INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Application Development For Mobile Devices

Kotlin

Emiliano González López

19/junio/2020

Contenido

¿Qué es Kotlin?

Kotlin es un lenguaje de programación moderno mantenido como un proyecto de código abierto por JetBrains, un líder mundial en IDE. En 2011, JetBrains estaba escribiendo principalmente código Java y buscando un nuevo lenguaje de programación para ayudar a hacerlos programadores más eficientes. En última instancia, decidieron crear un nuevo lenguaje de programación llamado Kotlin.

Con los años Kotlin ha crecido hasta convertirse en un reemplazo directo de Java, lo que permite a los desarrolladores de Java actualizar fácilmente a un lenguaje más expresivo. No sólo eso, en mayo de 2017 Google anunció que Kotlin sería un lenguaje oficialmente compatible en la plataforma Android. Con JetBrains y ahora Google apoyando el proyecto, no es de extrañar que Kotlin haya visto un aumento tan grande en popularidad.

Variables

Para crear una variable en Kotlin usamos la palabra clave var.

Además, normalmente puede omitir el argumento type. Si Kotlin puede averiguar qué tipo de algo es, no es necesario especificarlo explícitamente.

Kotlin hay dos tipos de variables: mutable e inmutable. Mutable significa que la variable se puede cambiar, e inmutable significa que NO SE PUEDE cambiar. Para crear una variable mutable usamos la palabra clave var como hemos estado haciendo. Sin embargo, para crear una variable inmutable necesitamos usar la palabra clave val.

Nulabilidad

En la mayoría de los idiomas, cuando se crea una variable pero no se establece igual a algo, obtiene un valor de NULL, que se puede considerar que no es igual a nada. Los valores NULL causan problemas en muchos programas cuando el programa intenta hacer algo con un valor NULL.

En Kotlin manejamos NULL un poco diferente. En lugar de que las variables se valores por defecto sean NULL, en realidad es un error no inicializar una variable no puede establecer una variable igual a NULL. En primer lugar, debe marcar el tipo de esa variable como que acepta valores NULL mediante la marca de interrogación:

- val x: Int = null // Error - null cannot be a value for non-null type Int
- val x: Int? = null // OK
- val y: String = null // Error - null cannot be a value for non-null type String
- val y: String? = null // OK

Desarrollo de la práctica

Primeramente, creamos un proyecto basado en kotlin, en la práctica se menciona que se se tiene que descargar un plugin, pero en las nuevas versiones de Android Studio solo hace falta seleccionar como lenguaje principal kotlin, y todo estará listo para desarrollar.

Ejercicio 1

Desarrollo

Para este ejercicio, es parecido a uno de los primeros que realizamos de practica en Android, me refiero a el ejercicio de calcular las raíces de una ecuación de segundo grado.

Comenzando, diseñando una sencilla interfaz donde hay 3 edit text y un botón, para poder calcular nuestro resultado esperado.

```
val valueA = findViewById<EditText>(R.id.valorA)
val valueB = findViewById<EditText>(R.id.valorB)
val valueC = findViewById<EditText>(R.id.ValorC)
val botonCalcular = findViewById<Button>(R.id.btnCalcular)
```

En esta primera imagen podemos ver la declaración de nuestras variables que recuperamos desde el layout, podemos observar que la declaración es un poco diferente que con java.

```
botonCalcular.setOnClickListener {
    val a = valueA.text.toString().toFloat()
    val b = valueB.text.toString().toFloat()
    val c = valueC.text.toString().toFloat()
    val aux = ((b*b)-(4*a*c))
```

Posteriormente le agregamos una escucha a nuestro botón para poder calcular al presionar nuestro botón, seguido recuperamos los valores de los editText y comenzamos los cálculos.

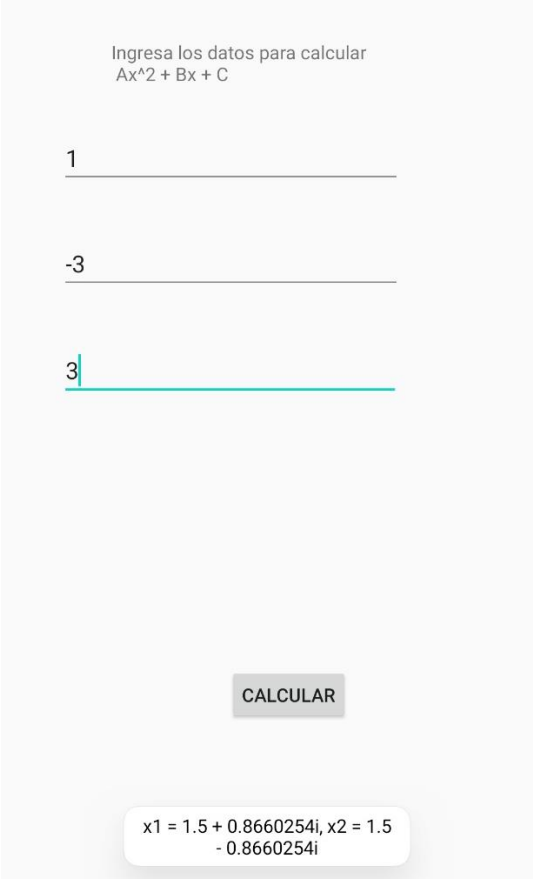
```
if (aux < 0){
    val res1 = ((-1) * b) / (2 * a)
    val res2 = sqrt(aux.absoluteValue) / (2 * a)
    Toast.makeText(this, "x1 = "+res1+" + "+ res2.toString() + "i, "+
            "x2 = "+res1+" - "+ res2.toString()+"i", Toast.LENGTH_LONG).show()
}else{
    val res1 = (((-1) * b) + sqrt(aux)) /(2 * a)
    val res2 = (((-1) * b) - sqrt(aux)) /(2 * a)
    Toast.makeText(this, "x1 = "+ res1.toString() + ", "+
            "x2 = "+res2.toString(), Toast.LENGTH_LONG).show()
}
```

