Desenvolvimento de Apps para Mobile

Fabrício Tonetto Londero

Aula 02 – Ciclo de vida de uma Activity

View

- Bloco básico de construção para componentes de interface com o usuário;
- Uma **view** ocupa uma área retangular na tela e é responsável por desenhar e manipular eventos;
- É a classe básica de *widgets*, que são utilizados para a criação de componentes gráficos interativos (botões, campos de texto, etc.);

Activity

- De acordo com a documentação do Android, o conceito de Activity é:
  - "*An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with* ***`setContentView(View)`***"
  - "Uma *activity* é uma coisa única e focada que o usuário pode fazer. Quase todas as *activities* interagem com o usuário, então a classe Activity cuida de criar uma janela para você na qual você pode colocar sua UI com setContentView(View)"
- Portanto, uma Activity faz referência, principalmente, a uma tela de uma aplicação.

Ciclo de vida de uma Activity

- O **ciclo de vida** de uma activity irá definir os **possíveis estados** em que ela se encontra:
  - Executando, temporariamente interrompida, segundo plano ou completamente destruída;

- O **Sistema Operacional** é **responsável** por **cuidar** deste ciclo de vida;

- Porém, ao desenvolver **aplicações** mais **robustas**, é importante levar cada estado possível em consideração.

Ciclo de vida de uma Activity

- Exemplo:
  - Você desenvolveu um jogo para Android, e enquanto o **usuário está jogando**, **alguém** faz uma **ligação** para ele;
  - Nesse caso, o **usuário** irá **parar** para **atender** a ligação, mas **o que acontece com o Jogo**?
  - O **Sistema Operacional** colocará o jogo em **segundo plano** enquanto o **usuário atende a ligação**.
  - Depois de terminar a ligação, o sistema operacional reiniciará a aplicação do jogo.
  - A grande pergunta é: o **jogo** vai conseguir **continuar** da **onde parou?** O estado e informações do jogo **foram salvos**?

Ciclo de vida de uma Activity

- Uma activity tem um **ciclo de vida bem definido**;
- Quando ela é **iniciada**, é **inserida** no **topo de uma pilha** chamada "*activity stack*";
- Sempre que uma nova activity é inserida no topo da pilha, a activity anterior que estava em execução fica logo abaixo da nova;
- A **activity** que está **no topo da pilha** é a activity que **está em execução** no momento, e as **demais podem estar executando** em **segundo plano**, estar no estado **pausado** ou totalmente **paradas**.

Estados de uma Activity

- Uma activity possui essencialmente 4 estados:
  - Se uma activity está em **primeiro plano**, então ela está **ativa** ou **executando**;
  - Se uma activity perdeu o foco mas ainda está visível (ou seja, uma outra activity que não esteja ocupando toda a tela ou que seja transparente), ela se encontra **pausada.** Uma activity pausada é considerada completamente “viva”, porém, ela pode ser destruída pelo sistema operacional em casos extremos de baixa memória;

