Stream Programming Summer 2019

Lecture Introduction — 26.02, 2017

Dr inż. Wojciech Macyna *Scribe: Krzysztof Agieńczuk*

1 Acknowledgements

This lecture notes was made using notes provided by Adam Grygielski.

2 Overview

In this lecture we discussed contents of the lectures and scala programming language.

3 Contents

Class consists of lectures and laboratories. Grade is calculated using formula

$$Grade = 0.4 * Lecture + 0.6 * Lab$$

Topics of the lectures include:

1. Introduction to Scala (5 lectures)
2. Stream processing, stream architecture, Data Stream models
3. Sampling approaches to data streams
4. Frequency counter algorithms
5. Filtering streams
6. Stream databases
7. Application of stream processing

Literature As a suggested reading book "Programming in Scala" by M.Oderski, L.Spoon and B.Venners was suggested.

4 Scala

Some basic facts about Scala. The name comes as an abbreviation of "Scalable Language". It is a type-safe, multi paradigm language. It means it can be used as a functional language, as well as an object oriented one. Typical environments for Scala development are Eclipse or Intellij Idea.

4.1 Short tutorial

Below is shown short know-how for Scala. This is prepared using M. Odersky's book called "Programming in Scala". After installing Scala, the simplest way to start it is to type `scala` in command prompt. This will result in turning on Scala interpreter, as shown in above mentioned book[1]. Typing simple arithmetic equation `1+2` will result in `res0: Int = 3` being printed. This means result 0 (scala prompt enumerates results like eg. Jupter) of type Int is equal to 3. Same goes for other mathematical activities like subtraction etc.

Sample function signature looks like `def max(x: Int, y: Int): Int = {}`. This is function that takes two integer arguments, x and y, and returns integer argument. It is worth noting, that unlike in Java, in Scala, last line in function is always a return value. Keyword return doesn't have to be stated explicitly.

Scala allows also looping and conditional statements, but they are very similar to the way they work in Java, so they are not described here. For further reference I encourage you to take a look at Odersky's book, or at Scala documentation at Scala webpage.

References

[1] Odersky Martin, Spoon Lex, Venners Bill. Programming in Scala *artima*, PrePrint Ed:47–62, 2007-2008.