

Introduction

In the previous video, we saw that Java has eight primitive data types, and that the wrapper classes give us extra options.

Of these primitive types, half are used to store whole numbers (numbers without a fractional or decimal component), one of which we've explored already, the integer, or the int data type.

In this video, we'll take a look at the other three whole number primitive data types, these are the byte, the short, and the long.

byte, short, int, long

We've previously said that Java has four primitive data types used to store whole numbers, these are the byte, the short, the int, and the long.

| Whole number Data Type | Wrapper Class | What's noteworthy |
|------------------------|---------------|--|
| byte | Byte | Has the smallest range |
| short | Short | |
| int | Integer | Java's default data type for whole numbers |
| long | Long | Has the largest range |

They are listed here in this table, by the range of values the type will support, the byte supports the smallest range, and the long supports the largest range.

The byte data type

The minimum value of a byte is -128.

The maximum value of a byte is 127.

Given its small range, you probably won't be using the byte data type a lot.

The byte wrapper class is the Byte with a capital B.

The short data type

The minimum value of a short is -32768.

The maximum value of a short is 32767.

The short wrapper class is the Short with a capital S.

byte and short overflow/underflow

Both the byte and the short, have the same overflow and underflow issue as the int data type has, but obviously with their own range of numbers.

Size of Primitive Types and Width

Size, or Width, is the amount of space that determines (or limits) the range of values we've been discussing:

| Data Type | Width (in bits) | Min Value | Max Value |
|-----------|-----------------|-------------|------------|
| byte | 8 | -128 | 127 |
| short | 16 | -32768 | 32767 |
| int | 32 | -2147483648 | 2147483647 |

A byte, can store 256 numbers and occupies eight bits, and has a width of 8.

A short, can store a large range of numbers and occupies 16 bits, and has a width of 16.

An int, has a much larger range as we know, and occupies 32 bits, and has a width of 32.

Using a numeric literal character suffix

The number 100, by default, is an int.

Java allows certain numeric literals to have a suffix appended to the value, to force it to be a different data type from the default type.

The long is one of these types and its suffix is an 'L'.

This is one of the few instances Java is not case sensitive, a lowercase 'l' or an uppercase 'L' at the end of a whole number mean the same thing – the number is a long.

How big is the difference between an int and a long?

How big is the difference, in the range of values that a long can store, compared to the int?

You can see, from this table, that the difference is quite significant.

| Data Type | Width (in bits) | Min Value | Max Value |
|-----------|-----------------|----------------------|---------------------|
| int | 32 | -2147483648 | 2147483647 |
| long | 64 | -9223372036854775808 | 9223372036854775807 |

When is L required?

A numeric literal that exceeds `Integer.MAX_VALUE` must use the 'L' suffix.

We cannot create a numeric literal in Java, that exceeds `Integer.MAX_VALUE`, without using the 'L' suffix, we'll always get the error 'integer number too large'.