

Abbreviating Operators

In the previous video, I talked about operators, operands, and expressions. In this video, I'll continue the discussion about operators, and you'll see how they can be abbreviated to simplify code.

In the next section of this course, we'll be transitioning to coding in a code editor called IntelliJ. I want to continue to prepare you for that environment by using the curly brace feature in JShell.

Why do we want to use multiple statements in curly braces {}?

So, why use multiple statements in curly braces?

- First, it's a way to group statements together before executing them.
- It allows us to put statements on multiple lines which is more natural and readable.
- We can execute the group of statements as a whole, which more closely resembles running code in Java.

Incrementing by One

Incrementing by one is a very common requirement in programming.

Obviously we can do the following:

```
result = result + 1;
```

But we also have two other shorthand ways to do this same thing.

| Shorthand (or Abbreviating) Operator | Code Sample |
|--|-------------|
| Post-fix Increment Operator | result++; |
| Compound Assignment Operator with + sign | result+=1; |

Decrementing by One

Decrementing by one is also very common.

We can decrement simply by using the equation:

```
result = result - 1;
```

But we also have two other shorthand ways to do this same thing.

| Shorthand (or Abbreviating) Operator | Code Sample |
|--|-------------|
| Post-fix Decrement Operator | result--; |
| Compound Assignment Operator with - sign | result-=1; |

Compound Assignment Operator Challenge

Using the code we have been using, either by scrolling up and editing the group of statements, or creating a new group.

- Initialize an **int** variable, named **result**, to the value of 10, rather than 1.
- Next, use the compound assignment operator, with the minus sign, to subtract a number from **result**, using a value of your choice.
- Print the result out, using the `System.out.print` statement.

Compound Operator Assignment

When **result** is an **int**, the compound operator assignment

```
result -= 5.5;
```

give us a different result from what we expected, which was

```
result = result - 5.5;
```

Compound Assignment Operator

The compound assignment operator

$x -= y$

is often said to be

$x = x - y$

but that's not entirely true if y is not the same data type as x .

Compound Assignment Operator

`x -= y`

is really

`x = (data type of x) (x - y)`

An implicit cast is done when using this operator, so no error occurs, but unexpected results may happen, as we have seen.

Compound Assignment Operator

So, summarizing, for our own sample of code:

```
result -= 5.5;
```

is really

```
result = (int) (result - 5.5);
```

The abbreviating operators

The abbreviating operators we've discussed so far are:

| Shorthand Operator | Code Sample |
|------------------------------------|---------------------------|
| Post-fix Increment Operator | <code>result++;</code> |
| Post-fix Decrement Operator | <code>result--;</code> |
| Addition Compound Assignment | <code>result += 5;</code> |
| Subtraction Compound Assignment | <code>result -= 5;</code> |
| Multiplication Compound Assignment | <code>result *= 5;</code> |
| Division Compound Assignment | <code>result /= 5;</code> |