

Software para Dispositivos Móviles
Grado en Ingeniería Informática del Software
Escuela de Ingeniería Informática – Universidad de Oviedo



Introducción a Kotlin

Dr. Juan Ramón Pérez Pérez
Departamento de Informática
jrpp@uniovi.es

Historia de Kotlin

- En **julio de 2011** JetBrains reveló el Proyecto Kotlin en la **JVM Language Summit**
- En **febrero de 2012**, JetBrains **liberó** el código fuente del proyecto bajo la Licencia Apache 2
- En **febrero de 2015** se lanza **Kotlin 1.0**
- En **mayo de 2017** en la conferencia **Google I/O** se anuncia que Android tendrá soporte para Kotlin al mismo nivel que Java
- **Google I/O 2019** comienza el enfoque **Kotlin first**

Kotlin ahora





- Lenguaje en expansión en todos los índices
- [PYPL Popularity of Programming Language](#) (p 12)
- [The State of Developer Ecosystem 2020](#) entornos de uso.

Worldwide, Oct 2023 :

Rank	Change	Language	Share	1-year trend
1		Python	28.05 %	+0.1 %
2		Java	15.88 %	-1.0 %
3		JavaScript	9.27 %	-0.3 %
4		C#	6.79 %	-0.2 %
5		C/C++	6.59 %	+0.3 %
6		PHP	4.86 %	-0.4 %
7		Lanzamiento 2014	4.45 %	+0.4 %
8			2.93 %	+0.1 %
9	↑↑	Swift	2.69 %	+0.7 %
10	↓	Objective-C	2.29 %	+0.2 %
11	↑↑	Lanzamiento 2017	2.05 %	+0.4 %
12	↓↓		1.95 %	-0.1 %
13	↓	Kotlin	1.76 %	-0.1 %
14		Matlab	1.53 %	+0.1 %
15		Ruby	1.04 %	-0.1 %
16	↑↑↑↑	Ada	1.04 %	+0.2 %
17	↓	VBA	0.94 %	-0.1 %
18		Dart	0.9 %	+0.1 %
19	↓↓↓	Powershell	0.89 %	-0.0 %
20		Scala	0.69 %	-0.0 %

Versiones y soporte

- Actualmente (sept 2024) versión 2.0
- Kotlin 2.0 incorpora el nuevo compilador K2
- Distintas Apps de Android desarrolladas en Kotlin
- Soporte en distintos entornos de desarrollo:

			
USE IntelliJ IDEA	USE Android Studio	USE Eclipse	STANDALONE Compiler
Bundled with Community Edition or IntelliJ IDEA Ultimate	Bundled with Studio 3.0 plugin available for earlier versions	Install the plugin from the Eclipse Marketplace	Use any editor and build from the command line
Instructions	Instructions	Instructions	Download Compiler

¿Qué es Kotlin?

- Kotlin es un lenguaje de programación fuertemente tipado desarrollado por JetBrains (los creadores de IntelliJ IDEA).
- Ha sido fuertemente **influenciado por lenguajes como Groovy, Scala o C#**.
- Compatible e interoperable con Java
- **Permite generar código para:**
 - **la JVM** (máquina virtual de Java)
 - **Versión multiplataforma** y permite **ejecutables nativos (web, escritorio, iOS)** en distintos niveles de desarrollo.

Características destacadas

Programming language for



JVM



Android



Browser



Native

Why Kotlin?



Concise

Drastically reduce the amount of boilerplate code.

[See example](#)



Safe

Avoid entire classes of errors such as null pointer exceptions.

[See example](#)



Interoperable

Leverage existing libraries for the JVM, Android, and the browser.

[See example](#)



Tool-friendly

Choose any Java IDE or build from the command line.

[See example](#)



Sintaxis, variables, clases y métodos

Simplificando la sintaxis y ayudando al compilador

Sintaxis (Java) →

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        char interrogacion= '?';  
        int anio= 2017;  
        String mes= "agosto";  
        String oficial= "se convirtio en lenguaje oficial de  
android";  
        TextView textviewjr=  
(TextView)findViewById(R.id.textView);  
        textviewjr.setText("¿Sabias que Kotlin "+oficial+" en  
"+mes+" de "+anio+interrogacion);  
    }  
}
```



Versión Java

Sintaxis (Kotlin)

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main) ;  
        var interrogacion: Char= '?'  
        val anio= 2017  
        val mes= "agosto"  
        val oficial= "se convirtio en lenguaje oficial de  
android"  
        textViewjr.text= "¿Sabias que Kotlin $oficial en $mes de  
$anio$interrogacion"  
    }  
}
```



Versión Kotlin

Tipos básicos, variables y campos

- Los tipos primitivos son clases: Int, Float, Char
- No existe void, algo equivalente sería Unit
- No hay conversión automática entre tipos, hay que usar métodos **.to<Tipo>()**
- Variables mutables (var) e inmutables (val)
- Los campos de las clases son propiedades

