



Desenvolvimento Mobile

Activity

Professor MSc. Antônio Catani
antonio.catani@faculdadeimpacta.com.br

Introdução

Introdução

- Na aula anterior foi criado um aplicativo simples para ver o funcionamento básico de um aplicativo Android.
- Ele continha basicamente uma tela (arquivo xml, dentro da pasta layout) e um arquivo Kotlin, onde foram definidos a tela e os eventos.
- A classe no arquivo Kotlin é conhecida como Activity.

Introdução

- A classe Activity é uma das classes mais importantes no Android.
 - Geralmente ela representa uma tela no aplicativo.
 - É responsável por definir qual será a View que desenha a interface gráfica.
 - É responsável por controlar os eventos da tela.

Introdução

- Usualmente um aplicativo tem mais de uma tela, e consequentemente mais de uma Activity.
 - Sempre que for criar uma nova tela no aplicativo, é necessário ter uma Activity relativa a esta tela.
- Activity = Atividade = Ações e funcionalidades que o usuário pode fazer no app.

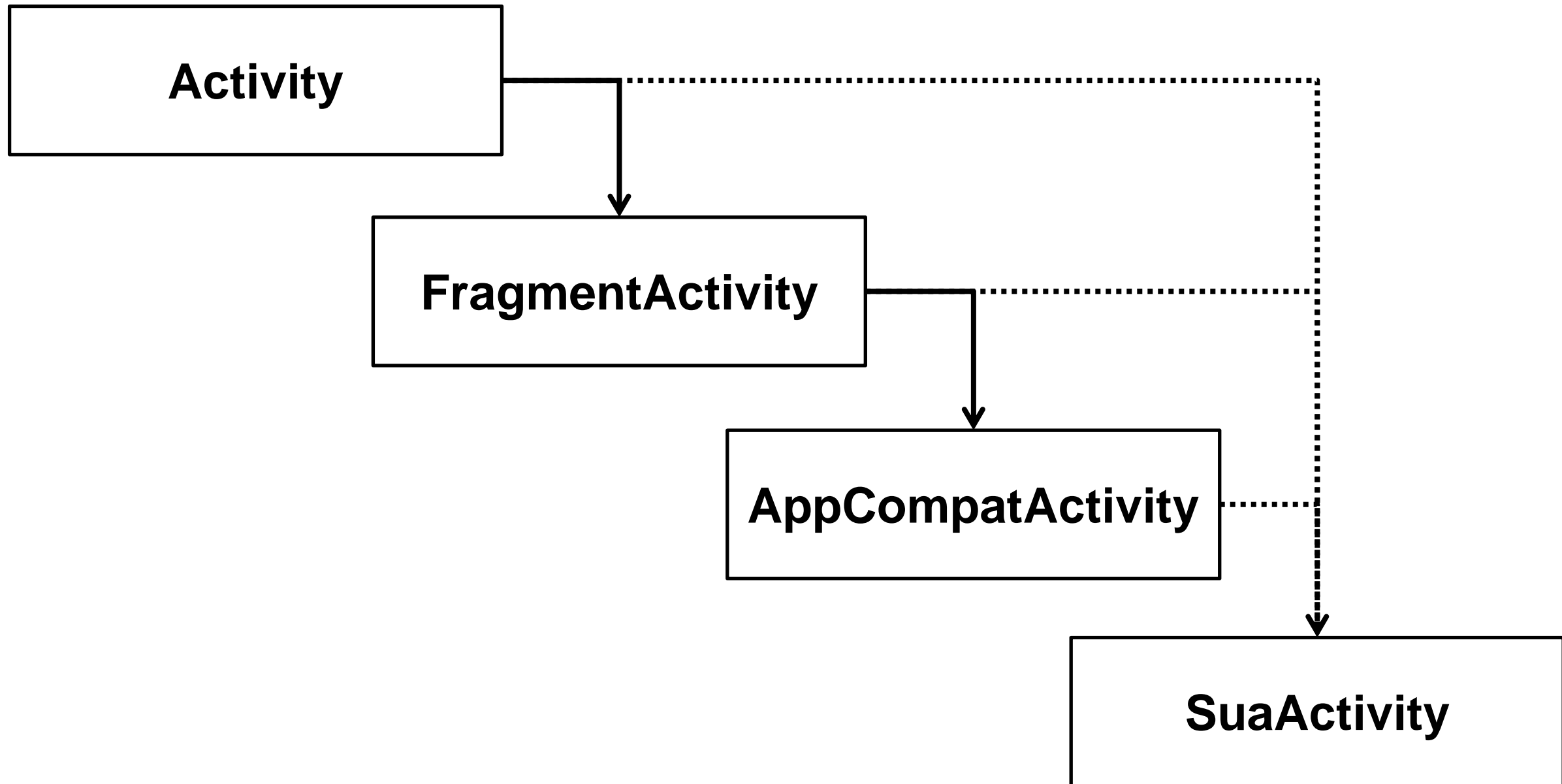
Activity

Activity

- Uma classe de Activity deve herdar todas as características (atributos) e comportamento (métodos) da classe `android.app.Activity`, ou alguma subclasse desta.
 - Por exemplo, `AppCompatActivity` ou `FragmentActivity` (ambas são subclasses de `Activity`).

Activity

- FragmentActivity permite utilizar fragments em versões antigas do Android.
- AppCompatActivity possibilita que a ActionBar (barra superior dos aplicativos) funcione e versões antigas do Android.
 - É subclasse de FragmentActivity.



Activity

- Toda Activity deve:
 - Sobrescrever o método onCreate() da superclasse.
 - Responsável por realizar a inicialização para executar a aplicação, como definir a interface do usuário.

Activity

`<activity android:name=".MainActivity" />`

- A declaração da Activity no arquivo de configuração é feita com a sintaxe do ponto.
