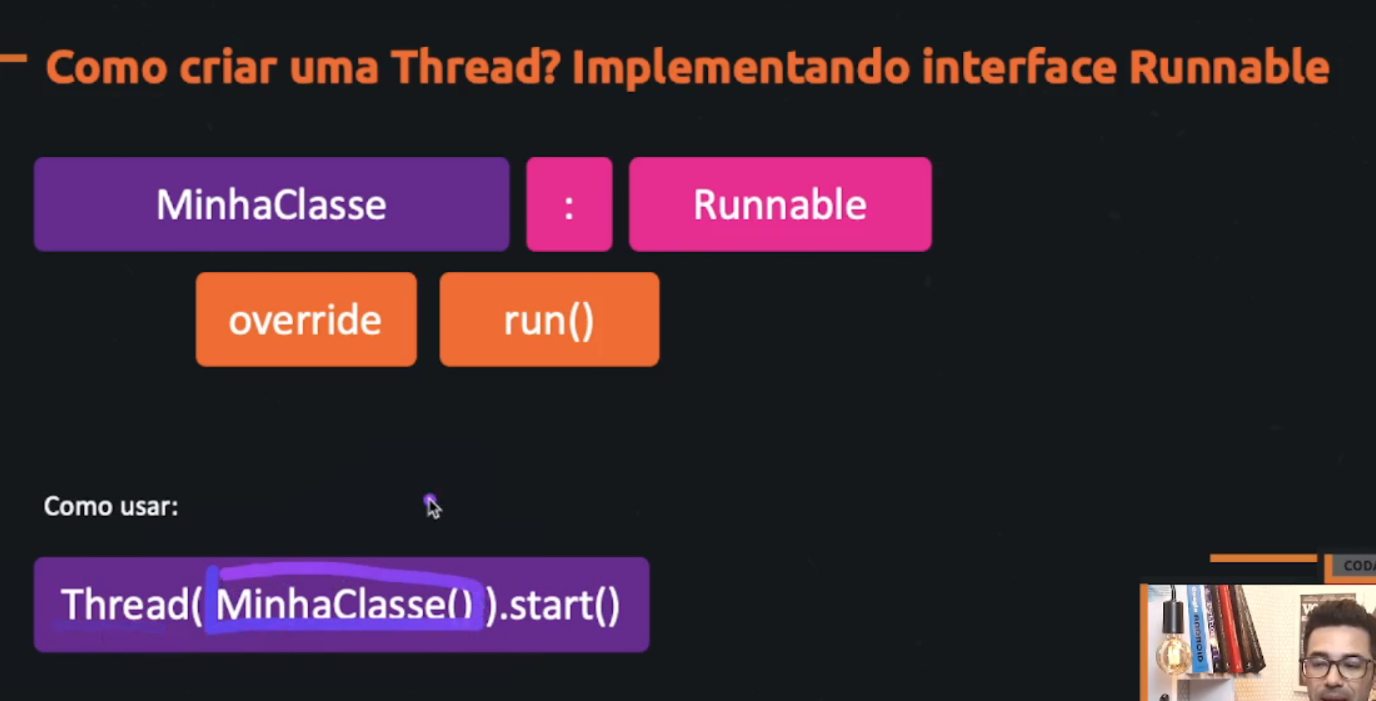
# Opção 1 de criar a thread



# Opção 2 de criar a thread

Repara a diferença



Agora não vamos instanciar nossa classe e passar o método start, vamos passar uma classe Thread no construtor dela que vamos passar nossa classe e depois o método start.

