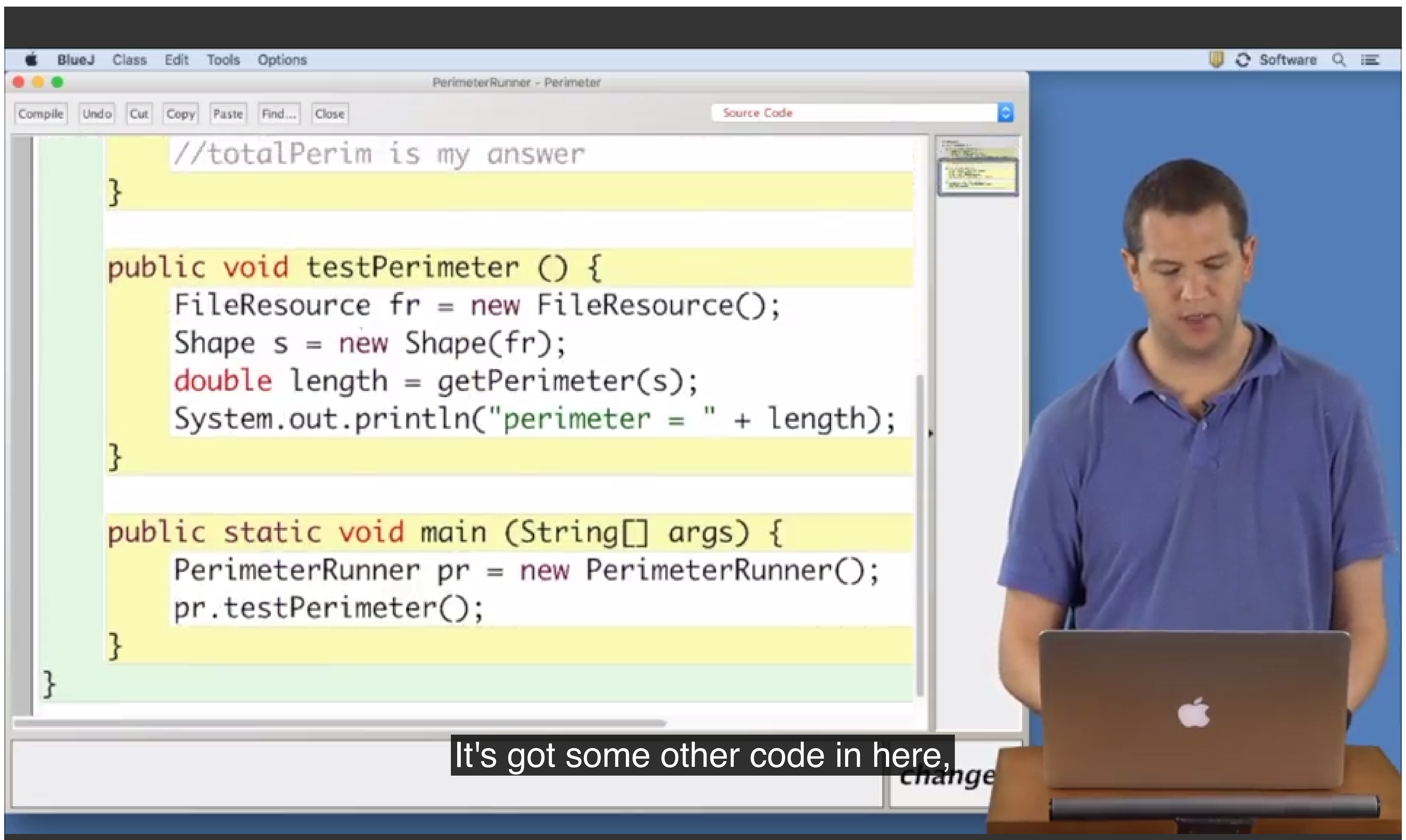
coursera Catalog Search catalog For Enterprise Java