Estados de uma Activity

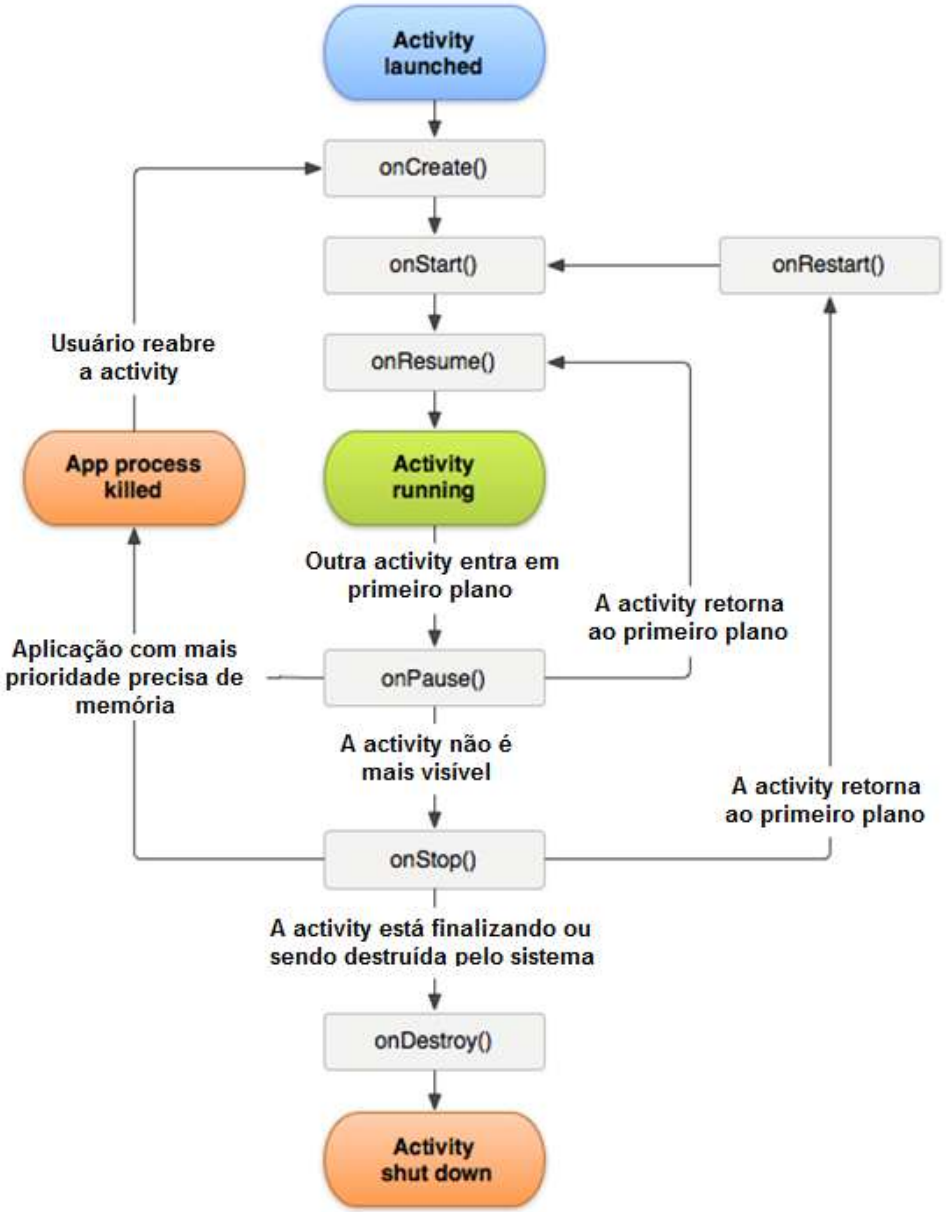
- Se uma activity está completamente obscurecida por outra activity, ela está **parada**. Neste caso, ela ainda mantém seu estado e suas informações, no entanto, já não é mais visível para o usuário, e muitas vezes o sistema operacional a destrói quando a memória é necessária em outras aplicações.

Ciclo de vida de uma activity

- Na **prática**, existem alguns **métodos** da **classe Activity** que podem ser utilizados para **controlar o estado** da aplicação;

- Já aprendemos a utilizar um dos estados: o **`onCreate(bundle)`**, que é chamado quando aquela activity está inicializando.

- Porém, existem outros:
  - `onCreate(bundle), onStart(), onRestart(), onResume(), onPause(), onStop(), onDestroy.`


Activity launched
onCreate()
onStart()
onRestart()
onResume()
Usuário reabre a activity
App process killed
Activity running
Outra activity entra em primeiro plano
A activity retorna ao primeiro plano
Aplicação com mais prioridade precisa de memória
onPause()
A activity não é mais visível
A activity retorna ao primeiro plano
onStop()
A activity está finalizando ou sendo destruída pelo sistema
onDestroy()
Activity shut down

