Option

Lesson Objectives

- After completing this lesson, you should be able to:
  - Describe the relevance of Option in the Scala type system
  - Outline how to use Option in your types

Algebraic Data Types (ADTs)

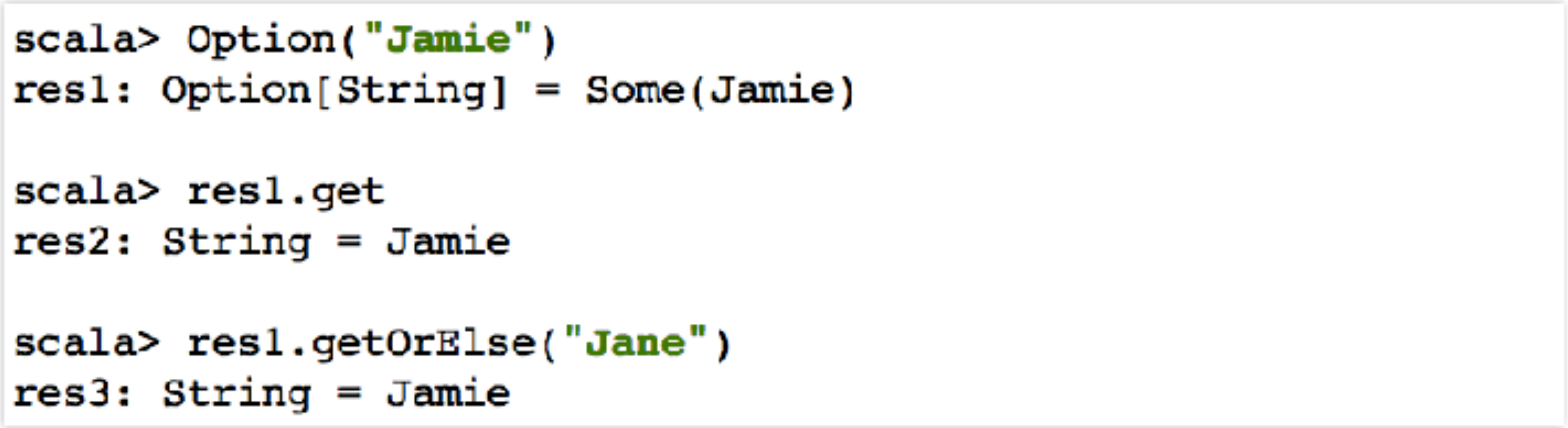
- A distinct set of possible types
- Intuition:
  - Days of the week
  - Binary light switches

Option

- Not a collection, but a container
- An ADT representing the existence of a value
- `Some` is the representation of a value
- `None` is the representation of the absence of a value
- Allows us to avoid `null` on the JVM

Typesafe

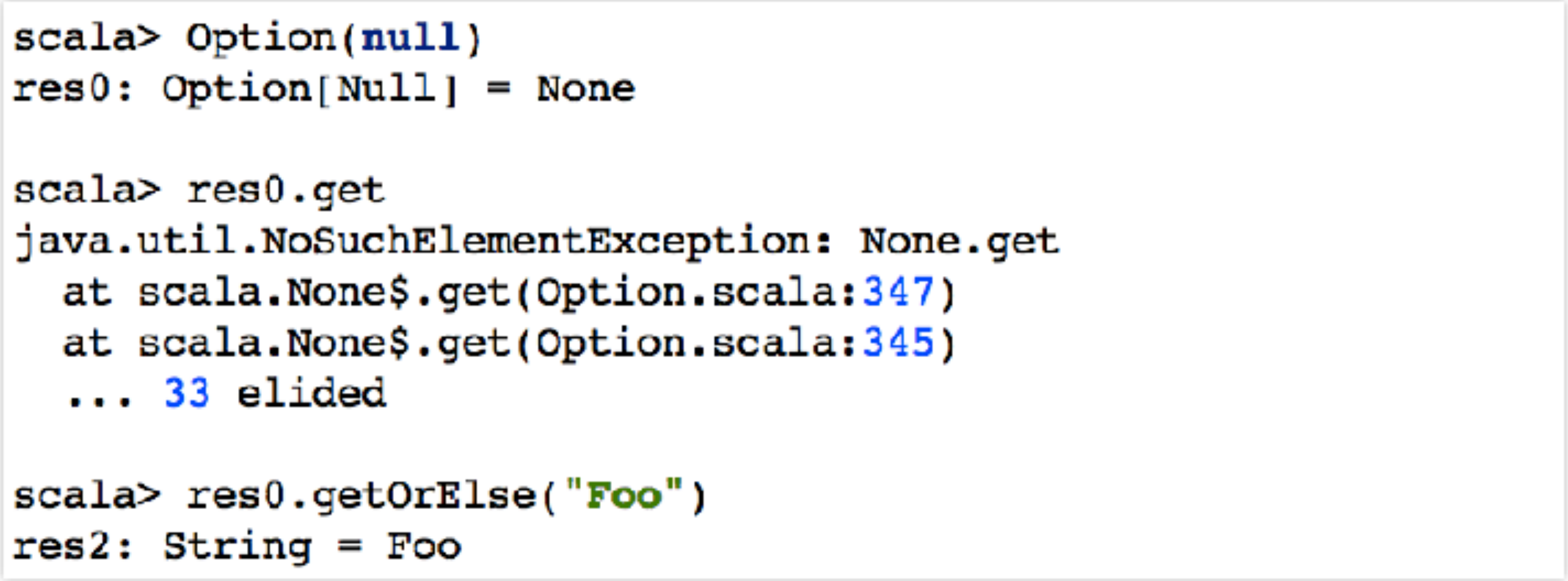
Option



```
scala> Option("Jamie")
res1: Option[String] = Some(Jamie)

scala> res1.get
res2: String = Jamie

scala> res1.getOrElse("Jane")
res3: String = Jamie
```

