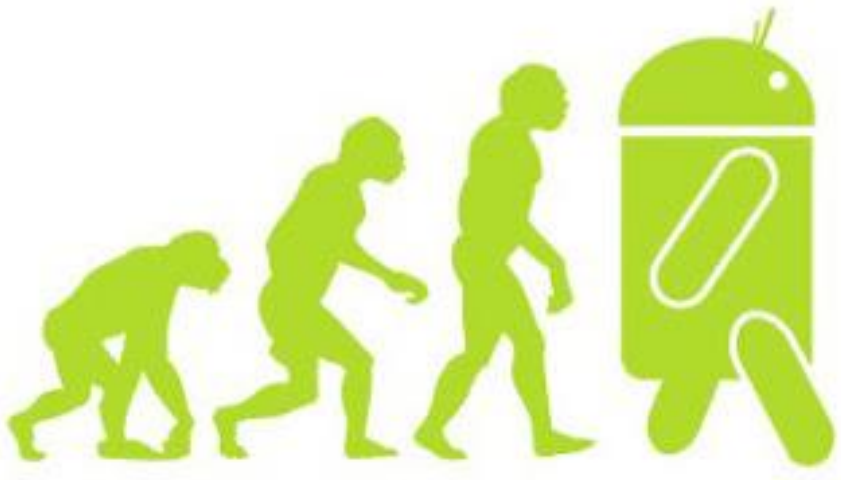
Programação para Android

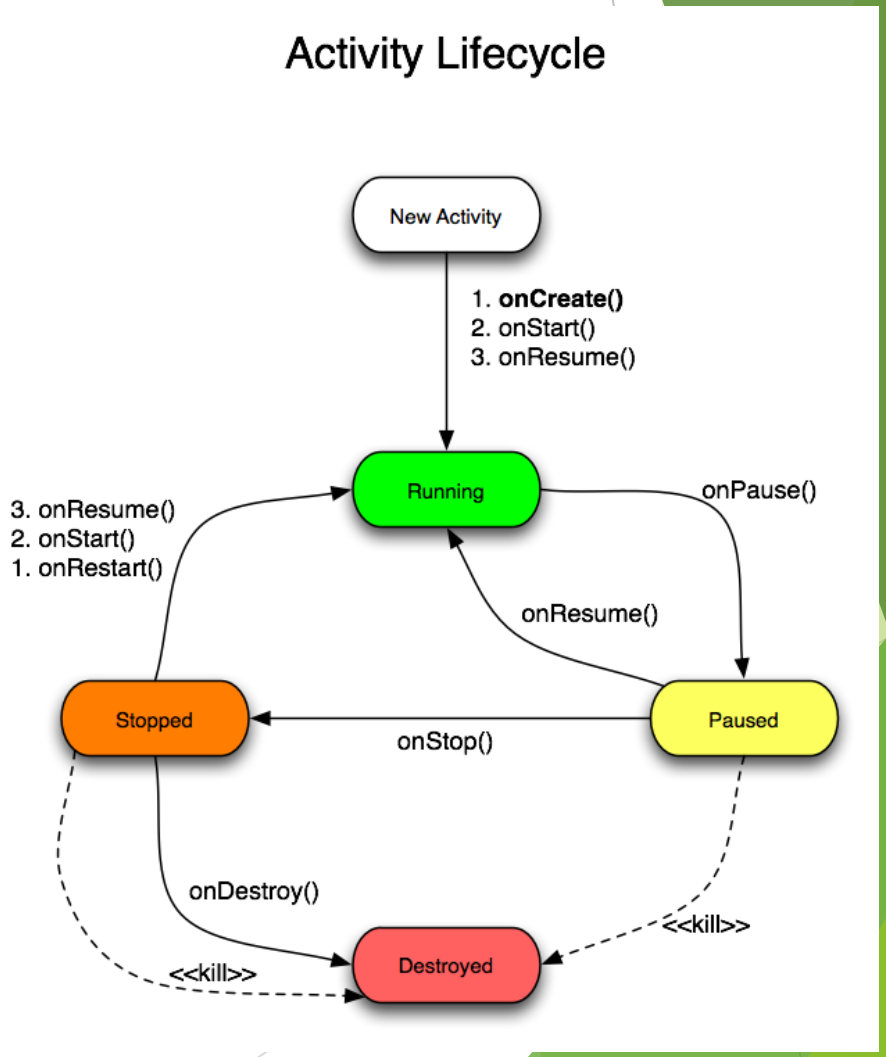
Aula 06: Activity, menus e action bar

Activity

- A classe Activity é quem gerencia a interface com o usuário. Ela quem recebe as requisições, as trata e processa.
- Na programação de aplicativos para Android, uma Activity está relacionada a uma Tela, mas é preciso entender também o ciclo de vida de uma Activity.
- Imagine que nossa aplicação seja interrompida de forma inesperada, seja por que o usuário abriu uma outra Activity e com isso houve algum fator que fez a Activity ser fechada, e até mesmo quando o Android finaliza a mesma quando vê a necessidade de liberar memória. Desta forma é preciso entender cada ciclo de vida de uma Activity para realizar a programação necessária a cada ciclo de uma Activity.
- Uma Activity possui vários estados em seu ciclo de vida, conheceremos os diversos estados nos slides a seguir.