Clases en Kotlin

```
open class Animal(open val name: String)
class Person(override val name: String, val surname:
String = "García"): Animal(name)

val pepe= Person("José", "López")
val paco= Person("Francisco")
```



Versión Kotlin

Constructores

```
class Person(name: String, surname: String) {  
    init {  
        name= name.toUpperCase()  
    }  
  
    constructor(name: String, parent: Person): this(name) {  
        parent.children.add(this)  
    }  
  
    //...  
}
```



Funciones elementos de
primer nivel

Funciones elementos de primer nivel

// Simplificación de una función en Kotlin

```
fun add(x: Int, y: Int): Int {  
    return x+y;  
}
```

```
fun add(x: Int, y: Int): Int = x+y
```

```
fun add(x: Int, y: Int) = x+y
```



Versión Kotlin

Métodos de extensión

```
fun Double.changeSign(): Double = this * -1

fun Float?.roundTwoDecimals(): Float = Math.round(this ?: 0F) *
100F / 100F

fun Long.getAsPrintableDate(): String {
    val simpleDateFormat=
        SimpleDateFormat("dd/MM/yyyy",Locale.ENGLISH)
    return simpleDateFormat.format(Date(this))
}

fun View.visible() { this.visibility = View.VISIBLE }

fun View.invisible() { this.visibility= View.INVISIBLE }

fun View.gone() { this.visibility= View.GONE }
```



Funciones lambda: Funciones que reciben y devuelven funciones

```
// funciones que reciben y devuelven funciones
fun calCircumference(radius: Double) = (2 * Math.PI) *
radius

fun calArea(radius: Double): Double = (Math.PI) *
Math.pow(radius, 2.0)

fun circleOperation(radius: Double, op: (Double) ->
Double): Double {
    val result = op(radius)
    return result
}

// Llamada
println("Calcula el área del círculo: " +
    circleOperation(3.0, ::calArea))
```


Programación funcional -> funciones lambda (I)

Paso de funciones como parámetros (Java)

```
public class Activity extends AppCompatActivity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        View boton = findViewById(R.id.botonID);  
  
        boton.setOnClickListener(new OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Toast.makeText(v.getContext(),  
                    "Has pulsado el botón",  
                    Toast.LENGTH_SHORT).show(); }  
        }  
    }  
}
```



Versión Java

Programación funcional -> funciones lambda (II)

Paso de funciones como parámetros (Java)

```
public class Activity extends CompactActivity implements OnClickListener{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View boton = findViewById(R.id.botonID);

        boton.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        Toast.makeText(v.getContext(), "Has pulsado el botón",
            Toast.LENGTH_SHORT).show();
    }
}
```



Versión Java

Programación funcional -> funciones lambda (III)

Paso de funciones como parámetros (Kotlin)

```
class Activity : CompactActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.main)  
        boton= find<Button>(R.id.botonID)  
  
        boton.setOnClickListener(object : OnClickListener() {  
            override fun onClick(v:View) {  
                toast("Has pulsado el botón")  
            }  
        })  
    }  
}
```

Con interfaces



Versión Kotlin

Programación funcional -> funciones lambda (IV)

Paso de funciones como parámetros (Kotlin)

```
class Activity : CompactActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.main)  
        boton= find<Button>(R.id.botonID)  
  
        boton.setOnClickListener({ view -> toast("Has pulsado el botón")})  
    }  
}
```



Versión Kotlin

Programación funcional -> funciones lambda (V)

Paso de funciones como parámetros (Kotlin)

```
class Activity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.main)  
        boton= find<Button>(R.id.botonID)  
  
        | boton.setOnClickListener { toast("Has pulsado el botón")}  
    }  
}
```

**Con expresiones
lambda**



Estructuras de control,
colecciones y prog. funcional

When

```
when (x) {  
  1 -> print("x == 1")  
  2 -> print("x == 2")  
  else -> { // Se puede crear un bloque  
    print("x no es ni 1 ni 2")  
  }  
}
```

When con rangos

```
when (x) {  
  in 1..10 -> print("x dentro del rango")  
  in validNumbers -> print("x es válido")  
  !in 10..20 -> print("x fuera del rango")  
  else -> print("Ninguno de los de arriba")  
}
```


When sin argumentos

```
when {  
  x.isOdd() -> print("x is odd")  
  x.isEven() -> print("x is even")  
  else -> print("x is funny")  
}
```

Condiciones y bucles simples

- **While()** y **do ... while()** funcionan como en Java.
- **if** y **when** son expresiones

for (iterador, rangos)

```
for (item: Int in enteros) {  
    // ...  
}  
  
for (i in 1..3) {  
    println(i)  
}  
  
for (i in 6 downTo 0 step 2) {  
    println(i)  
}  
  
for (i in array.indices) {  
    println(array[i])  
}
```

