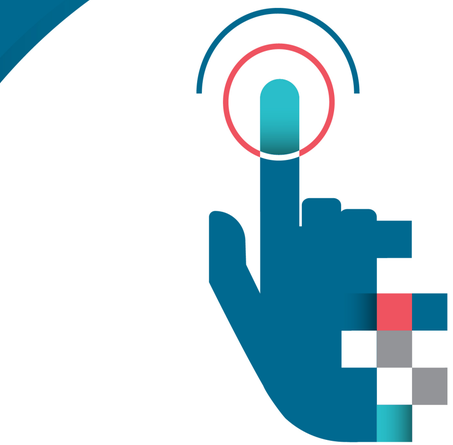
2019

1° ESCUELA DE ACTUALIZACIÓN EN TECNOLOGÍAS DE INFORMÁTICA

PROGRAMACIÓN EN KOTLIN PARA PLATAFORMAS JAVA Y ANDROID

LIC. EMMANUEL LAGARRIGUE LAZARTE



eati

DCIC DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

UNS UNIVERSIDAD NACIONAL DEL SUR

Sobrecarga de Operadores y otras convenciones



```
data class Point(val x: Int, val y: Int)
```

Sobrecarga de Operadores Aritméticos

Operaciones Aritméticas Binarias

```
data class Point(val x: Int, val y: Int) {
    operator fun plus(other: Point): Point {
        return Point(x + other.x, y + other.y)
    }
}

>>> val p1 = Point(10, 20)
>>> val p2 = Point(30, 40)
>>> println(p1 + p2)
Point(x=40, y=60)
```

Adds the coordinates and returns a new point

Defines an operator function named “plus”

Calls the “plus” function using the + sign



```
operator fun Point.plus(other: Point): Point {
    return Point(x + other.x, y + other.y)
}
```

Expression	Function name
`a * b`	`times`
`a / b`	`div`
`a % b`	`mod`
`a + b`	`plus`
`a - b`	`minus`



```
operator fun Point.times(scale: Double): Point {
    return Point((x * scale).toInt(), (y * scale).toInt())
}

>>> val p = Point(10, 20)
>>> println(p * 1.5)
Point(x=15, y=30)
```


```
operator fun Char.times(count: Int): String {
    return toString().repeat(count)
}

>>> println('a' * 3)
aaa
```

Operadores de Asignación compuesta

```
>>> var point = Point(1, 2)
>>> point += Point(3, 4)
>>> println(point)
Point(x=4, y=6)
```

Operadores Unarios

```
operator fun Point.unaryMinus(): Point {
    return Point(-x, -y)
}

>>> val p = Point(10, 20)
>>> println(-p)
Point(x=-10, y=-20)
```

The unary minus function has no parameters.

Negates the coordinates of the point, and returns it

Expression	Function name
`+a`	`unaryPlus`
`-a`	`unaryMinus`
`!a`	`not`
`++a, a++`	`inc`
`--a, a--`	`dec`

```
operator fun BigDecimal.inc() = this + BigDecimal.ONE

>>> var bd = BigDecimal.ZERO
>>> println(bd++)
0
>>> println(++bd)
2
```

Increments after the first println statement executes

Increments before the second println statement executes

Operadores de Comparación

```
class Point(val x: Int, val y: Int) {
    override fun equals(obj: Any?): Boolean {
        if (obj === this) return true
        if (obj !is Point) return false
        return obj.x == x && obj.y == y
    }
}

>>> println(Point(10, 20) == Point(10, 20))
true
>>> println(Point(10, 20) != Point(5, 5))
true
>>> println(null == Point(1, 2))
false
```

Overrides the method defined in Any

Optimization: checks whether the parameter is the same object as "this"

Checks the parameter type

Uses a smart cast to Point to access the x and y properties

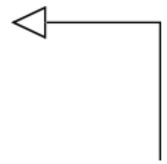
```
class Person(
        val firstName: String, val lastName: String
) : Comparable<Person> {
    override fun compareTo(other: Person): Int {
        return compareValuesBy(this, other,
            Person::lastName, Person::firstName)
    }
}
```

