# ФУНКЦИОНАЛЬНОЕ ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Санкт-петербургский государственный политехнический университет
Институт Компьютерных Наук и Технологий
Кафедра: Информационные и Управляющий Системы
Автор: Лукашин Антон Андреевич

2016-2017

# Содержание

- Инфраструктура
  - *Установка и настройка*
  - *IDE*
- Основы синтаксиса языка Scala
  - *Scala REPL, Worksheet*
  - *типы данных*
  - *val, var, def*
  - *выражения*
  - *class, trait, case class, object*
- Evaluation Model

# Установка и настройка

- Полезные сайты
  - *http://www.scala-lang.org/ - официальный сайт*
  - *https://www.lightbend.com/ - мощный контрибьютер (+книги и видео)*
  - *https://github.com/odersky - аккаунт Мартина Одерского на гитхабе*
- Установка
  - *Скачать и установить бинарные файлы*
  - *IDE - может содержать встроенный scala-компилятор*
  - *Lightbend Activator – веб/консольное приложение для разработки*

# Установка и настройка

- Полезные сайты
  - *http://www.scala-lang.org/ - официальный сайт*
  - *https://www.lightbend.com/ - мощный контрибьютер (+книги и видео)*
  - *https://github.com/odersky - аккаунт Мартина Одерского на гитхабе*
- Установка
  - *Скачать и установить бинарные файлы*
  - *IDE - может содержать встроенный scala-компилятор*
  - *Lightbend Activator – веб/консольное приложение для разработки*

# Установка бинарных файлов

- Скачать бинарные файлы (http://www.scala-lang.org/download/)
- Установка пакета:
  - *MSI – для Windows*
  - *Deb - для debian-based (Debian, Ubuntu, Mint etc)*
  - *rpm – для rpm-based (red-hat, centOS, fedora etc)*
- Установка вручную
  - *Скачать и распаковать архив (tar.gz, zip)*
  - *Добавить поддиректорию bin в PATH*

# IDE

- Scala IDE (http://scala-ide.org/)
- Eclipse with plugin
- Intellij IDEA (https://www.jetbrains.com/idea/)

# Сборщики проектов

- SBT (http://www.scala-sbt.org/)
  - *Лаконичный синтаксис*
  - *Быстрый Scala компилятор*
  - *Можно собирать Java и Scala*
- Gradle with Scala plugin (https://gradle.org/)
- Maven with Scala plugin (https://maven.apache.org/)
- Ant with manual tasks (https://ant.apache.org/)

На самостоятельное изучение!

# Scala REPL & Worksheet

- REPL - read-eval-print loop (цикл чтение-исполнение-вывод)
  - *Сохраняет написанное выражение*
  - *Компилирует и исполняет его*
  - *Автоматически импортирует ранее написанные выражения*
- Вызов при помощи команды Scala
- Используется для проверки идей, написания учебных примеров, вычислений
- Scala Worksheet
  - *инструмент в IDE Eclipse*
  - *Расширение идеи REPL*
  - *Файл, который постоянно компилируется и исполняется*

# Типы данных в Scala

- Numeric (Byte, Short, Int, Long, Float,  Double)
- Boolean = true/false
- Char
- String
- Function
- Unit (no-value)
- Null – empty reference
- Nothing – subtype of any type (What about Unit?) – has no instance
- Any – supertype of any types
- AnyRef – supertype of any reference types

# val, var, def

- val <name>:<type> = <expression>
  - *Неизменяемое именованное значение*
- var <name>:<type> = <expression>
  - *Изменяемая именованная переменная*
- lazy val/var <name>:<type> = <expression>
  - *Инициализация значения произойдет в момент первого обращения*
- def <name>(<parameters>):<type> = <expression>
  - *Именованная функция*

# val, var, def

- val first:Int = 15
- var second = “Hello”; second= second+ “World”
- lazy val veryLazyToDoItNow = {heavyFunction() +15}
- def willCalculateItEveryCall = 15*15
- val complexVal = {val tmp=15; println(tmp+1)}
  - *Будет ли напечатано 16?*
  - *Какого типа будет значение complexVal?*
  - *Время жизни значения tmp?*
- Бесконечные циклы и dead locks

