Why KOTLIN is my Favourite example of Functional Programming?

Jayashree S Kumar, IBM
Agenda

• What? Kotlin, Functional Programming
• Why? Kotlin, Function Programming
• My Favourite FP concepts with Kotlin
• Conclusion
About Me

- Java Classes Library developer
- Worked Extensively on JDK’s Testing
- Runtimes team @ IBM Software Labs
WHAT ?
• Statically Typed
• JVM Targeted
• Open Source
• OO & FP features
• 100% Interoperable with Java

Borrowed: https://www.slideshare.net/abreslav/introduction-to-kotlin-brief-and-clear
- Functions first class citizens
- Immutable data
- Mathematical (Pure) functions
- Easier to write, reuse & test
- No loops => lots of recursion, less efficient.
fun main(args : Array<String>) {
    val name : String = "World"
    println("Hello, $name!")
}

name = "Hi"

Compile error: Val cannot be reassigned

//Top-level functions
//Optional “;”. no "new"
//Types on the right
//Strong type inference
//Val & Var(traditional way) to declare types
WHY?
FP now?

- Highly parallel multicore architectures -> high concurrency -> immutability in code design
- Better compilers and runtimes -> more expressive
- Because Java got it to mainstream!

Fact: Whatsapp needs only 50 engineers for its 900M user because Erlang is used to implement its concurrency needs. Facebook uses Haskell in its anti-spam system.
Kotlin?

- Modern, Concise, SAFE
- NETFLIX
- PYPL - 16th rank, SOF - least disliked, RedMonk - Raising.
- Top-Notch Tools
- SIMPLE learning curve
- ~3.4 million apps on Google Play
- GDG Milano
HOW?
fun double (n : Int) : Int = n * 2

val double = { n : Int -> n * 2}
or
double{ it * 2 }

println(double(2))
Where Lambda?

Execute a block of code later—> once or multiple times.

- Run code in separate thread.
- Run it multiple times
- Run only when necessary
- Run it at specific point in an algorithm
  Eg. comparison operation while sorting
- Run code when some event happens
  Eg. on click of a link
fun higherOrder(func : (Int, Int) -> Int) {
}

fun sum (x: Int, y : Int) = x + y

fun main(args: Array<String>)
{
    higherOrder(sum)       //Pass existing functions
    higherOrder(x,y -> x +y)  //Write lambda
}

Higher-Order Functions
Building blocks of HOF

```scala
val numbers = 1..10

numbers.filter{ it % 5 == 0 }
  .map { it * 2 }
  .forEach{ println(it) }
```

10
20
fun closureMaker(): () -> Unit {
    var num = 0
    return { println(num++)}
}

fun main(args: Array<String>){
    val myCounter1 = closureMaker()
    val myCounter2 = closureMaker()
    myCounter1()
    myCounter1()
    myCounter1()
    myCounter1()
    myCounter2()
    myCounter2()
    myCounter1()
}
fun MutableList<Int> .swap(index1: Int, index2: Int) {
    val tmp = this[index1]
    this[index1] = this[index2]
    this[index2] = tmp
}

fun main(args: Array<String>) {
    val abc = mutableListOfOf(1, 2, 3)
    abc.swap(0, 2)
    abc = (3, 2, 1)
Composition, Currying

```kotlin
import org.funktionale.composition.*
import org.funktionale.currying.*

fun functional() {

    val add5 = {i: Int -> i + 5}
    val multiplyBy2 = {i: Int -> i * 2}
    val multiplyBy2andAdd5 = add5 compose multiplyBy2
    val composeResults = multiplyBy2andAdd5(10)
    println("multiplyBy2andAdd5(10) = \$composeResults")

    val sum2ints = { x: Int, y: Int -> x + y }
    val curried: (Int) -> (Int) -> Int = sum2ints.curried()
    val cur = curried(2)(4)
    println("curried(2)(4) = \$cur")
}
```
val loggingEnabled = System.getProperty("log") != null

fun log(s: String): Unit = if (loggingEnabled) println(s)

fun log(ls: () -> String): Unit = if (loggingEnabled) println(ls())

log("now!")
log { "calculate me later" }
Convert Java File to Kotlin File

CTRL + ALT + SHIFT + K
(or CMD + ALT + SHIFT + K)

Command-line

> vi HelloWorld.kt
> kotlinc HelloWorld.kt -include-runtime -d HelloWorld.jar
> java -jar HelloWorld.jar
Conclusion

• FP is a journey.

• Kotlin is making your life easier to on-board by making usage of functional concepts easier..

• Java, owing to its massive adoption, bears much more responsibility while making changes…

• Happier to use my Java and Kotlin in conjunction!! :)
To each his own
Evaluate the Sessions

Sign in and vote at eclipsecon.org
Thank U!