2024-1

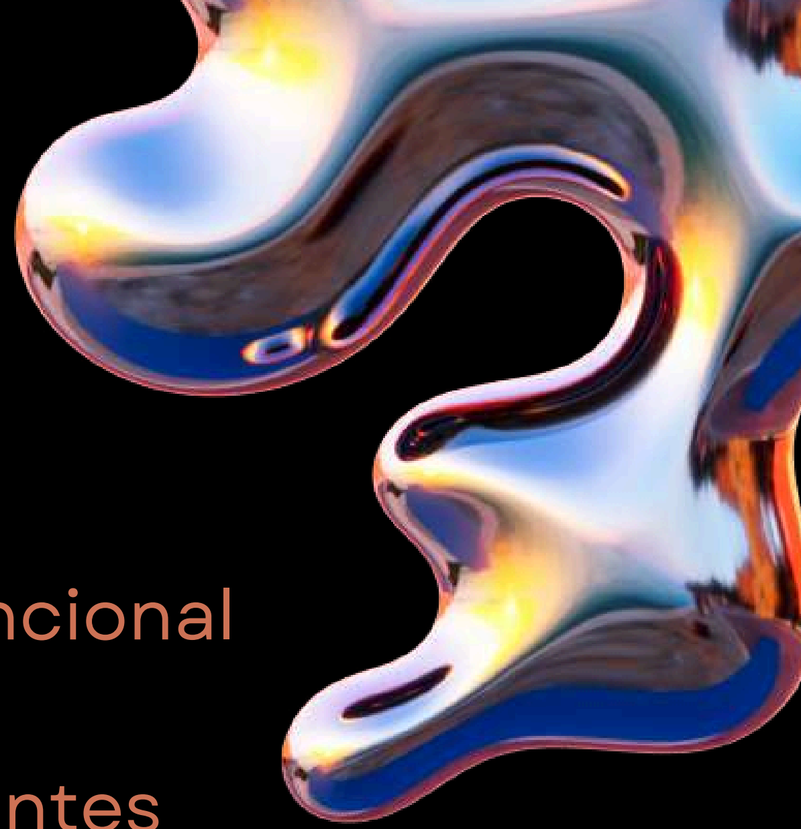
Scala 1

Teoría de Lenguajes de Programación

Sara Acevedo Maya
saacevedom@unal.edu.co
Universidad Nacional de Colombia

UNIVERSIDAD NACIONAL DE COLOMBIA

Contenido



01/ Programación funcional
02/ Tipado
03/ Variables/constantes
04/ Funciones anónimas

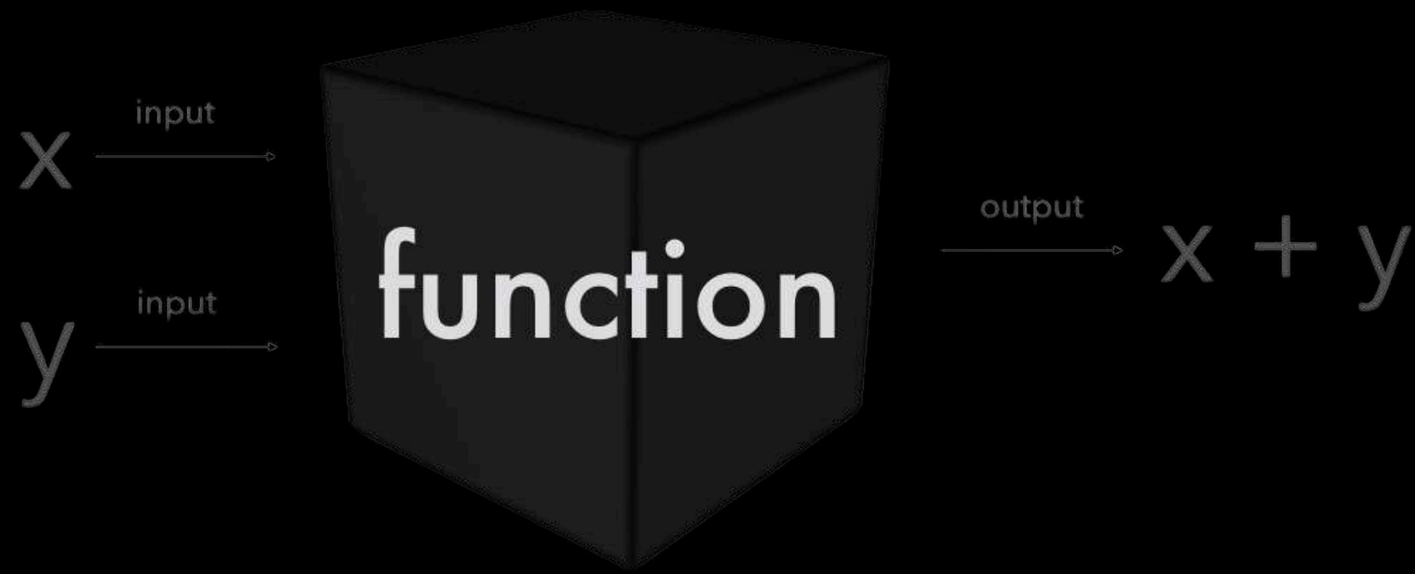
UNIVERSIDAD NACIONAL DE COLOMBIA

01 - Programación funcional

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Programación funcional

Es un paradigma de programación que trata la computación como la evaluación de funciones matemáticas y evita cambiar el estado y los datos mutables. Es una forma de pensar en la construcción de software donde las funciones son las unidades fundamentales de composición.



