**TRABAJO DE INVESTIGACION KOTILO**

****

**INTRUCTOR (A)**: Zulema Leon

**APRENDIZ**: Sebastian Cotazo Guacas

SERVICIO NACIONAL DE APRENDIZAJE – **SENA**

**TECNOLOGO ADSO**

**FICHA: 2923604**

POPAYAN – CAUCA

2024

DESARROLLO ACTIVIDADES Y ESTRATEGIAS DE APRENDIZAJE

¿Qué es Kotlin y cuáles son sus principales características?

## **-¿Qué es Kotlin?**

## **Kotlin es un lenguaje de programación** moderno y multiplataforma desarrollado por JetBrains que fue lanzado en 2011. Aunque se diseñó para funcionar sobre la máquina virtual de Java (JVM), Kotlin ya soporta plataformas como Android, iOS y la web.

Se utiliza mucho **Kotlin para Android** debido a su interoperabilidad con Java, su concisión y sus excelentes características de seguridad​.

-**Características que lo diferencian de otros lenguajes de programación**

Este lenguaje de programación se distingue de otros por varias características de Kotlin que deberían tener muy en cuenta a la hora de elegirlo.

* Es totalmente***compatible con Java****, lo que permite usar sus bibliotecas y frameworks.*
* *Su sintaxis permite escribir menos código, haciéndolo****más legible y mantenible****.*
* *Incorpora mecanismos para****evitar errores comunes****de punteros nulos mediante su sistema de tipos*nullable*y*non-nullable*.*
* *Permite crear aplicaciones que****funcionan en diferentes plataformas****.*
* *Aún****programación funcional y orientada a objetos***​​.

¿Cuáles son las ventajas de usar Kotlin en comparación con Java?

Kotlin:

Kotlin es un lenguaje de programación moderno, pero ya maduro, de código abierto. Es conciso, seguro, interoperable con Java y otros lenguajes, y ofrece múltiples maneras de reutilizar código entre múltiples plataformas para una programación productiva. Tiene tipado estático y admite tanto la programación orientada a objetos como la funcional. Su sintaxis y características son similares a las de otros lenguajes como C#, Java y Scala.

**Java:**

Java es un lenguaje de programación de código abierto, de propósito general y orientado a objetos. Al ser un lenguaje multiplataforma, Java puede ejecutarse en prácticamente cualquier dispositivo, sistema operativo o servidor. Al estar compilado en código de bytes, puede ejecutarse en cualquier Máquina Virtual Java (JVM).

Java tiene tipado estático, lo que significa que realiza la comprobación de tipos en tiempo de compilación. Su sintaxis es similar a la de C y C++, aunque ofrece menos funciones de bajo nivel.

Como se usan:

-Kotlin

**Desarrollo de Android:** preferido para el desarrollo de aplicaciones Android modernas debido a su sintaxis concisa y funciones avanzadas, y al soporte oficial de Google.

**Desarrollo del lado del servidor:** cada vez más utilizado para aplicaciones del lado del servidor gracias a sus características modernas y su interoperabilidad perfecta con Java.

**Proyectos multiplataforma:** adecuado para proyectos que apuntan a múltiples plataformas (por ejemplo, JVM, JavaScript, WASM, nativo) utilizando [Kotlin](https://bugfender.com/blog/how-to-debug-a-kotlin-multiplatform-mobile-app-from-scratch/" \t "_blank) Multiplatform.

-Java

**Aplicaciones empresariales:** se utilizan ampliamente para crear aplicaciones del lado del servidor robustas y escalables en grandes organizaciones.

**Desarrollo de Android:** lenguaje preferido desde hace mucho tiempo para desarrollar aplicaciones de Android.

**Desarrollo web:** se utiliza comúnmente para crear aplicaciones web dinámicas utilizando marcos como Spring.

**Big Data:** Se utiliza en tecnologías y plataformas de big data como Apache Hadoop y Apache Spark.

Explica la diferencia entre val y var. Proporciona un ejemplo de uso.

En Kotlin se utiliza ***val*** y ***var*** para conseguir que nuestras variables **sean de solo lectura o**

**mutables** respectivamente.

| **Característica** | **val (Inmutable)** | **var (Mutable)** |
| --- | --- | --- |
| **Mutabilidad** | No se puede cambiar una vez inicializada. Es inmutable. | Se puede cambiar su valor después de la inicialización. |
| **Uso recomendado** | Ideal para valores que no cambiarán, como constantes. | Útil para valores que necesitas modificar. |
| **Ejemplo** | val edad = 25 | var nombre = "Juan" |
| **Errores comunes** | Intentar reasignar provoca error de compilación. | Es susceptible a cambios no deseados si no se controla bien. |

Ejemplos de códigos:

Con val no se puede cambiar

val pi = 3.14 // El valor de pi no cambiará

println(pi)

// Si intentas hacer: pi = 3.15 → Esto dará un error.

Con var se puede cambiar

var contador = 5 // El valor puede cambiar

println(contador)

contador = 10 // Ahora contador vale 10

println(contador)

Enumera y describe los principales tipos de datos en Kotlin.

