SCALA
Programming language

Lucas Castro (PED) MC346 2021.2
lucas.castro@ic.unicamp.br

PED CLASSES

01 RUST LANGUAGE
09/11

02 GOLANG
11/11

03 CLOJURE
16/11

04 SCALA
18/11

SCALA HIGHLIGHTS

JVM

HIGH LEVEL PROGRAMMING

✓ FUNCIONAL
✓ OR. OBJETOS

Código em bytecodes

Multi-paradigma

SCALA NA SUA MÁQUINA

- Scala 3 - Compilador - REPL
  - Cousier tool
    - Instalar o Consier
    - Exportar o cousier no seu path
    - $ cs install scala3-compiler
    - $ cs install scala3
- Scala3 - SBT (Scala Built Tool)
- Scala3 Binaries
- Scala Web

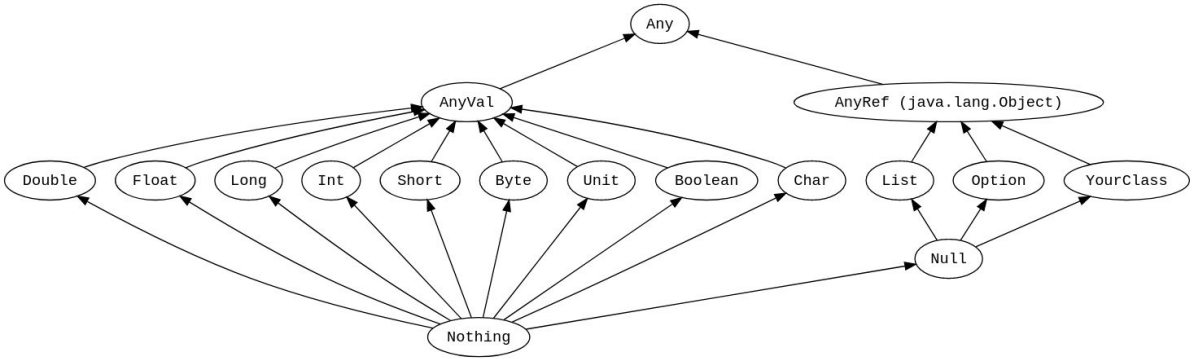
SCALA FEATURES

- Scala - **Sca**lable **La**nguage
- Martin Odersky
- Funcional e orientada a objetos
  - Híbrida
- Tipagem estática
- Inferência de tipos pelo compilador
- REPL
- Interoperabilidade com o Java
- Design patterns ofertados como objetos ou traits
  - Singleton, ...

SCALA DATA TYPE

- Any, Anyref, Anyval

- Variáveis e valores
  - **var** <name> : type = <value>
  - var a : int = 10
  - **val** <b> : type = <value>
  - val b : float = 10.2



- *Last expression*
  - val x = {var a:Int =100 ; var b:Int = 200; a+b}

Scala Data types

- Lazy variável
  - lazy val x = 500
  - ...x:Int = lazy

SCALA, HELLO WORLD!

- Object, def
- HelloWord.main()
- S,f,raw -> Interpolação de String
- Escopo das variáveis
    - public, private, protected

hello.scala

```scala
object HelloWorld {
    def main(args: Array[String]) = {
        val name:String = "mc346"
        val year:Int = 2021
        println("Hello " + name +" - Welcome to " + year + " Scala!")
        println(s"Hello $name  - Welcome to  $year  Scala!")
        println(f"Hello $name%s  - Welcome to  $year%d  Scala!")
        println(raw"Hello $name%s  - Welcome to  $year%d  Scala \n\n!")
        //comment line
    }
}
```

