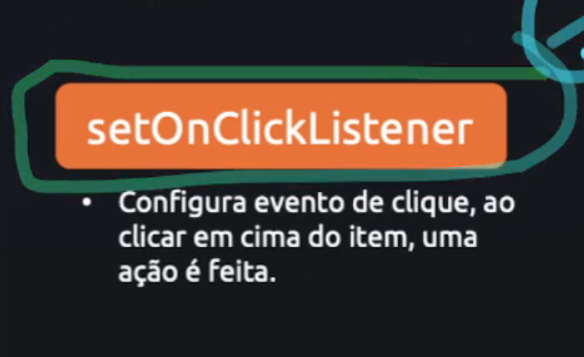
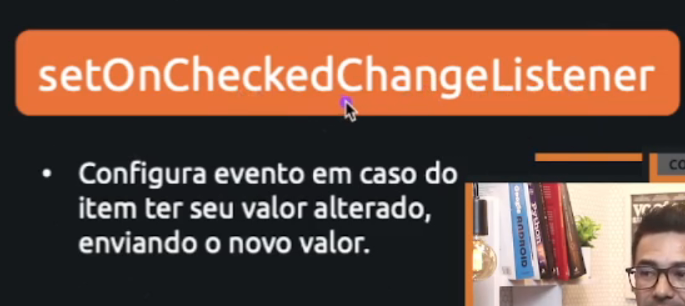


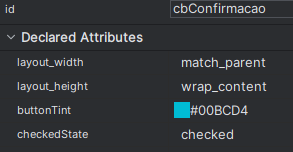
Com o checkbox voce pode marca 1 opcao ou varias opções



Vamos utilizar mais esse para o checkbox

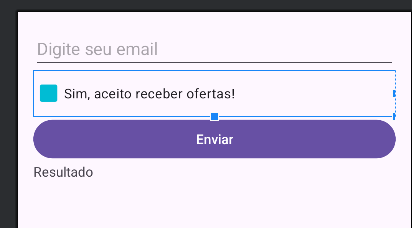
Vamos imaginar que existe uma câmera no mercado, e ela fica analisando quando um item e retirado na prateleira

Mesma coisa vai ser aqui, ele vai ficar ouvindo, e vai fazer a ação quando o item mudar.



buttonTint = mudar a cor do checkbox

checkedState = vai deixar o botão sempre marcado como se tivesse selecionado



# Usando com o setOnClickListerner

package com.allephnogueira.checkbox  
  
import android.os.Bundle  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import com.allephnogueira.checkbox.databinding.ActivityMainBinding  
  
class MainActivity : AppCompatActivity() {  
  
  
 */\*\* Adiconando o viewBinding  
 \*  
 \*/* private val binding by *lazy* **{** ActivityMainBinding.inflate(*layoutInflater*)  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(binding.*root*)  
 */\*\* View Binding \*\*/* ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** */\*\*  
 \* Criamos um metodo para quando o usuario clicar no botao ele vai chamar o metodo checkbox  
 \* Ai la dentro do metodo checkbox, vamos recuperar se o usuario clicou ou não  
 \* E exibir na tela  
 \*/  
 with*(binding) **{** btnEnviar.setOnClickListener **{** //checbox()  
  
 **}** cbConfirmacao.setOnClickListener **{** */\*\*  
 \* Podemos fazer diretamente aqui dentro, quando a marcação estiver selecionada  
 \* Antes estavamos chamando o metodo, agora vamos deixar aqui dentro mesmo.  
 \*  
 \* Agora quando o check for marcado, ele vai fazer a mesma coisa que um evento de click  
 \* E ja vai atualizar o resultado na mesma hora  
 \*/* val selecionado = binding.cbConfirmacao.*isChecked* val resultado =  
 if (selecionado) "Foi selecionado = Verdadeiro" else "Não selecionado = Falso"  
  
 binding.textResultado.*text* = resultado  
 **}  
   
   
  
