Programación IV

Programación Funcional.

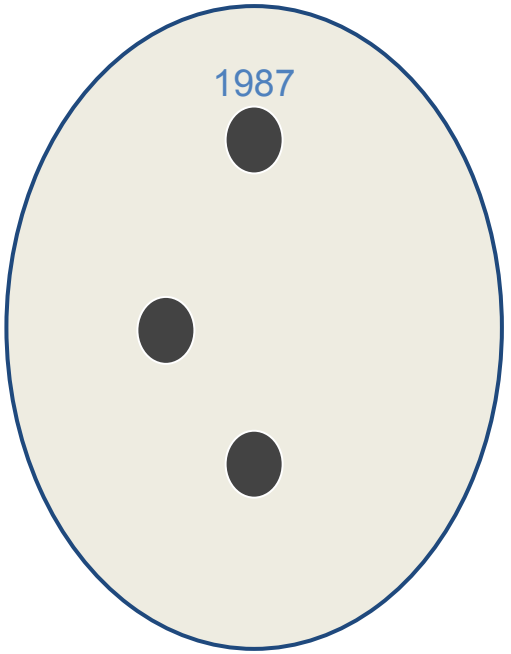
Sets • Membresía

Ejemplo:

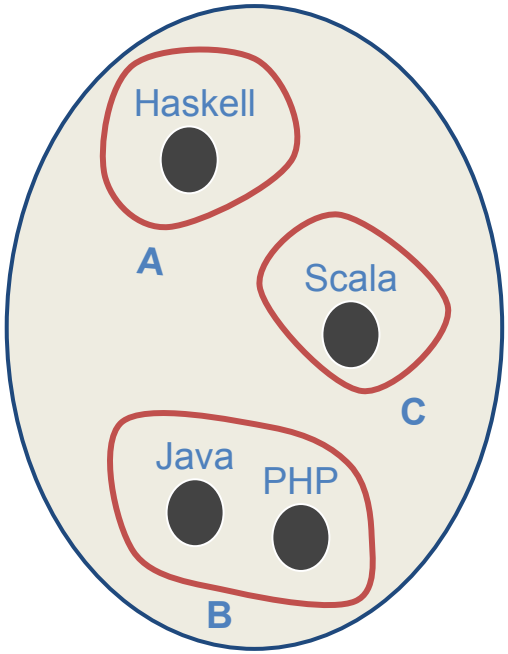
“1987 es elemento de Año”

Notación:

