CS 3710 Vocational Languages

November 5, 2012

Scala

- “Scalable Language”, a language developed by Martin Odersky’s team at EPFL in Switzerland.
  - Executes on the Java platform
  - Integrates with Java
- Multiparadigm language
  - Object-oriented (with generics and mixin class composition)
  - Statically typed with Hindley-Milner type inference (parametric polymorphism)
  - Functional (similar to Haskell and SML)
  - Pattern Matching and Parsing
  - Actor-based concurrency-oriented programming

Define Hello World

```scala
object HelloWorld {
  def main(args: Array[String]) {
    println("Hey world!")
  }
}
```

- Call object constructor to create a single class instance named HelloWorld
- Define method main
- Specify parameter args of type Array[String]
- Array is generic class with type parameter

Running in Interpreter Environment

```
scah victory //the software is install in victory
scala
Welcome to Scala version 2.9.2-unknown-unknown (Java HotSpot(TM) 64-Bit Server VM, Java 1.6.0_33).
Type in expressions to have them evaluated.
Type :help for more information.
scah>
scala> object HelloWorld {
  | def main(args: Array[String]) {
  |   println("Hey world!")
  | }  

defined module HelloWorld
scala> HelloWorld.main(null)
Hey world!
unnamed0: Unit = ()
```

Compile and Execute Hello World

- `ssh victory`
- `scalac HelloWorld.scala`
  
  //scalac compiles the program into bytecode
- `ls` 
  
  HelloWorld$.class HelloWorld.class 
  
  Hello$.scala 
- `scala HelloWorld`
  
  //scala executes the bytecode in the Java virtual machine.
  
  Hey world!

var and val

- `var` declares variable while `val` declares constant value
- It is an error to declare a `val` or `var` without initialization.
- Scala infers the `val` and `var` values types based on the initial values.

```
scala> val area=3*4
area: Int = 12

scala> area=30
<console>:9: error: reassignment to val
area=30
  ^

scala> var area=3*4
area: Int = 12

scala> area=30
area: Int = 30
```

Numerical Type Classes

- Byte, Char, Short, Int, Long, Float, Double and Boolean.
- Invoke methods on numbers:
  
  `-1.toString()` // “1”
- Converting numerical types through method calls, not casting.

Invoke Built-In Class Methods

- `a method b;` is a short-hand of `a.method(b);`
- `1+10;` is the same as `1.+(10);`
- "Hello" intersect "World"; is the same as "Hello.interset ("World");
- `++` and `—` are replaces with `+=1` and `-=1`
  
  `-cont+=1`
Expressions

- If-expression has a return value `exp1` or `exp2` depending on the evaluation of the condition:
  
  ```scala
  if (condition) exp1 else exp2
  ```
  
  scala
  ```scala
  scala> if (area > 0) 1 else -1
  Res3: Int = 1
  scala> if (area == 0) 1
  res4: AnyVal = ()
  ```

- There is no need of ";" at the end of each expression.

Block Expressions

- One or more expression can be grouped into a block using {} with ";" to separate the expressions.
- The last expression value if the value of the block.

```scala
var distance = {
  val dx = x - x0;
  val dy = y - y0;
  sqrt(dx * dx + dy * dy)
}
```

Loops

```scala
while(n > 0) {...; n -= 1}
for(i <- 1 to n){...}
// i traverses from 1 to n
for(i <- 0 until n){...}
// i traverses from 0 to n-1
for(i <- 1 to 3; j <- 1 to 3)
  println (i + "," + j)
//?
```

Loops - Continue

```scala
for(i <- 1 to 10) yield i % 3
//?
```

- Yield produces a vector.
- If there is only 1 value, the result is a vector of 1 value.
def fac(n:Int=0):Int = if(n <=0)
  1 else n *fac(n-1)
fac() //?
fac(3) //?
def sum(args: Int*:Int = if(args.length==0) 0 else
  args.head+sum(args.tail:_*)
sum(1,2,3,4)//?