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

CARLOS LORÍA-SÁENZ LORIACARLOS@GMAIL.COM

SETIEMBRE 2017

EIF/UNA

Objetivos

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

Material

- En el sitio
- Kotlin referencia

Básico

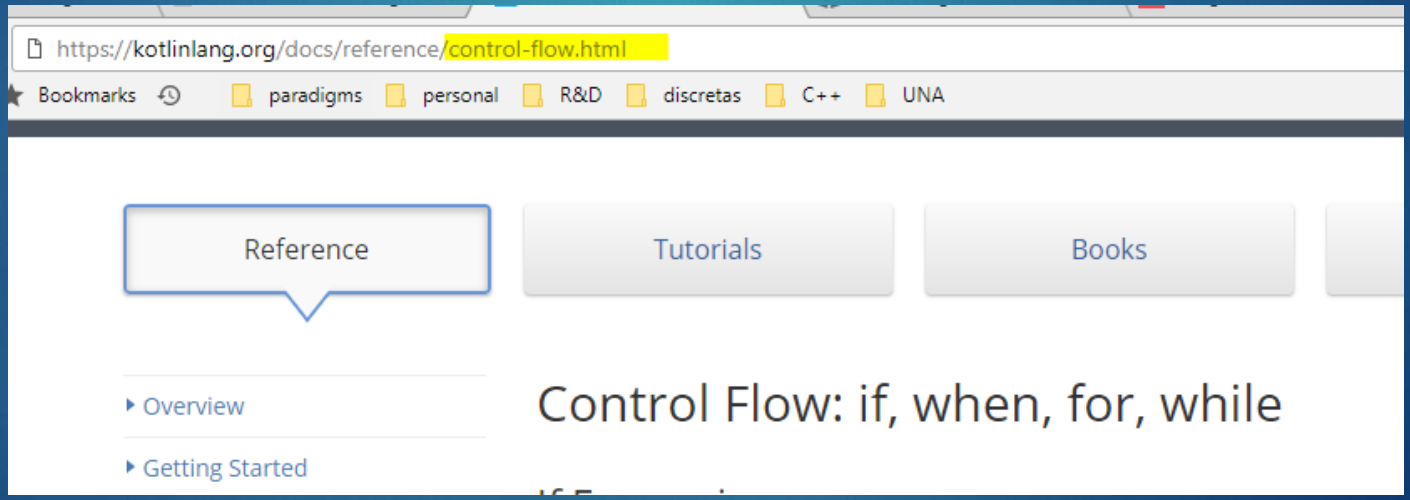
- Cubrir basics

Estructuras comunes

- Todo dato es un objeto
- Declaraciones val (no asignable) vs var (asignable)
- Hay const (se calcula en compilación)
- Loops imperativos while, do-while como en Java
- If-the-else como expresiones
- when
- Formas basadas en for
- Colecciones Array, List, MutableList
- ClosedRange (n..m)

Ejercicios

- Haremos algunos ejercicios inicialmente usando programación imperativa
- Flujo de control usual en tal caso



Ejercicio

- Modifique `misc.kt` para que imprima cuáles de los argumentos de línea de comando son números
- Para compilar y correr

```
22/09/2017  13:33    <DIR>          .
22/09/2017  13:33    <DIR>          ..
20/09/2017  13:46                90 misc.kt
               1 archivos             90 bytes
               2 dirs  19.141.083.136 bytes libres

PP:kotlinc -d classes misc.kt

PP:kotlin -cp classes MiscKt
>>> Testing <<<
10

PP:
```

Ejercicio…

- Salida

```
fun printOnlyNumbers(args: Array<String>){
    println("Print only Numbers")
    for (value in args){
        val num = value.toIntOrNull();
        if (num != null)
            println("num=$num")
        else println("*** $value ***")
    }
}
fun main(args: Array<String>){
    println(">>> Testing <<<")
    var n = 10
    println(n)
    printOnlyNumbers(args);
}
```

```
PP:kotlin -cp classes MiscKt 10 20 5 $$$ a a 1
>>> Testing <<<
10
Print only Numbers
num=10
num=20
num=5
*** $$$ ***
*** a ***
*** a ***
num=1
```

Ejercicio…

- Modifique para que salga el índice de cada número (numero de argumento)
- Use `args.withIndex` en el `for` que retorna `(index, value)`

```
PP:kotlin -cp classes MiscKt 10 20 5 $$$ a a 1
>>> Testing <<<
10
Print only Numbers
0) num=10
1) num=20
2) num=5
*** $$$ ***
*** a ***
*** a ***
6) num=1
```

Ejercicio…

- Imprima el máximo y el mínimo entre todos los números tecleados
- Haga primero `selectOnlyNumbers` que retorna solo los números tecleados en una lista (List)
- Haga `maxMinOfList` que recibe esa lista y la imprime

```
fun printOnlyNumbers(args: Array<String>){
fun selectOnlyNums(args: Array<String>): List<Int>{

fun maxMinOfList(nums: List<Int>): List<Int>{

fun main(args: Array<String>){
    println(">>> Testing <<<")
    var n = 10
    println(n)
    printOnlyNumbers(args);
    val maxMin = maxMinOfList(selectOnlyNums(args))
    println("max=${maxMin[1]} min=${maxMin[0]}")
}
```

```
PP:kotlin -cp classes MiscKt 10 20 5 $$$ a a 1
>>> Testing <<<
10
Print only Numbers
num=10
num=20
num=5
*** $$$ ***
*** a ***
*** a ***
num=1
max=20 min=1
```

Ejercicio…

- Haga `inRange(num, min, max)` que retorne true si `num` está entre `min` y `max`. Modifique para que salga lo siguiente
- Use operador `..` de rango

```
PP:kotlin -cp classes MiscKt 10 20 5 $$$ a a 1
>>> Testing <<<
10
Print only Numbers
0) num=10 inRange(1,10)=true
1) num=20 inRange(1,10)=false
2) num=5 inRange(1,10)=true
*** $$$ ***
*** a ***
*** a ***
6) num=1 inRange(1,10)=true
max=20 min=1
```

Ejercicio…

- Haga gcd(a, b) que retorne el máximo común divisor entre a y b. Use when en su implementación. Modifique para que salga lo siguiente (el gcd de los números tecleados)

```
PP:kotlin -cp classes MiscKt 10 a 20 b 75 c
>>> Testing <<<
10
Print only Numbers
0) num=10 inRange(1,10)=true
*** a ***
2) num=20 inRange(1,10)=false
*** b ***
4) num=75 inRange(1,10)=false
*** c ***
max=75 min=10
gcd=5
```

Continuará…

- Con clases y objetos