CURSO DE PROGRAMACION SCALA
Sesión 3

Sergio Couto Catoira
ingscc00@gmail.com

Índice

Paquetes

- Declaración
  - package com.mypackage
- Importar
  - Una clase => import java.io.File
  - Todas las clases de un paquete => import java.io._
  - Varias clases => import java.io.{File, IOException, FileNotFoundException}

Paquetes 2

- Creación alias
  - Import java.util.{Date => UtilDate}

- Eliminar clases de un paquete
  - import java.util.{Random => _, _} (el segundo _ importa todas las demás clases)

Clases - creación

- Definicion
  - class Person(var name: String, val age: Int, salario: Int)
  - var => parametro accesible y reasignable
  - val => parametro accesible
  - Nada => parametro innacesible fuera de la clase
- Creación, lectura y modificación
  - val p = new Person("Pedro", 42, 1100)
  - p.age
  - p.name = "Pedro Garcia"

Clases - constructores

- Constructores => el constructor por defecto es la cabecera de la clase
- Constructores extra, 2 alternativas

```
class Person (val name: String, val age: Int) {

  def this(name: String) {
    this(name, 0)
  }
}
```

```
class Person (val name: String, val age: Int = 0)
```

Clases – métodos privados

- Por defecto los miembros accesibles son públicos. Se pueden hacer privados:

```
class Person (private val _name: String, private var _age: Int = 0) {

 def name = _name

 def age = _age

  def age_(newAge: Int) = _age = newAge

}
```

Ejercicio

- Define una clase Alumno con los atributos Nombre y apellidos
- Define una clase Asignatura con los atributos Nombre, limite de alumnos (por defecto 30) y descripcion (opcional)
- Define una clase Administración (signatura en código)
- Define los métodos alta y baja (signatura en código)

Sobreescritura de métodos

- Ejercicio: Sobreescribe el método toString en las clases Alumno y Asignatura definidas anteriormente

Currificación

- Una función puede recibir varias listas de argumentos para:
  - Parámetros implícitos (siguiente sesión)
  - Ayudar en inferencia de tipos (siguiente sesión)
  - Currificar
- Transformar una función que usa n elementos en una que usa un único argumento
  - Sin currificar: def uncurriedSum(x: Int, y: Int) = x+y
  - Currificada: def curriedSum(x: Int)(y: Int) = x+y

Currificación - ejemplo

> def sum (a: Int) (b:Int) (c: Int, d : Int) = a+b+c+d

> val f = add(5)

- f: Int => ((Int, Int) => Int)

> val g = f(2, 4)

- g: (Int, Int) => Int

> g(3,4)

Ejercicio

- Define una función uncurry con la siguiente signatura
  - def uncurry(f: Int => Int => Int) : (Int, Int) => Int
- Define una función curry con la siguiente signatura
  - def curry (f: (Int, Int) => Int) : Int => Int => Int

Composición de funciones

- Una función podría definirse como val, pero es necesario indicar el tipo
  - val duplicate: Int => Int = x => x*2
  - val print: Int => String = x => x.toString
- Palabra clave compose
  - val printDuplicate: Int => String = print compose duplicate

Ejercicio

- Implementa la función composicion con la siguiente signatura
  - def composicion(f: Int => String, g: Int => Int): Int => String

Funciones genéricas

- Funciones que varían segun el tipo de parámetro:
  - def add[A, B] (x: A, y: A)(f: (A, A) => B): B = f(x, y)

- Ejercicio: Implementa las funciones curry, uncurry y compose de forma genérica