Welcome

Getting Started with BlueJ

Variables and Mathematical Operators

▤ Let's learn some basic Java syntax! — 1 min

▶ Shapes: Collections of Points — 2 min

▶ Why Semantics: Motivation to Read Code — 1 min

▶ Variables — 2 min

▶ Mathematical Operators — 2 min
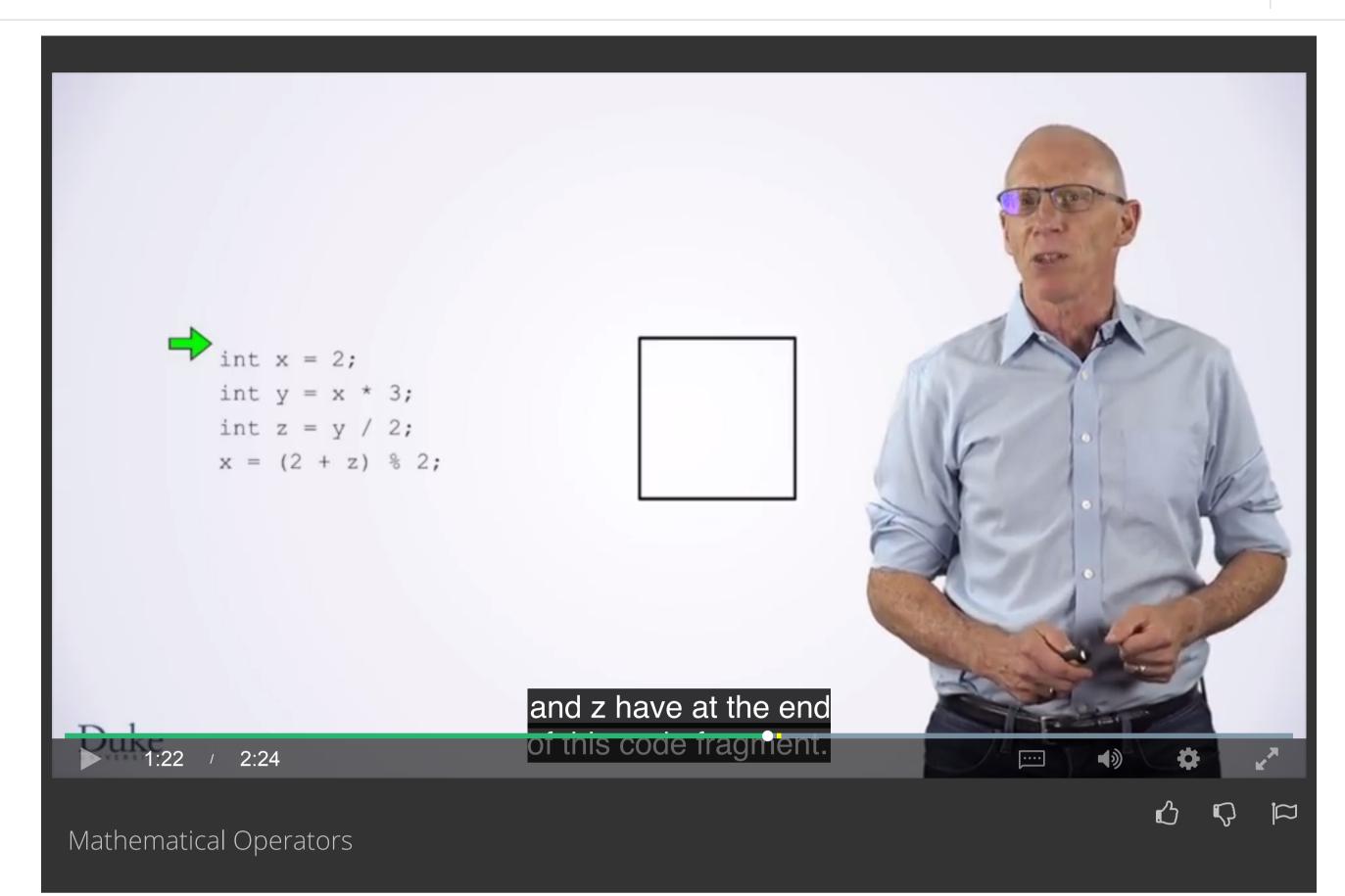
★ Practice Quiz: Variables and Mathematical Operators — 3 questions

Functions and Conditionals

Classes, Types, and For Each Loops

Seven Steps for Solving Programming Problems

```
int x = 2;
int y = x * 3;
int z = y / 2;
x = (2 + z) % 2;
```

and z have at the end of this code fragment.

1:22 / 2:24

Mathematical Operators

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

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

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

Interactive Transcript

Search Transcript | English ▾

0:03
Now that you know a little bit more about expressions, let's see them in action. This code example starts with the declaration of an integer, or int variable, called x, which behaves exactly as you've already seen. Next we initialize x to 4 + 3 * 2. As you know from math, times has higher precedence than plus. So this expression evaluates to 4 + 6, which is 10. So we put 10 in the box for x. Next we declare another variable, y of type int. And initialize it to x- 6, which is 10- 6, which is 4. So we create a box for y and put 4 in it. The last statement says, x gets x * y. Sometimes novice programmers expect statements which look like this to behave like algebraic equations with an equal sign, where you might solve for x. However, that's not what happens. Instead, you follow the rules you've already learned. The right hand side evaluates to 10 * 4, which is 40. And you put 40 in the box for x. Now, let's see another example. Before we work through this, take a moment to pause the video. And see if you can figure out what values x, y, and z have at the end of this code fragment.

1:25
Okay, let's step through it. First, we declare and initialize x. Next, we evaluate x * 3, which is 6 and initialize y to that value.

1:39
Next we compute y/2, which is 3 and initialize z to that value. The last statement says x gets (2+z) mod 2. Since the 2+z is in parentheses, we compute that first and get 5. Next, we compute 5 mod 2. Remember from your reading that 5 mod 2 means we divide 5 by 2, but take the remainder, not the quotient. So this expression evaluates to 1. So we update the value in x's box to be 1. Okay, great, now you should be able to evaluate code involving a wide variety of mathematical expressions.