# Oque é uma activity?

É um componente do android que representa a tela do nosso aplicativo, onde vamos ter: textos, botões, menus, listas entre outros.

# Uma activity ela é composta por;

Uma class Kotlin e um arquivo XML onde vai ter a parte gráfica.

Os dois juntos vão formar uma acitivity

# Navegando entre activity

Antes de ir para a navegação entre as activity o ideal é entender oque é uma **PILHA** de activity

Uma pilha de activity é; conforme você vai abrindo, uma vai ficando sobre a outra, então o ideal é ir fechando-as para não ocupar muito espaço na memória.

Vamos imaginar que temos 2 activity e você abriu a segunda, então você não precisa voltar para a primeira, você pode apenas fechar a segunda e automaticamente vai retornar para a primeira.

# Como abrir uma nova activity

Para abrir uma nova activity vamos utilizar o método startActivity

Agora para abri precisamos utilizar como parâmetro o objeto **intent**

**startActivity(intent)**

Intent = intenção || (resolve ou faz determinada coisa)

startAcitivity(intente(**contexto**, classQueVaiSerAberta))

**contexto** = contexto que será executado

## Oque é o CONTEXTO

É um ponto de acesso para informações globais do seu App.

Vamos imaginar aqui em casa, tem várias janelas.

- Aí eu falo Fernanda abra a janela (**intente = intenção**)

E aí eu preciso passar o **contexto**

Oque seria o contexto; seria qual janela Fernanda vai abrir, porque tem mais de uma janela

- Fernanda abra a janela(**intente**) da cozinha (**contexto**)

E aí eu passo dessa forma

startActivity(intente(contexto, janelaSegundoActivity::class.java))

## Outro exemplo

startActivity(intent(contexto, JanelaCaronaActivity::class.java))

Vamos imaginar que falo, abra a janela, mas não sabemos qual, então precisamos passar um pouco mais de contexto

Abra janela (intenção)?

Qual? (contexto)

Do carona (JanelaCaronaActivity::class.java)

Resumo a **intent** ela faz algo que você esta mandando, mas ela precisa do contexto para saber onde ela vai fazer.

Então no exemplo de cima a **intent** esta abrindo uma nova Activity, mas ela precisa saber qual (**contexto**) ela vai abrir, ai você usa o uma virgula e passa a activity que precisa ser aberta.

## Abrindo nova Activity na pratica.

class MainActivity : AppCompatActivity() {  
  
 lateinit var btnAbrirSegundaTela : Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_main*)  
 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  
 **}** btnAbrirSegundaTela = findViewById(R.id.*button\_abrir*)  
 btnAbrirSegundaTela.setOnClickListener {  
 startActivity(Intent(this, DetalhesActivity::class.*java*))  
 }}  
  
}

Estamos adicionando uma variável que ainda não inicializamos, vamos inicializar depois **lateinit**

Agora estamos transformando a variável em um objeto do tipo **button\_abrir**

E depois estamos fazendo uma função lambda para quando o usuário clicar nesse botão ele abrir uma nova tela.

**setOnClickListener** = um ouvinte que fica esperando ser clicado para disparar um evento, nesse caso o evento vai ser abrir a segunda tela.

This = é para dizer qual contexto vai ser executado no nosso caso, pedimos para ele executar apenas o DetalhesActivity, mas poderíamos pedir para ele executar algo dentro do DetalhesActivity.

## Fechando uma activity

Aqui estamos fechando a segunda activity então ele vai voltar automaticamente para a tela anterior

Não faça isso para a primeira activity se não ele vai fechar o aplicativo.

class DetalhesActivity : AppCompatActivity() {  
  
 lateinit var btnFechar : Button  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*)  
 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  
 **}** btnFechar = findViewById(R.id.*button\_fechar*)  
 btnFechar.setOnClickListener {  
 finish()  
 **}** }  
}

# Clico de vida de uma Activity

Primeiro começamos pelo onCreate (Onde tudo que estiver la vai ser feito primeiro)

Ele vai automaticamente configurar nosso layout

class DetalhesActivity : AppCompatActivity() {  
  
 lateinit var btnFechar : Button  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*) < configurar nosso layout  
 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  
 **}**

btnFechar = findViewById(R.id.*button\_fechar*)

Configurar nosso evento de click no botão.  
 btnFechar.setOnClickListener **{** finish()  
 **}** }  
}Aqui conseguimos manipular elementos da interface.