 - Ou seja, o pacote da classe é relativo ao pacote do projeto (definido quando o projeto foi criado), por exemplo `br.com.ahcatani.lmsapp`)

Activity

- Caso a Activity esteja em outro pacote, basta colocar o caminho do pacote:
 - Um pacote dentro do pacote principal
(br.com.ahcatani.lmsapp.telas)


```
<activity android:name=".telas.MainActivity" />
```
 - Outro pacote, fora do pacote principal:
- ```
<activity
android:name="br.com.ahcatani.lmsapp.activities.
MainActivity" />
```

# Activity

---

- Uma Activity geralmente representa uma tela do aplicativo.
- Deve implementar o método onCreate().
- Deve ser declarado em AndroidManifest.xml.

# Ciclo de Vida da Atividade

# Ciclo de Vida de uma Atividade

---

- Ciclo de vida está relacionado aos estados que uma Activity se encontra.
  - Executando.
  - Temporariamente interrompida (segundo plano).
  - Destruída.
- O sistema operacional é responsável por cuidar deste ciclo de vida.
- Entretanto, um aplicativo robusto se preocupa em tratar estes estados.

# Ciclo de Vida de uma Atividade

---

- Exemplo:
  - O usuário está utilizando um aplicativo de jogo no Android e enquanto isso ele recebe uma ligação.
  - Ao atender a ligação o SO interrompe o jogo temporariamente e o coloca em segundo plano.



# Ciclo de Vida de uma Atividade

---

- Exemplo (...):
  - Quando a ligação termina, o SO reinicia o jogo.
    - O jogo vai continuar de onde parou?
    - O estado e informações foram salvos ou foi perdido?
  - O Android fornece a estrutura necessária para tratar estes casos.
    - Para isso é necessário entender o ciclo de vida da Activity.

## Ciclo de Vida de uma Atividade

---

- Uma Activity tem um ciclo de vida bem definido.
- Cada Activity iniciada é inserida no topo da pilha de Activities.
- Aquela que está no topo da pilha está em execução.
  - As outras abaixo dela podem ser “pausadas” ou estar totalmente paradas.

