EPFL

Priorities Between Given Instances

Effective Programming in Scala

Priorities

Actually, several given instances matching the same type don't generate an ambiguity if one is **more specific** than the other.

A definition `given a: A` is more specific than a definition `given b: B` if:

- `a` is in a closer lexical scope than `b`,
- `a` is defined in a class or object which is a subclass of the class defining `b`,
- type `A` is a subtype of type `B`,
- type `A` has more "fixed" parts than `B`.

Priorities: Example (1)

Which given instance is summoned here?

```
given universal[A]: A = ???
given int: Int = ???

summon[Int]
```

Priorities: Example (2)

Which given instance is summoned here?

```
trait A:
  given x: Int = 0

trait B extends A:
  given y: Int = 1

object C extends B:
  summon[Int]
```

Priorities: Example (3)

Which given instance is summoned here?

```
given x: Int = 0
def foo() =
  given y: Int = 1
  summon[Int]
```

Priorities: Example (4)

Which given instance is summoned here?

```
class General()
class Specific() extends General()

given general: General = General()
given specific: Specific = Specific()

summon[General]
```

Summary

Several given instances matching the same type don’t generate an ambiguity if one is **more specific** than the others.