An Exercise in Program Structuring  
From Java to Scala 
Jacques Noyé 
27 may 2013

1 Objective

The objective is to get acquainted with basic Scala syntax and explore different ways to structure a program in Scala, starting from a Java top down design of complex numbers (see figure [1] where Complex is itself a composition of interfaces.

2 Compositions en Scala

1. Project ComplexV1: reproduce the previous structure in Scala by using traits instead of interfaces (see figure [2]). You can also get rid of getters and setters, and use operators. Pi, floor, abs... are in the package scala.math.

2. Project ComplexV2: implement a new version following figure [3] where Polar and Cartesian include both abstract and concrete fields. A large part of the previous code can be reused (sometimes moved).

   What about including methods equals and toString in traits Polar and Cartesian?

3. Project ComplexV3: implement a variant of the previous version which restricts the use of complex numbers to an API consisting of the trait Complex which its companion object.

4. Project ComplexV4: introduce a trait Field[T], similar to the initial Java interface ComplexField, and compose the trait Complex with traits Polar, Cartesian and Field[T].

   Define in Field[T] the power operator ^ as well as the neutral elements one and zero in the traits Polar and Cartesian, respectively.

   Check, for instance, that Complex(2, new Angle(Pi / 4)) ^ 8 == Complex(256, 0).

5. Project ComplexV5: consider two classes ComplexeCar and ComplexPol that only implement fields and operations that can be locally managed:

   - Cartesian coordinates, sum, inverse and equality for complex numbers with polar coordinates.
   - Polar coordinates, product, square root and equality for complex with cartesian coordinates.

   Complete with implicit conversions pol2car and car2pol.
interface inheritance
class inheritance

Figure 1: Top-down design in Java
Figure 2: Top-down design in Scala
Complex CarComplex PolComplex

uses

polar coordinates with the concrete operation *
cartesian coordinates with concrete operation +

Complex

factories

Polar Complex Cartesian

CarComplex PolComplex

uses

Figure 3: Traits with concrete fields