Syntaxe de Base

Jean-Luc Falcone

HEPIA-2013

Variables

Java

```java
int i = 3;
String s = "Hello !";
```

Variables

Java

```
int i = 3;
String s = "Hello !";
```

Scala

```
var i: Int = 3;
var s: String = "Hello !";
```

Variables

Java

```
int i = 3;
String s = "Hello !";
```

Scala

```
var i: Int = 3;
var s: String = "Hello !";
```

Grâce à l'inférence des types et des points virgules:

```
var i = 3
var s = "Hello !"
```

Valeurs

Java

```java
final int i = 3;
final String s = "Hello !";
```

Valeurs

Java

```
final int i = 3;
final String s = "Hello !";
```

Scala

```
val i = 3
val s = "Hello !"
```

Blocs

Scala

```scala
val a = 1
val b = 2
val c = {
  println( "Computing" )
  val d = 10 - a
  d*d + b
}
```

Méthodes

Java

```java
double pythagore( double a, double b ) {
  final double c2 = a*a + b*b;
  return Math.sqrt(c2);
}
```

Méthodes

Java

```java
double pythagore( double a, double b ) {
  final double c2 = a*a + b*b;
  return Math.sqrt(c2);
}
```

Scala

```scala
def pythagore( a: Double, b: Double ): Double = {
  val c2 = a*a + b*b
  return math.sqrt(c2)
}
```

Méthodes

Java

```java
double pythagore( double a, double b ) {
  final double c2 = a*a + b*b;
  return Math.sqrt(c2);
}
```

Scala

```scala
def pythagore( a: Double, b: Double ): Double = {
  val c2 = a*a + b*b
  return math.sqrt(c2)
}

def pythagore( a: Double, b: Double ) = {
  val c2 = a*a + b*b
  math.sqrt(c2)
}
```

Remarques

- Inférence, pratique sauf:
  - recursion
  - return
  - *méthodes publiques*
  - *méthodes non-triviales*
- Eviter le return, plus lents, effets de bords

Scala

```scala
def add( i: Int, j: Int ) = i+j
```

Invocation de méthode

```
val sum = add( 2, 51 )
val hypothenus = pythagore( 4, 1.0 )
```

Multi-parameter lists

```scala
def add( i: Int )( j: Int ) = i+j

add(2)(5)
```

Méthodes Génériques

```scala
def isSizeEven[A]( as: List[A] ) =
  as.size % 2 == 0
```

Procédures (java)

```java
void saveAndQuit( State state ) {
  System.out.println( "Saving state...");
  save( state );
  System.out.println( "Bye" );
  quit();
}
```

Procédures (scala)

```scala
def saveAndQuit( state: State ): Unit = {
  println( "Saving state...")
  save( state )
  println( "Bye" )
  quit()
}
```

Procédures (scala)

```scala
def saveAndQuit( state: State ) {
  println( "Saving state...")
  save( state )
  println( "Bye" )
  quit()
}
```

Paramètres par défaut (scala)

```scala
def brew( coffee: String = "Expresso" ) {
  println( "Brewing: " + coffee  )
}

brew()

brew( "Capuccino" )
```

Paramètres nommés (scala)

```scala
def fullname( surname: String, firstname: String ) =
  firstname + " " + surname

fullname( "Knuth", "Donald" )

fullname( firstname="Donald", surname="Knuth" )
```

Paramètres nommés et par défaut (scala)

```scala
def readCSV(
  fileName: String,
  columnSeparator: String = ",",
  lineSeparator: String = "\n"
): List[List[String]] = ???

val rows1 = readCSV( "bar.csv" )
val rows2 = readCSV(
  fileName="foo.csv",
  columnSeparator=";"
)
```

While (java)

```java
double average( int[] ary ) {
  final int n = ary.length;
  int i = 0;
  int sum = 0;
  while( i < n ) {
    sum += ary[i];
    i++;
  }
  return 1.0*sum / n;
}
```