Evaluates the given callbacks in order, and compares values

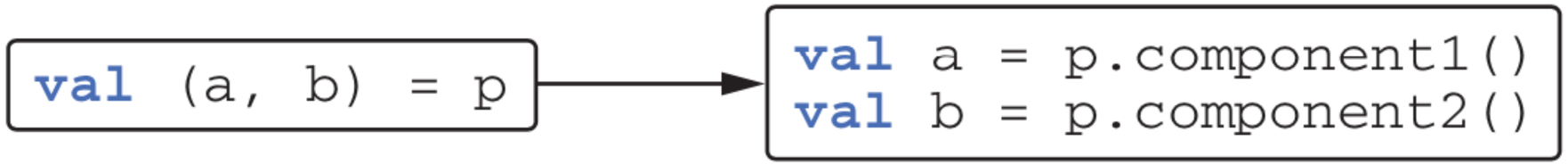
```
>>> val p1 = Person("Alice", "Smith")
>>> val p2 = Person("Bob", "Johnson")
>>> println(p1 < p2)
false
```

Des-estructurando Declaraciones

```
>>> val p = Point(10, 20)
>>> val (x, y) = p
>>> println(x)
10
>>> println(y)
20
```



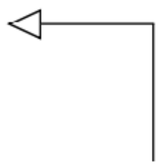
Declares variables x and y, initialized with components of p




```
class Point(val x: Int, val y: Int) {
    operator fun component1() = x
    operator fun component2() = y
}
```



```
fun printEntries(map: Map<String, String>) {
    for ((key, value) in map) {
        println("$key -> $value")
    }
}
```



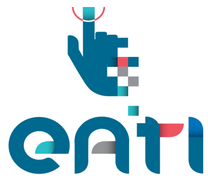
Destructuring declaration in a loop

```
>>> val map = mapOf("Oracle" to "Java", "JetBrains" to "Kotlin")
>>> printEntries(map)
Oracle -> Java
JetBrains -> Kotlin
```

Coroutines


```
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.0.0'
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.0.0'
```

```
import kotlinx.coroutines.*

fun main() {
    GlobalScope.launch { // launch new coroutine in background and continue
        delay(1000L) // non-blocking delay for 1 second (default time unit is ms)
        println("World!") // print after delay
    }
    println("Hello,") // main thread continues while coroutine is delayed
    Thread.sleep(2000L) // block main thread for 2 seconds to keep JVM alive
}
```



