CEFET-MG
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Aplicações Móveis

Aula 5 - Ciclo de vida e passagem de parâmetros

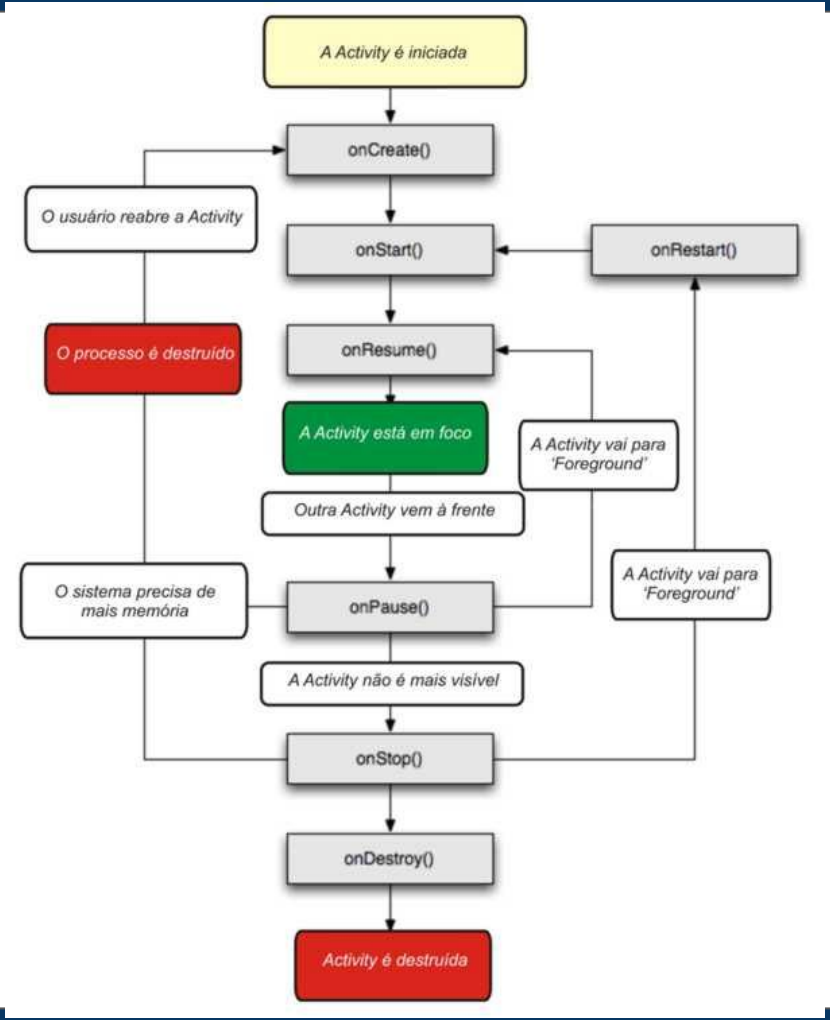
Prof. Dr. Wendell Fioravante da Silva Diniz
3º Ano - Informática Integrado
Centro Federal de Educação Tecnológica de Minas Gerais
Unidade Varginha

Recapitulando...

- Uma Activity é um conjunto de ações que o usuário pode realizar, reunidos em uma interface (tela)
- Pode ser entendido como um gerenciador de interface
- Carrega um layout associado a si
- Responde à interação do usuário nos controles
- Retorna resultados e informa o usuário
- São executadas em um pilha

Ciclo de vida de uma Activity

- Uma Activity pode assumir diversos estados durante sua execução
  - Em execução
  - Pausada
  - Parada
- A transição entre estes estados dispara eventos específicos
- Estes eventos são usados para sincronização e salvamento de dados
- Uma Activity pode ser restaurada, desde-que os dados da sessão estejam salvos


A Activity é iniciada
onCreate()
O usuário reabre a Activity
onStart()
onRestart()
O processo é destruído
onResume()
A Activity está em foco
A Activity vai para 'Foreground'
Outra Activity vem à frente
O sistema precisa de mais memória
onPause()
A Activity vai para 'Foreground'
A Activity não é mais visível
onStop()
onDestroy()
Activity é destruída

Métodos do ciclo de vida

São métodos herdados da classe Activity que são disparados na transição dos estados

- `onCreate(Bundle)` - principal método, é chamado quando a Activity é iniciada. Neste método, deve-se fazer todas as inicializações necessárias à aplicação. Recebe como parâmetro um objeto do tipo `Bundle`. Este objeto serve para empacotar o estado da Activity para que se possa restaurá-lo depois
- `onStart()` - este método é chamado logo antes de a Activity ficar visível. Pode-se usá-lo para tratar os valores passados no `Bundle`. Pode ser seguido por `onResume()` ou `onStop()`

