Recap

Saturday, August 6, 2022 10:07 PM

1. Scala Variables
2. Control Structures
3. Scala Functions
4. Array
5. Array Buffer
6. List
7. List Buffer

Topics for Today

Sunday, July 3, 2022 7:52 PM

1. Tuples
2. Sets
3. Maps
4. Auxiliary Constructor
5. Single Ton Objects
6. Companion Objects
7. Case Class
8. Pattern Matching
9. Inheritance
10. Traits
11. Layered Traits
12. Higher Order Functions
13. Spark Components

Tuple

04 July 2020 14:06

Why Tuples?
Used to stores different data types

Declaration

val lst = List(1,true,"str")

lst.foreach(println)

lst(0) - First element in list

a:Any

Tuples

val a = (1,4,"Bob",true, 'a')

a: (Int, Int, String, Boolean, Char) = (1,4,Bob,true,a)

a.productIterator.foreach(println)

Access Elements of Tuples

a._1 - First Element of Tuple

Index starts with position 1

offsets
Offset starts with 1 and not from 0

Iteration
productIterator

a.productIterator.foreach(println)

Swap Elements

Sets

04 July 2020 14:08

Why Sets?

Only Unique values are stored in sets

Declaration

```
val s  = Set(1,2,3,4,4,5)
val t = Set(4,5,6,7,8)
```

Intersection

```
val u = s.intersect(t)
```

Union

Maps

04 July 2020 14:14

Why Maps?
Dictionary in Python

Collection of Key Values Pairs
Keys should be unique and Values can be same
val map1 = Map(1 -> "Finance", 2-> "Operations" , 2 -> "Accounts")

Option
AnyRef Type

1) Store the value Some(Value)
   1) It can be empty None

val mp1 = Map(1 -> "Operations" , 2 -> "Dispatch" , 3 -> "Finance")

Keys should always be unique
Keys should be stored nternally as sets

val mp1 = Map(1 -> "Operations" , 2 -> "Dispatch" , 3 -> "Finance" , 2 -> "Store")
[Joe(Male,36,USA), Tim(Male,40,UK), Jade (Female,25,Germany)]

val a = Map("Joe" -> ("Male",36,"USA"), "Tim" -> ("Male",40,"UK"), "Jade" -> ("Female",25,"Germany"))

Declaration

contains

getOrElse

Usage in Our Project
Location of file

("FileName" -> "Location of File")

https://www.scala-lang.org/api/2.10.7/#package

https://spark.apache.org/docs/2.3.0/api/scala/index.html#org.apache.spark.sql.functions$

Classes and Objects

04 July 2020 16:08

```
classlearnClass{
private var value=0//fieldsmustbeinitialized
def incr(){value+=1}
def curr()=value
}
```

Keyword

Class counter

```
classlearnClass{

privatevarvar1=10
//defincr(){value+=1}//Null
//defcurr()=value//Returningthevalue
defcustGetter()=var1//customisedGetter
defcustSetter(x:Int){var1=x}//customisedsetter

}

valobj1=newlearnClass

obj1.custGetter()//CallingtheCustomisedgetter
obj1.custSetter(13)//CallingtheCustomisedsetter
obj1.custGetter()//CallingtheCustomisedgetter

valobj2=newlearnClass
```

Example

Object is instance of the class

```
val ctr1 = new cntr

ctr1.incr           cntr1 = 1
ctr1.incr           cntr1 = 2
ctr1.incr           cntr1 = 3
ctr1.incr           cntr1 = 4
ctr1.curr           4

class birds {
var color = "green"
def changecolor (newColor : String)  { color =
newColor }
def findColor = { color}
}

val brd1 = new birds
brd1.findColor
brd1.changecolor("pink")
brd1.findColor
```

Getters and Setter

05 July 2020 11:14

Why Getter and Setters?

Used to expose class properties/variables

Getter Example

```
class learnGetter {
  val size = 1
}

val f = new learnGetter
val a =  f.size
println("Printing after geting value: " + f.size)
```

Getter and Setter Example

```
class learnGetterSetter {
  var size = 1
}

val f = new learnGetterSetter
val a =  f.size
println("Printing before setting value: " + f.size)
f.size
```

```
f.size_=(10)
println("Printing after setting value: " + f.size)
```

Another Getter and Setter Example

```
class learnGetterSetter2 {

  private var privateAge = 0
  def age = privateAge //getter
  def age_=(newAge: Int) { if (newAge > privateAge)
privateAge = newAge } //setter
}
val a = new learnGetterSetter2

a.privateAge
a.privateAge_
a.age
a.age_=(10)
a.age
```