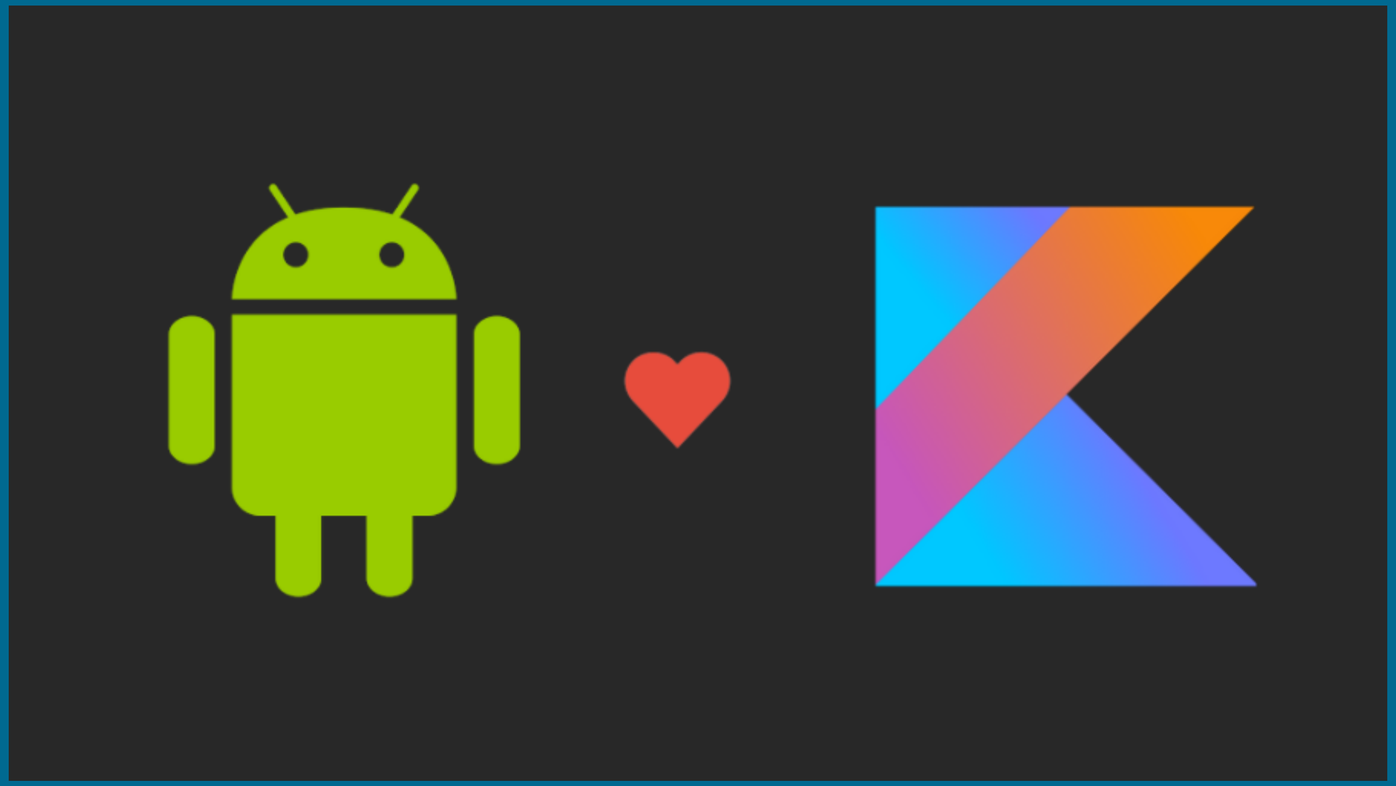


```kotlin
import kotlinx.coroutines.*

fun main() {
    GlobalScope.launch { // launch new coroutine in background and continue
        delay(1000L)
        println("World!")
    }
    println("Hello,") // main thread continues here immediately
    runBlocking {     // but this expression blocks the main thread
        delay(2000L)  // ... while we delay for 2 seconds to keep JVM alive
    }
}
```

