Paradigmas de Programación (EIF-400) FP Kotlin Parte 04

CARLOS LORÍA-SÁENZ LORIACARLOS@GMAIL.COM

SETIEMBRE-OCTUBRE 2017

EIF/UNA

Objetivos

- Estudiar el paradigma de OOP-FP en el caso de Kotlin
- Contrastar con JS/Java8
- Clases/Objetos continuación

Material

- En el sitio
- Kotlin referencia

Funciones Resumen

- Funciones estilo método
  - Tipo de una función (`Unit` es default de retorno)
  - Parámetros default, nominados
  - Cuerpo de una expresión
  - Infijas
  - Recursiva de cola
- Lambdas `{…}`
  - Retornos etiquetados
  - Retorno no-local por defecto
  - Colecciones y lambdas
  - Inlining
- Funciones anónonimas `fun(…){…}`
  - Retorno local por defecto
- Funciones de extensión

Colecciones (Clásicas, eager)

- Array<T> (mutable, invariante)
- Hay `X`array `X` in {Int, Double,…}. No son subtipo de Array pero comparten interfaz
- List<T> (inmutable, covariante). Actualmente es un ArrayList de Java.
- MutableList<T> (mutable)
- Set<T>
- MutableSet<T>
- Map<K, V>
- MutableMap<K, V>
- HashMap (Map)
- "Constructor": typeOf(…) type in {Array, List, …}

Combinadores FP de Array

- Ver API
- Propiedades especiales lastIndex, índices
- Método útil withIndex() para obtener los elementos indexados
- Usuales, map, filter, fold (alias de reduce)
- Note: “every” de JS es “all”; “some” es “any”
- Revise drop, take y similares (comparables a slice)

Combinadores de List

- Ver API
- Similares a Array
- Recuerde que es inmutable

Colecciones Especiales

- `Range`: para representar sucesiones de enteros, chars, flotantes
- Son rangos cerrados (inclusive el extremo)
- Permiten el operador . .
- Igual tienen combinadores como map, filter, etc
- `Pair`: para representar un par ordenado
- Funcionan con desestructuración
- En general funciona con cualquier data class

Ejercicio

- Usando de base `fp.kt` (en `work`)
- Escriba `stats` que retorna el máximo y el mínimo y el promedio de una lista de enteros
- Hágala de dos maneras:
  - Usando `fold` (no use max, min, average del API)
  - Usando el API
- Haga un `data class` Stats que contenga los tres resultados y reemplace al `Pair` . Use desestructuración
- Haga una función $summation(numbers)$ que calcule $\sum_{i=1}^{n-1} i * numbers[i]$ siendo n el largo de $numbers$.

Ejercicio

- Considere el ejercicio de ES6 sobre grafos adjunto (graphs_es6)
- Mueva ambos archivos a Kotlin así como están
- Una vez logrado lo anterior haga lo pedido en el pdf adjunto