Aqui no onCreate pode ser carregado o progressBar, ele vai ficar rodando ate ela carregar.

Em seguida é chamado o método **onStart**()

Imagina um jogo onde temos que pegar os dados do banco de dados, então é aqui que ele vai ficar carregando para a gente.

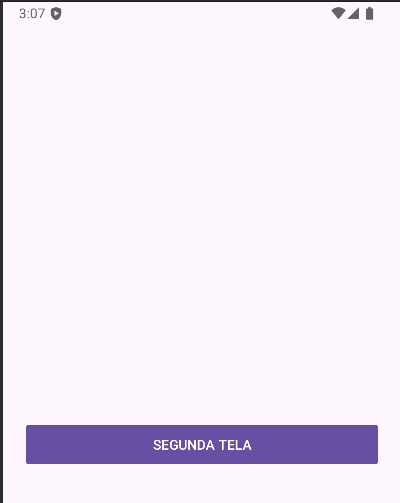
Depois o **onResume**()

Imagina assim : onCreate exibo o ícone de carregando

onStart : puxo os dados do banco

onResumo eu retiro o ícone de carregando

Após isso e renderizado a imagem no **Activity Running**



Agora vamos refazer todos esses métodos na pratica

## Refazendo o clico na pratica

package com.allephnogueira.aulaacitivityfragment  
  
import android.os.Bundle  
import android.util.Log  
import android.widget.Button  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
  
class DetalhesActivity : AppCompatActivity() {  
  
 lateinit var btnFechar : Button  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 Log.i("clico\_vida", "onCreate")  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*)  
 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  
 **}** btnFechar = findViewById(R.id.*button\_fechar*)  
 btnFechar.setOnClickListener **{** finish()  
 **}** }  
  
 override fun onStart() {  
 super.onStart()  
 //Carregar os dados que vem do servidor  
 //Imagina o alta pressão gnv, aqui é onde vamos carregar os dados do posto.  
 Log.i("cliclo\_vida", "onStart")  
 }  
  
 override fun onResume() {  
 super.onResume()  
 Log.i("clico\_vida", "onResume")  
 }  
  
 override fun onPause() {  
 super.onPause()  
 Log.i("clico\_vida", "onPause")  
 }  
  
 override fun onStop() {  
 super.onStop()  
 Log.i("clico\_vida", "onStop")  
 }  
  
 override fun onRestart() {  
 super.onRestart()  
 Log.i("clico\_vida", "onRestart")  
 }  
  
 override fun onDestroy() {  
 super.onDestroy()  
 Log.i("clico\_vida", "onDestroy")  
 }  
  
  
}

Qualquer duvida tem o material em PDF também.

# Passando dados de uma actvivity para outra.

Imagina o aplicativo Netflix, onde temos vários filmes, então quando clicamos em 1, vamos para a segunda tela, a tela de detalhes, mas detalhes de quem? Então precisamos passar as informações.

Vamos usar o

Intent.putExtra(“NOME\_FILME”, “Harry Potter”)

No primeiro parâmetro vamos passar o identificador

No segundo parâmetro vamos passar a informação que queremos utilizar.

Logo após fazer isso vamos chamar a classe, veja no código.

val intent = Intent(this, DetalhesActivity::class.*java*)  
  
btnAbrirSegundaTela = findViewById(R.id.*button\_abrir*)  
btnAbrirSegundaTela.setOnClickListener **{** // Vamos passar os dados para a proxima activity por parametro.  
 // Primeiro valor é o identificador do que voce quer passar  
 // Segundo é o dado  
  
   
 intent.putExtra("filme", "Harry Potter")  
 intent.putExtra("classificacao", 8)  
 intent.putExtra("avaliacoes", 300)  
 intent.putExtra("exibicao", false) // exibição nos cinemas?

startActivity(intent) // Estamos chamando o intente já configurado com o nome da classe.  
**}**

