Introdução à Programação Android

Prof. Samir Bonho

Aula 4

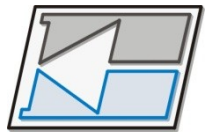
Florianópolis, 19 de Março de 2014

Sumário

- ✓ Activity

- ✓ Fontes:
  - ✓ Departamento de Computação | ICEB | Universidade Federal de Ouro Preto
  - ✓ Android Developers. Disponível em http://developer.android.com/
  - ✓ ECHETA, Ricardo R. **Google Android:** aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2. ed. rev. e ampl. São Paulo: Novatec, 2010.

Activity



Activity

- Activity
  - ✓ Representa uma tela da aplicação.
  - ✓ Controla os eventos e define quem tem direito a desenhar na tela.
  - ✓ Obrigatoriamente deve herdar da classe android.app.Activity (ou qualquer outra subclasse).
  - ✓ Cada *activity* deve implementar o método onCreate(): Inicializações necessárias para carregar uma dada *activity.*

Activity

- Activity

✓ onCreate(): Pode, por exemplo, definir a view a ser utilizada através da chamada ao setContentView(...).

✓ Cada *Activity* deve ser obrigatoriamente declarada no *AndroidManifest.xml*.

✓ Uso da *tag <activity>:*

```
<activity
    android:name="com.example.camera_hardware.MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />

      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
 </activity>
```

Activity

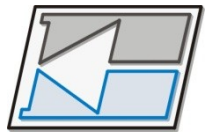
- Ciclo de vida de uma *Activity*

✓ Uma *Activity* é governada através de estados específicos:

- Executando;
- Interrompida;
- Em segundo plano;
- Destruída.

✓ SO é quem coloca/retira a aplicação em cada um dos possíveis estados.

✓ Desenvolvedor deve levar em conta cada um dos estados quando criando uma aplicação Android.



