UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA: Tipos de Datos en Kotlin

PRESENTADO POR: Ramírez Aispuro Juan José

GRUPO: 9-B

MATERIA: Desarrollo para Dispositivos Inteligentes

PROFESOR: Ray Brunett Parra Galaviz

Tijuana, Baja California, 25 de septiembre del 2024

Tipos de Datos en Kotlin

En Kotlin, los tipos de datos son fundamentales para la estructura del lenguaje, ya que permiten almacenar y manipular diferentes tipos de valores. Kotlin es un lenguaje de programación fuertemente tipado y ofrece tanto variables mutables como inmutables, que se definen usando las palabras clave var y val, respectivamente.

1. Tipos Primitivos o Básicos

Estos tipos representan los datos más simples y están directamente soportados por el lenguaje de programación. En Kotlin, aunque todos los tipos son técnicamente objetos, los tipos primitivos se optimizan a nivel de JVM para mejorar el rendimiento.

- **Números enteros:**
    - Byte: 8 bits, representa valores de -128 a 127.
    - Short: 16 bits, valores de -32,768 a 32,767.
    - Int: 32 bits, valores de -2^31 a 2^31-1.
    - Long: 64 bits, valores de -2^63 a 2^63-1.
- **Números decimales:**
    - Float: 32 bits, para números de coma flotante.
    - Double: 64 bits, mayor precisión que Float.
- **Carácter (Char):**
    - 16 bits, representa un solo carácter Unicode.

- **Booleano (Boolean):**
    - Tiene dos valores posibles: true o false.

Estos tipos se manejan como objetos en Kotlin, aunque a nivel de JVM se optimizan como primitivos para mejorar el rendimiento.

2. Tipos Referenciados o de Referencia

Los tipos de referencia en Kotlin son tipos de datos más complejos, representados como objetos o clases. Estos incluyen estructuras de datos que no son primitivos y permiten trabajar con entidades más sofisticadas.

- **String (Cadena de caracteres):**
    - Un objeto que representa una secuencia de caracteres.
- **Arrays:**
    - Colecciones de datos del mismo tipo. Se definen usando Array<T>.
- **Colecciones (Collections):**
    - Estructuras como listas (List), conjuntos (Set), y mapas (Map), que permiten agrupar y organizar elementos.
- **Clases:**
    - Definen tipos personalizados y estructuras complejas de datos. Una clase puede tener propiedades, métodos y comportamientos específicos.
- **Tipos Anulables (Nullable Types):**
    - Se manejan con ?, permitiendo que una variable de referencia pueda tener un valor null, lo que previene errores de puntero nulo.

En resumen, los **tipos primitivos** son optimizados por la JVM para mejorar el rendimiento, mientras que los **tipos referenciados** incluyen objetos y estructuras más complejas que permiten manejar datos de forma más flexible y estructurada.