JACEK KUNICKI

@rucek

SOFTWAREMILL

TYPE CLASSES FROM ZERO TO HERO

POLYMORPHISM

POLYMORPHISM

- type independent - same implementation for all types, e.g. `List.head`
- type dependent - different implementations for each type:
  - ad-hoc/compile-time (e.g. method overloading)
  - runtime (subclassing)

BETTER AD-HOC POLYMORPHISM

- add a `show` method to any type
- but we can’t (or don’t want to) overload methods
- a single method that somehow knows how to „show” different types

Talk is cheap
show me the
CODE

SOFTWARE**MILL**

LIBRARIES

- cats/Scalaz
- Shapeless/Magnolia for derivation
- simulacrum - less boilerplate with `@typeclass`

REAL LIFE

SCALA COLLECTIONS

```scala
def sum[B >: A](implicit num: Numeric[B]): B =
  foldLeft(num.zero)(num.plus)
```

JSON

```
def toJson(implicit writer: JsonWriter[T]): JsValue =
  writer.write(any)
```

```
def toJson[T](o: T)(implicit tjs: Writes[T]): JsValue
```

FOR COMPREHENSIONS

```
def addOptions[T: Numeric](a: Option[T], b: Option[T]): Option[T] =
  for {
    x <- a
    y <- b
  } yield x + y
```

```
def addFutures[T: Numeric](a: Future[T], b: Future[T]): Future[T] =
  for {
    x <- a
    y <- b
  } yield x + y
```

FOR COMPREHENSIONS

```
def add[T: Numeric, FM[_] : FlatMap](a: FM[T], b: FM[T]): FM[T] =
  for {
    x <- a
    y <- b
  } yield x + y
```

```
add(1.some, 3.some)
add(Future.successful(1), Future(3))
```

https://typelevel.org/cats/typeclasses.html

DOTTY (AKA SCALA 3.0)

- no more implicit classes for syntax, just:
  - a polymorphic trait
  - instances defined with `given`
- simulacrum not supported anymore (macros)

TAKEAWAYS

- ad-hoc polymorphism, also for existing types
- powerful DSLs when combined with implicits
- context bounds vs. implicit parameter

BE MORE FUNCTIONAL

Read

SCALA TIMES

Est. 2014

★ ★

WEEKLY SOURCE OF SCALA NEWS

http://scalatimes.com

THANK YOU!

rucek/type-classes-from-zero-to-hero

@rucek

SOFTWAREMILL