What's a Method Reference?

Java gives us an alternative syntax to use for this second kind of lambda, that uses named methods.

These are called method references.

These provide a more compact, easier-to-read lambda expression, for methods that are already defined on a class.

For the last couple of videos for example, I've been ignoring Intelli-J's warnings and hints, whenever I've used System.out.println in a lambda expression.

That's because it can be replaced with a method reference.

Why are these statements equal?

At first glance, it's not really obvious why a method reference has this syntax.

Lambda Expression	**Method Reference**
`s -> System.out.println(s)`	`System.out::println`

A method reference abstracts the lambda expression even further, eliminating the need to declare formal parameters.

We also don't have to pass arguments to the method in question, in this case println.

A method reference has double colons, between the qualifying type, or object, and the method name.

In this example of a Consumer interface, not only is the method inferred, but the parameters are as well.

What methods can be used in method references?

Methods which can be used, are based on the context of the lambda expression.

This means the method reference, is again dependent on the targeted interface's method.

You can reference a static method on a class.

You can reference an instance method from either an instance external to the expression, or an instance passed as one of the arguments.

Or you can reference a constructor, by using new as the method.

Method references can be used to increase the readability of your code.

Deferred Method Invocation

When you create variables that are lambda expressions or method references, it's important to remember that the code isn't invoked at that point.

The statement or code block gets invoked at the point in the code that the targeted functional method is called.