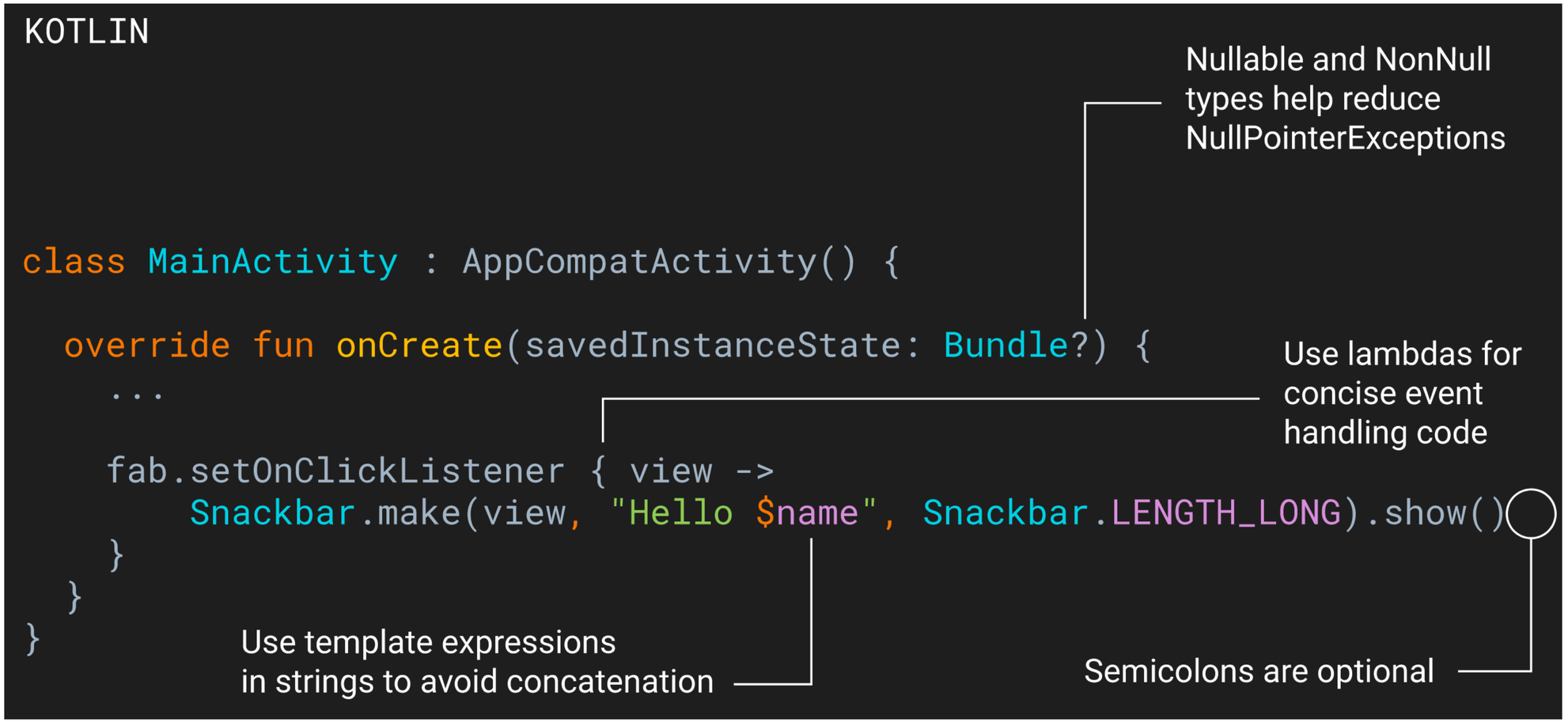
Waiting for a job

```kotlin
import kotlinx.coroutines.*

fun main() = runBlocking {
    val job = GlobalScope.launch { // launch new coroutine and keep a reference to its Job
        delay(1000L)
        println("World!")
    }
    println("Hello,")
    job.join() // wait until child coroutine completes
}
```

Kotlin en Android



- Kotlin es soportado por el android studio desde 3.0.
- No solo nos permite desarrollar en kotlin, disponemos de herramientas como el conversor de java a kotlin.



```
buildscript {
    ext.kotlin_version = '1.3.0'
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.2.1'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
```

_main.xml | MainActivity.kt | app

```
apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

apply plugin: 'kotlin-android-extensions'
```

```
implementation fileTree(dir: 'libs', include: ['*.jar'])
implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
```

Extensions

```
// Using R.layout.activity_main from the 'main' source set
import kotlinx.android.synthetic.main.activity_main.*

class MyActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Instead of findViewById<TextView>(R.id.textView)
        textView.setText("Hello, world!")
    }
}
```

Algunas librerías hechas en Kotlin

Fuel https://github.com/kittinunf/Fuel

- Cliente HTTP para Kotlin/Android

```
//an extension over string (support GET, PUT, POST, DELETE with httpGet(), httpPut(
"https://httpbin.org/get".httpGet().responseString { request, response, result ->
  //do something with response
  when (result) {
    is Result.Failure -> {
      val ex = result.getException()
    }
    is Result.Success -> {
      val data = result.get()
    }
  }
}
```

```
val (request, response, result) = "https://httpbin.org/get".httpGet().responseString()
```

