Paradigmas de Programación (EIF-400) FP Kotlin Parte 01

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

Material

- En el sitio
- Kotlin referencia

Básico

- Cubrir basics

Clases Resumen

- Clases
  - Constructores y propiedades
  - Bloque inicializador (`init`)
  - Constructores múltiples
  - Creación: No hay operador `new`
  - `public` por defecto
- Herencia (no `extends`)
  - Closed por defecto
  - `open`, `override`
- Funciones de extensión

Ejercicio (en `lcJS`)

- Convertir el modelo a Kotlin (ver slide siguiente para más detalles)

```
PP:node test_lc.js

>>> Testing Lambda Calculus (LC) <<<


Test Case: "toString of T"
>>> term.toString( ) = (((\X.(\Y.(X Y))) a) b) <<<


Test Case: "T equals T"
>>> term.equals( (((\X.(\Y.(X Y))) a) b) ) = true <<<


Test Case: "not equals G"
>>> term.equals( (\X.(\Y.(X Y))) ) = false <<<


Test Case: "equals clone"
>>> term.clone( (((\X.(\Y.(X Y))) a) b) ) = (((\X.(\Y.(X Y))) a) b) <<<


Test Case: "isClone of TC"
ERROR *** Term.isClone not (correctly) implemented! ***


Test Case: "freeVars of T"
ERROR *** Term.freeVars not (correctly) implemented! ***


Test Case: "depth of T"
ERROR *** Term.depth not (correctly) implemented! ***


Test Case: "replace Y by b in F"
>>> term.replace( {what:X,  by:b , except: } ) = (\Y.(b Y)) <<<


Test Case: "reduce T Lazy"
>>> term.reduce( {level:10}} ) = (a b) <<<
```

Ejercicio… (en work)

- Estudie y ponga a correr `lc.kt` y `testLc.kt`
- Son versiones convertidas de `lc.js` `test_lc.js`

```
KT:kotlinc -nowarn -d classes lc.kt

KT:kotlinc -nowarn -cp classes -d classes testLc.kt

KT:kotlin -cp classes eif400.test.TestLcKt

>>> Testing Lambda Calculus (LC) <<<


Test Case: 'toString of T'
>>> term.toString( [] ) = X <<<


Test Case: 'T equals T'
>>> term.equals( [X] ) = true <<<
```

Ejercicio…

- Termine de convertir los métodos faltantes
- Implemente de manera nueva los que no estaban implementados

Continuará…

- Más sobre interfaces
- Lambdas
- FP en colecciones