

DESAFÍO 2 - PRÁTICA KOTLIN

ACTIVITY INICIAL - ESCOLHA UMA DAS OPÇÕES

- São 5 aplicações demos diferentes
- Cada uma ilustra a utilização de uma ou mais das features solicitadas no desafio



COROUTINE

- Uma **Animation** de rotação é exibida na tela.
- Ao clicar no **Button** “Rodar em uma Corotina”, uma **List** com 100.000 elementos é criada e não há impacto na animação.
- Ao clicar no **Button** “Rodar na Thread Gráfica”, a mesma **List** é criada, porém há um pequeno travamento na animação



SCOPED FUNCTIONS

- Após apertar o botão OK, uma instância da classe **Person** é criada com os dados entrados nos **EditText**.
- Abaixo do botão, um **TextView** tem seu conteúdo modificado pelo usuário das scoped function **apply**, **run**, **with**, **let** e also sobre a instância de **Person** e um **StringBuilder**.



EXTENTIONS E DELEGATES

- Um novo atributo do tipo booleano (**isUnderage**) é adicionado via extention na classe **Person**. O comportamento deste atributo é delegado a classe **UnderAgeCheck**
- Se $\text{Age} < 18$, **isUnderage** é **true**; caso contrário **false**.
- Preenchendo o formulário e clicando no **Button** OK, uma instância de **Person** é criada e através do atributo **isUnderage**, um **Toast** informativo diz se a pessoa é menor de idade ou não.

SimpleAppKotlin

A classe Person será extendida com um novo atributo 'isUnderAge', cujo valor será delegado a uma função que verifica o atributo age.

Entre com os dados abaixo e clique em OK. Uma mensagem TOAST apropriada será exibida dependendo da idade.

Nome: caio

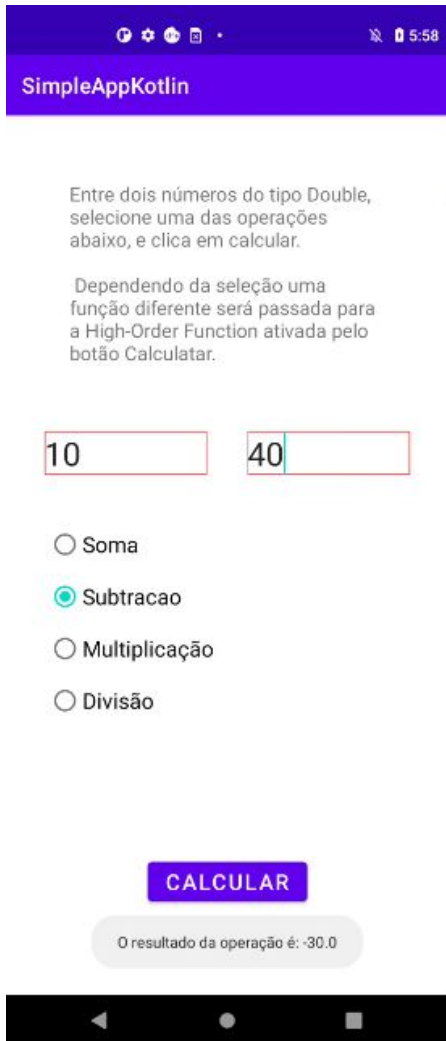
Sobrenome: viel

Idade: 30

OK

HIGH-ORDER FUNCTIONS

- Após entrar dois valores número e clicar no **Button** Calcular, a high-order function **performOperation** é chamada, realizando a operação e exibindo um **Toast** na tela com o resultado.
- **performOperation** recebe como argumento dois valores e a função que ela chama dentro dela com os valores recebidos.
- A função passada como parâmetro é do tipo (Double, Double) -> Double, e é selecionado dependendo **RadioButton** selecionado.
- Foi adicionado um novo método (extensions) no **RadioGroup** para ajudar a encontrar a opção selecionada.



GENERICIS

- A aplicação simula um console
- Foi criado um método genérico screenlog para printar informações neste console

```
fun <T> screenlog(obj : T) {  
    val timeStamp: String = SimpleDateFormat( pattern: "yyyyMMdd HH:mm:ss").format(Date())  
    var sb = StringBuilder().apply { this: StringBuilder  
        append(console.text.toString())  
        append("\n")  
        append("$timeStamp - ${obj.toString()}")  
        append("\n")  
    }  
    console.text = sb.toString()  
}
```

- Diversos tipos de objetos são então passados como parâmetro para a função genérica

SimpleAppKotlin

—SCREEN CONSOLE - Generics Test—

20210130_060436 - Vamos começar colocando strings no console...

20210130_060436 - String de teste

20210130_060436 - Agora vamos colocar um número inteiro

20210130_060436 - 1

20210130_060436 - Agora vamos colocar um número de ponto flutuante

20210130_060436 - 1.0

20210130_060436 - Agora vamos colocar o próprio TextView

20210130_060436 - com.google.android.material.textview.MaterialTextView(5703590 V.ED.....ID 0,0-0,0 #7f0800c7 app:id/!bl_console)

20210130_060436 - Agora vamos colocar a própria activity

20210130_060436 - com.example.simpleappkotlin.GenericActivity@1ea3063

20210130_060436 - Agora vamos colocar uma instância de Person

20210130_060436 - com.example.simpleappkotlin.Person@e78de89