Métodos do ciclo de vida

- `onResume()` - chamado logo antes de a Activity estar pronta para interagir com o usuário. É a responsável de fato a restaurar o estado anterior, se for o caso. Sempre é seguida de `onPause()`
- `onPause()` - é chamado sempre que uma outra Activity for tomar o lugar da Activity ativa. Aqui, devemos cuidar da persistência, ou seja, salvar os dados da sessão para restaurarmos a atividade quando ela voltar a ter o foco.
- `onStop()` - chamado quando a Activity não vai mais ficar visível, isto é, quando ela vai para o segundo plano. Caso a Activity que entrou na frente saia, segue para `onRestart()`. Caso o aplicativo seja fechado, pelo sistema ou pelo usuário, segue para `onDestroy()`

Métodos do ciclo de vida

- `onRestart()` - chamado quando uma Activity que foi parada vai ser restaurada e voltar à execução. É sempre seguida de `onStart()`
- `onDestroy()` - chamado quando a Activity vai ser fechada definitivamente. O método `isFinishing()` retorna true neste ponto, quando a Activity é encerrada pelo usuário ou pela chamada ao método `finish()`. Caso a Activity tenha sido encerrada pelo sistema para liberar memória, o método retorna false. Uma vez destruída, uma Activity é reiniciada no método `onCreate()`

Salvando o estado da aplicação

- Mesmo quando uma Activity é pausada ou parada, seu estado é mantido na memória
- No entanto, o sistema pode destruir a Activity para liberar memória, **sem avisar o usuário**
- No entanto, o sistema deve ser capaz de recriar a Activity exatamente como ela estava, no caso do usuário retornar a ela
- Existem dois métodos específicos para o registro do estado da aplicação

Salvando o estado da aplicação

- `onSaveInstanceState(Bundle)` - chamado quando o sistema vai destruir a Activity, mas **sabe** que deve restaurá-la, por exemplo, quando a tela é bloqueada. **Não** é chamado quando o usuário clica em “voltar”. É chamado antes de `onStop()` ou antes ou depois (varia…) de `onPause()`. Salva os conteúdos das Views identificadas (@+id)
- `onRestoreInstanceState(Bundle)` - chamado logo após o usuário solicitar novamente a Activity destruída. A restauração também pode ser feita pelo método `onCreate(Bundle)`

Salvando o estado da aplicação

```
1      @Override
2      protected void onSaveInstanceState(Bundle outState) {
3          super.onSaveInstanceState(outState);
4          outState.putInt("numDaSorte",numDaSorte);
5          outState.putInt("tentativas",tentativaAtual);
6          outState.putInt("estado",estado);
7      }
```

Salvando o estado da aplicação

```
1 @Override
2 protected void onRestoreInstanceState(Bundle estadoSalvo) {
3     super.onRestoreInstanceState(estadoSalvo);
4     estado = estadoSalvo.getInt("estado", Idle);
5     tentativaAtual = estadoSalvo.getInt("tentativaAtual", 0);
6     numDaSorte = estadoSalvo.getInt("numDaSorte",gerador.nextInt(10)+1);
7 }
```

Passagem de dados entre Activities

- Cada Activity cuida de seus próprios dados
- Algumas vezes, é preciso passar dados para outra Activity
- Uma Activity pode ser iniciada pelo método startActivity(Intent)
- Um Intent transporta os dados entre as atividades

O que é um Intent?

- Um Intent é uma intenção de se fazer alguma coisa (no caso, uma Activity)
- Um Intent pode ser usado para comunicar dados entre os seguintes componentes:
  - Para iniciar uma Activity
  - Para iniciar um Service
  - Para entregar um Broadcast
- Um Intent contém:
  - Nome do componente
  - Ação
  - Dados
  - Extras
  - Categoria
  - Flags

CEFET-MG

O que é um Intent?

