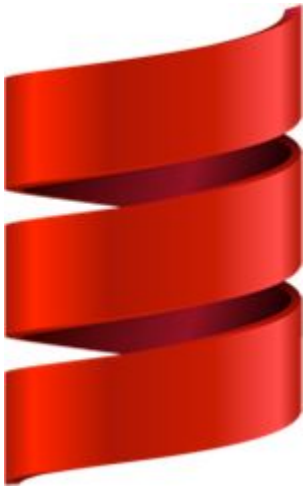

Scala

compared to SML/NJ

by djls45

Functional Object-Oriented (FOO) Programming

Outline

- Introduction
- Getting Started
  - Code Layout
  - Example Program
- Language Features
  - Types
  - Variables
  - Expressions
  - Control Structures
  - Subprograms
- Summary

Introduction

- History
  - Designed in 2001
  - Created in 2003
  - By Martin Odersky, a professor at Ecole Polytechnique Fédérale de Lausanne
  - He wanted to merge functional and object-oriented programming.
  - Came from Funnel, a language for describing functional nets
  - Compiles to Java bytecode

Code Layout

- Files follow standard Java package layout.
- Whitespace is not usually significant.
- Comments are the same as C/C++/Java.
- Semicolons are optional.
- Parentheses may be optional.
- All terms are case-sensitive.

Example Program

```
//package factorials
object Factorials {
   def fact(num: Int): Int = {
      if (num < 2) 1 else num * fact(num-1)
   }
   def main(args: Array[String]) {
      for( ln <- io.Source.stdin.getLines )
            println( fact(ln.toInt) )
   }
}
```

```
fun fact x = if x<2 then 1 else x*fact(x-1);
let val keepgoing:bool ref = ref true
in
      while !keepgoing do
            let val num = valOf(TextIO.inputLine
            TextIO.stdIn)
            in
                  print(Int.toString(fact(
                  valOf(Int.fromString(num)))));
                  keepgoing := not( null(
                  explode(num)))
            end
end;
```

Types

Scala

- Strongly typed
- No primitives
- Type inference
- Explicit conversion
- No reference types
- Everything is an object.

SML

- Strongly typed
- Basic primitives
- Type inference
- Explicit conversion
- Special ref types
- Objects and functions

Variables

Scala

- Static typing
- val/var
- All variables must be given a value at declaration.
- No null.
- Function parameters
  - Call by value (eager)
  - Call by name (lazy)

SML

- Static typing
- val
- All variables must be defined at declaration.
- [] is generic null.
- Function parameters
  - Call by value (eager)

Expressions

Scala

- Standard order of operations
  - 2-arity operators are function calls with optional infix notation.
- No implicit conversions
- Newlines or semicolons can end expressions.
- Parentheses are usually optional, but are recommended.

SML

- Standard order of operations
  - Operators are function calls.
- No implicit conversions
- Newlines end expressions.
- Parentheses are required for setting order of evaluation and for tuples.

Control Structures

Scala

- Lots of loop constructs
  - while
  - for
    - comprehensions
    - filters
    - yield
- Exceptions
  - throw
  - try-catch
  - finally

SML

- Only simple while loop
- Exceptions
  - raise
  - handle

Subprograms

Scala

- Parameters
  - By value (eager evaluation)
  - By name (lazy evaluation)
- Static scope
- Persistency
  - class variables
  - derive from Unit
- Huge library

SML

- Parameters
  - By value (eager evaluation)
- Static scope
- No persistency
  - Can be emulated with monads
- Very small library

Summary

Scala

- Readability +
- Writability +
- Reliability +

- Academic
- Commercial
  - Twitter
  - Novell
  - Xerox
  - The Guardian
  - Sony
  - FourSquare
  - Siemens
  - Électricité de France

SML

- Readability =
- Writability =
- Reliability -

- Academic