ORACLE

Lesson 1-6: Referencing External Variables in Lambda Expressions

Local Variable Capture

- Lambda expressions can refer to *effectively final* local variables from the surrounding scope
  - Effectively final: A variable that meets the requirements for final variables (i.e., assigned once), even if not explicitly declared final
  - Closures on values, not variables

```
void expire(File root, long before) {
    root.listFiles(File p -> p.lastModified() <= before);
}
```

What Does ‘this’ Mean In A Lambda

- ‘this’ refers to the enclosing object, not the lambda itself
- Think of ‘this’ as a final predefined local
- Remember the Lambda is an *anonymous function*
  - It is not associated with a class
  - Therefore there can be no ‘this’ for the Lambda

Referencing Instance Variables

Which are not final, or effectively final

```
class DataProcessor {
  private int currentValue;

  public void process() {
    DataSet myData = myFactory.getDataSet();
    dataSet.forEach(d -> d.use(currentValue++));
  }
}
```

Referencing Instance Variables

Which are not final, or effectively final

```
class DataProcessor {
  private int currentValue;

  public void process() {
    DataSet myData = myFactory.getDataSet();
    dataSet.forEach(d -> d.use(this.currentValue++));
  }
}
```

‘this’ (which *is* effectively final) inserted by the compiler

Section 6

Summary

- Variables in the surrounding scope can be used in a Lambda expression
  - But they must be either final, or effectively final
- `this` in a Lambda refers to the object of the surrounding scope
  - The compiler will insert a reference to this for you where required


ORACLE®