Back to Week 1 Lessons Prev Next



```
        //totalPerim is my answer
    }

    public void testPerimeter () {
        FileResource fr = new FileResource();
        Shape s = new Shape(fr);
        double length = getPerimeter(s);
        System.out.println("perimeter = " + length);
    }

    public static void main (String[] args) {
        PerimeterRunner pr = new PerimeterRunner();
        pr.testPerimeter();
    }
}
```

It's got some other code in here,

## Seven Steps in Action: Translating to Code

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

## Interactive Transcript

Search Transcript English

0:02
All right, now you've developed the algorithm to find the perimeter of a shape, and we want to turn it into Java code. You've seen a lot of Java code as we've talked about the syntax and semantics. And so what we're going to do is step by step go through and take the algorithm that we wrote, and turn it into code that has semantics that match what we wanted our algorithm to do. So here I have a class called Perimeter Runner, it's going to have the method getPerimeter, which takes a shape and return to double. It's got some other code in here, so it has a main that we can run, which will make an instance of this class and call testPerimeter, which will use FileResource to read it from a file and create a shape and call the perimeter and then print that out. We're going to go through and translate this to code, and we've written our algorithm here as comments, which are things for people that the compiler will ignore. So the first thing we say is, start with totalPerim equals zero. So this sounds like we need a variable, we called it totalPerim, we're going to set it equal to zero. We need to think about what type we want. Since we said it equals zero, we might want an int. But if we think a little more carefully, we might realize we want to double, since we're working with floating point numbers that might have fractions. So double, totalPerim equals zero. And if we'd written this as int we, hopefully, would find that out in testing, when we come up with answers that should have fractions and we don't have them. Then, we say start with previous point equals the last point. So we need another variable, previous point. What type of variable is this? Well, this is going to be a point, and it's going to be the last point. We meant in s even though we didn't write that down, so we'll call get last point. How do I know that was there? I looked at the documentation for shape before I started doing this video and saw that it has a getLastPoint method. Then, we save for each point, which we wanted to call curve point in the shape. So this sounds like a for each loop. Each point, curr point in the shape.getPoints, which is going to give us all of the point. We want to do these steps, and I'm going to go ahead and put them in curly braces. We want to find the distance from previous point to the current point and name it curr distance. So anytime we want to name a quantity, we want a variable. And so that's going to be the distance from prev point to curr point. Then, we're going to update totalPerim to be totalPerim, plus current distance. And then we're going to update previous point to be current point. Last we said, the totalPerim is my answer, whenever we know our answer, we're going to return that because that's how we give our answer back to whoever called us. So now that I've written this code, I'm going to come up here, and I'm going to click compile, and it says cannot find a symbol variable shape. That's because my shape was called s. And now it says class compiled with no syntax errors. So now, I'm going to shrink this window down a little bit smaller. I'm going to come over here, and I'm going to do, main, even though that's how we run programs outside of BlueJ. If we've written a main, we can do that. We're going to give it no argument, so I'm just going to click okay, and it's going to ask me, "What input file has the points for the shape?". So, each of these files has some point. For example, example one has negative one three, negative one negative one, four negative one one three, which is what we used when we developed our algorithm. So that's a good first check, make sure we get the answer we expected. And that gives us 16, which you may recall is what we came up with whom we worked this example ourselves. Of course, using that might be bad, because if we made a mistake, we wouldn't necessarily catch it, since we've already used those values in developing this. We might want another one, and so this says that this other shape with points of negative three, four, negative three, negative four and three negative four has a parameter of 24. If you work that out yourself, you'll find that that's the right answer. And so we become more and more confident that this code that we just wrote is correct. So that's how you turn your algorithm into code. You go through step by step, take each step, write down the code that corresponds to what you wrote.

## Downloads

**Lecture Video** mp4

**Subtitles (English)** WebVTT

**Transcript (English)** txt

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