Recorrido sobre una colección

```
val fruits = listOf("platano", "aguacate", "manzana",  
"pera")
```

```
fruits.forEach { println(it) }
```

Expresión
lambda

Colecciones y operaciones funcionales

```
val fruits = listOf("platano", "aguacate", "manzana",  
"pera")  
fruits  
    .filter { it.startsWith("p") } //sólo empiezan por "p"  
    .sortedBy { it }               // ordena elementos  
    .map { it.toUpperCase() }      // los convierte todos a mayus  
    .forEach { println(it) }      // imprime todos
```

Operaciones típicas con colecciones

- Aggregate operations: any, all, max, min, forEach, reduce, sumBy
- Filtering operations (filtra elementos): drop, filter, take
- Mapping operations (modifica cada elemento): map
- Elements operations: elementAt, contains, first, indexOf
- Ordering operations: reverse, sort, sortBy



Clases Data, static y singleton

Personalización de clases

Clase POJO / JavaBean (Java)

```
public class Forecast {  
    private Date date;  
    private float temperature;  
    private String details;  
  
    public Forecast(Date date, float temperature, String details) {  
        this.date= date;  
        this.temperature= temperature;  
        this.details= details;  
    }  
  
    public Date getDate() { return date; }  
    public void setDate(Date date) { this.date= date; }  
    public float getTemperature() { return temperature; }  
    public void setTemperature(float temperature) { this.temperature=  
temperature; }  
    public String getDetails() { return details; }  
    public void setDetails(String details) { this.details= details; }  
  
    @Override  
    public String toString() {  
        return "Forecats{" + "date= " + date + ',' +  
        "temperature= " + temperature + ',' + "details= \"" + details + "\"\" +  
        '}'';  
    }  
}
```



Clases data (Kotlin)

```
data class Forecast(val date: Date, val temperature: Float,  
                    val details:String)
```



Versión Kotlin

- Sólo contiene estado y no realiza ninguna operación.
- Además, incluye los métodos:
 - equals() y además se sobrecarga el operador == para que sea equivalente
 - hashCode()
 - copy(), parecido a clone(), pero permitiendo cambios.
 - Funciones componentX()

Personalización de propiedades

```
data class ForecastList(val id: Long, val city: String, val
country: String, val dailyForecast: List<Forecast>) {
    // Sobrecarga del operador []
    operator fun get(position: Int): Forecast =
        dailyForecast[position] // Listas lo tienen sobrecargado
    // Otra propiedad que no tiene set y el get personalizado
    val size: Int
    get() = dailyForecast.size
}

// Propiedades de extensión
var TextView.textColor: Int
get() = currentTextColor
set(v) = setTextColor(v)
```

Sobrecarga de operadores

```
data class ForecastList(val city:String, val country: String,  
                        val dailyForecast: List<Forecast>) {  
    operator fun get(position: Int): Forecast =  
        dailyForecast[position]  
}
```

Operador	Función equivalente	Operador	Función equivalente
a++	a.inc()	a[i]	a.get(i)
a+b	a.plus(b)	a[i,j]	a.get(i,j)
a*b	a.times(b)	a[i]= b	a.set(i,b)

Singleton

```
object DataProviderManager {  
    fun registerDataProvider(provider: DataProvider) {  
        // ...  
    }  
  
    val allDataProviders: Collection<DataProvider>  
        get() = // ...  
}
```

```
DataProviderManager.registerDataProvider(...)
```

Companion objects

```
class MyClass {  
    companion object Factory {  
        fun create(): MyClass = MyClass()  
    }  
}  
  
val instance = MyClass.create()
```

Null safety

```
val a: Int = null // no compila
```

```
val a: Int? = null
```

```
a.toString() // no compila
```

Null safety - Llamadas seguras

```
val a: Int? = null
if (a!=null) {
    a.toString()
}
```

```
val a: Int? = null
a?.toString()
```

Asignación segura

Operador Elvis?

```
val a: Int? = null
val cadena= if (a!=null) a.toString() else cadena= ""

val a: Int? = null
val cadena= a?.toString() ?: ""
```


Lazy & lateinit

```
val text: String by lazy  
    { intent.extras.getString(NAME_KEY) }  
lateinit var kotlin: KotlinDataClass
```

Corrutinas

- Las corrutinas son hilos pero con ventajas:
- Permiten escribir tu código asíncrono en forma de secuencia
- Varias corrutinas se pueden ejecutar utilizando el mismo hilo
- Son la base para la programación concurrente

Referencias

- Referencia del lenguaje:
 - <https://kotlinlang.org/docs/reference/>
- Blog de Antonio Leiva
 - <https://devexperto.com/crea-proyecto-android-kotlin/>
- Entorno web para probar ejemplos de Kotlin:
 - <https://try.kotlinlang.org>
- Canal Antonio Leiva
 - <https://youtu.be/Vgz1mvoQFCc>
- Kotlin en Android
 - <https://developer.android.com/kotlin?hl=es>