While (scala)

```scala
def average( ary: Array[Int] ): Double = {
  val n = ary.size
  var i = 0
  var sum = 0
  while( i < n ) {
    sum += ary(i)
    i += 1
  }
  sum.toDouble / n
}

def average2( ary: Array[Int] ) =
  ary.sum.toDouble / ary.size
```

If-Else (java)

```java
double x = f(1.0);
int a = 0;
if( x < 2.5 ) {
   a = 1;
} else {
   a = 2;
}
```

If-Else (scala)

```scala
val x = f(1.0)
val a = if( x < 2.5 ) 1 else 2

if( x < 0 ) {
  println( "x is negative..." )
}

val b = if( a == x ) {
  println("Idem")
  a
} else {
  val y = -x
  a * y
}
```

Switch/case (java)

```java
int day = getDay();
String dayString;
switch (day) {
    case 1:  dayString = "Lundi";  break;
    case 2:  dayString = "Mardi";  break;
    case 3:  dayString = "Mercredi";  break;
    case 4:  dayString = "Jeudi";  break;
    case 5:  dayString = "Vendredi";  break;
    case 6:  dayString = "Samedi";  break;
    case 7:  dayString = "Dimanche";  break;
}
System.out.println(dayString);
```

Match/Case (scala)

```scala
val day = getDay()
val dayString = day match {
    case 1 => "Lundi"
    case 2 => "Mardi"
    case 3 => "Mercredi"
    case 4 => "Jeudi"
    case 5 => "Vendredi"
    case 6 => "Samedi"
    case 7 => "Dimanche"
}
println( dayString )
```

Match/Case (scala)

```scala
val number: Int = f()
val kind = number match {
  case 0          => "Zero"
  case 1|2|3      => "Small"
  case i if i > 0 => "Positive: " + i
  case _          => "Negative"
}
```

Match/Case (scala)

```scala
def asInt( a: Any ) = a match {
  case s: String  => s.toInt
  case i: Int     => i
  case b: Boolean => if(b) 1 else 0
  case _ =>
        throw new Exception("Conversion error")
}
```

Try-catch (java)

```java
try {
  //...
} catch (FileNotFoundException e) {
    System.err.println(
        "FileNotFoundException: " + e.getMessage()
    );
    throw new SampleException(e);

} catch (IOException e) {
    System.err.println(
        "Caught IOException: " + e.getMessage()
    );
}
```

Try-catch (scala)

```scala
try {
  //...
} catch {
  case e: FileNotFoundException => {
    Console.err.println(
        "FileNotFoundException: " + e.getMessage()
    )
    throw new SampleException(e)
  }
  case e: IOException => Console.err.println(
    "Caught IOException: " + e.getMessage()
  )
}
```

Try-catch (scala)

```scala
def probability( s: String ) = try {
  val x = s.toDouble
  if( x <= 1 && x >= 0 ) x else -1.0
} catch {
  case _: NumberFormatException => -1.0
}
```

Imports

Imports (java)

```java
import java.util.Random;

import java.io.*;

import static java.lang.Math.PI;
```

Imports (scala)

```scala
import java.util.Random

import java.io._


import java.lang.Math.PI
```

Imports (scala)

```scala
import java.util.{List,ArrayList,Random}

import java.util.{List=>JList}

def write( msg: String ): Unit = {
  import java.io._
  val pw = new PrintWriter("output.txt")
  pw.println(msg)
  pw.close
}
```

Méthode `main` (java)

```java
public class Hello {

  private static void hello( String name ) {
    System.out.println( "Hello " + name );
  }

  public static void main( String[] args ) {
    hello( args[0] );
  }

}
```

Méthode `main` (scala)

```scala
object Hello {

  private def hello( name: String ): Unit = {
    println( "Hello " + name )
  }

  def main( args: Array[String] ) {
    hello( args(0) )
  }

}
```

Méthode `main` (scala)

```scala
object Hello extends App {

  private def hello( name: String ): Unit = {
    println( "Hello " + name )
  }

  hello( args(0) )

}
```