# **Bootcamp - Android**

Roshka - Agosto 2022

# Null

El null básicamente es un valor para representar algo que NO tiene valor.

Sabemos que Kotlin es un lenguaje *null safety* como vimos la primera clase y tiene métodos que ayudan a evitar posibles errores de este tipo. Aún así podemos utilizar los valores nulos en Kotlin indicando explícitamente que lo

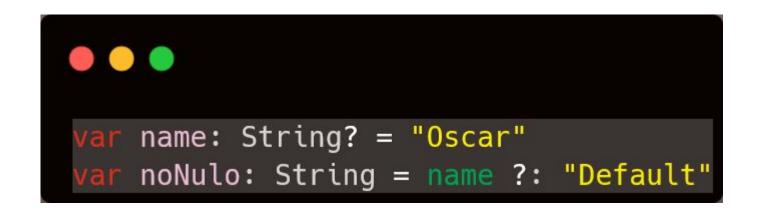
haremos.

```
var name: String? = "Oscar"

name = null
// al agregar el operador ?
// hacemos que se ejecute la propiedad
// o si se trata de un valor nulo, retorna null
print(name?.length)
```

# Null

Hay veces que necesitamos asignar una variable no nullable, con una que podría serlo, para eso utilizamos el operador Elvis **?:** poniendo una opción válida cuando la variable es *null* 



# Null

También podemos convertir una variable nullable como una no nullable con el operador !!, pero esto no siempre es recomendable porque en el caso que contenga un valor nulo, tendremos una excepción en nuestro programa (KotlinNullPointerException)

```
var number: String? = "CINCO"
print(number!!.toLowerCase())
```

### **Diccionarios**

Un diccionario es una estructura de datos para almacenar valores en pares de clave : valor, es muy utilizado ya que nos permite acceder de manera rápida, ordenada y segura a la información que buscamos.

```
var hashMap = HashMap<String, Int>()
hashMap.put("IronMan", 3000)
hashMap.put("Thor", 100)
```

#### **Diccionarios**

Podemos utilizar distintos tipos de datos en los valores y en las claves, según las necesidades que tengamos, incluso podemos guardar otros arrays dentro de un valor.

Y tenemos la opción de iterar a través de la misma.

```
println("hashMap : " + hashMap + "\n")

for(key in hashMap.keys){
   println("Elemento en la llave $key : ${hashMap[key]}")
}
```

Las clases son tipos de datos que podemos definir, es una forma de representar objetos de la vida real en código.

Nos permite mantener propiedades y funciones en un mismo lugar, mantener un código más ordenado y organizado.

La POO es un paradigma de programación que nos permite mapear objetos de la vida real en código. Todos estos objetos tienen un **estado**(propiedades) y un **comportamiento**(métodos).

Por ejemplo:



Si comparamos un objeto de Kotlin con uno del mundo real, tienen muchas similitudes.

Los objetos tienen:

- Estados
- Comportamientos

Los estados de un objeto en software son las propiedades y los comportamientos se muestran a través de métodos

Las clases son un esquema o plano de los objetos.

Así tenemos que los objetos son instancias de las clases.

Para definir una clase en Kotlin, lo hacemos de la siguiente forma:

```
class Persona {
  class Perro(nombre: String, edad: Int) {}
```

Las **propiedades** definimos como variables dentro de nuestra clase.

Y los **métodos** definimos como funciones dentro de la clase.

```
class Perro(nombre: String, edad: Int) {
    var nombre = "First property: $nombre".also(::println)
    var edad: Int = 0
    fun ladrar(){
        println("Woof!")
```

También podemos inicializar las propiedades si queremos nuestra clase.

```
class Perro(nombre: String, edad: Int) {
    var nombre = "First property: $nombre".also(::println)
    var edad: Int = 0
   init {
        this.nombre = nombre
        this.edad = edad
    fun ladrar(){
        println("Woof!")
```