1987 ∈ Año



Sets · Subsets



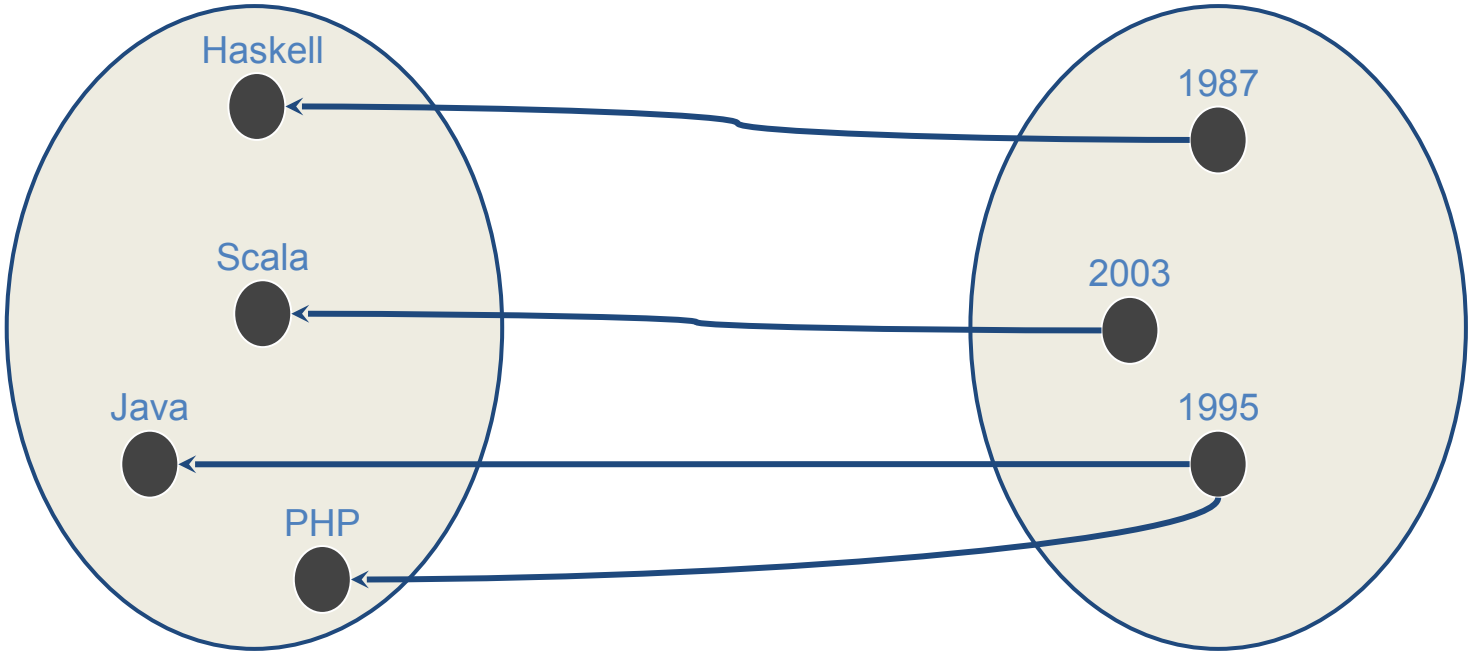
Ejemplo:

“B es subset de Lenguaje”

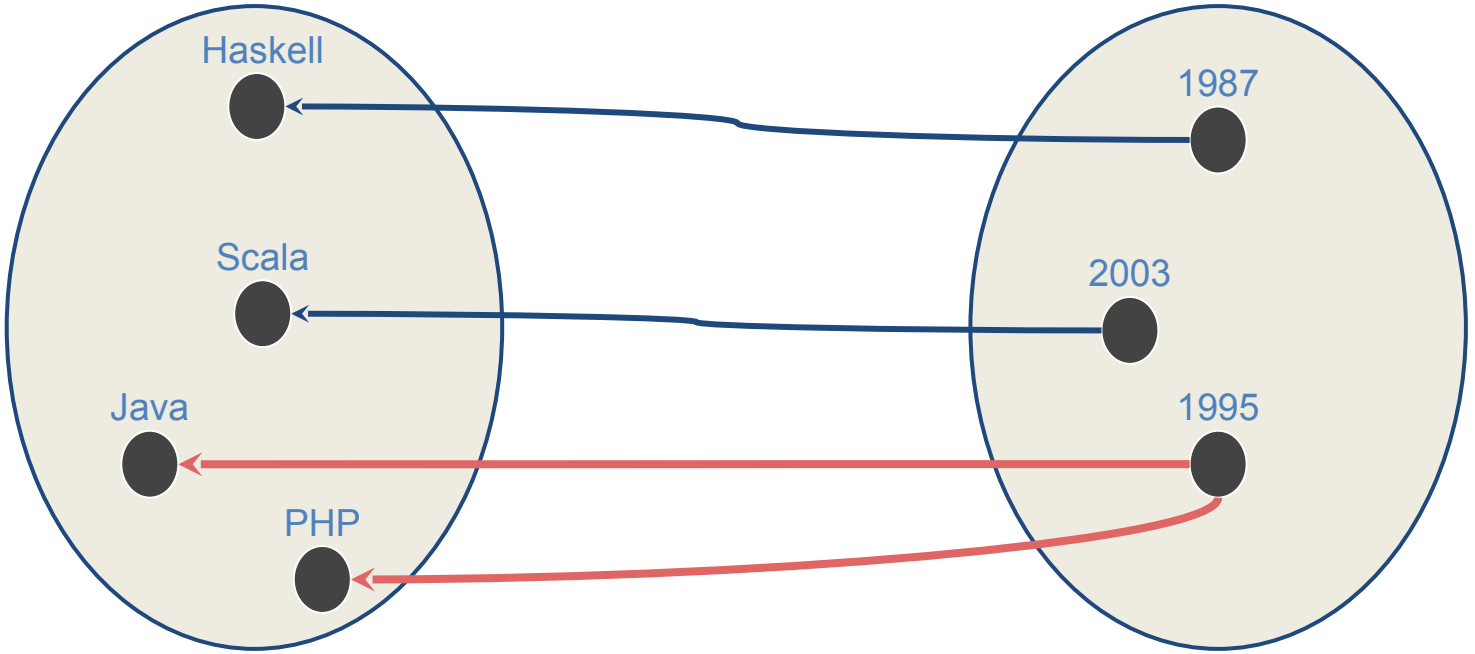
Notación:

