Desenvolvimento Mobile com Kotlin no Replit

Fundamentos de Kotlin

Sintaxe Básica

Kotlin é uma linguagem moderna, concisa e segura, desenvolvida pela JetBrains e adotada oficialmente pelo Google para o desenvolvimento Android. Vamos começar com os conceitos básicos da sintaxe.

Pontos importantes:

- Kotlin não exige ponto e vírgula no final das linhas
- A inferência de tipo permite declarar variáveis sem especificar o tipo explicitamente
- val para valores imutáveis (similar a final em Java)
- var para variáveis mutáveis

```
// Exemplo 1: Sintaxe Básica de Kotlin
// Este arquivo demonstra os conceitos básicos da sintaxe de Kotlin
// Função principal (ponto de entrada do programa)
fun main() {
  // Imprimindo texto no console
  println("Olá, Mundo! Bem-vindo ao Kotlin!")

  // Variáveis em Kotlin
  // val - para valores imutáveis
  val nome = "Maria"

  // var - para variáveis mutáveis
  var idade = 25
  idade = 26 // Isso é permitido
}
```

Estruturas de Controle de Fluxo

if/else como expressão

Kotlin permite usar if/else como expressão para atribuir valores a variáveis, tornando o código mais conciso.

```
val tipoNumero = if (numero % 2 == 0) "par" else "ímpar"
println("$numero é $tipoNumero")
```

when (similar ao switch)

Uma versão mais poderosa do switch de outras linguagens, permitindo múltiplas opções por branch.

```
val tipoDia = when (diaDaSemana) {
  1, 7 -> "Fim de semana"
  in 2..6 -> "Dia útil"
  else -> "Inválido"
}
```

Loops for

Usados para iterar sobre coleções, ranges ou arrays com sintaxe simples e intuitiva.

```
for (i in 1..5) { // Range inclusivo
  print("$i ")
}
```

while e do-while

Funcionam como em outras linguagens, permitindo repetições condicionais.

```
var contador = 0
while (contador < 5) {
  print("$contador ")
  contador++
}
```

Funções em Kotlin

Funções Simples

Funções são declaradas com a palavra-chave fun e podem ter parâmetros e retornos explícitos.

```
// Função com parâmetros e retorno explícito
fun somar(a: Int, b: Int): Int {
  return a + b
}

// Função com corpo de expressão
fun multiplicar(a: Int, b: Int) = a * b
```

Parâmetros com Valores Padrão

Kotlin permite definir valores padrão para parâmetros, tornando-os opcionais na chamada da função.

```
fun cumprimentar(nome: String, saudacao: String = "Olá") {
  println("$saudacao, $nome!")
}

// Chamada usando valor padrão
cumprimentar("Ana") // Usa o valor padrão "Olá"
```

Argumentos Nomeados

Permitem chamar funções com parâmetros fora de ordem, melhorando a legibilidade.

```
fun detalhesPessoa(nome: String, idade: Int, cidade: String) {
  println("Nome: $nome, Idade: $idade, Cidade: $cidade")
}

// Chamada com argumentos nomeados
detalhesPessoa(idade = 22, cidade = "São Paulo", nome = "Daniel")
```

Coleções em Kotlin

Listas (List)