## Ciclo de Vida de uma Atividade

---

- Uma Activity pausada pode ter seu processo encerrado pelo SO para liberar recursos.
  - Quando o SO decide encerrar o processo, o aplicativo pode salvar os dados para recuperar depois.
- Tudo no Android é uma Activity, inclusive a Tela Inicial (Home).

# Ciclo de Vida de uma Atividade

- Pilha de Activities

| Activity em Execução |
|----------------------|
| Activity Parada      |
| Activity Parada      |
| Activity Parada      |
| Activity Parada      |
| Activity Parada      |

# Ciclo de Vida de uma Atividade

- Usuário está na tela inicial no Android.
  - Activity da Home é colocada no topo da pilha.

| Activity Home |
|---------------|
|               |
|               |
|               |
|               |
|               |

# Ciclo de Vida de uma Atividade

- Usuário abre o Navegador.
  - Activity da Home é parada.
  - Activity principal do navegador é colocada no topo da pilha.

| Activity Principal - Navegador |
|--------------------------------|
| Activity Home                  |
|                                |
|                                |
|                                |
|                                |

# Ciclo de Vida de uma Atividade

- Usuário volta para Home.
  - Activity principal do navegador é parada.
  - Activity Home é colocada no topo da pilha.

| Activity Home                  |
|--------------------------------|
| Activity Principal - Navegador |
|                                |
|                                |
|                                |
|                                |

# Ciclo de Vida de uma Atividade

- Usuário abre o aplicativo de e-mail.
  - Activity Home é parada.
  - Activity Principal do aplicativo de e-mail é colocada no topo da pilha.

| Activity Principal - E-mail    |
|--------------------------------|
| Activity Home                  |
| Activity Principal - Navegador |
|                                |
|                                |
|                                |



# Ciclo de Vida de uma Atividade

- Usuário seleciona opção de escrever novo e-mail.
  - Activity Principal do aplicativo de e-mail é parada.
  - Activity de Escrever e-mail é colocada no topo da pilha.

|                                       |
|---------------------------------------|
| <b>Activity Escrever - E-mail</b>     |
| <b>Activity Principal - E-mail</b>    |
| <b>Activity Home</b>                  |
| <b>Activity Principal - Navegador</b> |
|                                       |
|                                       |


# Ciclo de Vida de uma Atividade

- Usuário volta para o navegador.
  - Activity de Escrever e-mail é parada.
  - Activity Principal do navegador volta para o topo da pilha.

| Activity Principal - Navegador |
|--------------------------------|
| Activity Excrever - E-mail     |
| Activity Principal - E-mail    |
| Activity Home                  |
|                                |
|                                |

# Ciclo de Vida de uma Atividade

- Se a pilha estiver cheia, o SO decide qual Activity destruir para colocar uma nova no topo da pilha.

| Activity Principal - Navegador |                                                                                       |
|--------------------------------|---------------------------------------------------------------------------------------|
| Activity Excrever - E-mail     |                                                                                       |
| Activity Principal - E-mail    |                                                                                       |
| Activity Home                  |                                                                                       |
| Activity                       |                                                                                       |
| Activity                       |  |

