TOUR OF SCALA

CLASS COMPOSITION WITH MIXINS

Mixins are traits which are used to compose a class.

```
abstract class A {
  val message: String
}
class B extends A {
  val message = "I'm an instance of class B"
}
trait C extends A {
  def loudMessage = message.toUpperCase()
}
class D extends B with C

val d = new D
println(d.message)  // I'm an instance of class B
println(d.loudMessage)  // I'M AN INSTANCE OF CLASS B
```

Class `D` has a superclass `B` and a mixin `C`. Classes can only have one superclass but many mixins (using the keywords `extends` and `with` respectively). The mixins and the superclass may have the same supertype.

Now let's look at a more interesting example starting with an abstract class:

```
abstract class AbsIterator {
  type T
  def hasNext: Boolean
  def next(): T
}
```

The class has an abstract type `T` and the standard iterator methods.

Next, we'll implement a concrete class (all abstract members `T`, `hasNext`, and `next` have implementations):

```
class StringIterator(s: String) extends AbsIterator {
  type T = Char
  private var i = 0
  def hasNext = i < s.length
  def next() = {
    val ch = s charAt i
    i += 1
    ch
  }
}
```

`StringIterator` takes a `String` and can be used to iterate over the String (e.g. to see if a String contains a certain character).

Now let's create a trait which also extends `AbsIterator`.

```
trait RichIterator extends AbsIterator {
  def foreach(f: T => Unit): Unit = while (hasNext) f(next())
```

```
  def foreach(f: T => Unit): Unit = while (hasNext) f(next())
}
```

This trait implements `foreach` by continually calling the provided function `f: T => Unit` on the next element (`next()`) as long as there are further elements (`while (hasNext)`). Because `RichIterator` is a trait, it doesn't need to implement the abstract members of AbsIterator.

We would like to combine the functionality of `StringIterator` and `RichIterator` into a single class.

```
class RichStringIter extends StringIterator("Scala") with RichIterator
val richStringIter = new RichStringIter
richStringIter.foreach(println)
```

The new class `RichStringIter` has `StringIterator` as a superclass and `RichIterator` as a mixin.

With single inheritance we would not be able to achieve this level of flexibility.

← previous next →

Contributors to this page:

ckipp01 mlachkar kk3399 ashawley cobbce fineconstant mghildiy martijnhoekstra mschweizer SethTisue Kobenko x0st crislinu voidance heathermiller