# Выражения

- Идентификатор (identifier)
- Значение (literal) (например, 5, “hello”, true, 7.5)
- Операторы (+, -, /, *, |, &, ^ etc) – является функцией 5.+(18)
- Вызов функции (sqrt(x))
- Условие if(<предикат>) <expression> else <expression>
  - *Всегда возвращает значение!*
  - *val testIF = if(5<3) 1*
  - *Каковы значение и тип переменной testIf?*
- Блок ({val x = math.abs(y) ; x * 2})
- Анонимные функции x => x + 1
- Pattern matching

# class

- Class – статическое описание набора полей и методов, которое может иметь множество сущностей
  - *Параметризован аргументами конструктора*

```
class myClass (val greeting:String, val name:String){
    def sayHello = println(greeting + name)
}
```

  - *Инстанциируется с помощью оператора new*

```
val hiWorld = new myClass(“Hello”, “World”)
hiWorld.sayHello
```

# object

- Scala object – singleton class
  - *Существует в единственном экземпляре*
  - *Не имеет параметров конструктора*
  - *Часто существует вместе с одноименным классом (объект X – компаньон класса X, companion object)*
  - *В языке Scala отсутствует модификатор static*
  - *Все что есть в object – является статичным*

# case class

- case class – scala class, который обладает следующими особенностями
  - *Может создавать через new или через apply*
  - *Автоматически переопределяется метод equals (структурный)*
  - *Используется для pattern matching*
  - *Используется в качестве DTO (POJO)*

# package object

- package object – scala object, который существует в рамках пакета
  - *Каждый пакет может иметь только один package object*
  - *Находящееся в package object доступно всем*

# trait

- trait - аналог interface
  - *Позволяет описывать функциональность классов*
  - *Может иметь реализованные методы*
  - *Может расширять class или другой trait*

```
trait Similarity {
  def isSimilar(x: Any): Boolean
  def isNotSimilar(x: Any): Boolean = !isSimilar(x)
}
```

# Особенности scala-синтаксиса

- ; не требуется
  - *Перенос строки считается концом выражения*
  - *Предыдущее неверно, если строка заканчивается на оператор*
  - *Несколько выражений в одной строке разделяются ;*
- return не требуется
  - *По умолчанию последнее выражение всегда считается возвращаемым*
- Указание типа не требуется
  - *Если компилятор по контексту может определить тип (компилятор определяет наиболее узкий из всех подходящих)*

# Evaluation model

- Выражение, не являющееся примитивным, исполняется следующим образом:
  - *Берется самый левый оператор(выражение)*
  - *Исполняются его операнды (левый раньше правого)*
  - *Применяется оператор(выражение) к операндам*
  - *Имена (значений, переменных, функций) – заменяются на присвоенное значение*
  - *Процесс останавливается как только получает значение*
- Исполнение функций
  - *Исполняются все аргументы функции*
  - *Заменяется имя функции на ее тело, и в то же время*
  - *Заменяются формальные параметры их значениями*

# Evaluation model

- sumOfSquares(3, 2+2)
- sumOfSquares(3, 4)
- square(3) + square(4)
- 3 * 3 + square(4)
- 9 + square(4)
- 9 + 4 * 4
- 9 + 16
- 25

# Evaluation Model

- call-by-value – параметр замещается значением
- call-by-name – параметр замещается телом

| **def something() = { println("calling something") 1 // return value }** | |
|---|---|
| def callByValue(x: Int) =<br>{ println("x1=" + x) println("x2=" + x) } | def callByName(x: => Int) =<br>{ println("x1=" + x) println("x2=" + x) } |
| callByValue(something()) | callByName(something()) |
| calling something<br>x1=1<br>x2=1 | calling something<br>x1=1<br>calling something<br>x2=1 |

# Спасибо за внимание!

- Планы на следующую лекцию:
  - *Коллекции (List, Map, Set, Range)*
  - *While loop*
  - *for loop*
  - *pattern matching*
- Самостоятельное изучение:
  - *build tools*
  - *Модификаторы видимости*
  - *import и implicit*