Local Classes

Local classes are inner classes, but declared directly in a code block, usually a method body.

Because of that, **they don't have access modifiers**, and are only accessible to that method body while it's executing.

Like an inner class, they have access to all fields and methods on the enclosing class.

They can also access local variables and method arguments, that are final or effectively final.

Local Class's 'Captured Variables'

When you create an instance of a local class, referenced variables used in the class, from the enclosing code, are 'captured'.

This means a copy is made of them, and the copy is stored with the instance.

This is done because the instance is stored in a different memory area, than the local variables in the method.

For this reason, if a local class uses local variables, or method arguments, from the enclosing code, these must be final or effectively final.

Final Variables and Effectively Final

The code sample on this slide shows:

- A method parameter, called methodArgument in the doThis method, declared as final.
- And a local variable, in the method block, Field30, also declared with the key word final.

In both these cases, this means you can't assign a different value, once these are initialized.

```
class ShowFinal {

    private void doThis(final int methodArgument) {

        final int Field30 = 30;
    }
}
```

These are **explicitly final**, and any of these could be used in a local class, because of this.

Effectively Final

In addition to explicitly final variables, you can also use **effectively final** variables in your local class.

A local variable or a method argument are effectively final, if a value is assigned to them, and then never changed after that.

Effectively final variables can be used in a local class.

Additional Local Types

As of JDK 16, you can also create a local record, interface and enum type, in your method block.

These are all implicitly static types, and therefore aren't inner classes, or types, but static nested types.

The record was introduced in JDK16.

Prior to that release, there was no support for a local interface or enum in a method block either.