Apply and Unapply

Lesson Objectives

- After completing this lesson, you should be able to:
  - Illustrate the difference between a type and a term
  - Describe how the apply method works in both objects and classes
  - Outline how unapply works

Types versus Terms

- A type is a description of a concept in an application
  - A class is a type
- A term is a concrete representation of a type
  - Any class instance (including an object) is a term
  - A method is a term, as it is also concrete and “callable”

Calling a Term

- Like some other languages, Scala allows you to "call" a term without specifying the method you want to call on it

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

scala> val time = Time(9, 0)
res0: Time = Time(9, 0)
```

Typesafe

How Did That Work?

- When you create a case class, the compiler generates a companion object for the class for you
- Calling **`Time(9, 0)`** is actually calling the companion object Time and delegating to the **`apply()`** method inside of it.

```
scala> Time(9)
res0: Time = Time(9,0)

scala> Time.apply(9)
res1: Time = Time(9,0)
```

Typesafe

An Example of apply

```
object Reverse {
  def apply(s: String): String =
    s.reverse
}

scala> Reverse("Hello")
res0: String = olleH
```

Another Example of apply

```
scala> Array(1, 2, 3)
res0: Array[Int] = Array(1, 2, 3)

scala> res0(0)
res1: Int = 1
```

Unapply Deconstructs a Case Class

```
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.unapply(time)
res2: Option[(Int, Int)] = Some((9,0))
```

Typesafe

Lesson Summary

- Having completing this lesson, you should be able to:
  - Illustrate the difference between a type and a term
  - Describe how the apply method works in both objects and classes
  - Outline how unapply works