吴雪峰@THOUGHTWORKS 2014.06.08

SCALA 语法糖

CASE CLASS

- equals
- toString
- copy

APPLY

- List(1,2,3) = List.apply(1,2,3)
- List(1)(0) = List.apply(1).apply(0)

UPDATE

- Map(1 -> "a")(1) = "b"
- scala.collection.mutable.Map(1 -> "a")(1) = "b"

```
final class Array[T](_length: Int)
extends java.io.Serializable with java.lang.Cloneable {

def length: Int = …
def apply(i: Int): T = …
def update(i: Int, x: T): Unit = … override def clone(): Array[T] = …

}
```

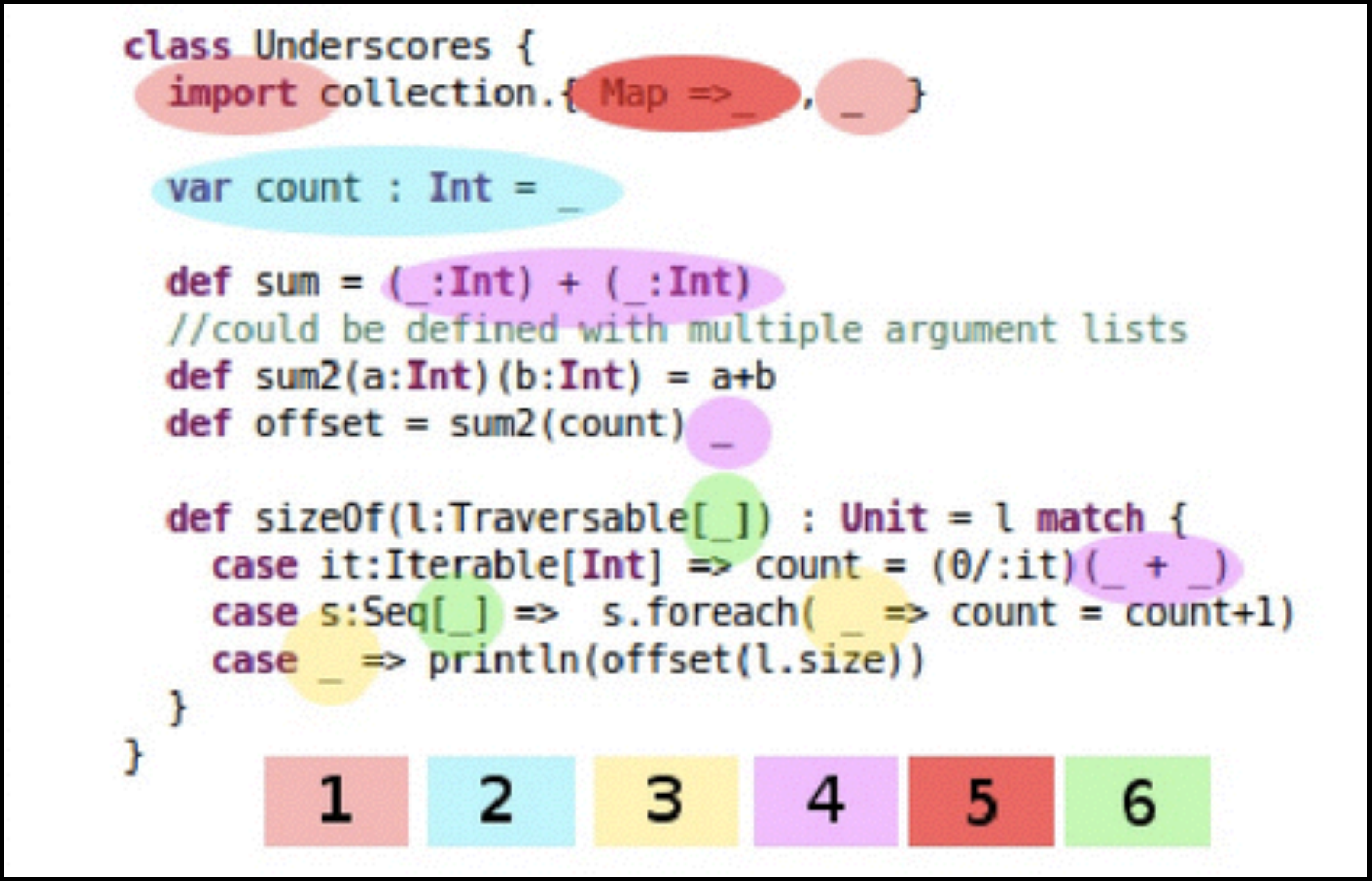
_

- Placeholder syntax: List(1, 2, 3) map (_ + 2)
- Wildcard patterns: Some(5) match { case Some(_) => println("Yes") }
- Wildcard imports: import java.util._

```scala
class Test {
  private var x0: Int = 0
  def x = x0
  def x_=(a: Int) = x0 = a
}
```

```
scala> val t = new Test
t: Test = Test@4166d6d3

scala> t.x = 1
t.x: Int = 1
```



```
class Underscores {
  import collection.{ Map =>_ , _ }

  var count : Int = _

  def sum = (_:Int) + (_:Int)
  //could be defined with multiple argument lists
  def sum2(a:Int)(b:Int) = a+b
  def offset = sum2(count) _

  def sizeOf(l:Traversable[_]) : Unit = l match {
    case it:Iterable[Int] => count = (0/:it)(_ + _)
    case s:Seq[_] =>  s.foreach( _ => count = count+1)
    case _ => println(offset(l.size))
  }
}
```

THE PREDEF OBJECT

// *classOf* --

*/** Returns the runtime representation of a class type. */*

def classOf[T]: Class[T] = **null**

// this is

// Standard

type String
type Class[T] = java.lang.Class[T]

// Miscellaneous --

type Function[-A, +B] = Function1[A, B]
type Map[A, +B] = collection.immutable.Map[A, B]

type Set[A] = collection.immutable.Set[A] **val** Map = collection.immutable.Map

val Set = collection.immutable.Set
// Manifest types, companions, and incantations for summoning ---------

type ClassManifest[T] **type** Manifest[T]
type OptManifest[T] **val** ClassManifest

val Manifest **val** NoManifest

= scala.reflect.ClassManifest[T] = scala.reflect.Manifest[T]
= scala.reflect.OptManifest[T]
= scala.reflect.ClassManifest

= scala.reflect.Manifest
= scala.reflect.NoManifest

a dummy, classOf is handled by compiler.
type aliases ---

= java.lang.String

def manifest[T](**implicit** m: Manifest[T]) = m

def classManifest[T](**implicit** m: ClassManifest[T]) = m

def optManifest[T](**implicit** m: OptManifest[T]) = m

def implicitly[T](**implicit** e: T) = e // for summoning implicit values from the nether

@inline **def** locally[T](x: T): T = x

Predefined Implicit Definitions