

DATA WAREHOUSE & MINING

TA COURSE #04

SCALA

TA COURSE #04

SCALA

**WHAT IS
SCALA**

WHAT IS SCALA

- ▶ Functional language
- ▶ Typed and Obj-Oriented
- ▶ Born from Java, be beyond Java

TA COURSE #04

SCALA

KUSOMISO TECHNIQUE

Mainly copied from [here](#)

BASICS

- ▶ val & var

- ▶ (lambda) functions

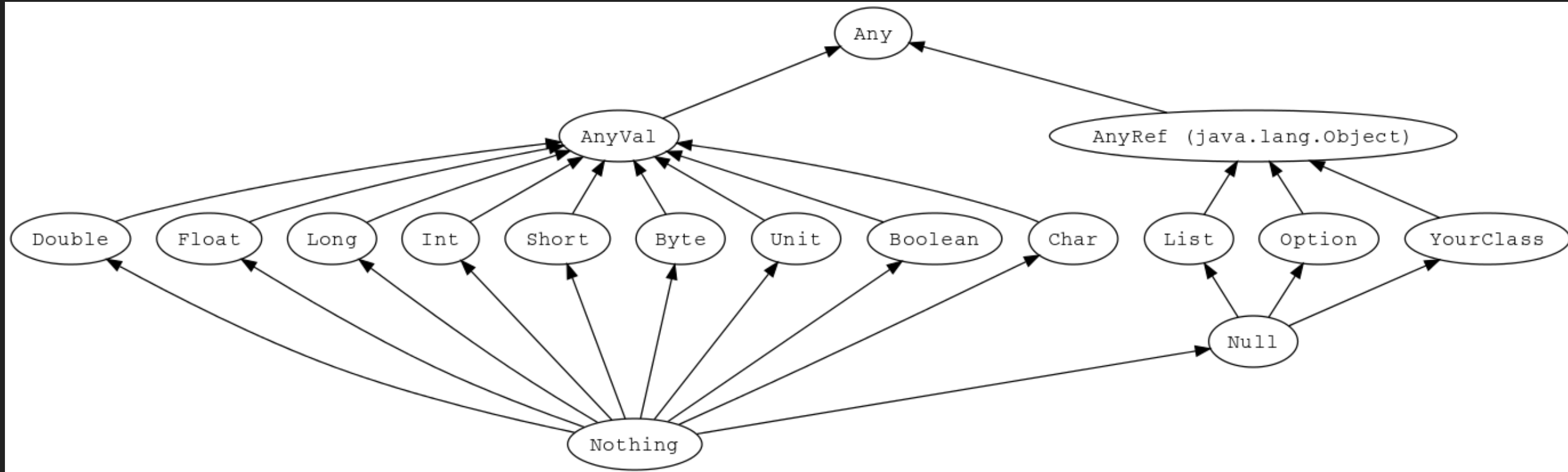
```
1 (x: Int) => x + 1
2 val addOne = (x: Int) => x + 1
3 val add = (x: Int, y: Int) => x + y
4 val getTheAnswer = () => 42
```

- ▶ (customized parameter list) methods

```
6 def add(x: Int, y: Int): Int = x + y
7 def addThenMultiply(x: Int, y: Int)(multiplier: Int): Int = {
8     (x + y) * multiplier
9 }
10 def name: String = System.getProperty("name")
```

- ▶ Entry called "main"

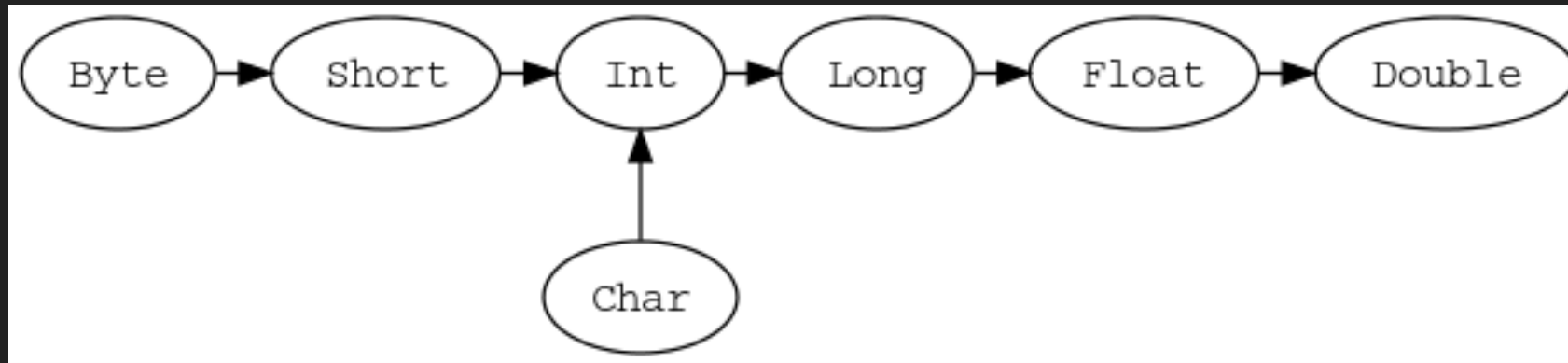
UNIFIED TYPES



```
12 val list: List[Any] = List(  
13     "a string",  
14     732, // an integer  
15     'c', // a character  
16     true, // a boolean value  
17     () => "an anonymous function returning a string"  
18 )  
19 list.foreach(element => println(element))
```