## Ciclo de Vida de uma Atividade

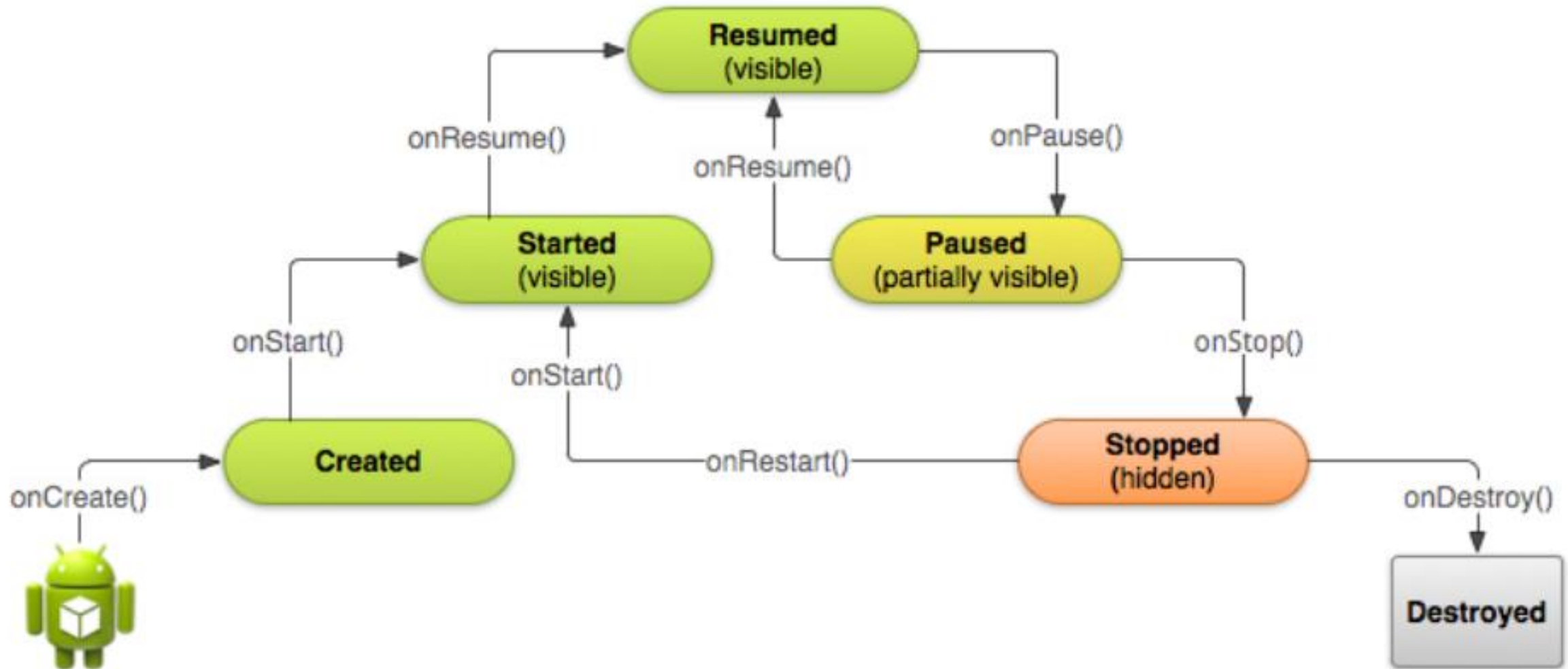
---

- Mesmo os aplicativos nativos do dispositivo, como navegador, tela inicial, agenda e telefone são definidos por uma Activity.
  - Estas vão ser inseridas na mesma pilha de atividades que uma Activity de um aplicativo instalado ou desenvolvido por você.
  - Lembrando: todos os aplicativos funcionam da mesma forma, sobre uma mesma arquitetura.

# Ciclo de Vida de uma Atividade

---

- A superclasse Activity define métodos para controlar estes estados do aplicativo:
  - onCreate(bundle?).
  - onStart().
  - onRestart().
  - onResume().
  - onPause().
  - onStop().
  - onDestroy().
- Estes métodos são sobrescritos (override) pela Activity do seu projeto



<http://developer.android.com/training/basics/activity-lifecycle/starting.htm>

# Ciclo de Vida de uma Atividade

---

- onCreate(bundle?):
  - Obrigatório em toda Activity.
  - É chamado somente uma vez, até o ciclo de vida ser encerrado.
  - Se houver uma View, é neste método que deve ser criada e exibida.
  - Assim que é finalizado, chama automaticamente onStart()

# Ciclo de Vida de uma Atividade

---

- onStart():
  - É chamado quando a tela está ficando visível, após o método onCreate() ou onRestart().