Finalmente calculamos basándonos en la parte de la raíz para saber si es un resultado imaginario

Pruebas

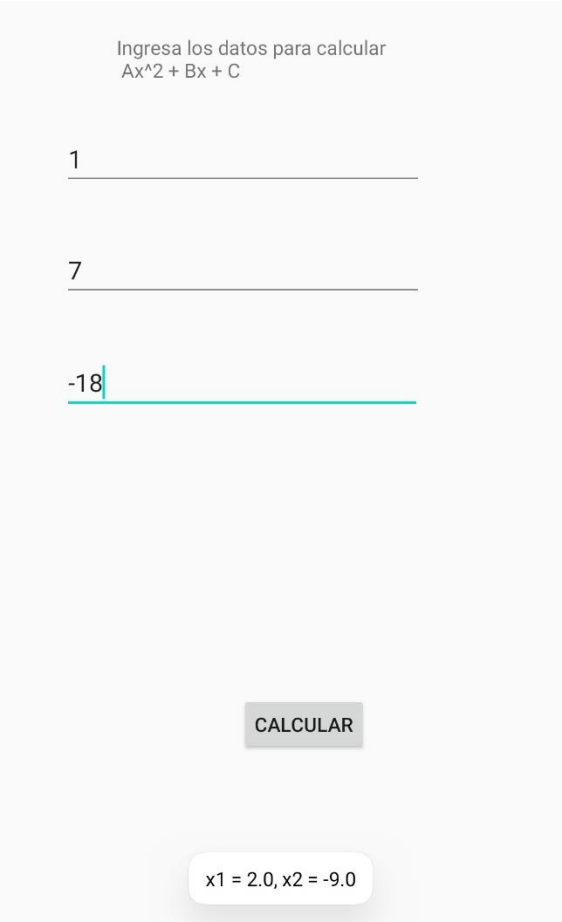
Prueba 1 con numero imaginario.

Podemos observar la pantalla de la aplicación corriendo y su comprobación de la ecuación.



Prueba 2 ejercicio sin número imaginario.

Podemos observar que los resultados no son desplegados en un textView, si no que se muestran en un mensaje Toast



Ejercicio 2

Desarrollo

Para el segundo ejercicio, decidí realizar un ejemplo para cada caso dentro de la misma aplicación y actividad.

```
var ejemplo: String? = null
txtNull.setText("Prueba de null: "+ejemplo?.length)
```

Comenzamos con la nulidad valida, creamos una variable con la nulidad habilitada y lo desplegamos en un TextView.

```
setName(ejemplo)
```

```
private fun setName (name: String?) {
    val username = name ?: "Valor por defecto"
    txtElvis.setText("El texto por defecto es: " + username)
}
```

Posteriormente creamos una función donde podemos ejecutar el operador Elvis que nos permite asignar un valor por defecto a una variable si esta es nula.

```
boton.setOnClickListener( {view -> toast("Toast lambda y extension")} )
```

```
fun AppCompatActivity.toast(msg: String){
    Toast.makeText(this, msg, Toast.LENGTH_LONG).show()
}
```

Para el ejemplo de la función lamba asignamos a la vista un mensaje toast, este mensaje toast es una extensión directa, o sobre escritura del método toast de AppCompatActivity.

```
txtSingletone.setText(kotlinSingleton.dispMsg())
```

```
object kotlinSingleton {
    var ejemplo = "Emiliano"
    fun dispMsg(): String{
        return ejemplo
    }
}
```

Para el ultimo caso, creamos un objeto singletone y desplegamos una función de ese objeto en un TextView.

Pruebas


Kotlin2
Hello World!
Prueba de null: null
El texto por defecto es: Valor por defecto
TOAST CON LAMBDA
Emiliano
Toast lambda y extension