Advanced Pattern Matching

Takeaways

We can define our own patterns!

```scala
class Person(val name: String, val age: Int)
object Person {
  def unapply(person: Person): Option[(String, Int)] =
    Some((person.name, person.age))
}
```

the object we want to decompose

results as an Option or Option(tuple)

```scala
person match {
  case Person(name, _) => println(s"Hi, I'm $name.")
}
```

the compiler searches the appropriate unapply for us

Patterns are independent of classes we decompose.

Takeaways

Infix patterns

```
numbers match {
  case head :: Nil => println("single element" + head)
  // equivalent:
  case ::(head, Nil) => println("single element" + head)
}
```

Unapply sequences

```
object MyList {
  def unapplySeq[A](list: MyList[A]): Option[Seq[A]] = // ...
}

myList match {
  case MyList(1,2,_*) => // ...
}
```

Custom return types for unapply (really rare!)

Scala rocks