 }** }  
  
 private fun checbox() {  
 */\*\*  
 \* Nosso metodo para recuperar o checkbox  
 \* isChecked é um metodo para verificar se o item esta marcado  
 \* Ele vai retornar sempre um verdadeiro ou falso  
 \*  
 \* Atençao repara que estamos acessando o ID do checkbox e atribuindo o valor dele direto em uma variavel  
 \* O retorno dele vai ser sempre TRUE ou FALSE  
 \*  
 \*  
 \* Podemos fazer diretamente la dentro do cbConfirmação tambem  
 \*/* val selecionado = binding.cbConfirmacao.*isChecked* val resultado =  
 if (selecionado) "Foi selecionado = Verdadeiro" else "Não selecionado = Falso"  
  
 binding.textResultado.*text* = resultado  
 }  
}

# Usando com o setOnCheckedChangeListener

package com.allephnogueira.checkbox  
  
import android.os.Bundle  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import com.allephnogueira.checkbox.databinding.ActivityMainBinding  
  
class MainActivity : AppCompatActivity() {  
  
  
 */\*\* Adiconando o viewBinding  
 \*  
 \*/* private val binding by *lazy* **{** ActivityMainBinding.inflate(*layoutInflater*)  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(binding.*root*)  
 */\*\* View Binding \*\*/* ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** */\*\*  
 \* Criamos um metodo para quando o usuario clicar no botao ele vai chamar o metodo checkbox  
 \* Ai la dentro do metodo checkbox, vamos recuperar se o usuario clicou ou não  
 \* E exibir na tela  
 \*/  
 with*(binding) **{** btnEnviar.setOnClickListener **{** //checbox()  
  
 **}** */\*\*  
 \* Aqui estamos utilizando com o setonClickListener  
 \*/*// cbConfirmacao.setOnClickListener {  
// /\*\*  
// \* Podemos fazer diretamente aqui dentro, quando a marcação estiver selecionada  
// \* Antes estavamos chamando o metodo, agora vamos deixar aqui dentro mesmo.  
// \*  
// \* Agora quando o check for marcado, ele vai fazer a mesma coisa que um evento de click  
// \* E ja vai atualizar o resultado na mesma hora  
// \*/  
// val selecionado = binding.cbConfirmacao.isChecked  
//  
// val resultado =  
// if (selecionado) "Foi selecionado = Verdadeiro" else "Não selecionado = Falso"  
//  
// binding.textResultado.text = resultado  
// }  
  
 */\*\*  
 \* Agora vamos usar o outro metodo  
 \*/* cbConfirmacao.setOnCheckedChangeListener **{** \_, isChecked **->** */\*\*  
 \* Aqui vamos ter 2 parametro buttonView = o item que esta sendo clicado  
 \* isChecked o valor se esta sendo falso ou verdadeiro  
 \*  
 \* Agora esse metodo, nao vamos mais precisar usar o isChecked  
 \* Porque ele vai ficar o tempo todo monitorando para ver se existe alguma mudança  
 \*  
 \* Atençao, porque no primeiro parametro tem o \_  
 \* Porque não vamos utilizar esse parametro, então o \_ nao fica ocupando espaço na memoria  
 \* Ele é uma variavel, mas uma variavel "oculta"  
 \* Se voce quiser utilizar o parametro/variavel ai voce deve por um nome nela  
 \*/* val resultado =  
 if (isChecked) "Foi selecionado = Verdadeiro" else "Não selecionado = Falso"  
  
 binding.textResultado.*text* = resultado  
  
 */\*\* Resumindo então, aqui vamos ficar ouvindo se tem alguma coisa sendo alterada  
 \* Se tiver o isCheckd vai alterar automaticamente e vai nos retornar um TRUE  
 \* ai podemos pegar o TRUE e fazer oque a gente bem entender.  
 \*/* **}  
  
  
  
 }** }  
  
 private fun checbox() {  
 */\*\*  
 \* Nosso metodo para recuperar o checkbox  
 \* isChecked é um metodo para verificar se o item esta marcado  
 \* Ele vai retornar sempre um verdadeiro ou falso  
 \*  
 \* Atençao repara que estamos acessando o ID do checkbox e atribuindo o valor dele direto em uma variavel  
 \* O retorno dele vai ser sempre TRUE ou FALSE  
 \*  
 \*  
 \* Podemos fazer diretamente la dentro do cbConfirmação tambem  
 \*/* val selecionado = binding.cbConfirmacao.*isChecked* val resultado =  
 if (selecionado) "Foi selecionado = Verdadeiro" else "Não selecionado = Falso"  
  
 binding.textResultado.*text* = resultado  
 }  
}