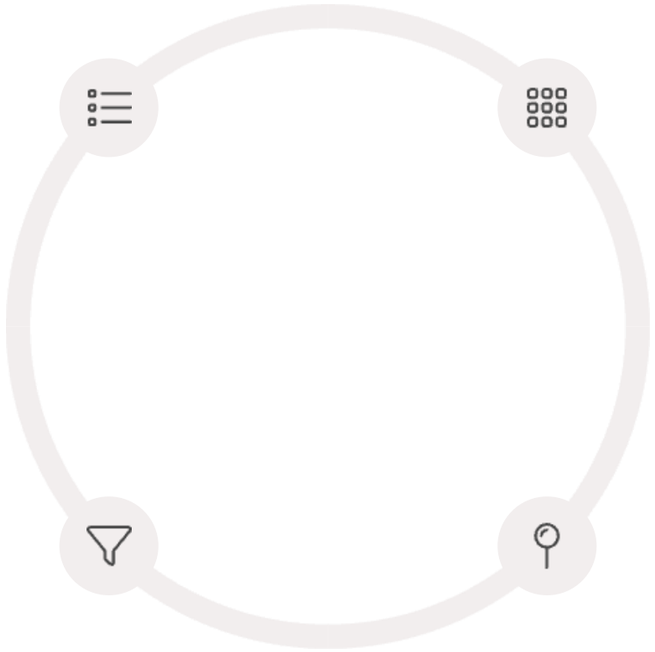
Coleções ordenadas que podem conter elementos duplicados.

- listOf() - cria lista imutável
- mutableListOf() - cria lista mutável
- Métodos: add(), remove(), removeAt()

Operações Funcionais

Manipulação de coleções com funções de alta ordem.

- filter{} - filtra elementos
- map{} - transforma elementos
- sorted() - ordena elementos



Conjuntos (Set)

Coleções que não permitem elementos duplicados.

- setOf() - cria conjunto imutável
- mutableSetOf() - cria conjunto mutável
- Métodos: add(), remove()

Mapas (Map)

Coleções de pares chave-valor.

- mapOf() - cria mapa imutável
- mutableMapOf() - cria mapa mutável
- Acesso: mapa["chave"]

Classes e Objetos

Definição de Classes

Classes são declaradas com a palavra-chave class e podem ter propriedades e métodos.

```
class Pessoa(val nome: String, var idade: Int) {
  var cidade: String = "Desconhecida"

  fun apresentar() {
    println("Olá, meu nome é $nome, tenho $idade anos.")
  }
}
```

Herança

Classes podem herdar de outras classes usando o operador : (dois pontos).

```
open class Animal(val nome: String) {
  open fun fazerSom() {
    println("O animal faz um som genérico.")
  }
}

class Cachorro(nome: String, val raca: String) : Animal(nome) {
  override fun fazerSom() {
    println("$nome, o cachorro da raça $raca, diz: Au au!")
  }
}
```

Interfaces e Classes Abstratas

Interfaces

Definem um contrato que as classes devem seguir. Em Kotlin, interfaces podem conter declarações de métodos abstratos e métodos com implementação padrão.

```
interface Clicavel {
  fun onClick() // Método abstrato (sem implementação)
  fun onLongClick() { // Método com implementação padrão
    println("Clique longo detectado (padrão)")
  }
}

class Botao(val id: String) : Clicavel {
  override fun onClick() {
    println("Botão '$id' foi clicado!")
  }

  override fun onLongClick() {
    println("Botão '$id' recebeu um clique longo!")
  }
}
```

Classes Abstratas

Podem conter métodos abstratos e não abstratos, além de propriedades. Não podem ser instanciadas diretamente, servem como base para outras classes.

```
abstract class FormaGeometrica(val nome: String) {
  // Propriedade comum
  var cor: String = "Branco"

  // Método abstrato (deve ser implementado pelas subclasses)
  abstract fun calcularArea(): Double

  // Método não abstrato (com implementação)
  fun descrever() {
    println("Esta é uma forma geométrica chamada $nome com cor $cor.")
  }
}

class Retangulo(nome: String, val largura: Double, val altura: Double) :
FormaGeometrica(nome) {
  override fun calcularArea(): Double {
    return largura * altura
  }
}
```

https://replit.com/

EXERCÍCIO PARA ENTREGA

- Criar um mutable list com NOME, IDADE E TELEFONE
- Adicione uma função a seu código chamada adicionarPessoa()
- Adicione uma função a seu código chamada listarPessoa() que imprima todos os cadastrados
- No main() já adicione pelo menos 3 pessoa a sua mutablelist.