- Para salvar os dados para iniciar a próxima Activity, use o método putExtra(), usando um par chave-valor
- Para recuperar os dados, use o método getStringExtra(String chave) na segunda Activity

Passando dados entre Activities

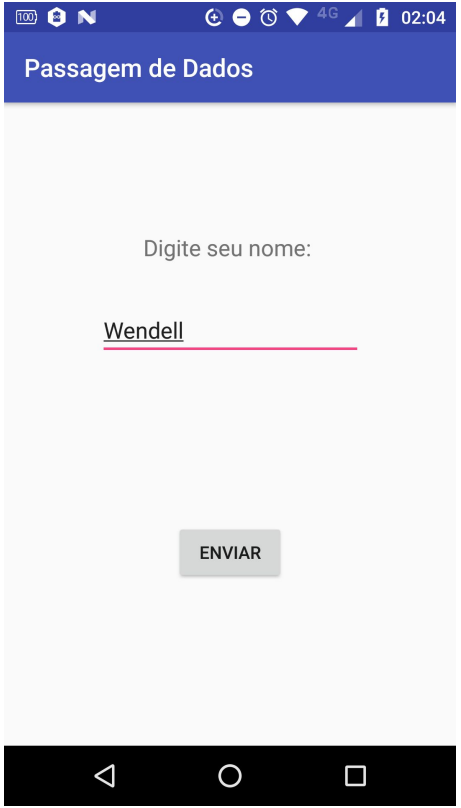
```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.wendell.passagemdedados.MainActivity">

    <TextView
        android:id="@+id/lblMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Digite seu nome: "
        android:textAppearance="@android:style/TextAppearance.Material.Medium"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.211" />

    <EditText
        android:id="@+id/txtNome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.503"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/lblMsg"
        app:layout_constraintVertical_bias="0.095" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:onClick="mandaMensagem"
        android:text="Enviar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtNome" />

</android.support.constraint.ConstraintLayout>
```



Passando dados entre Activities

```java
package com.example.wendell.passagemdedados;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void mandaMensagem(View v) {
        EditText nome = findViewById(R.id.txtNome);
        String oNome = nome.getText().toString();
        Intent i = new Intent(MainActivity.this, Resultado.class);

        Bundle bundle = new Bundle();
        bundle.putString("nome", oNome);
        i.putExtras(bundle);

        startActivity(i);
    }
}
```

CEFET-MG

Passando dados entre Activities

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.wendell.passagemdedados.Resultado">

    <TextView
        android:id="@+id/txtResultado"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="68dp"
        android:text="Olá,!"
        android:textAppearance="@android:style/TextAppearance.Material.Medium"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Passando dados entre Activities

```java
package com.example.wendell.passagemdedados;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class Resultado extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_resultado);

        Intent i = getIntent();

        Bundle b = i.getExtras();

        String nome = b.getString("nome");

        TextView txtResultado = (TextView) findViewById(R.id.txtResultado);

        txtResultado.setText(String.format("Olá, %s!", nome));
    }
}
```

Tipos de Intent

- A Intent que acabamos de ver é do tipo **explícita**
- Intenções explícitas definem um componente que será executado pelo sistema, fazendo referência à sua classe Java correspondente
- São utilizadas, tipicamente, para inicializar outras atividades ou serviços

Tipos de Intent

- Uma Intent **implícita** não faz referência à uma classe Java. Em vez disso, ela especifica um tipo de ação. O sistema verifica se existe um aplicativo capaz de lidar com aquela ação específica
- Os tipos de Intent com as quais uma aplicação pode lidar devem ser declaradas no manifesto da aplicação, usando a tag *<intent-filter>*
  Ex: uma Intent solicitando a abertura de uma url no navegador:

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
```

  uma Intent solicitando a câmera para captura de uma imagem:

```
Intent i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE));
```

Tipos de Intent

- Se existir apenas um aplicativo capaz de lidar com a requisição, ele é executado
- Caso exista mais de um, um seletor é exibido para que o usuário possa escolher qual aplicativo ele deseja usar

Recebendo uma Intent implícita

- Cada componente deve declarar filtros separados para os tipos de ação que podem processar
- Cada filtro é definido usando um ou mais dos elementos a seguir:
  - `<action>` Declara a ação do intent aceito, no atributo `name`. O valor deve ser o valor literal da string de uma ação, e não a constante da classe.
  - `<data>` Declara o tipo de dados aceitos usando um ou mais atributos que especificam diversos aspectos da URI de dados (`scheme`, `host`, `port`, `path`etc.) e do tipo MIME.
  - `<category>` Declara a categoria do intent aceito, no atributo `name`. O valor deve ser o valor literal da string de uma ação, e não a constante da classe.

Recebendo uma Intent implícita

```
1 <activity android:name="ShareActivity">
2     <intent-filter>
3         <action android:name="android.intent.action.SEND"/>
4         <category android:name="android.intent.category.DEFAULT"/>
5         <data android:mimeType="text/plain"/>
6     </intent-filter>
7 </activity>
```

Sub-Activities

- Se são necessários dados que serão gerados em uma Activity que foi iniciada, pode-se usar o método startActivityForResult()
- Neste caso, a Activity iniciada passa a ser uma Sub-Activity
- A Sub-Activity deve ser projetada para retornar um resultado, o que é feito através de um novo Intent
- O Intent é retornado no método onActivityResult(), onde deve-se programar o tratamento do resultado
- Um argumento específico, o requestCode, deve ser passado no Intent, para identificar o retorno

Sub-Activities