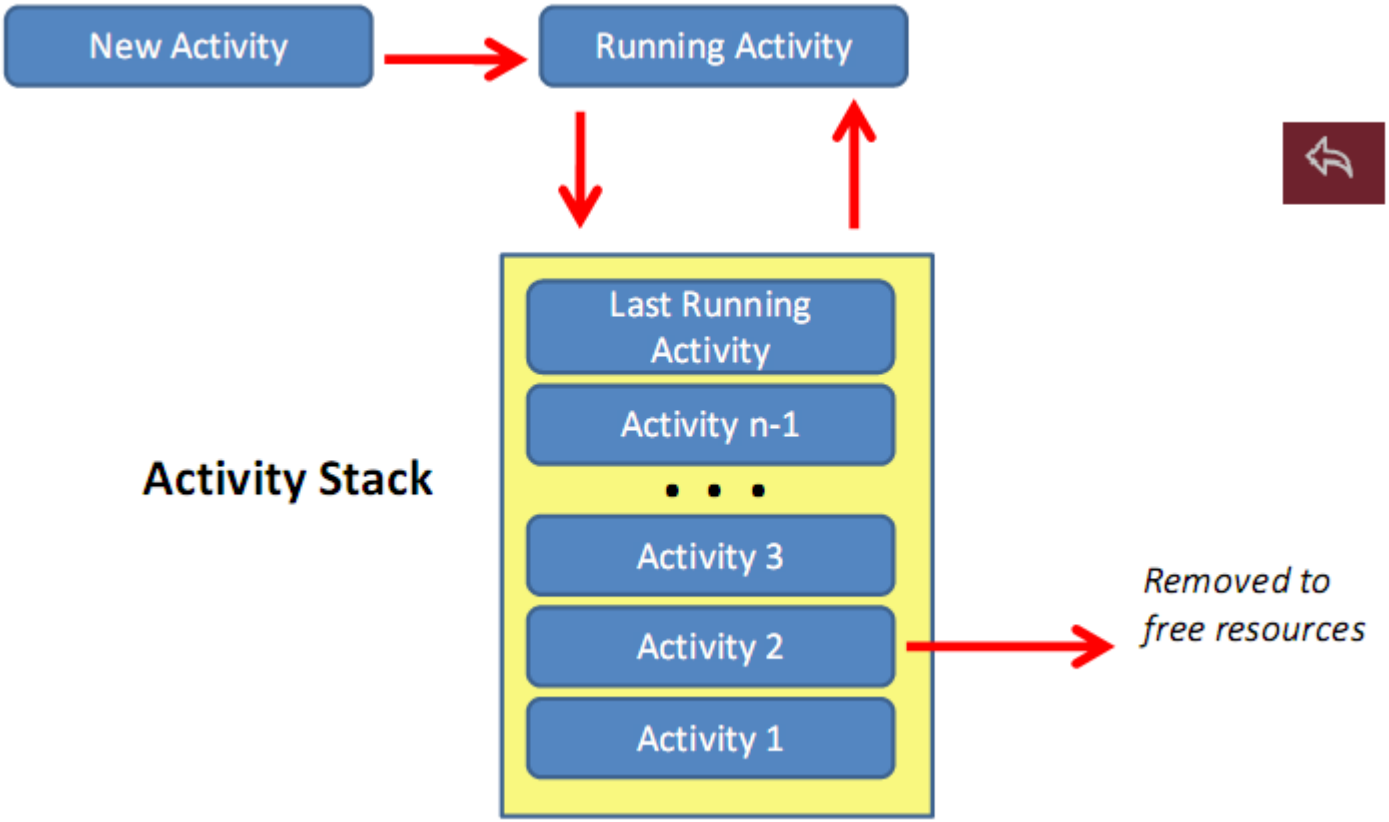
Activity

- Ciclo de vida de uma *Activity*
  - ✓ Caso de exemplo: Usuário assistindo um vídeo no *youtube* recebe uma ligação telefônica.
  - ✓ O que acontece?
  - ✓ Android oferece suporte à troca de aplicações em execução.
  - ✓ O desenvolvedor deve implementar corretamente a aplicação.

Activity

- Ciclo de vida de uma *Activity*

✓ Cada *activity* iniciada é colocada no topo de uma pilha, chamada de "*activity stack*".



Activity

- Ciclo de vida de uma *Activity*

✓Sempre que uma *activity* está pausada o SO pode decidir encerrar o processo para liberar recursos - por exemplo.

✓ O conceito de *activity stack* é aplicado a qualquer aplicação:

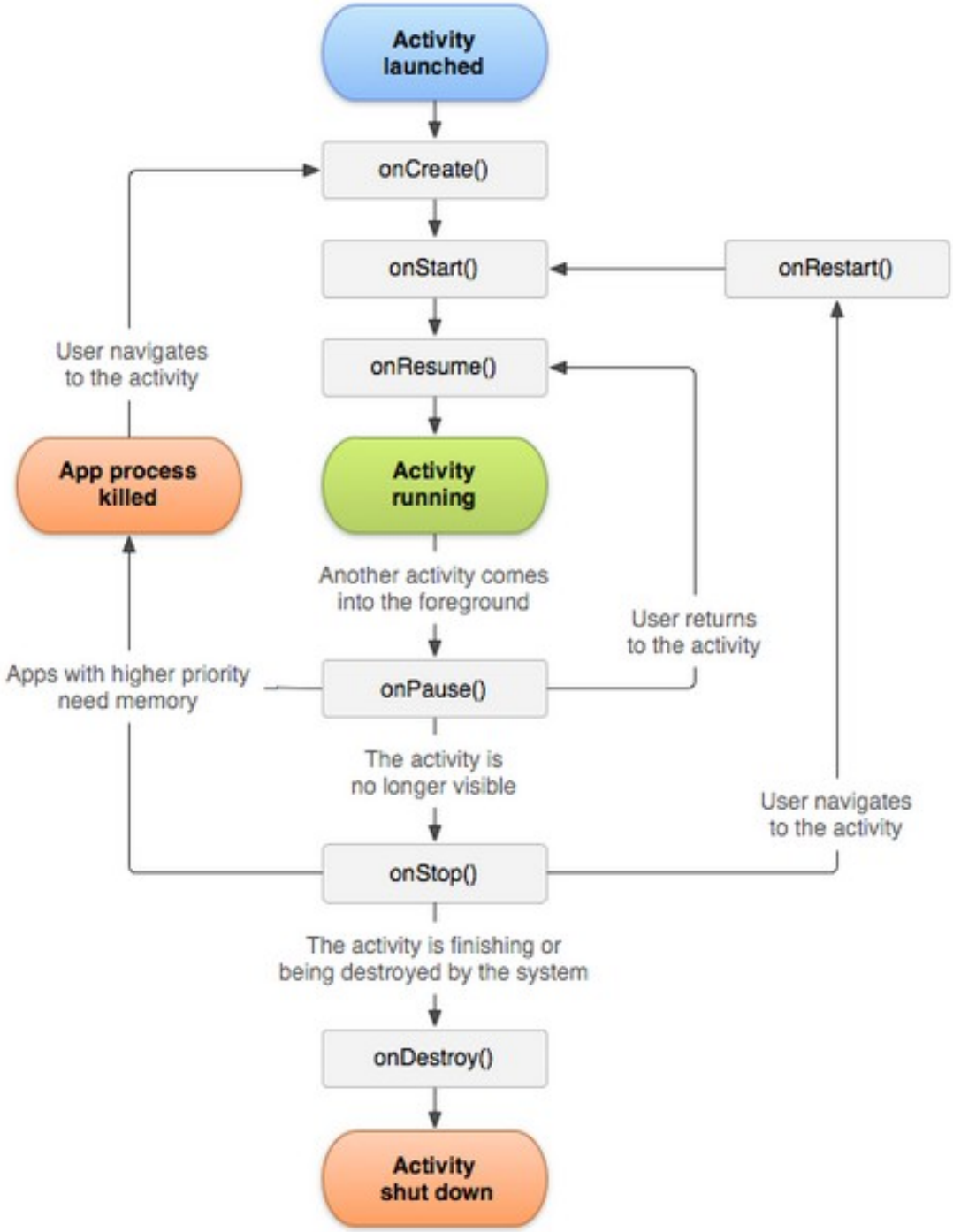
Home, *launcher*, *caller*, jogos, etc.