Ciclo de vida de uma activity

- **`onCreate(bundle)`**:
  - É obrigatório e é chamado uma única vez. Neste método deve-se criar uma View e chamar o método **`setContentView(view)`** para exibi-la na tela. Logo após a finalização do método, o método **`onStart()`** é chamado automaticamente;

- **`onStart()`**:
  - É chamado quando a activity está ficando visível ao usuário e já tem uma view associada. Pode ser chamado depois dos métodos **`onCreate()`** ou **`onRestart()`**, dependendo do estado da aplicação;

- **`onRestart()`**:
  - É chamado quando uma activity fica parada temporariamente e está sendo iniciada outra vez. O método **`onRestart()`** chama o método **`onStart()`** automaticamente.

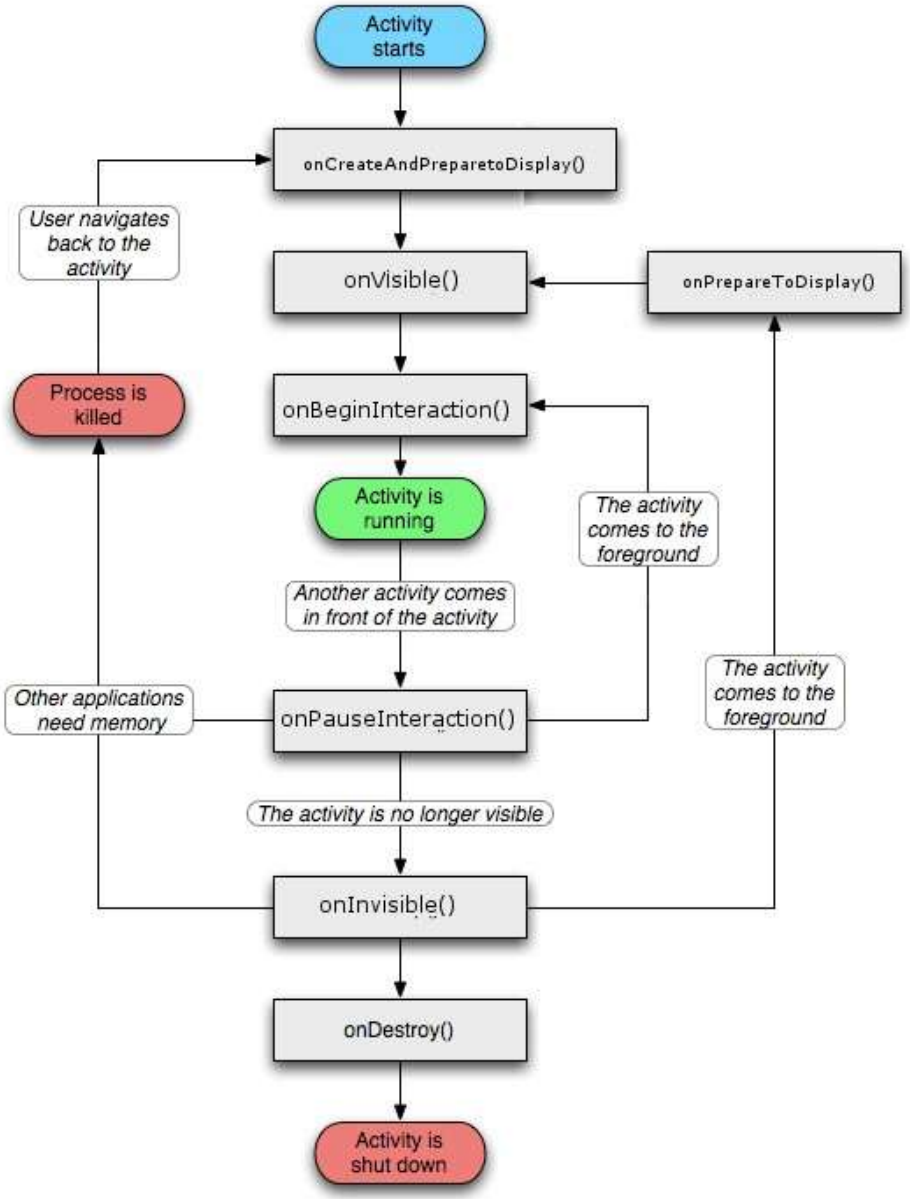
Ciclo de vida de uma activity

- **`onResume()`**:
  - É chamado quando a activity está no topo da pilha. Podemos dizer que este método representa o estado de que a activity está executando e começa sua interação com o usuário.