B ⊆ Lenguaje

Sets · Relaciones



Sets · Relaciones

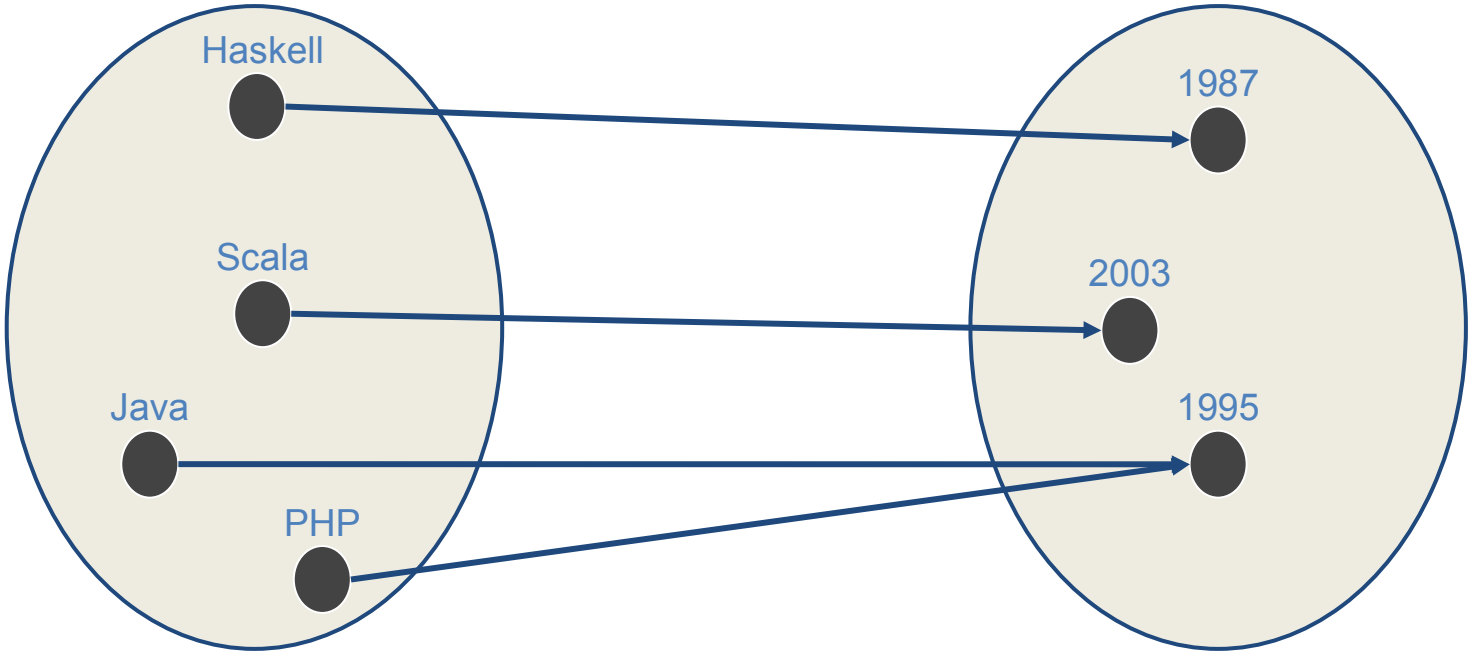


Funciones

Una función es una relación entre un set de inputs y un set de outputs permisibles...