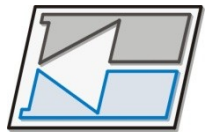
✓ Métodos da classe *Activity* são utilizados para se controlar o ciclo de vida de uma *activity* -> aplicação.

✓ onCreate(), onStart(), onRestart(), onPause(), onResume(),onStop(), onDestroy().

Activity

- Ciclo de vida de uma *Activity*
  - ✓ O ciclo de vida compreende início, meio e fim da *activity.*
  - ✓ Existem três subníveis do ciclo de vida principal durante a execução da aplicação:
    - *entire lifetime;*
    - *visible lifetime;*
    - *foreground lifetime.*
  - ✓ Cada um desses ciclos é iniciado durante a chamada de algum dos métodos que controla o ciclo de vida da aplicação.


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down



Activity

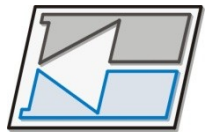
- *Callbacks* - Visão Geral:

✓ onCreate(): Chamado uma única vez durante a criação da *activity*.

✓ onStart(): Chamado quando uma activity se faz visível ao usuário. Pode ser chamado mais de uma vez.

✓ onRestart(): Chamado quando uma activity foi parada anteriormente e está sendo reiniciada.

✓ onResume(): Chamado quando uma activity foi pausada anteriormente e está sendo resumida.



Activity

- *Callbacks* - Visão Geral:

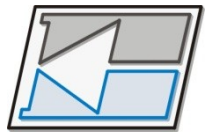
✓ onPause(): Chamado quando uma *activity* em interação com o usuário é interrompida por outra de maior prioridade.

✓ onStop(): Chamado quando uma *activity* está sendo interrompida e não está mais visível ao usuário.

Activity pode ser reiniciada depois de algum tempo: onRestart().

onDestroy() pode ser chamado caso haja necessidade de mais memória no sistema.

✓ onDestroy(): Chamado quando uma *activity* tem seus recursos definitivamente encerrados pelo SO.



