Programming Languages Recitation

Scala : Installation and Basics

Arpit Jain

Computer Science Department
Courant Institute of Mathematical Sciences
New York University

arpit.jain@cs.nyu.edu

November 13, 2014

Overview

Scala Introduction

- Concise and expressive

Scala Introduction

- Concise and expressive
- Functional (First class values)

Scala Introduction

- Concise and expressive
- Functional (First class values)
- Closures

Scala Introduction

- Concise and expressive
- Functional (First class values)
- Closures
- Type inference

Scala Introduction

- Concise and expressive
- Functional (First class values)
- Closures
- Type inference
- Strongly typed

Scala Introduction

- Concise and expressive
- Functional (First class values)
- Closures
- Type inference
- Strongly typed
- Static typing

Scala Introduction

- Concise and expressive
- Functional (First class values)
- Closures
- Type inference
- Strongly typed
- Static typing
- Object-functional Programming Language

Scala Introduction

- Concise and expressive
- Functional (First class values)
- Closures
- Type inference
- Strongly typed
- Static typing
- Object-functional Programming Language
- Compiles to java bytecode

Scala Introduction

- Concise and expressive
- Functional (First class values)
- Closures
- Type inference
- Strongly typed
- Static typing
- Object-functional Programming Language
- Compiles to java bytecode
- Works with any standard JVM

Scala Introduction

Companies using Scala ?

Scala Introduction

Companies using Scala ?

- Foursquare

Scala Introduction

Companies using Scala ?

- Foursquare
- Gilt

Scala Introduction

Companies using Scala ?

- Foursquare
- Gilt
- Coursera

Scala Introduction

Companies using Scala ?

- Foursquare
- Gilt
- Coursera
- **Twitter**

Scala

- For Linux sudo apt-get install scala
- For Mac download scala-2.11.x from here, move it to /usr/local/scala and add /usr/local/scala/bin to bash profile
- For Windows download scala-2.11.x from here, move to program files and configure path variables. (check scala resources on NYU classes for detailed procedure)

- Run scala in Terminal/Command Prompt and the scala interactive environment should be launched

Using IDE (Optional)

- Get Scala IDE for Eclipse
- Scala plugin for IntelliJ
- Or Scala plugin for Netbeans

Expressions

- 1+1 ⇒
  res0: Int = 2
- “new” + “york”; ⇒
  res1: String = newyork

Values

Gives the results of an expression a name

- *val* num=9 $\Rightarrow$
  num: *Int* = 9
- *val* two = 1 + 1 $\Rightarrow$
  two: *Int* = 2
- *val* str= "newyork" $\Rightarrow$
  str: *String* = newyork
- *va* bool=true $\Rightarrow$
  bool: *Boolean* = true

Variables

If binding needs to be changed use *var*

- *var* name = "steve" ⇒
  name: *String* = steve
- *var* name = "marius" ⇒
  name: *String* = marius
- *var* i=9 ⇒
  i: *Int* =9
- *var* i=0 ⇒
  i: *Int* =0
- But cannot do *var* i="hello"

Functions

- Function declaration : *def* name parameter = body
- Anonymous function : parameter ⇒ body

Functions

- *def* addOne(m: *Int*): *Int* = m + 1 ⇒
  addOne: (m: *Int*)*Int*
- *def* addOne(m: *Int*) = m + 1 ⇒
  addOne: (m: *Int*)*Int*
- Need to state m: *Int*, rest can be inferred.
- *val* three = addOne(2) ⇒
  three: *Int* = 3
- *def* three() = 1 + 2 ⇒
  three: ()*Int*

Currying and partial application

- def multiply(m: Int)(n: Int): Int = m * n ⇒
  multiply: (m: *Int*)(n: *Int*)*Int*
- multiply(2)(3) ⇒ res0: *Int* = 6
- *val* timesTwo = multiply(2)_ ⇒
  timesTwo: (*Int*) ⇒ *Int* = ⟨ function1⟩
- timesTwo(3) ⇒
  res1: *Int* = 6

Anonymous Functions

- (x: *Int*) $\Rightarrow$ x + 1 $\Rightarrow$
  (*Int*) $\Rightarrow$ *Int* = $\langle$function1$\rangle$
- val addOne = (x: *Int*) $\Rightarrow$ x + 1 $\Rightarrow$
  addOne: (*Int*) $\Rightarrow$ *Int* = $\langle$function1$\rangle$