Synthetic Methods

Lesson Objectives

- After completing this lesson, you should be able to:
  - Describe how the Scala compiler generates functionality for you
  - Explain what the synthetic **`equals()`**, **`hashCode()`**, **`toString()`** and **`copy()`** methods do
  - Outline how you would use immutable case classes in a program where state is changing

Typesafe

Coding and Maintenance are Expensive

- Writing and maintaining the source code required by the JVM for simple data classes is difficult
- To support the features of case classes, a comparable Time class in Java would be over 70 lines of code

What are Synthetic Methods?

- Scala's compiler generates this "boilerplate" for you
- The implementations are rock solid and proven
  - `equals()`
  - `hashCode()`
  - `toString()`
  - `copy()`

`equals()`

- This method is required by the JVM, but the default implementation only compares whether an instance of the class is the exact same instance
- Scala provides value-based equivalence, allowing you to compare whether two different instances of a class have the same state

equals()

```
scala> case class Time(hours: Int = 0, minutes: Int = 0)
defined class Time

scala> val time = Time(9, 0)
time: Time = Time(9,0)

scala> time == Time(9)
res0: Boolean = true

scala> time == Time(9, 30)
res1: Boolean = false
```

Typesafe

hashCode()

- This method is required for any class that you might want to put into a hashed collection, such as a HashMap or HashSet

```
scala> case class Time(hours: Int = 0, minutes: Int = 0)
defined class Time

scala> Time(9, 45).hashCode()
res0: Int = 471971180
```

Typesafe

`toString()`

- This method is required by the JVM, but the default implementation prints out a virtual representation of the instance location in memory
- The synthetic **`toString()`** provided by Scala's case class shows you the values inside of the class
- You can override this to make it even better

Typesafe

toString()

```
scala> class Time(hours: Int = 0, minutes: Int = 0)
defined class Time

scala> new Time()
res0: Time = Time@4d591d15

scala> case class Time(hours: Int = 0, minutes: Int = 0)
defined class Time

scala> Time()
res1: Time = Time(0,0)
```

`copy()`

- This method is not required by the JVM
- The synthetic `copy()` provided by Scala's case class allows you to remain immutable and use "snapshots" of case classes when state needs to change

copy()

```
scala> case class Time(hours: Int = 0, minutes: Int = 0)
defined class Time

scala> var time = Time(9, 45)
time: Time = Time(9,45)

scala> time = time.copy(minutes = 50)
time: Time = Time(9,50)
```

Typesafe

Lesson Summary

- Having completing this lesson, you should be able to:
  - Describe how the Scala compiler generates functionality for you
  - Explain what the synthetic **`equals()`**, **`hashCode()`**, **`toString()`** and **`copy()`** methods do
  - Outline how you would use immutable case classes in a program where state is changing