- **`onPause()`**:
  - Se ocorrer algum evento (celular entrar em modo de espera para economizar energia), a activity do topo da pilha pode ser temporariamente interrompida com o método **`onPause()`**, salvando o estado da aplicação, para que posteriormente quando a activity voltar a executar, tudo possa ser recuperado no método **`onResume()`**;

Ciclo de vida de uma activity

- **`onStop()`**:
  - É chamado quando a **activity está sendo encerrada ou colocada em segundo plano**, mas não está mais visível ao usuário. Por exemplo, quando abrimos outra aplicação, ou em uma chamada ao método **`finish()`**;
  - Caso ela seja reiniciada, o método **`onRestart()`** é executado;
  - Caso ela fique muito tempo parada e o Sistema Operacional precisar limpar os recursos para liberar memória, e o método **`onDestroy()`** será automaticamente chamado para remover completamente a activity da pilha;
- **`onDestroy()`**:
  - Encerra a execução de uma activity. Pode ser chamado automaticamente pelo SO para liberar recursos ou ser chamado pela aplicação através do método **`finish()`**.

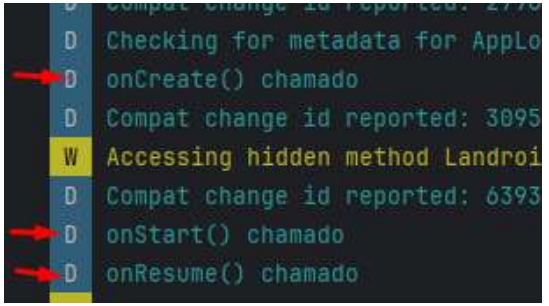

Activity starts
onCreateAndPreparetoDisplay()
User navigates back to the activity
onVisible()
onPrepareToDisplay()
Process is killed
onBeginInteraction()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPauseInteraction()
The activity comes to the foreground
The activity is no longer visible
onInvisible()
onDestroy()
Activity is shut down

Exemplo

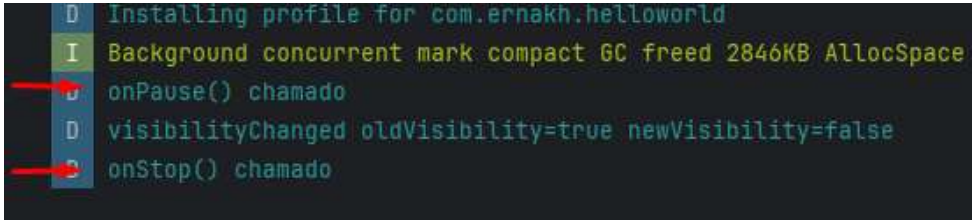
- No exemplo que iremos elaborar, utilizaremos o LogCat. Certifique-se de que o mesmo está visível no seu **AndroidStudio** com o seu emulador já aberto ou **debugando** uma aplicação:

```java
@Override
protected void onCreate(Bundle savedInstanceState){
   super.onCreate(savedInstanceState);
   Log.d("CicloDeVida", "onCreate() chamado");

   ...
}

@Override
protected void onStart(){
   super.onStart();
   Log.d("CicloDeVida", "onStart() chamado");
}

@Override
protected void onRestart(){
   super.onRestart();
   Log.d("CicloDeVida", "onRestart() chamado");
}

@Override
protected void onResume(){
   super.onResume();
   Log.d("CicloDeVida", "onResume() chamado");
}

@Override
protected void onPause(){
   super.onPause();
   Log.d("CicloDeVida", "onPause() chamado");
}

@Override
protected void onStop(){
   super.onStop();
   Log.d("CicloDeVida", "onStop() chamado");
}

@Override
protected void onDestroy(){
   super.onDestroy();
   Log.d("CicloDeVida", "onDestroy() chamado");
}
```

