Trait Parameters

Scala 3 allows traits to have parameters, just like classes have parameters.

```
trait Greeting(val name: String):
  def msg = s"How are you, $name"

class C extends Greeting("Bob"):
  println(msg)
```

Arguments to a trait are evaluated immediately before the trait is initialized.

One potential issue with trait parameters is how to prevent ambiguities. For instance, you might try to extend `Greeting` twice, with different parameters.

```
class D extends C, Greeting("Bill") // error: parameter passed twice
```

Should this print "Bob" or "Bill"? In fact this program is illegal, because it violates the second rule of the following for trait parameters:

1. If a class `C` extends a parameterized trait `T`, and its superclass does not, `C` *must* pass arguments to `T`.
2. If a class `C` extends a parameterized trait `T`, and its superclass does as well, `C` *must not* pass arguments to `T`.
3. Traits must never pass arguments to parent traits.

Here's a trait extending the parameterized trait `Greeting`.

```
trait FormalGreeting extends Greeting:
  override def msg = s"How do you do, $name"
```

As is required, no arguments are passed to `Greeting`. However, this poses an issue

when defining a class that extends `FormalGreeting`:

```
class E extends FormalGreeting // error: missing arguments for `Greeting`.
```

The correct way to write `E` is to extend both `Greeting` and `FormalGreeting` (in either order):

```
class E extends Greeting("Bob"), FormalGreeting
```

Traits With Context Parameters

This "explicit extension required" rule is relaxed if the missing trait contains only context parameters. In that case the trait reference is implicitly inserted as an additional parent with inferred arguments. For instance, here's a variant of greetings where the addressee is a context parameter of type `ImpliedName`:

```
case class ImpliedName(name: String):
  override def toString = name

trait ImpliedGreeting(using val iname: ImpliedName):
  def msg = s"How are you, $iname"

trait ImpliedFormalGreeting extends ImpliedGreeting:
  override def msg = s"How do you do, $iname"

class F(using iname: ImpliedName) extends ImpliedFormalGreeting
```

The definition of `F` in the last line is implicitly expanded to

```
class F(using iname: ImpliedName) extends
  Object,
  ImpliedGreeting(using iname),
  ImpliedFormalGreeting(using iname)
```

Note the inserted reference to the super trait `ImpliedGreeting`, which was not mentioned explicitly.

Reference

For more information, see Scala SIP 25.

‹ Other ... Transp... ›