# Ciclo de Vida de uma Atividade

---

- `onRestart()`:
  - Chamado quando a Activity foi parada temporariamente e foi reiniciada.
  - Chama automaticamente o método `onStart()`.

## Ciclo de Vida de uma Atividade

---

- onResume():
  - Chamado quando a Activity está no topo da pilha de execução.
  - A Activity está executando e pronta para interação.
  - Normalmente utilizado para disparar Threads que consultam WS ou BD para atualizar a tela, por exemplo.

# Ciclo de Vida de uma Atividade

---

- onPause():
  - Chamado quando algum evento ocorrer no celular.
  - Activity é temporariamente interrompida.
  - Neste método que os dados devem ser salvos para recuperar depois em onResume().

# Ciclo de Vida de uma Atividade

---

- onStop().
  - Chamado quando a Activity for encerrada, não mais visível pelo usuário.
  - Pode ser reiniciada e chama onRestart().
  - Caso fique muito tempo parada e o SO precise de recursos, pode chamar onDestroy() para remover da pilha.

# Ciclo de Vida de uma Atividade

---

- `onDestroy()`:
  - Encerra a execução da Activity e remove da pilha.
  - Processo no SO é encerrado.
  - Pode ser chamado pelo SO (liberar recursos) ou pelo aplicativo (método `finish()` da Activity).

# Exemplo Prático

## Exemplo Prático

---

- Abra seu projeto da aula passada.
- Crie arquivo chamado DebugActivity.kt.
  - Clicar com o botão direito no pacote principal do seu projeto.
  - Utilize a opção new → Activity → Empty Activity.
  - Desmarque a opção que cria um arquivo de layout.
  - Neste novo arquivo Kotlin, sobrescreva os métodos da Activity que controlam o ciclo de vida.
    - onCreate, onStart, onRestart, onResume, onPause, onStop, onDestroy.
    - Todos os métodos devem chamar o método equivalente da superclasse.

## Exemplo Prático

- No corpo de cada método, coloque a seguinte linha para mostrar uma mensagem no LogCat.

**Log.d(TAG, className + ".onMetodoCicloVida()  
chamado")**

- Troque onMetodoCicloVida pelo nome do método onde está a mensagem.
- Na classe, crie as constants TAG e className da seguinte forma:



## Exemplo Prático

---

```
private val TAG = "LMSApp"
private val className: String
 get() {
 val s = javaClass.name
 return s.substring(s.lastIndexOf("."))
 }
```

## Exemplo Prático

---

- Por exemplo, sobrescrever o método onStart():

```
override fun onStart() {
 super.onStart()
 Log.d(TAG, className + ".onStart() chamado")
}
```

## Exemplo Prático

---

- Faça a MainActivity estender DebugActivity.
- Execute o aplicativo olhe o LogCat: quais métodos foram chamados?
- Simule a chamada de outros métodos do ciclo de vida para verificar os métodos chamados:
  - Pressione o botão voltar do emulador/dispositivo.
  - Abra novamente o aplicativo.
  - Clique no ícone para voltar à tela inicial.
  - Volte a abrir o aplicativo.

# Navegação

# Navegação

---

- Um aplicativo usualmente tem mais de uma tela.
- A navegação entre telas é feita a partir de uma Activity que está sendo executada (tela que está sendo mostrada no app), chamando a Activity que deve ser aberta
- Existem 2 métodos da classe `android.app.Activity` para iniciar outra Activity:
  - `startActivity(intent)`: inicia a próxima Activity.
  - `startActivityForResult(intent, codigo)`: inicia a próxima Activity e envia um código, identificando a Activity de origem.
    - Possibilita retornar informações para a primeira Activity

# Navegação

---

- Estes métodos recebem um parâmetro do tipo `android.content.Intent`.
  - Representa a “intenção” de realizar uma tarefa.
- A Intent contém as informações e parâmetros sobre a activity/tela que será chamada.
- Vamos criar um exemplo para:
  - Navegar da MainActivity para uma nova tela (TelainicialActivity).
  - Enviar um parâmetro de MainActivity para TelainicialActivity.