IF ELSE, FUNÇÕES, ANÔNIMAS

- Função sem retorno
  - *def printAlgumaCoisa(x:String, y:Int): **Unit** = {println(s"$x $y"}*
- Default values
  - *def functionName(a: Int =1, b:int) : Int = {}*

helloIf.scala

```scala
object HelloIf{
    def main(args: Array[String]) = {
        if(isGreater(args(1).toInt))
            println("Maior de idade!")
        else
            println("Menor de Idade - Faltam " + restantePara18(args(1).toInt) + " anos.")
        converteIdadeToDias(args(1).toInt)
    }

    def restantePara18(age:Int): Int = 18-age;

    def converteIdadeToDias(age:Int): Unit = {
        var newAge = (a:Int) => a * 365
        println("Sua idade em dias é " + newAge(age))
    }

    def isGreater(age:Int): Boolean =
        return (if (age >= 18) true else false);
}
```

LOOPS/ FILTER/ YIELD

```
object For{

    def main(args:Array[String]) = {
        var lista = List(10,20,30,50,100)

        for(i <- 1.to(10))
            println(i)

        for(i <- lista; if i > 20)
            println(i)

        var resultado = for {i <- lista; if i > 20} yield
            i * i

        println("Resultado: " + resultado)
    }
}
```

FUNÇÕES DE ALTA ORDEM

- Aninhamento de funções
  - Como fazer a soma de 3 números usando as funções de alta ordem?

highorder.scala

```scala
object HighOrder{

    def math(x: Double, y:Double, f: (Double, Double) => Double) : Double = f(x,y);

    def main(args: Array[String]) = {
        val r1 = math(50,10, (x,y) => x * y)
        val r2 = math(50,10, _*_)
        println(r1)
        println(r2)
    }
}
```

SCALA, COLLECTIONS

- Lista
  - var lista = List(1,2,3,4,5)
- Array, Arraybuffer

```
var array = Array(1,2,3,4,5)
array(1) = 10
for(i <- array)
    println(i)

var arrayInt = new Array[Int](20)

for(j <- 0 to (arrayInt.length -1))
    arrayInt(j) = j

val sortedArray = arrayInt.sortWith(_>_)
print("Sorted Array: ")
sortedArray.foreach(print)

var sortedArrayEven = for(i <- sortedArray if i % 2 == 0) yield i
println("\nSorted Array (Evens): ")
sortedArrayEven.foreach(println)

var arrayBuffer = ArrayBuffer[String]()

arrayBuffer.insert(0,"campinas")
arrayBuffer += "sp"
arrayBuffer ++= Array("rio","bh","cwb")

for(i <- arrayBuffer)
    println(i)
```

SCALA, COLLECTIONS

- Mapas e tuplas

```
var tupla = ("ic",2021,"unicamp")
println("\nTupla: " + tupla._2 + ", " + tupla._1 + "-" + tupla._3)
tupla.productIterator.foreach{ i => println(i)}
println(tupla.toString())

val mapa = Map("a" -> 8.5, "b" -> 7, "c" -> 6)

if(mapa.contains("a"))
    println(mapa("a"))

var mapaMutavel = collection.mutable.Map(9->"a", 7->"b", 6 -> "c")
mapaMutavel(5) = "d"

for((k,v) <- mapaMutavel)
    printf("%d : %s\n",k,v)
```

ORIENTAÇÃO OBJETOS

```
class Animal(var nome: String, var som: String){
    this.setNome(nome)
    var id = Animal.newIdNum


    def getNome(): String = nome
    def getSom(): String = som

    def setNome(nome:String)= {
        if(!(nome.matches(".*\\d+.*")))
            this.nome = nome
        else
            this.nome = "Sem nome"
    }

    def setSom(som:String) = {
        this.som = som
    }

    def this(nome:String) = {
        this("Sem nome","Sem som")
        this.setNome(nome)
    }

    def this() = {
        this("Sem nome","Sem som")
    }

    override def toString() : String = {
        return "%s (ID: %d) fala %s".format(this.nome,this.id,this.som)
    }
}

object Animal {
    private var idNum = 0
    private def newIdNum = {idNum += 1; idNum}
}
```

HERANÇA

```
    class Passaro(nome: String, som:String, tipoVoo:String) extends Animal(nome,som){

        def this(nome:String, som:String)= {
            this("Sem nome",som,"sem voo")
            this.setNome(nome)
        }

        def this (nome:String) = {
            this("Sem nome","sem som","sem voo")
            this.setNome(nome)
        }

        def this () = {
            this("Sem nome","sem som","sem voo")
        }

        override def toString(): String = {
            return "%s (ID: %d) fala %s - %s".format(this.nome,this.id,this.som,this.tipoVoo)
        }
    }
}
```