...con la propiedad de que cada input está relacionado con exactamente un output.

Funciones



->

¿Que es la programación funcional?

La programación funcional es un paradigma de programación declarativa basado en la utilización de funciones aritméticas que no maneja datos mutables o de estado. Enfatiza la aplicación de funciones, en contraste con el estilo de programación imperativa, que enfatiza los cambios de estado

Programación funcional

- No mantiene estados.
- Enfatiza la aplicación de funciones
- Las funciones no tienen efecto secundario
- Uso de recurrencia

Ventajas de usar un paradigma funcional

- Ausencia de efectos colaterales (transparencia referencial)
- Proceso de depuración menos problemático
- Pruebas de unidades más confiables
- Mayor facilidad para la ejecución concurrente

Lenguajes funcionales

Entre los lenguajes funcionales puros, cabe destacar a Haskell y Miranda.

Los lenguajes funcionales híbridos más conocidos son Scala, Lisp, Clojure, Scheme, Ocaml, SAP y Standard ML (estos dos últimos, descendientes del lenguaje ML). Erlang es otro lenguaje funcional de programación concurrente.

Mathematica permite la programación en múltiples estilos, pero promueve la programación funcional. R también es un lenguaje funcional dedicado a la estadística.

Recientemente Microsoft Research está trabajando

Lisp

- Lisp (LISt Processing)
- Es el lenguaje más conocido de Programación Funcional. Aún así, no es un lenguaje funcional puro ya que posee asignación (SETF) e iteración (DO).
- Se utiliza la notación prefijo para cualquier función, inclusive para las expresiones aritméticas
- (defun cuadrado(n) (* n n))

Scala

- Scala es multiparadigma.
- Se puede utilizar el Paradigma Funcional y el orientado a Objetos
- Es de tipado estático
- Tiene genéricos similar a Java
- Todo es un Objeto (casi todo)
- Fue influenciado por Java, Ruby, Haskell, Erlang,
- etc...

Scala un lenguaje funcional y orientado a objetos

Smalltalk + Haskell = Scala

Inferencia de Tipos

- Inferencia de tipos : La inferencia de tipos asigna automáticamente un tipo de datos a una función sin necesidad de que el programador lo escriba.
- En C++ o Java debemos escribir lo siguiente:
- int i = 5;
- El compilador podría inferir el tipo, en scala podemos hacer:
- var i = 5

Todo retorna un valor y todo tiene un tipo

- En scala toda expresión retorna un valor y todo tiene un tipo.
- Por ejemplo un if puede retornar un valor y retorna lo ultimo que ejecuta:
- var i = if (a == 5) 4 else 10
- El compilador infiere el tipo y i va ser un entero.
- Si tenemos una expresión que retorna diferentes tipos, scala lo tipara por el padre común.
- Si tenemos la expresión: if (x > 0) 1
- En el caso del “else” no retornamos nada, en scala todo retorna un valor y la nada es un valor que tiene un tipo similar a void de c o java, este tipo es la clase Unit.

Val y var

- Val es la palabra clave que permite declarar un valor (similar a una constante) y var permite definir una variable.
- scala> val answer = 8 * 5 + 2
- answer: Int = 42
- scala> answer = 0
- <console>:6: error: reassignment to val
- Las varibles pueden ser asignadas con diferentes valores:
- var counter = 0
- counter = 1 // OK, can change a var

Todo es un Objeto

- Como Java, scala cuenta con tipos tales como Byte , Char , Short , Int , Long , Float y Double, además Boolean. Sin embargo no existe el concepto de valores primitivos, es decir todo es un objeto, en serio. Veamos un ejemplo:
- 1.toString() //"1"
- O
- 1.to(10) // Yields Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
- La sobrecarga de operadores en scala es más simple, dado que un operador no es más que un método de un objeto. Es decir:
- a + b
- Es la forma simplificada de :
- a.+(b)

Ya hablamos mucho de Scala, ahora la Práctica... Recursividad


UNIVERSIDAD
ADVENTISTA
DEL PLATA