1. a string
2. 732
3. c
4. true
5. <function>

TYPE CASTING



TRAIT

- ▶ Class method template
- ▶ To implement to class before to use
- ▶ Different from abstract class since it can't be instantiated

```
21 trait Iterator[A] {  
22     def hasNext: Boolean  
23     def next(): A  
24 }  
25  
26 class IntIterator(to: Int) extends Iterator[Int] {  
27     private var current = 0  
28     override def hasNext: Boolean = current < to  
29     override def next(): Int = {  
30         if (hasNext) {  
31             val t = current  
32             current += 1  
33             t  
34         } else 0  
35     }  
36 }  
37  
38 val iterator = new IntIterator(10)  
39 iterator.next() // prints 0  
40 iterator.next() // prints 1
```


CLASS MIXIN

- ▶ Mix multiple classes and traits into one

```
42 abstract class A {  
43     val message: String  
44 }  
45  
46 class B extends A {  
47     val message = "I'm an instance of class B"  
48 }  
49  
50 trait C extends A {  
51     def loudMessage = message.toUpperCase()  
52 }  
53  
54 class D extends B with C  
55 val d = new D  
56 d.message // I'm an instance of class B  
57 d.loudMessage // I'M AN INSTANCE OF CLASS B
```

HIGHER-ORDER FUNCTION

► Function as parameter

```
59 class Decorator(left: String, right: String) {  
60   def layout[A](x: A) = left + x.toString() + right  
61 }  
62  
63 object FunTest extends App {  
64   def apply(f: Int => String, v: Int) = f(v)  
65   val decorator = new Decorator("[", "]")  
66   println(apply(decorator.layout, 7))  
67 }
```

PATTERN MATCHING

► Value matching

```
69 def matchTest(x: Int): String = x match {  
70   case 1 => "one"  
71   case 2 => "two"  
72   case _ => "many"  
73 }  
74  
75 matchTest(3) // many  
76 matchTest(1) // one
```

► Type matching

```
78 abstract class Device  
79 case class Phone(model: String) extends Device {  
80   def screenOff = "Turning screen off"  
81 }  
82  
83 case class Computer(model: String) extends Device {  
84   def screenSaverOn = "Turning screen saver on..."  
85 }  
86  
87 def goIdle(device: Device) = device match {  
88   case p: Phone => p.screenOff  
89   case c: Computer => c.screenSaverOn  
90 }
```

SINGLETON OBJECTS

► Single implementation classes

```
92 object Blah {  
93   def sum(l: List[Int]): Int = l.sum  
94 }
```

► Companions

```
96 class IntPair(val x: Int, val y: Int)  
97 object IntPair {  
98   import math.Ordering  
99   implicit def ipord: Ordering[IntPair] =  
100     Ordering.by(ip => (ip.x, ip.y))  
101 }
```

TYPE BOUNDS

- ▶ Upper type bound
 - ▶ $B <: A \iff B$ is a subtype of A
- ▶ Lower type bound
 - ▶ $B >: A \iff B$ is a supertype of A

TA COURSE #04

SCALA

PUMP IT!

sparkgraphxsrcmainscalaorgapachesparkgraphxlibTriangleCount.scala

Project

github

idea

assembly [spark-assembly_2.12]

bin

build

common

conf

core [spark-core_2.12]

data

dev

docs

examples [spark-examples_2.12]

external

graph

graphx [spark-graphx_2.12]

src

main

java

scala

org

apache

spark

graphx

impl

lib

package

ConnectedComponents

LabelPropagation

package-info.java

PageRank

ShortestPaths

StronglyConnectedComponents

SVDPlusPlus

TriangleCount

util

package

Edge

EdgeContext

EdgeDirection

EdgeRDD

EdgeTriplet

Graph

GraphLoader

GraphOps

GraphXUtils

package-info.java

PartitionStrategy

Pregel

VertexRDD

test

resources

scala

org

apache

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

set.add(nbrs(i))

}

i += 1

}

set

}

// join the sets with the graph

val setGraph: Graph[VertexSet, ED] = graph.outerJoinVertices(nbrSets) {

(vid, _, optSet) => optSet.getOrElse(null)

}

// Edge function computes intersection of smaller vertex with larger vertex

def edgeFunc(ctx: EdgeContext[VertexSet, ED, Int]) {

val (smallSet, largeSet) = if (ctx.srcAttr.size < ctx.dstAttr.size) {

(ctx.srcAttr, ctx.dstAttr)

} else {

(ctx.dstAttr, ctx.srcAttr)

}

val iter = smallSet.iterator

var counter: Int = 0

while (iter.hasNext) {

val vid = iter.next()

if (vid != ctx.srcId && vid != ctx.dstId && largeSet.contains(vid)) {

counter += 1

}

}

ctx.sendToSrc(counter)

ctx.sendToDst(counter)

}

// compute the intersection along edges

val counters: VertexRDD[Int] = setGraph.aggregateMessages(edgeFunc, _ + _)

// Merge counters with the graph and divide by two since each triangle is counted twice

graph.outerJoinVertices(counters) { (_, _, optCounter: Option[Int]) =>

val dblCount = optCounter.getOrElse(0)

// This algorithm double counts each triangle so the final count should be even

require(dblCount % 2 == 0, "Triangle count resulted in an invalid number of triangles.")

dblCount / 2

}

}

}

TriangleCount

runPreCanonicalized(graph: Graph[VD, ED])

edgeFunc(...)

6: TODO

Tool Output

Terminal

9: Version Control

ANTLR Preview

sbt project detected: Import sbt project (20 minutes ago)

TA COURSE #04: SCALA

THX!