Koin

https://github.com/InsertKoinIO/koin

- Framework de inyección de dependencias

```
// Given some classes
class Controller(val service : BusinessService)
class BusinessService()

// just declare it
val myModule = module {
  single { Controller(get()) }
  single { BusinessService() }
}
```

```
class MyApplication : Application() {
  override fun onCreate(){
    super.onCreate()
    // start Koin!
    startKoin(this, listOf(myModule))
  }
}
```

```
// Just inject in a simple Activity
class MyActivity() : AppCompatActivity() {

    // lazy inject BusinessService into property
    val service : BusinessService by inject()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // or directly get any instance
        val service : BusinessService = get()
    }
}
```

```
// Controller & BusinessService are declared in a module
class Controller(val service : BusinessService){

  fun hello() {
     // service is ready to use
     service.sayHello()
  }
}
```

Mockito-Kotlin

https://github.com/nhaarman/mockito-kotlin

- Mocks de objetos para unit testing

```kotlin
@Test
fun doAction_doesSomething(){
  /* Given */
  val mock = mock<MyClass> {
    on { getText() } doReturn "text"
  }
  val classUnderTest = ClassUnderTest(mock)

  /* When */
  classUnderTest.doAction()

  /* Then */
  verify(mock).doSomething(any())
}
```

Kluent

https://github.com/MarkusAmshove/Kluent

- Asserts para unit testing

assertEquals

```
"hello" shouldEqual "hello"
```

assertNotEquals

```
"hello" shouldNotEqual "world"
```

Assert that an Array/Iterable contains something

```
val alice = Person("Alice", "Bob")
val jon = Person("Jon", "Doe")
val list = listOf(alice, jon)
list shouldContain jon
```

Stubbing

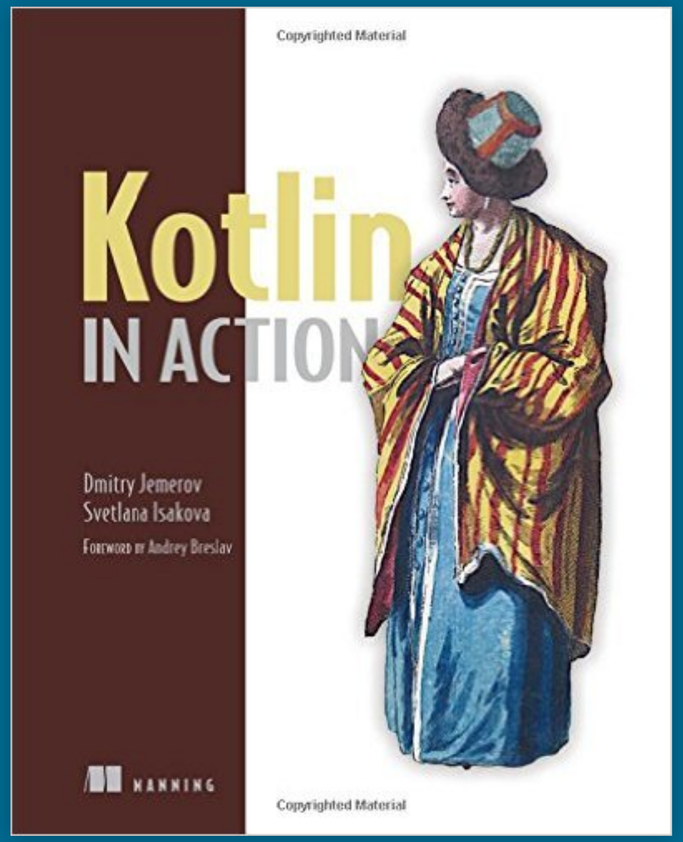
```
val stub = mock(Database::class)
val bob = Person("Bob", "Guy")
When calling stub.getPerson() itReturns bob
```

assertEquals

```
"hello" `should equal` "hello"
```

assertNotEquals

```
"hello" `should not equal` "world"
```



https://kotlinlang.org


ANDROID
JAVA
KOTLIN