Other languages compiled to Java byte-code
Overview

- Scala
  - object and functional language
- Closure
  - functional language
    - a Lisp dialect
- Groovy
  - “skripting” language
- Kotlin
  - “new” Java
- Jython
  - Java implementation of Python
- JRuby
  - Java implementation of Ruby
Scala

- SCAlable LAnguage
- a mix of object and functional languages
- statically typed
- compiled to Java byte-code
  - also a version for .NET
    - now abandoned
- it is used
  - Twitter developed in Scala
    - since 2009, originally was in Ruby
  - LinkedIn
  - ...
Scala

- usage
  - compiler scalac
  - executing scala -classpath . Třída

- variables
  - var identifikator[:type] = hodnota

- constants
  - val identifikator[:type] = hodnota

- data types
  - classes Int, Double, Boolean,...
  - primitive Java types used when compiled
Scala

- **methods**
  - def max(x: Int, y: Int): Int = if (x > y) x else y
    - semicolon is not mandatory
    - no need for curly braces if the body has only a single command

- **arrays**
  - val anArray: Array[Type] = new Array[Type](size)
  - usage
    - anArray(0) = 5

- **lists**
  - val aList = List(1, 2, 3)
  - unmodifiable, all elements of the same type
  - Nil – empty list
  - methods for lists
    - head, tail, filter, sort, count, reverse,...
Scala

• tuples
  − val player = ("Novak", 68)
  − can contain elements of different types
  − unmodifiable
  − accessing elements
    • player._1 player._2

• functions are “first class” elements
  − a function is object; can be assigned
  − anonymous functions
  − ex.
    • funkce(i) = {x => Math.pow(x,i)}
    • seznam.filter(x => x > 4)
    • seznam.exists(x => x == 4)
    • seznam.sort((x,y) => x < y)
Scala

• classes
  – `object` ExampleClass {
    def foo() { … }
  }

  – `object` ~ singleton
    • a single instance
    • everything inside behave like static
      – but the static keyword does not exist

  – `class` Complex(re:Double, im:Double) {
      ...
  }

• only a single inheritance
  • but there are traits
Scala

- trait
  - partially implemented class
  - a trait is mix-in a class
    - class A extends T
  - several traits can be used
    - class B extends A with T1 with T2 with ...

trait Comparable {
  def <(co: Any): Boolean
  def <=(co: Any): Boolean = (this < co) || (this == co)
  def >(co: Any): Boolean = !(this <= co)
  def >=(co: Any): Boolean = !(this < co)
}
Scala

• generic data types
  class Moje[T] {
    ...
  }
  val m1 = new Moje[Int]
  val m2 = new Moje[Double]

• generic methods
  – def foo[T](i: Int) ...

Scala

- flexible syntax
  - semicolon unnecessary
  - methods can be used as an infix operator
    - "%d apples".format(num)
    - "%d apples" format num
  - curly braces can be used instead of parentheses in method calls
    - breakable { ... if (...) break() ... }
  - placeholder in anonymous functions
    - list map { x => sqrt(x) }
    - list map { sqrt(_) }
    - list map sqrt
  - ...
- easy building of new languages
**Groovy**

- object oriented language, dynamic
- dynamically compiled to byte-code
- syntax similar to Java and Ruby
  - most of Java code is syntactically correct Groovy code
- since 2003
  - in 2009, the original Groovy author announced
    “I can honestly say if someone had shown me the Programming in Scala book by Martin Odersky, Lex Spoon & Bill Venners back in 2003 I'd probably have never created Groovy. “
• examples

  class Song{
    length
    name
  }

  class Book{
    name
    author
  }

  def doSomething(thing){
    println "going to do something with a thing
    named = " + thing.name
  }

  mySong = new Song(length:90, name:"Burning Down the House")

  anotherSomething = doSomething
Groovy

- **closures**
  ```groovy
class Dog{
    action

    train(){
      action.call()
    }
}

sit = { println "Sit, Sit! Sit! Good dog"}
down = { println "Down! DOWN!" }

myDog = new Dog(action:sit)
myDog.train()  // prints Sit, Sit! Sit! Good dog

mollie = new Dog(action:down)
mollie.train()  // prints Down! DOWN!
```
Groovy

• collections
  
  ```groovy
  aCollect = [5, 9, 2, 2, 4, 5, 6]
  println aCollect.join(' - ')
      // prints 5 - 9 - 2 - 2 - 4 - 5 - 6
  println aCollect.count(2)  // prints 2
  println aCollect.sort()
      // prints [2, 2, 4, 5, 5, 6, 9]
  ```

• maps
  
  ```groovy
  myMap = ["name" : "Groovy", "date" : new Date()]
  println myMap["date"]
  println myMap.date
  ```

• ranges
  
  ```groovy
  myRange = 29...32
  myInclusiveRange = 2..5
  println myRange.size()  // prints 3
  println myRange[0]     // prints 29
  println myRange.contains(32)  // prints false
  println myInclusiveRange.contains(5) // prints true
  ```
Groovy

- GroovyBeans
  - similar to JavaBeans

- transformations of AST (Abstract Syntax Tree)
  - via annotations

```java
@ToString
class Person {
    String firstName
    String lastName
}
```

- adds the toString() method
Clojure

- functional language
- a Lisp dialect

- examples

```clojure
(println "Hello, world!")

(javax.swing.JOptionPane/showMessageDialog nil
"Hello World")
```