## Capturando na outra tela os dados que chegaram

class DetalhesActivity : AppCompatActivity() {  
  
 lateinit var btnFechar : Button  
 lateinit var textFilme : TextView // onde vamos colocar o nome do filme  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 Log.i("clico\_vida", "onCreate")  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*)  
 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  
 **}** btnFechar = findViewById(R.id.*button\_fechar*)  
 textFilme = findViewById(R.id.*textFilme*) // Onde estamos passando a referencia do objeto filme  
  
 */\*\* Bundle é um objeto do android que significa (PACOTE - EMBRULHAR)  
 Essa classe é utilizada para armazenar valores.  
 Esse objeto é uma maneira do android empacotar as informações e passar de uma tela para outra. \*/  
  
 /\*\* intent = Quando passamos uma intent(intenção) na tela anterior o proprio android studio ja passou esses dados e deixou guardado  
 \* so vamos precisar acessar eles agora  
 \*  
 \* extras = lembra que usamos o extra também? os dados estao dentro de inputExtra, vamos acessar eles também.  
 \*  
 \* agora o bundle tem todos os metodos passados da tela anterior pra ca.  
 \*/* val bundle = *intent*.*extras  
 /\*\*  
 \* Agora vamos recuperar o filme, ja passamos ele para dentro do bundle  
 \* Repara que estamos utilizando o getString porque o dado que passamos é do tipo String, mas poderia ser outro tipo tambem  
 \* como getInt para numeros inteiros.  
 \* vamos passar outros parametros para teste  
 \*  
 \* primeiro parametro é o nome que definimos la (key)  
 \*  
 \* Atenção: Devemos utilizar uma chamada segura (?) porque o objeto que esta vindo pode ser nulo.  
 \* bundle?.getString("filme")  
 \* Mas podemos também verificar se é diferente de nulo  
 \*  
 \* if (bundle != null) {  
 \* /\*\* Os dados podem vir nulo, por isso vamos verificar antes. \*/  
 \* bundle.getString("filme")  
 \* }  
 \*  
 \*/* if (bundle != null) {  
 */\*\* Os dados podem vir nulo, por isso vamos verificar antes. \*/* val filme = bundle.getString("filme")  
 textFilme.*text* = filme  
 }

# Passando objeto por activitys

Muitas das vezes vamos trabalhar com objetos em POO, então para passar objetos se chama serialização.

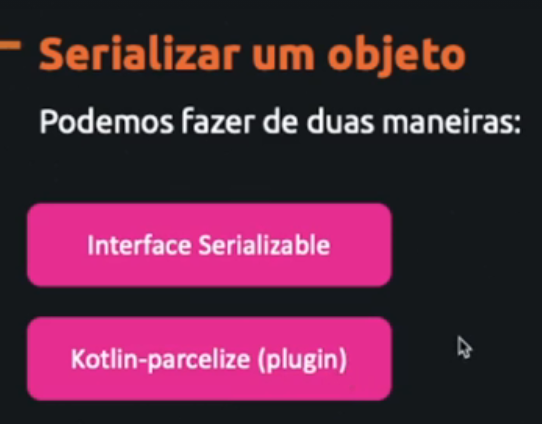
Significa salvar o estado atual do objeto em formato binário, podendo ser recuperado esse estado posteriormente.

Serializar é como se tivesse compactando esse objeto.

Passos



Como serializar o objeto, podemos utilizar duas maneiras



Kotlin-parceliza(plugin) é recomendando, é um método mais novo, e também é mais rápido.

## Passando os dados PRATICA

Temos os 2 métodos escolha um.