TRAITS

```
    def main(args: Array[String])= {

        val superman = new Heroi("Superman")
        println(superman.voar)
        println(superman.levarTiro)
        println(superman.defesa(2000))
    }

    trait Voar{
        def voar: String
    }

    trait ProvaBalas{
        def levarTiro : String

        def defesa(velocidade:Double) : String = {
            "A defesa foi realizada em %.1f ms".format(velocidade *.75)
        }
    }

    class Heroi(val nome:String) extends Voar with ProvaBalas{
        def voar = "%s voa pelos ares!".format(this.nome)
        def levarTiro = "O %s se defendeu do tiro".format(this.nome)
    }
}
```

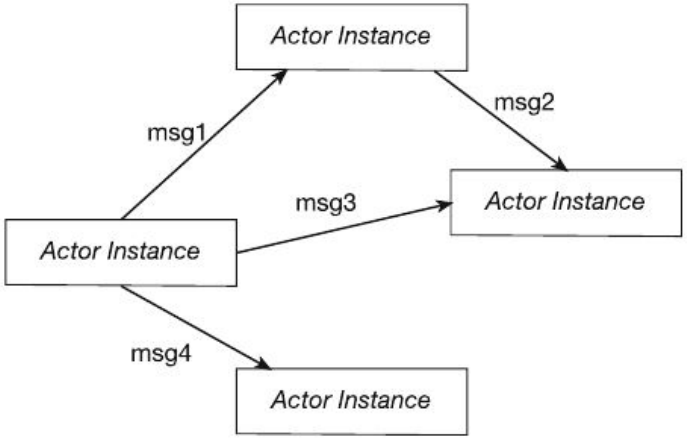
MAP, FILTER, FOLD, FLATTEN

```
object MapFilterExample{

    def main(args: Array[String]) = {

        var lista = (1 to 10).toList
        lista.map(x => x * 2).foreach(println)
        println("-----")
        lista.filter(x => x % 2 == 0).foreach(println)

        println("foldL " + lista.foldLeft(0)(_+_))
        println("foldR " + lista.foldRight(1)(_*_))

        var listaFlatten = (List(List(1,2,3),List(4,5,6)).flatten)
        println(listaFlatten)
    }
}
```

CONCORRÊNCIA

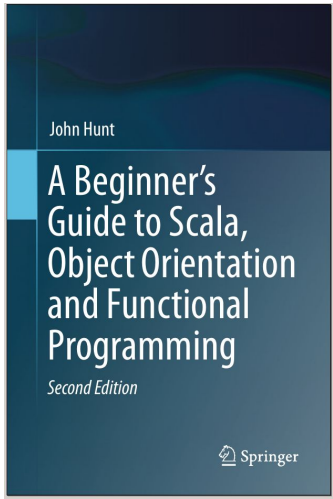
actor > actor.scala

```scala
import akka.actor.Actor
import akka.actor.ActorSystem
import akka.actor.Props

class HelloActor extends Actor {
    def receive = {
        case "hello" => println("olá de volta...")
        case _       => println("você disse algo?")
    }
}

object Main extends App {

    val system = ActorSystem("HelloSystem")
    val helloActor = system.actorOf(Props[HelloActor], name = "helloactor")

    helloActor ! "hello"
    helloActor ! "buenos dias"
    system.shutdown

}
```



APRENDER + SCALA

- Official Learning Resources
- Learn Y in X Minutes, where Y = Scala
- Scala exercises
- Exercism.org on Scala
- Vagas Scala Worldwide
  - +22k needed in US
  - Brasil - Muita vaga!
  - Salary



Springer Link

Scala Developer Salary

	Annual Salary	Weekly Pay
Top Earners	$174,000	$3,346
75th Percentile	$161,500	$3,105
Average	**$139,292**	$2,678
25th Percentile	$118,000	$2,269