Activity – Ciclo de Vida





Activity – Ciclo de Vida

- Métodos de uma Activity
  - *onCreate()* - É a primeira função a ser executada em uma Activity. Geralmente é a responsável por carregar os layouts XML e outras operações de inicialização. É executada apenas uma vez.
  - *onStart()* - É chamada imediatamente após a onCreate() – e também quando uma Activity que estava em background volta a ter foco.
  - *onPause()* - É a primeira função a ser invocada quando a Activity perde o foco (isso ocorre quando uma nova Activity é iniciada).
  - *onRestart()* - Chamada imediatamente antes da onStart(), quando uma Activity volta a ter o foco depois de estar em background.

Activity – Ciclo de Vida

- Métodos de uma Activity
  - onResume() - Assim como a onStart(), é chamada na inicialização da Activity e também quando uma Activity volta a ter foco. Qual a diferença entre as duas? A onStart() só é chamada quando a Activity não estava mais visível e volta a ter o foco, a onResume() é chamada nas “retomadas de foco”.
  - onDestroy() - A última função a ser executada. Depois dela, a Activity é considerada “morta” – ou seja, nao pode mais ser relançada. Se o usuário voltar a requisitar essa Activity, um novo objeto será contruído.

Activity - Criação

- Para se criar telas no Android, é necessário trabalhar com o conceito de activity. Assim como em programação Windows Desktop que você trabalha com form, onde cada um do mesmo equivale a uma tela. No Android aplica-se o termo de activity.
- Uma aplicação Android pode ter várias Activity´s, porém o usuário só interage com uma de cada vez. As inicializações e configurações da interface com o usuário devem ser feitas sempre no método onCreate(), pois ele é o primeiro método a ser executado quando a Activity é iniciada.
- Para criar uma activity basta criar uma classe no Java e herdar da classe android.app.Activity. Desta forma automaticamente já há torna uma possível atividade no Android.
- Após estendermos a classe é obrigatório rescrever o método **onCreate** para selecionar um arquivo XML, que será seu layout.

Activity - Criação

- No código Java, após estender a classe Activity devemos informar qual o layout relacionado conforme código abaixo:

```
public class Janela2Activity extends Activity
{
  @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Especificamos abaixo qual o layout
        setContentView(R.layout.janela2);
    }


}
```

Registrando Activity

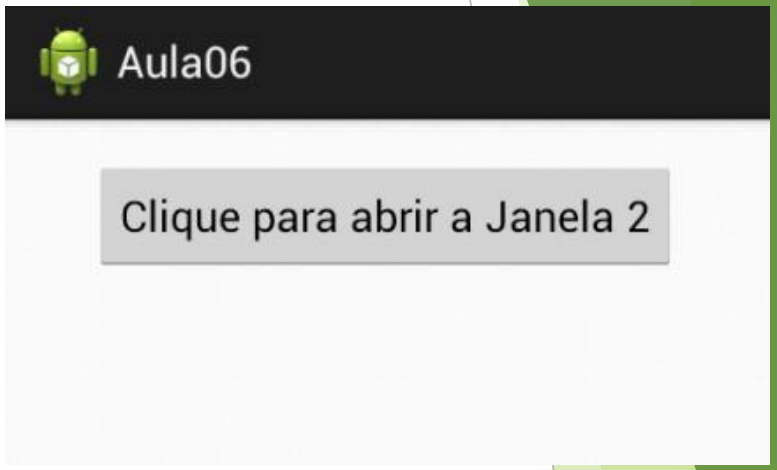
- Após criarmos a classe é necessário registrar a activity. O Android precisa saber quais classes são atividades, **não basta simplesmente criar uma classe e a estender, é necessário registra-la.**
- Para você registrar a activity, basta abrir o arquivo Manifest.XML, e no mesmo deve ser feito o registro da activity, ou seja, incluir uma nova activity.

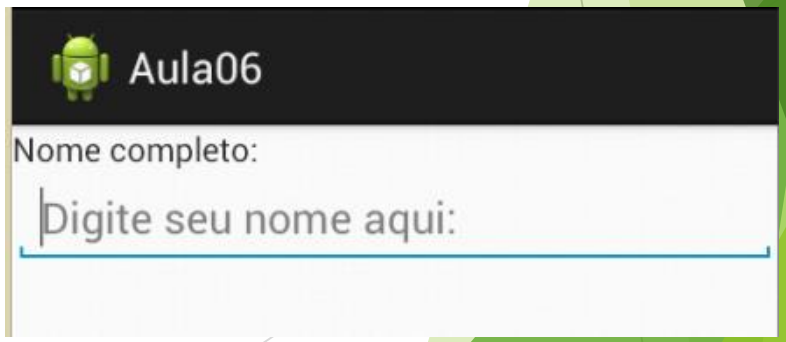
```
<activity
    android:name="com.example.aula06.Janela2Activity"
    android:label="Janela 2">
</activity>
```

- Em android:name é informado o nome da classe.
- OBS: As marcações intente filter é utilizado para definir qual a Activity inicial de nossa aplicação.