### Interface Serializable

Primeiro vamos criar um DTO (Data class)

package com.allephnogueira.passandoobjetosporactivity  
  
import java.io.Serializable  
  
data class Filme(  
 val nome: String,  
 val descricao: String,  
 val avaliacoes: Double,  
 val direitor: String,  
 val distribuidor: String  
) : Serializable // Precisamos dizer que filme é do tipo Serializable para poder usar no putExtra

#### Passando os dados

class MainActivity : AppCompatActivity() {  
  
 // Criamos uma variavel  
 // Adicionamos o tipo dela  
 lateinit var btnDetalhesFilme : Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_main*)  
 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 uma variavel  
 // Adicionamos dentro da variavel a classe que ela vai chamar  
 val intent = Intent(this, DetalhesActivity::class.*java*)  
  
 // Passamos para o botao onde esta o item que ele vai alterar  
 // Passamos para o botao a sua referencia  
 btnDetalhesFilme = findViewById(R.id.*button\_DetalhesFilme*)  
  
 // Criamos um evento de click, quando o usuario clicar vai chamar a variavel que esta com a classe dentro.  
 btnDetalhesFilme.setOnClickListener **{** startActivity(intent) **}** // Agora vamos usar nosso Data Class, passando os dados.  
 val filme = Filme (  
 "Harry Potter",  
 "Harry Potter é um bruxo que veio para lutar com um bruxo das trevas",  
 9.5,  
 "Alleph",  
 "Sony"  
 )  
  
 // Agora vamos passar os dados

// Aqui vamos passar a referencia como Filme e vamos passar os dados que são do tipo Filme

intent.putExtra("Filme", filme)  
  
  
 }  
}

#### Pegando dados na outra tela (Descontinuado)

Atenção esse modo foi descontinuado, mas ainda podemos achar códigos assim, se achar você deve alterar.

Ele foi descontinuado na versão 33

Atenção dessa forma também funciona, mas existe a maneira correta.

Intent.extra é onde esta nossos dados.

Mas vamos colocar ele em uma variável bundle.

class DetalhesActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*)  
 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  
 **}** val bundle = *intent*.*extras* if (bundle != null) {  
 // Aqui vamos passar a chave que esta vindo do outro servidor  
 // Aqui como o filme é um objeto precisamos usar o as  
 // as = como estou convertendo esse serializable para um Filme  
 val filme = bundle.getSerializable("Filme") as Filme  
  
 val nomeFilme: TextView = findViewById(R.id.*text\_nomeFilme*)  
  
 nomeFilme.*text* = "${filme.nome}"  
  
 }  
  
  
  
 }

#### Pegando dados na outra tela (Maneira correta)

class DetalhesActivity : AppCompatActivity() {  
  
 lateinit var nomeFilme : TextView  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*)  
 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  
 **}** nomeFilme = findViewById(R.id.*text\_nomeFilme*)  
  
  
 val bundle = *intent*.*extras  
  
  
 /\*\*  
 \* Podemos atribuir um metodo em uma variavel  
 \* Se a versao for maior que 33 a variavel vai receber o  
 \* bundle?.getSerializable("Filme", Filme::class.java)  
 \*/* val filme = if (Build.VERSION.*SDK\_INT* >= 33) {  
 Log.i("Build", "Versão do sistema é ${Build.VERSION.*SDK\_INT*}")  
 */\*\*  
 \* Lembrar que esse novo metodo é apenas para versoes de sdk superior a 33  
 \* Entao vamos usar o BUILD (BUILD é para saber os dados do celular que esta rodando nosso codigo)  
 \* Vamos testar a versão do SDK se for maior que 33, vamos utilizar o metodo novo, caso nao seja vamos utilziar o metodo antigo  
 \*  
 \* getSerializable  
 \* Primeiro parametro é a chave que colocamos la na outra tela, vamos acessar ela  
 \* Acessando a chave vamos ter os dados que estao dentro dela  
 \* Segundo parametro é a nossa classe DTO  
 \*/* bundle?.getSerializable("Filme", Filme::class.*java*)  
  
 }else {  
 */\*\*  
 \* Caso a versao seja menor que a 33 vamos utilizar o metodo antigo  
 \*/* bundle?.getSerializable("Filme") as Filme  
 }  
  
 */\*\*  
 \* Repara que filme ainda pode vir nulo, para isso vamos fazer uma verificação( ? )  
 \*/* nomeFilme.*text* = filme?.nome  
  
  
 }  
}

### Kotlin-parcelize (Plugin) (RECOMENDADO)

#### Primeiro devemos adicionar o plugin

plugins **{** alias(*libs*.*plugins*.*android*.*application*)  
 alias(*libs*.*plugins*.*kotlin*.*android*)  
 id("kotlin-parcelize")  
**}**