```
1 static final int PICK_CONTACT_REQUEST = 1;  // The request code
2 ...
3 private void pickContact() {
4     Intent pickContactIntent = new Intent(Intent.ACTION_PICK, Uri.parse("content://contacts"));
5     pickContactIntent.setType(Phone.CONTENT_TYPE); // Show user only contacts w/ phone numbers
6     startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
7 }
```

Sub-Activities

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.

            // Do something with the contact here (bigger example below)
        }
    }
}
```

SharedPreferences

- SharedPreferences são arquivos contendo pares chave-valor onde é possível salvar dados da atividade
- A API fornece os métodos para ler e gravar
- Útil quando o volume de dados é relativamente pequeno
- Você pode definir vários arquivos identificados por seus nomes

SharedPreferences

- É possível criar um novo arquivo de preferência compartilhada ou acessar um existente chamando um destes dois métodos:
  - getSharedPreferences() — use esse método se precisar identificar vários arquivos de preferência compartilhada por nome, que devem ser especificados com o primeiro parâmetro. É possível chamá-lo de qualquer Context em seu aplicativo.
  - getPreferences() — Use esse método de uma Activity se precisar usar apenas um arquivo de preferência compartilhada para a atividade. Como ele retorna um arquivo de preferência compartilhada padrão que pertence à atividade, não é necessário fornecer um nome.

Criando ou recuperando uma SharedPreference

```
1 Context context = getActivity();
2 SharedPreferences sharedPref = context.getSharedPreferences(
3         getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

Criando ou recuperando uma SharedPreference

```
1 SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

Gravando valores nas SharedPreferences

```
1 SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
2 SharedPreferences.Editor editor = sharedPref.edit();
3 editor.putInt(getString(R.string.saved_high_score), newHighScore);
4 editor.commit();
```

Lendo valores das SharedPreferences

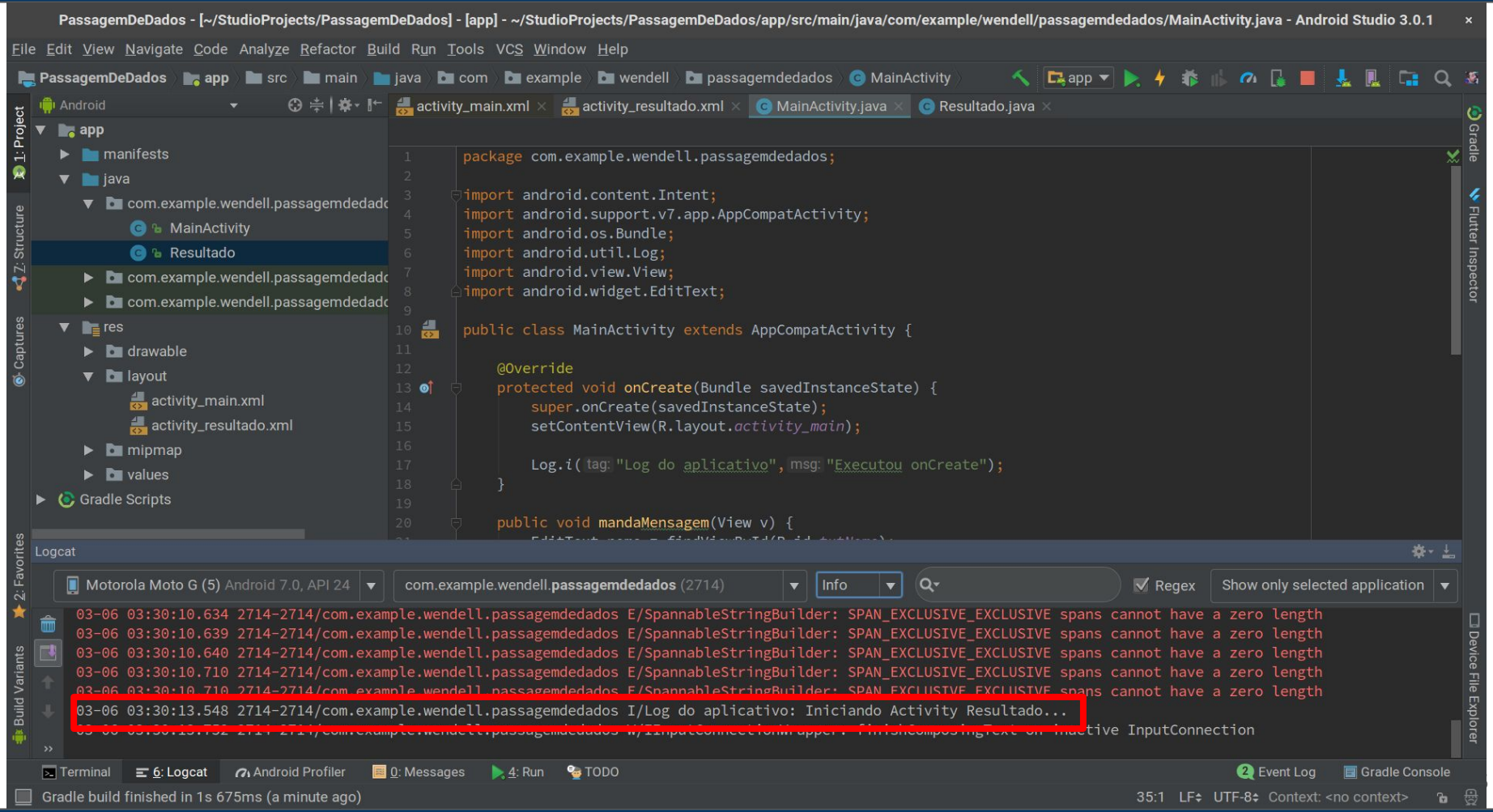
```
1 SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
2 int defaultValue = getResources().getInteger(R.string.saved_high_score_default);
3 long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

Logs do aplicativo

- O SDK do Android possui uma classe utilitária muito útil para enviar informações sobre o desenrolar da aplicação
- As saída de sistema no Android não são enviadas para a saída padrão, logo, métodos como print não funcionam
- O Logcat captura essas saídas e as exibe
- As mensagens são separadas por categoria:
  - Log.v - verbose, nível de saída detalhada
  - Log.d - debug, nível de saída para mensagens de depuração
  - Log.i - info, nível para informações sobre ações da aplicação
  - Log.w - warning, nível usado para aviso de erros não-críticos
  - Log.e - erro, nível para aviso de erros críticos

Logs de aplicativo