UNIVERSIDAD NACIONAL DE COLOMBIA

Características Principales:

- **Funciones puras:** son aquellas que siempre produce la misma salida para los mismos argumentos y no tiene efectos secundarios (no modifica variables externas, no realiza I/O, etc.).

Scala code - SAM<3

```scala
def suma(a: Int, b: Int): Int = a + b
```

- **Inmutabilidad:** En PF, los datos son inmutables, lo que significa que una vez creados, no pueden ser modificados. Cualquier cambio en los datos produce nuevos datos.

Scala code - SAM<3

```scala
val lista = List(1, 2, 3)
val nuevaLista = lista :+ 4  // lista sigue siendo List(1, 2, 3)
```

UNIVERSIDAD NACIONAL DE COLOMBIA

Características Principales:

- **Funcionesde Orden Superior y como Ciudadanos de Primera Clase:** Las funciones son tratadas como ciudadanos de primera clase, lo que significa que pueden ser pasadas como argumentos a otras funciones, devueltas como valores de otras funciones y asignadas a variables.

Scala code - SAM<3

```scala
val suma = (a: Int, b: Int) => a + b
val aplicarFuncion = (f: (Int, Int) => Int, x: Int, y: Int) => f(x, y)
aplicarFuncion(suma, 3, 4)  // Devuelve 7
```

UNIVERSIDAD NACIONAL DE COLOMBIA

Características Principales:

- **Composición de Funciones** es el proceso de combinar funciones pequeñas para construir funciones más complejas. Esto permite construir programas complejos a partir de funciones simples y reutilizables.

Scala code - SAM<3

```scala
val sumarUno = (x: Int) => x + 1
val duplicar = (x: Int) => x * 2
val sumarYDuplicar = sumarUno.andThen(duplicar)
sumarYDuplicar(3)  // Devuelve 8
```

UNIVERSIDAD NACIONAL DE COLOMBIA

Características Principales:

- **Recursión:** Dado que los bucles tradicionales (con estado mutable) no se usan comúnmente en PF, la recursión se convierte en una herramienta importante para iterar sobre datos.

Scala code - SAM<3

```scala
def factorial(n: Int): Int = {
  if (n == 0) 1
  else n * factorial(n - 1)
}
```

UNIVERSIDAD NACIONAL DE COLOMBIA

02 - Tipado

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Tipado

Características principales del tipado en Scala:

Inferencia de Tipos: Scala tiene un sistema de inferencia de tipos poderoso que puede deducir el tipo de una variable automáticamente si no se proporciona explícitamente. Por ejemplo:

```
val x = 10     // Scala infiere que x es de tipo Int
val y = "hello" // Scala infiere que y es de tipo String
```

Tipos explícitos: Aunque Scala puede inferir tipos, también permite especificar tipos explícitamente cuando sea necesario:

```
val z: Double = 3.1416
```

Tipos Genéricos: Scala soporta la definición de clases y funciones genéricas, lo que permite escribir código que funcione con tipos arbitrarios. Por ejemplo

```
class Box[T](val value: T) {
  def getValue: T = value
}
```

En este ejemplo, T es un tipo genérico que puede representar cualquier tipo de dato.

UNIVERSIDAD NACIONAL DE COLOMBIA

Tipado

Compatibilidad con Java: Scala se ejecuta en la JVM (Java Virtual Machine) y puede interoperar con código Java de manera fluida, permitiendo el uso de clases y librerías de Java directamente en Scala y viceversa.

Tipos de Datos Compuestos: Scala proporciona tipos de datos compuestos como Tuple, List, Option, entre otros, que permiten manejar colecciones de datos de manera segura y eficiente.

