

Universidad Tecnológica de Tijuana

Materia:

Desarrollo para dispositivos inteligentes

Grupo:

9-B

Nombre del Alumno:

Rendon Gurrola Alan Misael

Docente:

Ray Brunett Parra Galaviz

Fecha de Elaboración:

25/09/2024

Tipos de datos en Kotlin

Tipos de datos en Kotlin

En Kotlin, los tipos de datos se dividen en dos grandes categorías: **tipos primitivos** (similares a los de otros lenguajes como Java) y **tipos de referencia**.

Tipos de datos **primitivos**:

1. Números

Int: Números enteros de 32 bits. Rango: -2^{31} a $2^{31} - 1$.

Long: Números enteros de 64 bits. Rango: -2^{63} a $2^{63} - 1$.

Short: Números enteros de 16 bits. Rango: -32,768 a 32,767.

Byte: Números enteros de 8 bits. Rango: -128 a 127.

Double: Números de punto flotante de 64 bits para valores decimales.

Float: Números de punto flotante de 32 bits.

2. Caracteres y cadenas

Char: Representa un solo carácter. Se define entre comillas simples, como 'a'.

String: Cadena de texto. Se define entre comillas dobles, como "Hola".

3. Booleanos

Boolean: Solo tiene dos valores: true o false.

Tipos de datos por **referencia**:

4. Arreglos

Array: Estructura que almacena una secuencia de elementos del mismo tipo.

Ejemplo: `val arr = arrayOf(1, 2, 3)`.

5. Nulos y tipos opcionales

En **Kotlin**, los tipos pueden ser nullables añadiendo un **?** al final del tipo.

Ejemplo: **String?** es una cadena que puede ser null. Kotlin hace una diferenciación estricta entre tipos que pueden ser **null** y los que no.

6. Tipos de colecciones

List: Lista inmutable, no puede ser modificada después de su creación.

MutableList: Lista mutable, permite añadir, modificar o eliminar elementos.

Set: Conjunto inmutable, no contiene elementos duplicados.

MutableSet: Conjunto mutable.

Map: Mapa inmutable, una estructura de datos que asocia claves con valores.

MutableMap: Mapa mutable.

7. Otros tipos importantes

Any: Tipo raíz para todos los tipos no nulos en Kotlin.

Unit: Representa la ausencia de un valor útil, equivalente a void en Java. Las funciones que no devuelven nada, devuelven Unit.

Nothing: Tipo especial que indica que una función nunca retorna (por ejemplo, una función que lanza una excepción).


Resumen:

Kotlin maneja tanto tipos primitivos como tipos de referencia, pero a diferencia de otros lenguajes como Java, todos los tipos en Kotlin son objetos. Sin embargo, para mejorar el rendimiento, ciertos tipos (como Int, Boolean, Char, etc.) son optimizados internamente para comportarse como tipos primitivos en tiempo de ejecución. Los tipos de referencia, por otro lado, incluyen estructuras más complejas como String, Array, colecciones (List, Set, Map), y tipos relacionados con la nulabilidad.

¿Cómo se declaran y se usan estos tipos de datos?

1. Enteros (`Int` , `Long` , `Short` , `Byte`):


kotlin

 Copy code

```
val miEntero: Int = 10           // Declaración explícita
val otroEntero = 20              // Inferencia de tipos (Int)
val unLong: Long = 1000000000L  // Para Long se usa 'L'
val unShort: Short = 300        // Para valores pequeños se usa Short
val unByte: Byte = 100          // Byte
```

2. Números decimales (`Double` , `Float`):


kotlin

 Copy code

```
val unDouble: Double = 3.14      // Número de punto flotante de 64 bits
val unFloat: Float = 3.14F       // Para Float se usa 'F' al final
```

3. Booleanos (`Boolean`):


kotlin

 Copy code

```
val verdadero: Boolean = true    // Valor booleano verdadero
val falso = false               // Inferencia de tipos (Boolean)
```

4. Caracteres (`Char`):

kotlin

 Copy code

```
val unCaracter: Char = 'A'      // Carácter en comillas simples
```



1. Cadenas de texto (String):

kotlin

Copy code

```
val saludo: String = "Hola, Kotlin!" // Declaración explícita
val otroSaludo = "Bienvenido"        // Inferencia de tipos (String)

// Uso de variables String
println(saludo)                       // Imprime: Hola, Kotlin!
println(saludo.length)               // Acceso a métodos (imprime: 12)
```

2. Arreglos (Array):

kotlin

Copy code

```
val numeros: Array<Int> = arrayOf(1, 2, 3, 4, 5) // Array de enteros
val nombres = arrayOf("Juan", "Maria", "Pedro") // Array de cadenas (String)

// Acceso a elementos del array
println(numeros[0]) // Imprime: 1
numeros[1] = 10     // Modificar elemento del array
```

3. Listas (List, MutableList):

kotlin

Copy code

```
val listaInmutable: List<String> = listOf("Uno", "Dos", "Tres") // Lista inmutable
val listaMutable: MutableList<Int> = mutableListOf(1, 2, 3)    // Lista mutable

// Operaciones con listas
println(listaInmutable[0]) // Imprime: Uno
listaMutable.add(4)        // Añade un nuevo elemento a la lista mutable
```

5. Mapas (Map, MutableMap):

kotlin

Copy code

```
val mapaInmutable: Map<String, Int> = mapOf("Uno" to 1, "Dos" to 2) // Mapa inmutable
val mapaMutable: MutableMap<String, String> = mutableMapOf("A" to "Apple", "B" to "Banana")

// Acceder a un valor del mapa
println(mapaInmutable["Uno"]) // Imprime: 1
mapaMutable["C"] = "Cherry"   // Añade un nuevo par clave-valor
```