1. Tipo numéricos.

**Int:** Enteros, como 1, 23, o -456.

**Double:** Números decimales de doble precisión, como 3.14 o -0.99.

**Float:** Números decimales de precisión simple, como 2.5f (nota el sufijo f).

**Long:** Enteros largos, para valores grandes, como 123456789L (nota el sufijo L).

**Short:** Enteros pequeños, como 32767.

**Byte:** Enteros muy pequeños, como 127.

1. Booleanos

**Boolean:** Representa valores lógicos, como true o false.

1. Texto

**Char:** Representa un solo carácter, como 'A' o '7'.

**String:** Cadena de texto, como "Hola, Kotlin".

1. Otros

**Array:** Conjunto de elementos del mismo tipo. Ejemplo:

Val números = arrayof(1,2,3).

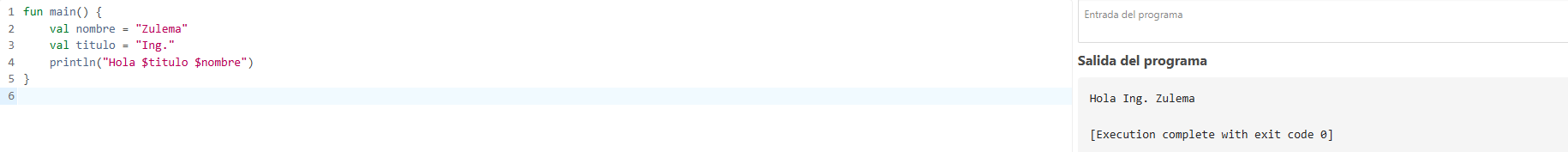
**List:** Similar a un array, pero más flexible. Puede ser mutable o inmutable.

**Any:** Tipo más general; puede contener cualquier cosa.

**Unit:** Similar a void en otros lenguajes, usado en funciones que no devuelven un valor.

**Null:** Representa un valor nulo. En Kotlin, necesitas usar tipos anulables con ? (por ejemplo, String?).

Escribe un programa en Kotlin que declare e inicialice variables de diferentes tipos de datos.



¿Cómo funcionan las estructuras de control if, when y for en Kotlin? Explica cada una con un ejemplo.

### if

if se usa para tomar decisiones basadas en una condición. Es similar a lo que ves en otros lenguajes y puede funcionar como una expresión (retornando un valor).

fun main() {

val numero = 5

if (numero > 0) {

println("El número es positivo")

} else {

println("El número es negativo o cero")

}

}

### when

when es una alternativa más poderosa y limpia que if-else if. Se utiliza para manejar múltiples condiciones de una forma más clara.

**Ejemplo básico:**

fun main() {

val dia = 3

when (dia) {

1 -> println("Lunes")

2 -> println("Martes")

3 -> println("Miércoles")

else -> println("Día desconocido")

}

}

### for

for se utiliza para iterar sobre elementos de una colección o rango.

**Ejemplo con rangos:**

fun main() {

for (i in 1..5) {

println("Número: $i")

}

}

Escribe un programa en Kotlin que solicite al usuario un número y determine si es par o impar usando when.

¿Cómo se declara una función en Kotlin? Explica la diferencia entre una función con retorno y una sin retorno.

En Kotlin, una función se declara utilizando la palabra clave fun, seguida del nombre de la función, los parámetros (si los hay) y el tipo de retorno (si aplica).

fun nombreDeLaFuncion(parametros: Tipo): TipoDeRetorno {

}

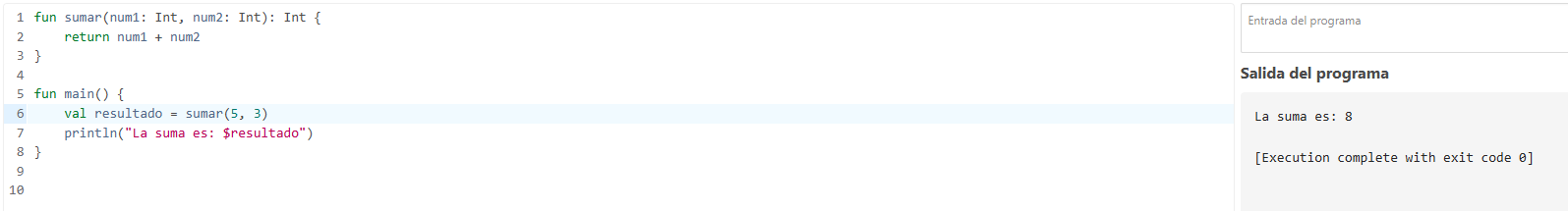
Ejemplo sin parámetros y sin retorno

fun saludar() {

println("¡Hola, bienvenido a Kotlin!")

}

Escribe una función en Kotlin que reciba dos números como parámetros y devuelva la suma de ambos.



### ¿Qué es una clase en Kotlin?

Una clase en Kotlin define una estructura que describe las características (propiedades) y comportamientos (métodos) de los objetos que se crean a partir de ella. Esencialmente, una clase es un "molde" para crear objetos.

### ¿Cómo se define una clase en Kotlin?

Para definir una clase en Kotlin, usamos la palabra clave class

Implementa un programa en Kotlin que simule el sistema de registro de estudiantes en un curso. Debe incluir:

* Una clase Estudiante con atributos nombre, código y edad.
* Un método que imprima la información del estudiante.
* Una lista de estudiantes y una función que agregue nuevos estudiantes a la lista.