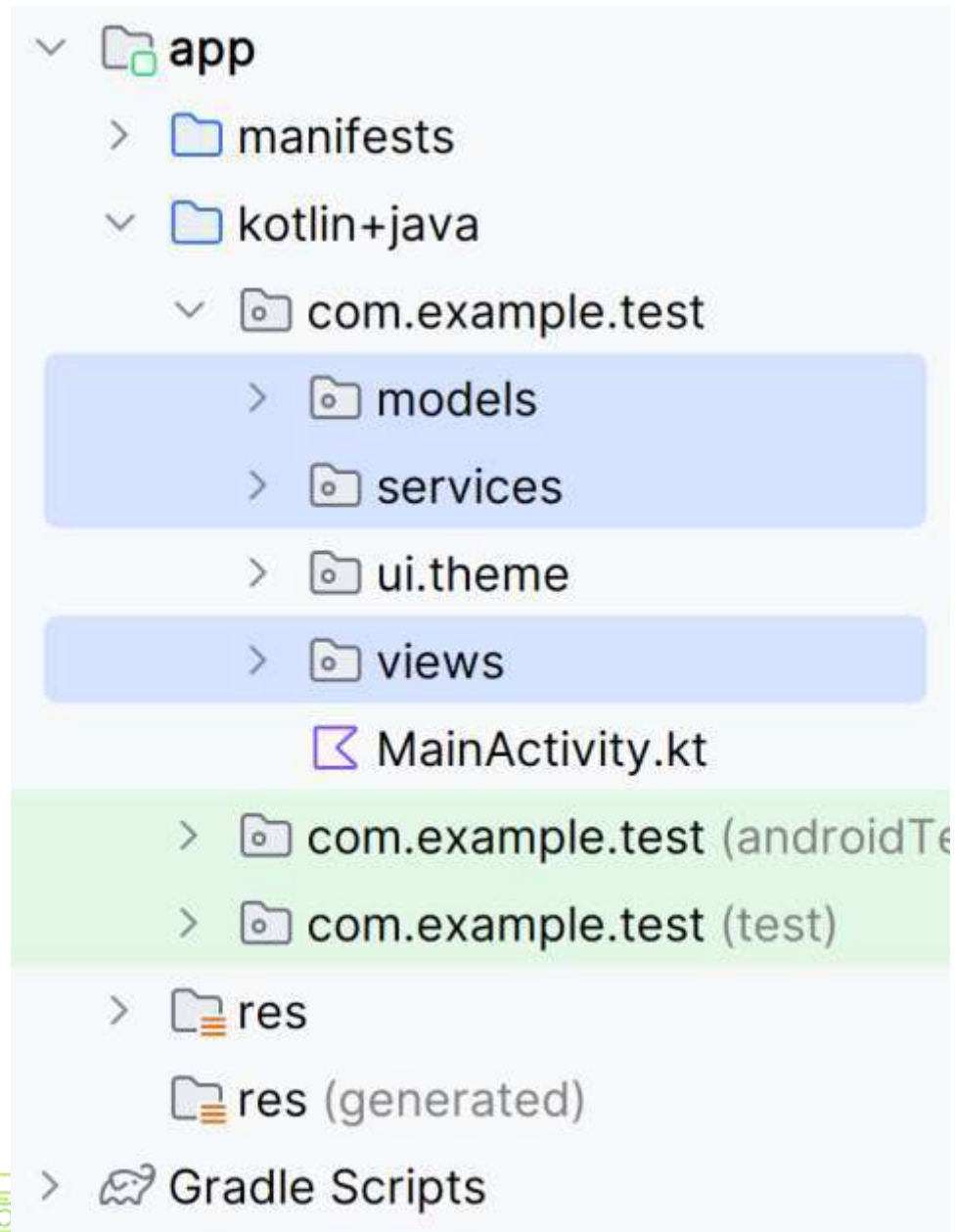




Checar o Android Studio e criar pastas.



Controle de Fluxo e Classes com Interface Jetpack Compose

data class → Classe com dados simples (modelo)

Use quando você só precisa guardar e organizar dados.

```
data class Pessoa(val nome: String, val idade: Int)
```

Métodos dentro da classe = dados com comportamento

```
data class Pessoa(val nome: String, val idade: Int) {  
    fun saudacao(): String {  
        return "Olá, meu nome é $nome e tenho $idade anos."  
    }  
}
```



Renderizando Dados de Classes

Usando um objeto dentro de um Composable:

O Compose lê o objeto e mostra seus dados na tela. Você pode passar objetos para componentes do mesmo jeito que passa variáveis.

```
@Composable
fun ExibirPessoa(pessoa: Pessoa) {
    Column {
        Text("Nome: ${pessoa.nome}")
        Text("Idade: ${pessoa.idade}")
    }
}
```

```
@Composable
fun SaudacaoDaPessoa(pessoa: Pessoa) {
    Text(pessoa.saudacao())
}
```



Relembrando - tipos

```
val idade: Int = 25
val anoAtual: Short = 2023
val distancia: Long = 15000000000L // Nota: O 'L' indica que é um Long
val altura: Float = 1.75f // Nota: O 'f' indica que é um Float
val peso: Double = 68.5
val inicial: Char = 'A'
val nome: String = "Carlos Silva"
val estaAtivo: Boolean = true
val numeros: Array<Int> = arrayOf(1, 2, 3, 4, 5)
val listaFrutas: List<String> = listOf("Maçã", "Banana", "Laranja")
val conjuntoNumeros: Set<Int> = setOf(1, 2, 3, 2) // O 2 duplicado será ignorado
val mapaCapitais: Map<String, String> = mapOf("Brasil" to "Brasília", "França" to "Paris")
var endereco: String? = null
```



Para pensar – Any, Unit e Nothing / Data Classes

Any: supertipo de todos os tipo não nulos.


