The Method

Java's description of the method is:

A method declares executable code that can be invoked, passing a fixed number of values as arguments.

The Benefits of the Method

A method is a way of reducing code duplication.

A method can be executed many times with potentially different results, by passing data to the method in the form of arguments.

Structure of the Method

One of the simplest ways to declare a method is shown on this slide.

This method has a name, but takes no data in, and returns no data from the method (which is what the special word **`void`** means in this declaration).

```
public static void methodName() {

    // Method statements form the method body

}
```

Executing a Method as a Statement

To execute a method, we can write a statement in code, which we say is calling, or invoking, the method.

For a simple method like calculateScore, we just use the name of the method, where we want it to be executed, followed by parentheses, and a semi-colon to complete the statement.

So for this example, the calling statement would look like the code shown here:

```
calculateScore();
```

Structure of the Method

Where we previously had empty parentheses after the method name, we now have method parameters in the declaration.

```
public static void methodName(p1type p1, p2type p2, {more}) {

    // Method statements form the method body

}
```

Parameters or Arguments?

Parameters and arguments are terms that are often used interchangeably by developers.

But technically, a parameter is the definition as shown in the method declaration, and the argument will be the value that's passed to the method when we call it.

Executing a Method with parameters

To execute a method that's defined with parameters, you have to pass variables, values, or expressions that match the type, order and number of the parameters declared.

In the calculateScore example, we declared the method with four parameters, the first a boolean, and the other three of int data types.

So we have to pass first a boolean, and then 3 int values as shown in this statement:

```
calculateScore(true, 800, 5, 100);
```

We can't pass the boolean type in any place, other than as the first argument, without an error.

Executing a Method with parameters

The statement below would cause an error.

```
calculateScore(800, 5, 100, true);
```

And you can't pass only a partial set of parameters as shown here.

This statement, too, would cause an error.

```
calculateScore(true, 800);
```