Tipos Primitivos

- Byte: Entero de 8 bits con signo (rango: -128 a 127)
- Short: Entero de 16 bits con signo (rango: -32,768 a 32,767)
- Int: Entero de 32 bits con signo (rango: -2,147,483,648 a 2,147,483,647)
- Long: Entero de 64 bits con signo (rango: -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807)
- Float: Número en punto flotante de 32 bits
- Double: Número en punto flotante de 64 bits
- Char: Caracter de 16 bits sin signo, que representa un solo carácter Unicode
- Boolean
- Unit: Similar a void en otros lenguajes, representa la ausencia de un valor útil. Tiene un único valor, ().
- String

UNIVERSIDAD NACIONAL DE COLOMBIA

Tipado

Tipos Básicos

- Any: La superclase de todos los tipos.
  - AnyVal: Subclase de Any que representa los tipos de valor.
    - Int: Entero de 32 bits.
    - Double: Número de punto flotante de 64 bits.
    - Float: Número de punto flotante de 32 bits.
    - Long: Entero de 64 bits.
    - Short: Entero de 16 bits.
    - Byte: Entero de 8 bits.
    - Char: Carácter de 16 bits.
    - Boolean: Valor booleano (true o false).
    - Unit: Tipo de valor vacío, que es similar a void en otros lenguajes. Su único valor es () (el literal de Unit).
  - AnyRef: Subclase de Any que representa los tipos de referencia.
    - String: Cadena de caracteres.
    - List, Option, Map, Set: Estructuras de datos inmutables.
    - Clases y objetos definidos por el usuario.

03-Variables/ constantes

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Val vs Var

- **val (valor):** Declara una variable inmutable, lo que significa que una vez que se le asigna un valor, no puede cambiar. Es similar a una constante en otros lenguajes.

- **var (variable):** Declara una variable mutable, lo que significa que su valor puede cambiar a lo largo de la vida del programa.

Scala code - SAM<3

```scala
// Usando `val`
val nombre = "Scala"
// nombre = "Java"  // Esto causaría un error de compilación porque `nombre` es inmutable

// Usando `var`
var edad = 25
edad = 26  // Esto es válido, `edad` es mutable y su valor puede cambiar
```

UNIVERSIDAD NACIONAL DE COLOMBIA

04-Funciones anónimas

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Funciones anónimas

es una función que no tiene un nombre explícito asociado a ella. Estas funciones se definen "al vuelo" y son frecuentemente utilizadas en programación funcional cuando se desea aplicar una operación específica de manera concisa, sin necesidad de crear una función nombrada.

Scala code - SAM<3

```
(valoresDeEntrada) => expresión
```

UNIVERSIDAD NACIONAL DE COLOMBIA

Funciones anónimas vs Funciones normales

- Funciones Normales son ideales para código reutilizable, modular y cuando la claridad es importante.
- Funciones Anónimas son útiles para casos simples, específicos y cuando se usan como argumentos en métodos de alto orden.

Scala code - SAM<3

```scala
// Función normal
def sumaNormal(a: Int, b: Int): Int = a + b

// Función anónima
val sumaAnonima = (a: Int, b: Int) => a + b

println(sumaNormal(3, 5))  // Imprime: 8
println(sumaAnonima(3, 5)) // Imprime: 8
```

UNIVERSIDAD NACIONAL DE COLOMBIA

Funciones anónimas en programación funcional

métodos funcionales comunes en colecciones como: map, filter, reduce, flatMap, y foreach.

map

Scala code - SAM<3

```scala
val numeros = List(1, 2, 3, 4, 5)
val multiplicados = numeros.map(x => x * 2)
println(multiplicados)  // Imprime: List(2, 4, 6, 8, 10)
```

UNIVERSIDAD NACIONAL DE COLOMBIA

filter

Scala code - SAM<3

```scala
val numeros = List(1, 2, 3, 4, 5)
val pares = numeros.filter(x => x % 2 == 0)
println(pares)  // Imprime: List(2, 4)
```

reduce

Scala code - SAM<3

```scala
val numeros = List(1, 2, 3, 4, 5)
val sumaTotal = numeros.reduce((a, b) => a + b)
println(sumaTotal)  // Imprime: 15
```

UNIVERSIDAD NACIONAL DE COLOMBIA

flatMap

Scala code - SAM<3

```scala
val palabras = List("Scala", "es", "genial")
val caracteres = palabras.flatMap(palabra => palabra.toList)
println(caracteres)  // Imprime: List(S, c, a, l, a, e, s, g, e, n, i, a, l)
```

foreach

Scala code - SAM<3

```scala
val numeros = List(1, 2, 3, 4, 5)
numeros.foreach(x => println(x))
```

UNIVERSIDAD NACIONAL DE COLOMBIA

Practica

- Escribe una función anónima en Scala que reciba una lista de enteros y devuelva una lista con los números incrementados en 10 y luego multiplicados por 2. Utiliza esta función para transformar la lista List(1, 2, 3, 4, 5).

- Escribir una clase que defina un método “procesarLista”, este método recibe una lista, y un proceso a realizarle a esta lista

UNIVERSIDAD NACIONAL DE COLOMBIA

Tarea

Grupo 3:

Plazo hasta el **Domingo 18 de Agosto a las 10am**:

https://forms.gle/K3f9MGVrTVD4pfkMA

UNIVERSIDAD NACIONAL DE COLOMBIA