Operators, Operands and Expressions

In the previous video, we did a recap of the primitive data types, as well as getting a formal introduction into String.

In this video, we're going to take a look at operators, operands, and expressions.

I've discussed very briefly what operators are, and we've used a few of them in some expressions.

Let's go deeper into that now.

What are Operators?

So what are operators?

Operators in Java are special symbols that perform specific operations on one, two, or three operands, and then return a result.

In the example below, which we saw in a previous video, we used the plus or addition operator, as well as the multiplication operator.

```
long longTotal = 50000L + 10L * (byteValue + shortValue + intValue);
```

But there are many other operators in Java.

What are Operands?

So what is an operand?

An operand is a term used to describe any object that is manipulated by an operator.

So if we consider this:

```
int myVar = 15 + 12;
```

The plus here is the operator, and 15 and 12 are the operands. Variables used instead of literals can also be operands.

Another example from a challenge we did previously:

```
long longTotal = 50000L + 10L * (byteValue + shortValue + intValue);
```

In the line above, **byteValue**, **shortValue** and **intValue** are operands, as are the numeric literals.

What are Expressions?

What's an expression?

An expression is formed by combining variables, literals, method return values, (which we haven't covered yet), and operators.

They are a way of forming and combining those values to produce a result.

In the line below, 15 plus 12 is the expression, which returns the value of 27.

```
int myVar = 15 + 12;
```

What are Expressions?

In the statement below, `byteValue + shortValue + intValue` is the expression.

```
int sumOfThree = byteValue + shortValue + intValue;
```

How may operators are in this code?

How many operators would you say, are in this statement?

```
int result = 1 + 2;  // 1 + 2 = 3
```

Well, we've actually got two, the equal operator, and the plus operator.

What is a Comment?

Comments are ignored by the computer, and are added to a program, to help describe something. Comments are there for humans to read.

We use two forward slashes in front of any code, or on a blank line. Anything after the two forward slashes, right through to the end of the line, is ignored by the computer.

So aside from describing something about a program, comments can also be used to temporarily disable code.

What is the effect of this code on the value in previousResult?

```
int result = 1 + 2;
int previousResult = result;
result = result - 1;
```

Before we move on, some of you may be wondering about the value that is now in the **previousResult** variable after the last statement.

To summarize, we assigned result to **previousResult**, and then we changed the value of **result**. But, did this also change the value in **previousResult**? That is a good question.

The + Operator on character data types

char	String
• Holds one, and only one, character • Literal enclosed in Single Quotes	• Can hold multiple characters • Literal enclosed in Double Quotes

The + Operator on char

You might remember that we said chars are stored as 2 byte numbers in memory.

When you use the plus operator with chars, it is these numbers in memory that get added together.

The character values don't get concatenated.

The Remainder Operator

The remainder operator is represented in Java by the % sign.

The remainder operator goes by several other names: modulus, modulo or just plain mod for short.

The remainder operator returns the remaining value from a division operation.

If there is no remaining value, the result is 0.

The Remainder Operator

This table shows some examples.

Division Result	Remainder Result	Explanation
10 / 5 = 2	10 % 5 = 0	Ten can be divided evenly by 5, so there is no remainder.
10 / 2 = 5	10 % 2 = 0	Ten can be divided evenly by 2, so there is no remainder.
10 / 3 = 3	10 % 3 = 1	Ten cannot be divided evenly by 3, but we get 3 from the division which gives us 9 with 1 remaining.
10 / 1 = 10	10 % 1 = 0	Using 1 on the right side of the remainder operate will always give a result of 0.

Summary of Operators

This tables shows the five operators we just reviewed. For all of the numeric types (whole numbers and decimals), the operators are mathematical operators as shown.

Operator	Numeric types	char	boolean	String
+	Addition	Addition	n/a	Concatenation
-	Subtraction	Subtraction	n/a	n/a
*	Multiplication	Multiplication	n/a	n/a
/	Division	Division	n/a	n/a
%	Remainder (Modulus)	Remainder (Modulus)	n/a	n/a

Summary of Operators

Only one operator, the + operator, is supported by String, but it means concatenation, not addition.

Operator	Numeric types	char	boolean	String
+	Addition	Addition	n/a	Concatenation
-	Subtraction	Subtraction	n/a	n/a
*	Multiplication	Multiplication	n/a	n/a
/	Division	Division	n/a	n/a
%	Remainder (Modulus)	Remainder (Modulus)	n/a	n/a

Summary of Operators

None of the operators are applicable to a boolean.

Operator	Numeric types	char	boolean	String
+	Addition	Addition	n/a	Concatenation
-	Subtraction	Subtraction	n/a	n/a
*	Multiplication	Multiplication	n/a	n/a
/	Division	Division	n/a	n/a
%	Remainder (Modulus)	Remainder (Modulus)	n/a	n/a

Summary of Operators

Because the char is stored as a whole number literal, all the operations are applicable to a char.

Operator	Numeric types	char	boolean	String
+	Addition	Addition	n/a	Concatenation
-	Subtraction	Subtraction	n/a	n/a
*	Multiplication	Multiplication	n/a	n/a
/	Division	Division	n/a	n/a
%	Remainder (Modulus)	Remainder (Modulus)	n/a	n/a