# Navegação

---

- No projeto crie uma nova Activity chamada TelaInicialActivity, filha de DebugActivity.
  - Monitorar o ciclo de vida da nova Activity.
  - Ela será chamada pela MainActivity.
  - Esta nova Activity deve ter um arquivo de layout.
- Coloque no arquivo de layout um TextView, sem nenhum texto por enquanto.

# Navegação

---

- Agora volte na MainActivity e implementar a navegação para TelaInicialActivity quando o usuário clicar no botão de login (evento onClick() implementado anteriormente):
- A navegação é implementada em 3 passos:
  - Criar uma nova instância de Intent, com 2 parâmetros .
    - Contexto: parâmetro do tipo android.content.Context, que é a Activity atual, que fará a chamada. Context é superclasse de Activity.
    - Activity que será chamada.
  - Colocar parâmetros, se for necessário.
  - Chamar o método startActivity(intent).



# Navegação

---

```
// criar intent
val intent = Intent(context,
 TelaInicialActivity::class.java)

// colocar parâmetros (opcional)
val params = Bundle()
params.putString("nome", "Carmen Regina")
intent.putExtras(params)

// fazer a chamada
startActivity(intent)
```

# Navegação

---

- O método `startActivity(intent)`, recebe como parâmetro a `intent`.
- Esta chamada delega ao SO a tarefa de encontrar e executar a `Activity` chamada.
- Execute o aplicativo, navegue entre as telas e veja o Log com as chamadas para os métodos do ciclo de vida.

# Passagem de Parâmetros

# Passagem de Parâmetros

---

- Em um aplicativo Android é possível enviar parâmetros entre as telas.
- A classe responsável por armazenar os parâmetros é `android.os.Bundle`.
  - Funciona como uma HashTable: chave=valor.
  - Os parâmetros são colocados em uma instância dessa classe.
- No exemplo, enviamos na chave “nome” o valor “Carmen Regina”

# Passagem de Parâmetros

---

- Em um aplicativo Android é possível enviar parâmetros entre as telas.
- A classe responsável por armazenar os parâmetros é `android.os.Bundle`.
  - Funciona como uma `HashTable`: `chave=valor`.
  - Os parâmetros são colocados em uma instância dessa classe.
- No exemplo, enviamos na chave “nome” o valor “Carmen Regina”

```
val params = Bundle()
params.putString("nome", "Carmen Regina")
intent.putExtras(params)
```

# Passagem de Parâmetros

---

- O método `putString(chave,valor)` é responsável por colocar no Bundle um parâmetro do tipo String com a chave “nome” e valor “Carmen Regina”.
- O método `putExtras(bundle)` da Intent é responsável por colocar os parâmetros (Bundle) na Intent.
- Para que o exemplo funcione corretamente, é necessário criar a constante `context` na `MainActivity`, para ela armazene a instância atual da classe.

```
private val context: Context get() = this
```

# Passagem de Parâmetros

---

- O objeto Bundle pode receber quantos parâmetros forem necessários, e de vários tipos diferentes.
  - putBoolean, putBooleanArray.
  - putByte, putByteArray.
  - putChar, putCharSequence, putCharArray, putCharSequenceArray.
  - putDouble, putDoubleArray.
  - putFloat, putFloatArray .
  - putInt, putIntArray.
  - putLong, putLongArray.
  - putShort, putShortArray.

