O que é?

Linguagem de programação **multiplataforma** que compila para: JVM; Javascript; Código Nativo.

- 2016: Kotlin 1.0 lançado
- 2017: Incluída no Android Studio 3.0
- 2019: Linguagem preferencial para Android.

Características

Linguagem **orientada a objetos** que possui **interoperabilidade** com Java. Na sintaxe o ponto e vírgula é **opcional** e os tipos de dados vem depois do nome da variável e suas variáveis podem ser somente leitura ou leitura e escrita.

Comentários

Em **bloco**: Usado no início do arquivo para *Copyright* e não deve ser usado para comentário de código.

Em **linha**: Usado para descrever lógica no código. Seu atalho é ***Ctrl + /***.

Em **documentação**: Usado para gerar documentação e para descrever classes, métodos e atributos.

Variáveis

Podem ser declaradas usando **val** e **var,** e a interferência de tipo é realizada **automaticamente** na declaração além de que **tipos** pode, ser informados na declaração.

Tipos de Dados

Tipo	Tipo	Tamanho	Valores
Inteiro	Byte	1 byte	-128 127
	Short	2 bytes	32.768
	Int	4 bytes	2.147.48...
	Long	8 bytes	
Real	Float	4 bytes	3.40E+38
	Double	8 bytes	1.80E+308

- **Booleano e Caractere**

Boolean e Char

Conversão de Tipos

A conversão precisa ser **explícita** e é feita através de **funções de suporte**.

Operadores:

Operador	Descrição
%	Resto
!=	Diferença
||	Ou lógico
&&	E lógico
in	Valor pertence à coleção

Entrada e Saída

```
print(“Digite um frase: ”)

val texto = readLine()
```

Estruturas de Decisão

IF, IF..ELSE, WHEN

```
if (numero > 0) {

println("$numero é positivo")

} else if (numero < 0) {

println("$numero é negativo")

}

when (operador) {

"+" -> println("$a + $b = ${a + b}")

"/" -> println("$a / $b = ${a / b}")

else -> println("Operador inválido")

}
```

- ## Funções

```
fun imprimeIntervalo(min: Int = 0, max: Int = 10) { for (i in min#..max) { print("$i, ") } println() }

!// Listas imutáveis val cores: List<String> = listOf("azul", "verde", "laranja")

!// Acessando um elemento print(cores[0]) !

// Percorrendo uma lista for (cor in cores) { print("${cor}, ") } !

// Tamanho da lista print(cores.size)

!// Listas mutáveis val frutas: MutableList<String> = mutableListOf("banana", "uva", "maçã")

!// Adicionando um elemento frutas.add("laranja")

!// Ordenado uma lista frutas.sort()

!// Removendo um elemento frutas.removeAt(0)
frutas.remove("uva")
```

- ## Paradigmas de Programação

Visão do programador em relação aos programas (Estrutura e Execução). Os principais paradigmas são o **Imperativo** (estruturado, procedural, orentado a objetos) e o **Declarativo** (funcionalista, lógico).

- ## Orientação a Objetos

Descreve o sistema com **elementos do mundo real** e considera que todos os componentes são **objetos**.

Objetos possuem uma **estrutura** e desempenham **ações**, além de serem **classificados** de acordo com suas **características**.

Suas principais vantagens são: **Abstração; Modularização; Extensibilidade** e **Reaproveitamento de Código**.

- ## Objetos e Classes

O universo é formado por **objetos** e cada um possui (e são classificados de acordo) **características** e **funções**.

Uma classe é um **agrupamento** de objetos semelhantes entre si

Classes possuem **Atributos** e **Métodos** e os relacionamentos entre as classes podem ser: **Associação; Agregação; Composição** e **Herança.**

- **Encapsulamento**

Todo objeto é responsável pelos seus atributos e o objetivo é **esconder** do mundo externo a estrutura interna dos objetos e detalhes da implementação. As interações são através de uma **interface pública**. Seus tipos são **Public**, **Private** e **Protected**

- **Herança**

Estabelece a relação **“É UM”** entre duas classes e permite **abstração** e **reaproveitamento de código.**

- **Polimorfismo**

Um objeto pode assumir diferentes formas e os tipos de polimorfismo são: **Inclusão**; **paramétrico**; **sobrescrita** e **sobrecarga.**