**Scala gerarchia core classes:**

**The Predef Object (tutorial a parte)**

For your convenience, whenever you compile code, the Scala compiler automatically

imports the definitions in the java.lang package (javac does this, too).   
  
The compiler also imports the definitions in the analogous Scala package, scala.

Similarly, a number of common, Scala-specific types are made available

without qualification, such as List.

Where there are Java and Scala type names that overlap, like String, the Scala version is imported last, so it “wins.”

The compiler also automatically imports the Predef object, which defines or imports

several useful types, objects, and functions.

**Package object: (tutorial a parte)**

**Sealed classes: (eliminare default case in un pattern match di classi stesso package (e file))**

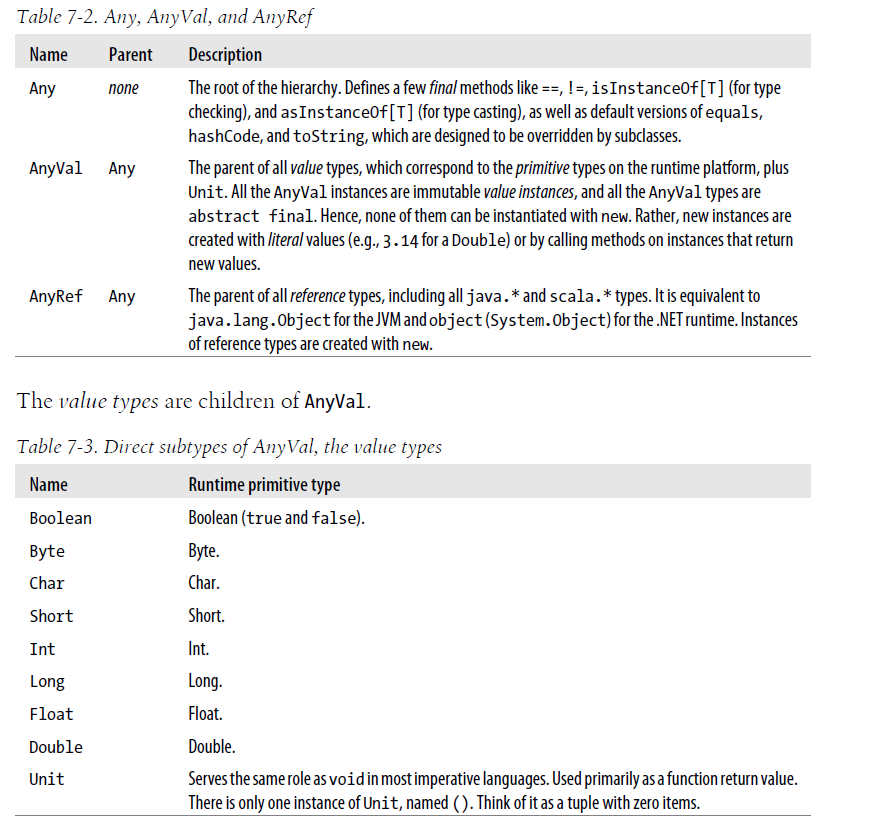
If you know that the case class hierarchy is unlikely to change and you can define the whole hierarchy in *one file*. In this situation, you can

add the sealed keyword to the declaration of the common base class.   
When sealed, the compiler knows all the possible classes that could appear in the match expression, because all of them must be defined in the same source file.   
So, if you cover all those classes in the case expressions (either explicitly or through shared parent classes), then you can safely eliminate the default case expression  
Conversely, if you omit one of the classes and you don’t provide a default case or a case for a shared parent class, the compiler warns you that the “match is not exhaustive.”

Utile solo se l’insieme di classi a cui e’ applicato il match non cambia mai.

**Gerarchia classi Core scala:**



**