## Passagem de Parâmetros

---

- Agora é preciso recuperar os parâmetros enviados na próxima tela do aplicativo (TelaInicialActivity).
  - O Parâmetro estará na variável herdada `intent`.
- Volte à `TelaInicialActivity` e no método `onCreate()` coloque o seguinte código:

```
val args = intent.extras
val nome = args.getString("nome")
Toast.makeText(context, "Parâmetro: "+ nome,
 Toast.LENGTH_LONG).show()
```

- O parâmetro enviado será armazenado na variável `nome`.



# Simplificando

# Simplificando

- O envio e recuperação dos parâmetros pode ser simplificado utilizando diretamente o objeto da Intent.
  - Na criação da intent e envio dos parâmetros.

```
// criar intent
val intent = Intent(context, TelaInicialActivity::class.java)
// enviar parâmetros simplificado
intent.putExtra("numero", 10)
// fazer a chamada
startActivity(intent)
```

- Na recuperação dos parâmetros.
  - Para Int, é preciso informar o valor padrão no segundo parâmetro;

```
val numero = intent.getIntExtra("numero", 0)
```

# Simplificando

- Resumo: Para recuperar os parâmetros os passos são os seguintes:
  - Acessar a variável intent.
  - Recuperar o objeto de parâmetros (Bundle) pelo atributo da intent extras.
  - Recuperar os parâmetros desejados de acordo com o tipo.
    - getString(chave).
    - getInt(chave).
    - getDoubleArray(chave).
  - Ou então diretamente da Intent.
    - getStringExtra(chave).
    - getIntExtra(chave, padrao).

# Exercício

---

- Pratique um pouco mais o conteúdo da última aula:
  - Altere o texto do campo de Texto da TelaInicialActivity com o nome do usuário enviado como parâmetro.

# Resultado de uma Activity

## Resultado de uma Activity

---

- É possível enviar dados para a Activity anterior quando a Activity atual é finalizada.
- Para isso, deve chamar o método `startActivityForResult`, ao invés de `startActivity`, quando for navegar entre as telas.

**`startActivityForResult(intent, 1);`**

- Repare no segundo argumento. É o `requestCode`, um valor inteiro utilizado para identificar a chamada.

## Resultado de uma Activity

---

- Na segunda Activity, é preciso informar que algo será retornado.
- Para praticar, vamos criar um botão “Sair” na view da TelaInicialActivity (activity\_tela\_inicial.xml) e enviar uma mensagem “Saída do LMSApp” para mostrar na tela de login.
- Trate o evento de clique no botão e no método onClick

## Resultado de uma Activity

```
override fun onCreate(savedInstanceState: Bundle?) {
 // código do onCreate
 botaoSair.setOnClickListener {
 cliqueSair()}
 }
 fun cliqueSair() {
 val returnIntent = Intent();
 returnIntent.putExtra("result", "Saída do LMSApp");
 setResult(Activity.RESULT_OK, returnIntent);
 finish();
 }
}
```



## Resultado de uma Activity

---

- Primeiro é preciso criar uma nova Intent.
- Nessa intent, coloque os dados que deseja retornar, utilizando o padrão chave=valor.
- Chame o método setResult, passando o tipo de resultado e a Intent.
- Por último, chame o método finish().
  - Este método força a destruição da Activity e volta para a Activity anterior.

## Resultado de uma Activity

---

- Os dados enviado pela Activity destruída podem ser recuperados pela Activity que fez a primeira chamada.
- Para isso, ela deve sobrescreve o método onActivityResult.
- Volte para a MainActivity e sobrescreva este método para mostrar os dados enviados no Toast, da seguinte forma.

# Resultado de uma Activity

```
override fun onActivityResult(requestCode: Int,
 resultCode: Int,
 data: Intent?) {
 if (requestCode == 1) {
 val result = data?.getStringExtra("result")
 Toast.makeText(context, "$result",
 Toast.LENGTH_LONG).show()
 }
}
```



**Obrigado!**

---

Prof. MSc. Antônio Catani  
[antonio.catani@faculdadeimpacta.com.br](mailto:antonio.catani@faculdadeimpacta.com.br)