Exemplo 01 – Criando múltiplas activities

- Iremos criar duas activities. A classe “MainActivity” é a activity (atividade) principal do projeto e a classe “Janela2Activity" que será chamada pela primeira tela.
- A primeira atividade é simplesmente um botão, que quando clicado abre a segunda atividade, que por sua vez consiste em um simples campo de texto editável.
- **A segunda activity não precisa de um botão "Sair", pois o usuário pode simplesmente clicar no botão "Voltar" do emulador.**





Exemplo 01 – Criando múltiplas activities

- Para carregar uma nova activity devemos criar uma Intent (Intenção). Intent são comandos que podemos enviar ao Sistema Operacional Android para realizar alguma ação.
- Com Intents você podemos enviar ou recuperar dados. O método startActivity(Intent intent) envia dados e requisições de telas, o método getIntent() recupera uma intent enviada por meio do startActivity(), o método putExtra("nome_de_identificação", valor) insere na intent algum valor, entre outras operações.
- Neste exemplo usaremos uma Intente para requisitar uma tela(ativity). O código da Intent para abrir uma activity está programado no Click do botão inserido na MainActivity

```
public void onClick(View v) {

//Cria uma nova intenção passando como parâmetro a classe atual e a nova classe
Intent i = new Intent(MainActivity.this,Janela.class);
//inicia uma atividade passando como parâmetro a inteção criada acima
MainActivity.this.startActivity(i);
}
```

Enviando e recebendo dados entre Activities

- É possível enviar e receber dados entre as activitys. Para enviar dados para uma activity que será carregada usamos o comando ***putExtra*** conforme o exemplo a seguir:

```
Intent dados = new
Intent(MainActivity.this,DadosActivity.class);
dados.putExtra("id", id);
dados.putExtra("nome", nome);
dados.putExtra("telefone", telefone);
```

Enviando e recebendo dados entre Activities

- Para receber os dados enviados por uma activity usamos o comando getStringExtra conforme exemplo a seguir.

```
 //cria uma nova intenção para buscar os dados enviados pela activity anterior
Intent valores = getIntent();
//pega os valores enviados da activity anterior e preenche os campos
String nome,telefone;

nome = valores.getStringExtra("nome");
telefone = valores.getStringExtra("telefone");
```

Menu e action bar

- Um menu é útil para disponibilizar acesso a diferentes partes de uma aplicação.
- Para adicionar um menu a uma aplicação Android devemos criar/utilizar um arquivo XML para os itens do menu.
- Definem-se as opções individuais usando o elemento **item**. Especificam-se os atributos **id**, **icon** e **title** para cada opção.
- **OBS: Durante a criação do projeto, caso o programador tenha optado por um template com ActionBar, uma pasta menu juntamente com um arquivo .XMl já estará criado no projeto.**

Menu e action bar

- Criação de itens de menu no arquivo menu.XML

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
 <item
        android:id="@+id/menuJanela2"
        android:title="Janela 2"/>
 <item
        android:id="@+id/menuJanela3"
        android:title="Janela 3"/>
 <item
        android:id="@+id/menuJanela4"
        android:title="Janela 4"/>
</menu>
```

- O atributo **showAsAction** permite especificar quando e como é que cada opção aparece-se na **ActionBar** (a **ActionBar** é a barra com o título que aparece no topo de uma aplicação Android a partir da versão Honeycomb). Os valores disponíveis são: **ifRoom**, **withText**, **never** e**always**.

Menu e action bar

- Depois é preciso atualizar a **Activity** onde o menu vai aparecer. É preciso criar o método **onCreateOptionsMenu** para construir o menu**. Esta função já está incluída na Activity.**

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the
action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
```

- O método getMenuInflater().infalte é responsável por “inflar” um menu com os itens definidos no arquivo .XML

Menu e action bar

- Usamos o método **onOptionsItemSelected** para programar a opção do menu selecionada pelo usuário

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
if (item.getItemId() == R.id.menuJanela2)
{
Toast.makeText(getBaseContext(), "Você clicou na Janela 2", Toast.LENGTH_SHORT).show();
return true;
}
else if (item.getItemId() == R.id.menuJanela3)
{
Toast.makeText(getBaseContext(), "Você clicou na Janela 3", Toast.LENGTH_SHORT).show();
}
return false;
}
```

Menu e action bar

- Podemos usar os itens do menu para chamar uma Activity, assim como mostrado no botão para chamar a Janela 2. Para isto usamos o método onOptionsItemSelected.

```
public boolean onOptionsItemSelected(MenuItem item)
{
if (item.getItemId() == R.id.menuJanela2)
{
     Intent i = new Intent(MainActivity.this,Janela.class);
     MainActivity.this.startActivity(i);
     return true;
}
else if (item.getItemId() == R.id.menuJanela3)
{
     Intent i = new Intent(MainActivity.this,Janela2Activity.class);
     MainActivity.this.startActivity(i);
     return true;
}

return false;
}
```

Na próxima aula...

- Persistência de dados: Shared Preferences e Internal Storage