Primary Constructor

05 July 2020 11:50

To construct our objects

note
Example

```
class learnPrimaryConstructor(firstname: String,
                              lastName: String,
                              middleName: String) {
  println(firstname +' '+ lastName +' '+ middleName)
  def first() { println(firstname) }
  def middle() { println(middleName) }
first()
middle()
 }

val p1 = new
learnPrimaryConstructor("Ram" ,"","Singh")
```

Auxiliary Constructor

05 July 2020 12:33

Used for Constructor Overloading

KeyWord This - Auxiliary Constructor
First line of Auxiliary - We must call Primary Constructor or Previously Defined Auxiliary Constructor this keyword

Example 1

```
class learnAuxiliaryConstructor(firstname: String,
                                lastName: String,
                                middleName: String) {

  /**def this - Define an Auxiliary Constructor
     * Each constructor must call one of the previously defined constructors
  */
  println("Complete Name is " + firstname + lastName + middleName)

def this(firstname: String) {
    this(firstname, "", "")
    println("First Name is " + firstname)
  }
}

val p1 = new learnAuxiliaryConstructor("Ram" ,"Singh","")
val p2 = new learnAuxiliaryConstructor("Ram")
```

Example 2

```
class learnMultipleAuxuliaryConstructor(firstName: String,
                                  lastName: String,
                                  middleName: String) {

  /**def this - Define an Auxiliary Constructor
    *Rule - First Line of Auxiliary Constructor ,you have to call primary
constructor
    * While calling primary constructor , you need to pass all the arguments
    */
  println("This is primary constructor")
  println("Complete Name is " + firstName + lastName + middleName)
//First Auxiliary Constructor
def this(firstname: String) {
    this(firstname, "", "")
    println("This is Auxiliary constructor with firstname")
    println("First Name is " + firstName)
  }

//Can this be allowed
 // def this(lastname: String) {
  //  this("", lastname, "")
//    println("This is Auxiliary constructor with lastname")
//    println("lastname  is " + lastname)
//  }
//Another Auxiliary Constructor
  def this(lastname: String,middlename: String) {
   // this("")
 this("",lastname,middlename)
    println("This is Auxiliary constructor with Lastname and MiddleName")
    println("Last Name is " + lastName)
    println("Middle Name is " + middleName)
  }
}
val p1 = new learnMultipleAuxuliaryConstructor("Ram","Sharma","Pawan")
val p2 = new learnMultipleAuxuliaryConstructor("Ram")
val p3 = new learnMultipleAuxuliaryConstructor("Ram","Sharma")
```

Singleton Objects

05 July 2020 12:56

Definition of SingleTon Object

An object that has got exactly one instance

Object SomeName

SingleTonObject Example

```
object learnSingletonObject {
  private var lastNum = 0
  def newReservation() =  {lastNum +=1 ; lastNum}

}
learnSingletonObject.newReservation()

learnSingletonObject.newReservation()

learnSingletonObject.newReservation()
learnSingletonObject.newReservation()

Class Reservation {

Def reservation () { }
Var x = 22

}
```

One instance of class reservation

```
Val a = new reservation
Val b = new reservation
Val c = new reservation
```

Companion Objects

05 July 2020 14:04

Definition

Same Name of Class and Object
Both Class and Object are defined in the same file

Companion Class
Companion Object

Usage of Companions

A companion object's apply method lets you create new instances of a class without using
the new keyword

List

```
val a = List(1,2,3,4,5)
val a = new List[Int](5)
```

```
class Person { var name = "" }
object Person {
    def apply(name: String): Person = {
        var p = new Person
        p.name = name
        p
        }
    }
```

Use Case : - Array

Case Class

05 July 2020 14:23

Defination

Used for comparison of objects
Instances of case classes are compared by structure and not by reference:

Example

case class learnCaseClass(isbn:String)

val frank = learnCaseClass("987-123")

How it is different from a class

class learnCaseClass1(isbn:String)

val frank = new learnCaseClass1("987-123")

Pattern Matching

05 July 2020 15:01

Used for matching values of Variables

Definition

Keyword
match
case

Example 1

```
def matchTest(x:Any) : Any = x match {
   case 1 => "one"
   case "two" => 2
   case y: Int => "scala.Int"
   case _ => "many"
      }

println(matchTest("two"))
println(matchTest("test"))
println(matchTest(1))
```

Example 2

```
case class Person(name: String, age: Int)
   val alice = Person("Alice", 25)
   val bob = Person("Bob", 32)
```

```
    val charlie = Person("Charlie", 32)
    for (person <- List(alice, bob, charlie)) {
      person match {
        case Person("Alice", 25) => println("Hi Alice!")
        case Person("Bob", 32) => println("Hi Bob!")
        case Person(name, age) => println("Age: " + age
+ "year, name: " + name + "?")
      }
    }
```

Inheritance

05 July 2020 15:48

What is Inheritance?

```
Class A
    Var 1
        Var 2
    Function 1
    Function 2
Class c
Var

Var
Func
func

Class B extends Class C
   var 3
   Function 3
```

Keyword ?