#### Depois vamos no nosso DTO e vamos fazer dessa forma

package com.allephnogueira.passandoobjetosporactivity  
  
import android.os.Parcelable  
import kotlinx.parcelize.Parcelize  
import java.io.Serializable  
  
@Parcelize   
data class Filme(  
 val nome: String,  
 val descricao: String,  
 val avaliacoes: Double,  
 val direitor: String,  
 val distribuidor: String  
) : Parcelable

#### Passando os dados

class MainActivity : AppCompatActivity() {  
  
 // Criamos uma variavel  
 // Adicionamos o tipo dela  
 lateinit var btnDetalhesFilme : Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_main*)  
 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 uma variavel  
 // Adicionamos dentro da variavel a classe que ela vai chamar  
 val intent = Intent(this, DetalhesActivity::class.*java*)  
  
 // Passamos para o botao onde esta o item que ele vai alterar  
 // Passamos para o botao a sua referencia  
 btnDetalhesFilme = findViewById(R.id.*button\_DetalhesFilme*)  
  
 // Criamos um evento de click, quando o usuario clicar vai chamar a variavel que esta com a classe dentro.  
 btnDetalhesFilme.setOnClickListener **{** startActivity(intent) **}** // Agora vamos usar nosso Data Class, passando os dados.  
 val filme = Filme (  
 "Harry Potter",  
 "Harry Potter é um bruxo que veio para lutar com um bruxo das trevas",  
 9.5,  
 "Alleph",  
 "Sony"  
 )  
  
 // Agora vamos passar os dados  
 // Aqui vamos passar a referencia como Filme e vamos passar os dados que são do tipo Filme  
  
 intent.putExtra("Filme", filme)  
  
  
 }  
}

#### Pegando os dados na outra Activity

class DetalhesActivity : AppCompatActivity() {  
  
 lateinit var nomeFilme : TextView  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*)  
 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  
 **}** nomeFilme = findViewById(R.id.*text\_nomeFilme*)  
  
  
 val bundle = *intent*.*extras  
  
  
 /\*\*  
 \* Podemos atribuir um metodo em uma variavel  
 \* Se a versao for maior que 33 a variavel vai receber o  
 \* Todos os metodo tanto o getSerializable e o Parcelable eles devem ser feito o teste  
 \*  
 \* bundle?.getSerializable("Filme", Filme::class.java)  
 \* bundle?.getParcelable("Filme", Filme::class.java)  
 \*  
 \* Essa forma que esta em cima é apenas para versões maiores que tiramisu(33)  
 \*/* val filme = if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  
 Log.i("Build", "Versão do sistema é ${Build.VERSION.*SDK\_INT*}")  
 */\*\*  
 \*  
 \* Aqui ele testa a versao se for maior que a 33  
 \* val filme = if (Build.VERSION.SDK\_INT >= 33)  
 \*  
 \* E aqui ele pega a versao pelo nome  
 \* val filme = if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.TIRAMISU)  
 \*  
 \* Lembrar que esse novo metodo é apenas para versoes de sdk superior a 33  
 \* Entao vamos usar o BUILD (BUILD é para saber os dados do celular que esta rodando nosso codigo)  
 \* Vamos testar a versão do SDK se for maior que 33, vamos utilizar o metodo novo, caso nao seja vamos utilziar o metodo antigo  
 \*  
 \* getSerializable ou getParcelable  
 \* Primeiro parametro é a chave que colocamos la na outra tela, vamos acessar ela  
 \* Acessando a chave vamos ter os dados que estao dentro dela  
 \* Segundo parametro é a nossa classe DTO  
 \*/* bundle?.getParcelable("Filme", Filme::class.*java*)  
  
 }else {  
 */\*\*  
 \* Caso a versao seja menor que a 33 vamos utilizar o metodo antigo  
 \*/* bundle?.getParcelable("Filme")  
 }  
  
 */\*\*  
 \* Repara que filme ainda pode vir nulo, para isso vamos fazer uma verificação( ? )  
 \*/* nomeFilme.*text* = filme?.nome  
  
  
 }  
}