Recap: Enums and Pattern Matching

Principles of Functional Programming

Martin Odersky

Recap: Case Classes

Case classes classes are Scala's preferred way to define complex data.

Example: Representing JSON (Java Script Object Notation)

```
{ "firstName" : "John",
  "lastName" : "Smith",
  "address": {
     "streetAddress": "21 2nd Street",
     "state": "NY",
     "postalCode": 10021
  },
  "phoneNumbers": [
    { "type": "home", "number": "212 555-1234" },
    { "type": "fax", "number": "646 555-4567" }
  ]
}
```

Representation of JSON with Case Classes

```
abstract class JSON
object JSON:
  case class Seq (elems: List[JSON])           extends JSON
  case class Obj (bindings: Map[String, JSON]) extends JSON
  case class Num (num: Double)                 extends JSON
  case class Str (str: String)                 extends JSON
  case class Bool(b: Boolean)                  extends JSON
  case object Null                             extends JSON
```

Representation of JSON with Enums

Case class hierarchies can be represented more concisely as enums:

```
enum JSON:
  case Seq (elems: List[JSON])
  case Obj (bindings: Map[String, JSON])
  case Num (num: Double)
  case Str (str: String)
  case Bool(b: Boolean)
  case Null
```

Example

```
val data = JSON.Obj(Map(
  "firstName" -> JSON.Str("John"),
  "lastName" -> JSON.Str("Smith"),
  "address" -> JSON.Obj(Map(
    "streetAddress" -> JSON.Str("21 2nd Street"),
    "state" -> JSON.Str("NY"),
    "postalCode" -> JSON.Num(10021)
  )),
  "phoneNumbers" -> JSON.Seq(List(
    JSON.Obj(Map(
      "type" -> JSON.Str("home"), "number" -> JSON.Str("212 555-1234")
    )),
    JSON.Obj(Map(
      "type" -> JSON.Str("fax"), "number" -> JSON.Str("646 555-4567")
    )) )) ))
```

Pattern Matching

Here's a method that returns the string representation JSON data:

```
def show(json: JSON): String = json match
  case JSON.Seq(elems) =>
    elems.map(show).mkString("[", ", ", "]")
  case JSON.Obj(bindings) =>
    val assocs = bindings.map
      (key, value) => s"$"$key$": ${show(value)}"
    s"{${assocs.mkString(", ")}}"
  case JSON.Num(num) => num.toString
  case JSON.Str(str) => s"$"$str$""
  case JSON.Bool(b)  => b.toString
  case JSON.Null     => "null"
}
```