Lecture 19: Scala, Variables, & Program Units

CSCI 131
Spring, 2011

Kim Bruce

Concrete Features

- Object (singleton) as well as class definitions
- Declarations as x: T, use Unit instead of void
- All defs start with key word
- No need for semicolons or "return"
- Local type inference allows omitting many types, including with generics.
- For comprehensions rather than traditional for

Running Scala

- If eclipse doesn't work for you (or you don't want to use it), add scalac, scala to your path.
  - /usr/local/share/scala/bin on xserv
- Type
  - > scalac MyProg.scala
  - > scala -classpath . MyProg

More Examples

- Complex Numbers
  - Constructor via parameters to class
  - Can drop "." between object and method
  - a.+(b) equivalent to a + b
  - Static methods go inside companion objects
  - "implicit" methods called automatically when type mismatch
  - Must use "override" key word when override method

More Examples

- MyList, MyArrayList, SinglyLinkedList
  - Val vs var
  - Can create Array[T] (unlike Java), though need implicit ClassManifest
  - forEach(f) uses iterator
  - Option type is like Maybe in Haskell
    - Avoids null
    - Access via pattern matching

Getting Rid of Null References

- "The Billion Dollar Mistake" – Tony Hoare
  - Hoare put them in Algol W because they were easy
- Option type is like Maybe in Haskell
  - Avoids null
  - Access via pattern matching

Higher-Order Functions

- Anonymous functions:
  - def twice(f: Int => Int) = (x:Int) => f(f(x))
- Easy to make expressions into functions
  - twice(_+1)(45) => 47
- Used in libraries extensively
  - List(2,4,6).map(_/2)
  - also filter, ...

Variables

Attributes of Variable

- Scope Done!
- Lifetime
- Location
- Value

Lifetime

- FORTRAN - all allocated statically - ∞
- Stack-based (C/C++/Java/Pascal/...)
  - local vbles/ parameters: method/procedure/block entry to exit
  - allocate space in activation record on run-time stack
- Heap allocated variable
  - lifetime independent of scope
- Static - global vbles or static vbles

Value & Location

- Sometimes referred to as l-value & r-value
  - x = x + 1 *What does each occurrence of x stand for?*
  - location normally bound when declaration processed
- Normally values change dynamically
  - if frozen at compilation then called constants
  - Java final variables frozen when declaration processed.
  - Java static final bound at compile time.

Aliases

- x and y are aliases if both refer to same location.
- If x, y are aliases then changes to x affect value of y.
- Java has uniform model where assignment is by "sharing", so create aliases.
- Language that mix are more confusing.
  - Common mistakes occur when not realize aliases. E.g, add elt to priority queue and then change it ...