# Pratica

Vamos criar outra inner class

binding.btnIniciar.setOnClickListener **{** */\*\* Aqui estamos encadeando metodo, estamos instanciando e ja usando seu metodo  
 \* Reparar que o metodo start ele vem de herança da classe Thread()  
 \*/* //MinhaThread().start() // FORMULA ANTIGA  
  
 */\*\* Nova formula de fazer \*/* Thread(MinhaRunnable()).start()  
 **}**}  
  
inner class MinhaRunnable : Runnable {  
 override fun run() {  
 *repeat*(30) **{** indice **->** Log.i("info\_thread", "MinhaThread: $indice - T: ${Thread.currentThread().*name*}")  
 runOnUiThread **{** */\*\* runOnUiThread = executar na Thread Principal  
 \* Aqui oque vai acontecer? somente essa linha de codigo que vai ser executada na thread principal  
 \*/* binding.btnIniciar.*text* = "Executando: $indice - T: ${Thread.currentThread().*name*}"  
 */\*\*Habilitando e desabilitando o botao para o usuario nao clicar e reiniciar  
 \*  
 \*/* binding.btnIniciar.*isEnabled* = false  
  
 if (indice == 29) {  
 */\*\* Agora quando a contagem terminar, vamos deixar o usuario clicar no botao novamente  
 \*  
 \*/* binding.btnIniciar.*text* = "Reiniciar a execução"  
 binding.btnIniciar.*isEnabled* = true  
 }  
 **}** Thread.sleep(1000)  
 **}** }  
  
}

Mais também temos a opção de fazer como uma função lambda que fica melhor ainda

# Fazendo thread com função lambda

Lembrar que essa formula só funciona quando estamos implementando a interface runnable

Dentro do setOnCLickListener fizemos diretamente la com uma função lambda

*/\*\* Fazendo com funçao lambda  
 \* Para deixar o codigo mais curto  
 \* Dessa forma nao vamos precisar criar uma classe  
 \*/* Thread**{** *repeat*(30) **{** indice **->** Log.i("info\_thread", "MinhaThread: $indice - T: ${Thread.currentThread().*name*}")  
 runOnUiThread **{** */\*\* runOnUiThread = executar na Thread Principal  
 \* Aqui oque vai acontecer? somente essa linha de codigo que vai ser executada na thread principal  
 \*/* binding.btnIniciar.*text* = "Executando: $indice - T: ${Thread.currentThread().*name*}"  
 */\*\*Habilitando e desabilitando o botao para o usuario nao clicar e reiniciar  
 \*  
 \*/* binding.btnIniciar.*isEnabled* = false  
  
 if (indice == 29) {  
 */\*\* Agora quando a contagem terminar, vamos deixar o usuario clicar no botao novamente  
 \*  
 \*/* binding.btnIniciar.*text* = "Reiniciar a execução"  
 binding.btnIniciar.*isEnabled* = true  
 }  
 **}** Thread.sleep(1000)  
 **}  
 }**.start()  
**}**

# Parando uma thread

Primeiro criamos uma variável como falso

private var pararThread = false

Porque usamos o false? Imagina assim:

Vamos parar a thread? Não/false

Mas quando a gente pressionar o botão de parar vamos alterar

Para isso vamos criar o setOnClickListener do botão parar.

Agora sim quando o usuário clicar no botão vamos alterar para sim, ex: Queremos parar a thread? Sim/true

binding.btnParar.setOnClickListener **{** pararThread = true  
**}**

Agora dentro do nosso código que faz a thread funcionar fizemos um if para verificar, se for verdadeiro adicionamos um return

inner class MinhaRunnable : Runnable {  
 override fun run() {  
 *repeat*(30) **{** indice **->** if (pararThread) {  
 return  
 }

Esse return ele vai fazer retornar e não deixar seguir o código.

Repara que clicamos no botão de parar e a thread ficou travada

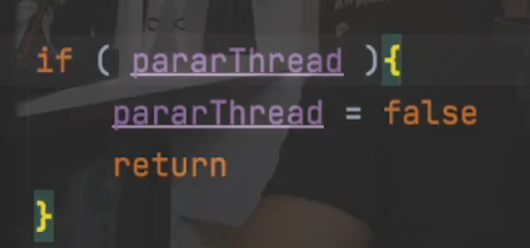


binding.btnParar.setOnClickListener **{** pararThread = true  
  
 binding.btnIniciar.*text* = "Execução parada"  
 binding.btnIniciar.*isEnabled* = true  
**}**



Quando clicamos em parar, ele altera o botão e ativa ele novamente.

Reiniciando a execução da thread



Quando o usuário clicar em parar, vamos retornar e configurar novamente o botão como false, para poder clicar nele novamente.