Activity

Exercícios:

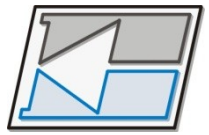
Crie um novo projeto que exiba mensagens no logcat a cada callback chamado. Exemplo:

```
public class MainActivity extends Activity {
        private static final String CATEGORIA = "ANDROID_IFSC";
        private TextView tView;
        @Override
                protected void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    Log.i(CATEGORIA, this.getLocalClassName() + ".onCreate() chamado!");
                   tView = new TextView(this);
                  tView.setText("Exemplo do ciclo de vida. Vide logcats!");
                  setContentView(tView);
}
```

Activity

Exercícios:

```
@Override
protected void onStart() {
super.onStart();
Log.i(CATEGORIA,this.getLocalClassName() + ".onStart() chamado!");
}
@Override
protected void onRestart() {
super.onRestart();
Log.i(CATEGORIA,this.getLocalClassName() + ".onRestart() chamado!");
}
@Override
protected void onPause() {
super.onPause();
Log.i(CATEGORIA,this.getLocalClassName() + ".onPause() chamado!");
}
```

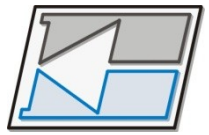


Activity

Exercícios:

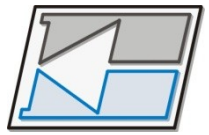
```
@Override
protected void onResume() {
super.onResume();
Log.i(CATEGORIA,this.getLocalClassName() + ".onResume() chamado!");
}
@Override
protected void onStop() {
super.onStop();
Log.i(CATEGORIA,this.getLocalClassName() + ".onStop() chamado!");
}
@Override
protected void onDestroy() {
super.onDestroy();
Log.i(CATEGORIA,this.getLocalClassName() + ".onDestroy() chamado!");
tView = null;
}
```

Trabalhando com várias Activities



Diversas Activities

- Aplicativos mais complexos possuem dezenas de *activities.*
- Necessário considerar o fluxo de navegação entre essas várias *activities.*
- Importante: Para cada nova tela, uma nova *activity*.
- Inicialização de novas *activities*:
  - ✓ startActivity(); e
  - ✓ startActivityForResult().



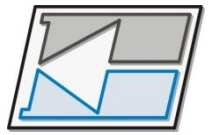
Diversas Activities

- Inicialização de novas *activities*:
  - ✓ startActivity()
    - Inicia a *activity* passada como parâmetro sem que no futuro haja qualquer vínculo com a mesma.
  - ✓ startActivityForResult()
    - Inicia a *activity* passada como parâmetro;
    - Recebe um parâmetro que identifica essa chamada;
    - No futuro, *activity* chamadora consegue recuperar valores da *activity* chamada.
  - ✓ Ambas recebem um objeto da classe android.content.Intent como parâmetro.

Diversas Activities

- Inicialização de novas *activities*:
  - ✓ Exemplo:

```
public class ExemploCicloVidaAbrirTela extends ExemploCicloVida implements
   OnClickListener {
private Button button1;
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   button1 = new Button(this);
   button1.setText("Clique para abrir a nova tela!!");
   button1.setOnClickListener(this);
   setContentView(button1);
   }
public void onClick(View arg0) {
   Intent it = new
   Intent(this,SegundaTela.class);
   startActivity(it);
}…
```



Diversas Activities

- Inicialização de novas *activities*:
  - ✓ Exemplo:

```
public class SegundaTela extends ExemploCicloVida {
@Override
public void onCreate(Bundle savedInstanceState)
        {
   super.onCreate(savedInstanceState);
   TextView view = new TextView(this);
   view.setText("Tela 2!");
   setContentView(view);
   }
}
```

Diversas Activities

✓ Exemplo: . AndroidManifest.xml

```
<activity
        android:name="com.example.app.ExemploCicloVidaAbrirTela"
        android:label="@string/app_name" >
     <intent-filter>
       <action android:name="android.intent.action.MAIN" />
       <category android:name="android.intent.category.LAUNCHER" />
     </intent-filter>
</activity>
<activity android:name="com.example.app.ExemploCicloVida">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
    </intent-filter>
</activity>
<activity android:name="com.example.app.SegundaTela">
   <intent-filter>
    <action android:name="android.intent.action.MAIN" />
</intent-filter>
</activity>
```

Diversas Activities

- Inicialização de novas *activities*:

✓ Uma atividade que fornecerá algum retorno deve ser iniciada com a chamada `startActivityForResult(<intent>,<request_code>)`.

✓ Obtenção dos resultados de outras atividades se baseia em uma chave.

✓ Ao final da execução, callback `onActivityResult(...)` é chamado na primeira atividade.

✓ `request_code` enviado anteriormente deve ser comparado no `onActivityResult(...)` para se ter a noção se é o tipo de valor esperado.