Example

```
class learnInheritance (speed:Int) {
  val mph: Int = speed
  def race() { println("Racing")}
println("This is vehicle")
}
```

Keyword Extends

```
class Car(speed:Int)  extends
learnInheritance(speed)
```

Requirement : to change some of the features not all

Keyword : Override - To modify existing property of method

```
class Car(speed:Int)  extends
learnInheritance(speed) {
  override val mph: Int = speed
  override def race() = println("Racing Car")

def carMethod () { println("I am in class Car")

}

val a = new Car(20)
println("Speed of Car: " + a.mph)
a.race()
```

Traits

05 July 2020 15:55

Definition
To Reuse Code

Example of Trait

```
trait learnTrait {

//abstract functions
//Name /Signature is given
//Implementation not given
  def hasNext:Boolean
  def next():Int
}

class IntIterator (to: Int) extends learnTrait {
  private var current = 0
  def hasNext:Boolean = current < to
 def next(): Int = {
    if (hasNext) {
      val t = current
      current += 1
      t
    } else 0 }

}
val iterator = new IntIterator(10)
iterator.next()
iterator.next()
```

layered Traits

05 July 2020 18:05

Definition
Multiple traits can be extended by a class or object

with - Keyword

super - keyword

Example of layered Traits

```
//trait 1
trait logger  {
//a simple method log
  def log (msg: String) {println(msg)}
}

//trait 2
trait TimestampLogger extends logger {
  override def log (msg: String) {
println("We are in Timestamp Logger")
    println(new java.util.Date() )
    super.log(new java.util.Date() + " " + msg)
  }
}

//trait 3
trait ShortLogger extends logger {
  val maxLength = 15
   override def log(msg: String) {
println("We are in Short Logger")
    super.log( if (msg.length <= maxLength) msg
    else msg.substring(0,maxLength -3) + "...")
  }
}
//Notice the keyword extends , with
class logging extends  TimestampLogger with ShortLogger
val a = new logging
a.log("My example")
```

Higher Order Functions

08 July 2020 22:18

Functions that take other functions as parameters or return a function as a result

Example

Problem Statement: Input 2 int

Output sum of cube of 2 integers

(1,2)

1+8 = 9

Functions That accepts Function

Higher Order Function

```
def sum(f:Int => Int,a:Int, b: Int) : Int =
   if (a > b) 0 else f(a) + sum(f,a+1,b)
```

```
def cube(x:Int): Int = x * x * x
```

```
def sumCubesHO(a:Int, b: Int) = sum(cube,a,b)
def square(x:Int) : Int = x *x
def semSquaresHO(a:Int,b:Int) = sum(square,a,b)
```

```
def fact(x: Int) : Int = if (x==0) 1 else x*fact(x-1)
def sumFactorialHO(a:Int, b: Int) = sum(fact,a,b)
def sumofnum(x :Int , y :Int, f:Int => Int) : Int = {if
( x > y) 0 else f(x) + sumofnum(x + 1,y,f)}
```

Anonymous Function

```
def sumCubesHOA(a:Int, b: Int) = sum(x=>
x*x*x,1,4)
```

Examples of Higher Oder Functions

Filter

```
(1 to 9).filter(_ % 2 == 0)
```

```
def EvenOdd (x:Int) = { x % 2 == 0}
(1 to 9).filter(EvenOdd)
```

ReduceLeft

```
(1 to 9).reduceLeft(_ * _)
```

```
def multiply(x :Int, y:Int) = { x * y}

(1 to 9).reduceLeft(multiply)
```

Functions That returns Functions

```
//Higher Order Function That returns Function

  def urlBuilder(ssl: Boolean, domainName:String) :
(String,String) => String = {
    val schema = if (ssl) "https://" else "http://"
    (endpoint:String, query:String) => s"$schema
$domainName/$endpoint?$query" }

  val domainName = "www.example.com"

  val endpoint = "users"
  val query = "id=1"
  def getURL = urlBuilder(ssl=true,domainName)
  val url = getURL(endpoint,query)
  println(url)
```

IntellijIdea Set up

05 September 2021 23:35

https://docs.scala-lang.org/getting-started/intellij-track/getting-started-with-scala-in-intellij.html#installation