Typesafe

Option



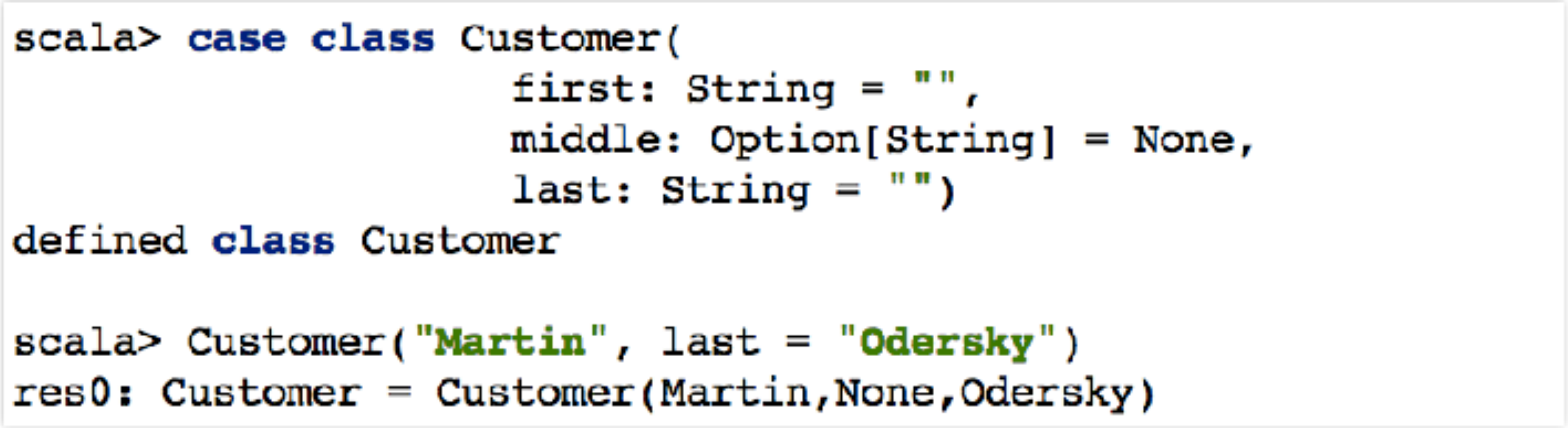
```
scala> Option(null)
res0: Option[Null] = None

scala> res0.get
java.util.NoSuchElementException: None.get
  at scala.None$.get(Option.scala:347)
  at scala.None$.get(Option.scala:345)
  ... 33 elided

scala> res0.getOrElse("Foo")
res2: String = Foo
```

Typesafe

Option

- Option allows us to create APIs where the possible absence of value is encoded in the type system
- We can then perform behavior without asking whether or not the value is `null` in advance

Option



```
scala> case class Customer(
                    first: String = "",
                    middle: Option[String] = None,
                    last: String = "")
defined class Customer

scala> Customer("Martin", last = "Odersky")
res0: Customer = Customer(Martin,None,Odersky)
```

Typesafe

Lesson Summary

- Having completing this lesson, you should be able to:
  - Describe the relevance of Option in the Scala type system
  - Outline how to use Option in your types