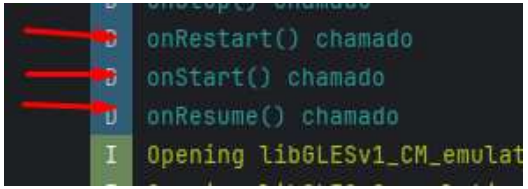
Ao **executar** a aplicação



Ao clicar em **home**



Ao **reabrir**:



Ciclo de vida de uma activity

- Ao **executar a aplicação** anterior, observe que os métodos `onCreate()`, `onStart` e `onResume()` foram chamados na sequência.

- Ao clicar o **botão voltar** (back) do emulador (para **sair da activity**), os métodos `onPause()`, `onStop()` **e** `onDestroy()` são chamados para **encerrar o ciclo** de vida da Activity;

- Ao **entrar novamente** na aplicação através do emulador, novamente os métodos `onCreate()`, `onStart()` e `onResume()` são chamados;

Ciclo de vida de uma activity

- Ao **clicar em home** para voltar a tela inicial, a activity é minimizada, e os métodos `onPause()` **e** `onStop()` são chamados, mas não o método `onDestroy()`;

- Isso acontece porque a **activity não foi completamente destruída**, apenas **retirada** do **topo** da **pilha**, estando **parada** em **segundo plano**.

- Ao **voltar a tela inicial** pelo ícone da aplicação no emulador, a **activity irá reiniciar**, fazendo com que ela volte ao topo da pilha. Assim, os métodos `onRestart()`, `onStart()` **e** `onResume()` são chamados.

Ciclo de vida de uma activity

- Observações:
  - O método `onCreate(bundle)` é chamado **uma única vez**;
  - Se a activity **já estiver executando** em segundo plano, o método `onCreate(bundle)` **não é chamado** novamente.
  - O mesmo ocorre com o método `onDestroy()` que **encerra o ciclo de vida.**

Ciclo de vida de uma activity

- O método `onCreate(bundle)` também recebe um parâmetro do tipo Bundle que é utilizado pelo sistema operacional para que, ao **recriar uma Activity**, ele possa restaurar o seu estado anterior antes de ser destruída:
  - por exemplo, quando o usuário gira a tela.

- Para salvar dados antes da Activity ser destruída, precisamos implementar o método `onSaveInstanceState()`, que é chamado antes de sua destruição. Ele também recebe por parâmetro um tipo `Bundle`, e é nesse parâmetro que salvamos as informações que desejamos.

Ciclo de vida de uma activity

- Observação:
  - O estado de uma *activity* só será salvo se a *activity* for **destruída pelo sistema operacional**, tais como nos seguintes casos:
    - Quando o usuário gira a tela (a *activity* é destruída e recriada)
    - Quando a *activity* entra em segundo plano, e o sistema operacional decide matá-la para liberar memória.
  - Quando o usuário clica no botão “voltar”, a *activity* é destruída mas **nenhum estado é salvo**!

Ciclo de vida de uma activity

- Ex:

```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
        // Salva o estado atual do jogador
        savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
        savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);
        // Invoca a super classe, para que seja possível salvar o estado
        super.onSaveInstanceState(savedInstanceState);
}
```

Ciclo de vida de uma activity

- Ex:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        // Invoca a super classe
        super.onCreate(savedInstanceState);
        // Verifica se estamos recriando uma instancia que foi destruída
        if (savedInstanceState != null) {
                // Restaura os valores do estado salvo
                mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
                mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
        } else {
                // Provavelmente inicializa as variáveis com valores padrão
        }
        ...
}
```