Unit: indica que uma função não retorna valor significativo (Java=void)

Nothing: indica que uma função não retorna normalmente

(por exemplo, lança uma exceção)

```
fun imprimeMensagem(): Unit {  
    println("Olá, mundo!")  
}  
  
fun lancaExcecao(): Nothing {  
    throw Exception("Ocorreu um erro")  
}
```

Classes usadas para armazenar dados, automaticamente geram métodos como **equals()**, **hashCode()**, **toString()**



```
data class Pessoa(val nome: String, val idade: Int)  
  
val pessoa = Pessoa("Mariana", 30)
```

Para pensar – Funções Lambda, Estruturas de decisão e repetição

Funções Lambda: Funções que podem ser tratadas como valores, atribuídas a variáveis e passadas como parâmetros.

```
val saudacao: (String) -> String = { nome -> "Olá, $nome!" }  
  
println(saudacao("Pedro")) // Saída: Olá, Pedro!
```

Estruturas de Decisão: If-else / When

```
if (idade >= 18) {  
    println("Maior de idade")  
} else {  
    println("Menor de idade")  
}
```

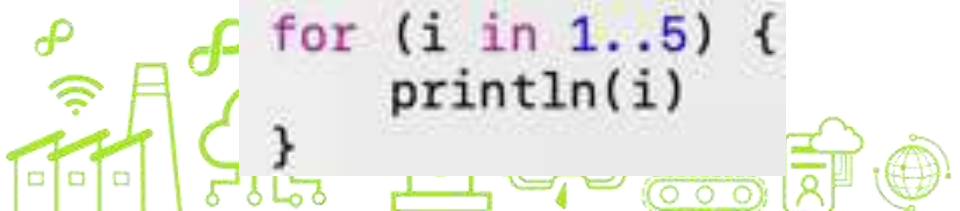
```
when (diaSemana) {  
    1 -> println("Domingo")  
    2 -> println("Segunda-feira")  
    else -> println("Outro dia")  
}
```

Estruturas de Repetição: For / While / Do-While

```
for (i in 1..5) {  
    println(i)  
}
```

```
var i = 0  
while (i < 5) {  
    println(i)  
    i++  
}
```

```
do {  
    println(i)  
    i++  
} while (i < 5)
```



Mão na Massa

Criar classe Pessoa com nome, idade, apresentar() arquivo separado

- Criar UI com PessoaCard() (arquivo separado)
 - usar when para mostrar faixa etária ("criança", "adulto", "idoso")
- Exibir lista de pessoas com idade e se podem votar

Models >

ClassPessoa.kt

```
package com.example.test

data class Pessoa(val nome: String, val idade: Int) {
    fun apresentar() = "Olá, meu nome é $nome e tenho $idade anos."
    fun podeVotar() = if (idade >= 16) "Pode votar" else "Não pode votar"
    fun faixaEtaria() = when {
        idade >= 60 -> "Idoso(a)"
        idade >= 18 -> "Adulto(a)"
        idade >= 12 -> "Jovem"
        else -> "Criança"
    }
}
```



Mão na Massa

Criar classe Pessoa com nome, idade, apresentar() arquivo separado

- Criar UI com PessoaCard() (arquivo separado)
 - usar when para mostrar faixa etária ("criança", "adulto", "idoso")
- Exibir lista de pessoas com idade e se podem votar

```
package com.example.test

import androidx.compose.foundation.border
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
```

```
@Composable
fun PessoaCard(pessoa: Pessoa) {
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .padding(16.dp)
            .border(1.dp, Color.Gray, RoundedCornerShape(8.dp))
            .padding(16.dp)
    ) {
        Text(pessoa.apresentar(), fontWeight = FontWeight.Bold)
        Spacer(modifier = Modifier.height(4.dp))
        Text(pessoa.podeVotar())
        Text(pessoa.faixaEtaria())
    }
}
```

Mão na Massa

Criar classe Pessoa com nome, idade, apresentar() arquivo separado

- Criar UI com PessoaCard() (arquivo separado)
 - usar when para mostrar faixa etária ("criança", "adulto", "idoso")
- **Exibir lista de pessoas com idade e se podem votar (NO MainActivity)**

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    PessoaCard(Pessoa("João", 17))
}
```

Extra: Que tal adicionar um campo de **profissão** ou **cidade**?



Notas - Modifier

Modifier é um objeto que define como um elemento da interface do usuário (UI) deve ser exibido, incluindo aspectos como **layout, tamanho, aparência e comportamento**.

Eles são utilizados na linguagem Jetpack Compose, que permite a criação de interfaces de usuário declarativas e flexíveis.

Os modifiers são aplicados a **@composables**, que são funções que retornam elementos de UI.

@Preview para testes visuais

```
@Composable
fun PessoaCard(pessoa: Pessoa) {
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .padding(16.dp)
            .border(1.dp, Color.Gray, RoundedCornerShape(8.dp))
            .padding(16.dp)
    ) {
        Text(pessoa.apresentar(), fontWeight = FontWeight.Bold)
        Spacer(modifier = Modifier.height(4.dp))
        Text(pessoa.podeVotar())
        Text(pessoa.faixaEtaria())
    }
}
```



Obrigado!



    @fpftech.educacional