Diversas Activities

- Inicialização de novas *activities*:
  - ✓ Exemplo: . MainActivity.java

```
public class MainActivity extends Activity {
   TextView textView1;
   Button button1;
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   textView1 = (TextView)findViewById(R.id.textView1);
   button1 = (Button)findViewById(R.id.button1);
   button1.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View arg0) {
         Intent intent=new Intent(MainActivity.this,SecondActivity.class);
              StartActivityForResult(intent, 2);// Activity com requestCode = 2
     }
        });
}
```

Diversas Activities

- Inicialização de novas *activities*:
  - ✓ Exemplo: . MainActivity.java (cont)

```
//Callback da segunda atividade para a primeira
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);
//Verifica se o requestCode recebido é o mesmo que o enviado
   if(requestCode == 2) {
      String message = data.getStringExtra("MESSAGE");
      textView1.setText(message);
   }
}

…
```

Diversas Activities

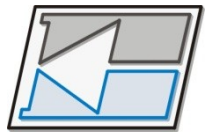
✓ Exemplo: . SecondActivity.java

```
public class SecondActivity extends Activity {
   EditText editText1;
   Button button1;
        protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.second_main);
           editText1 = (EditText)findViewById(R.id.editText1);
            button1 = (Button)findViewById(R.id.button1);
           button1.setOnClickListener(new OnClickListener() {
                public void onClick(View arg0) {
                  String message = editText1.getText().toString();
                   Intent intent = new Intent();
                   intent.putExtra("MESSAGE",message);
                   setResult(2,intent);
                   finish();//terminando a activity
                }
        });
  }
```

Diversas Activities

- Passagem de parâmetros para uma nova *Activity*:
  - Informação a ser passada para uma próxima *Activity* precisa ser encapsulada dentro de um objeto do tipo *android.os.Bundle*.
  - Os dados passados estão sempre em um formato de tabela hash.
    - Ex.: Função que coloca uma *String no bundle* recebe como parâmetros uma *chave* e um *valor*: *putString(chave, valor);*
  - *Para cada tipo de dado existe uma função put diferente:*

putBoolean(), putChar(), putByteArray(), putShort(), putInt(), putLong(), putFloat(), putDouble(), etc.

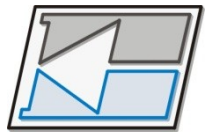


Diversas Activities: passagem de parâmetros

✓ Exemplo:

```
public class ExemploCicloVidaAbrirTela extends
ExemploCicloVida implements OnClickListener {
...
      @Override
      public void onClick(View arg0) {
            Intent it = new Intent(this,ExemploTela2.class);
            Bundle params = new Bundle();
            params.putString("msg", "Mensagem enviada para
outra activity");
            it.putExtras(params);
            startActivity(it);

}
```



Diversas Activities

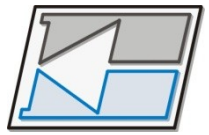
- Passagem de parâmetros para uma nova *Activity*:

  1) Método *getIntent()* é chamado para recuperar a Intent utilizada para chamar essa nova *activity*.

  2) Também, necessário validar se o retorno do método *getIntent()* é válido, i.e., diferente de null.

  3) Através da *Intent* consegue-se recuperar o *Bundle* com as informações passadas pela *activity* anterior: *getExtras()*

  4) Recupera-se o dado à partir do método get específico. No caso: *getString(chave)*.



Diversas Activities

- Passagem de parâmetros para uma nova *Activity*:
  - Para cada tipo de dado existe uma função get diferente: getBoolean(), getChar(), getByteArray(), getShort(), getInt(), getLong(), getFloat(), getDouble(), etc.
  - Classe android.content.Intent é utilizada basicamente para o envio de parâmetros entre duas *activities*.

Diversas Activities: passagem de parâmetros

✓ Exemplo:

```
public class ExemploTela2 extends ExemploCicloVida {
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      TextView view = new TextView(this);
      view.setText("Tela 2!");
      setContentView(view);
      Intent it = getIntent();
      if (it != null) {
            Bundle params = it.getExtras();
            if (params != null) {
            String msg = params.getString("msg");
            Log.i(CATEGORIA,"Mensagem : " + msg);
      } } }
```

Exercícios