```
1  public class MainActivity extends AppCompatActivity {
2
3      @Override
4      protected void onCreate(Bundle savedInstanceState) {
5          super.onCreate(savedInstanceState);
6          setContentView(R.layout.activity_main);
7
8          Log.i("Log do aplicativo","Executou onCreate");
9      }
10
11     public void mandaMensagem(View v) {
12         EditText nome = findViewById(R.id.txtNome);
13         String oNome = nome.getText().toString();
14         if(oNome.matches("")) {
15             Log.d("Log do aplicativo","Nome está vazio");
16         }
17         Intent i = new Intent(MainActivity.this, Resultado.class);
18
19         Bundle bundle = new Bundle();
20         bundle.putString("nome", oNome);
21         i.putExtras(bundle);
22         Log.i("Log do aplicativo","Iniciando Activity Resultado...");
23         startActivity(i);
24     }
25 }
```


PassagemDeDados - [~/StudioProjects/PassagemDeDados] - [app] - ~/StudioProjects/PassagemDeDados/app/src/main/java/com/example/wendell/passagemdedados/MainActivity.java - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PassagemDeDados app src main java com example wendell passagemdedados MainActivity
app
Android
app
manifests
java
com.example.wendell.passagemdedadc
MainActivity
Resultado
com.example.wendell.passagemdedadc
com.example.wendell.passagemdedadc
res
drawable
layout
activity_main.xml
activity_resultado.xml
mipmap
values
Gradle Scripts
activity_main.xml
activity_resultado.xml
MainActivity.java
Resultado.java
1 package com.example.wendell.passagemdedados;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.util.Log;
7 import android.view.View;
8 import android.widget.EditText;
9
10 public class MainActivity extends AppCompatActivity {
11
12 @Override
13 protected void onCreate(Bundle savedInstanceState) {
14 super.onCreate(savedInstanceState);
15 setContentView(R.layout.activity_main);
16
17 Log.i(tag: "Log do aplicativo", msg: "Executou onCreate");
18 }
19
20 public void mandaMensagem(View v) {
Logcat
Motorola Moto G (5) Android 7.0, API 24
com.example.wendell.passagemdedados (2714)
Info
Regex
Show only selected application
03-06 03:30:10.634 2714-2714/com.example.wendell.passagemdedados E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
03-06 03:30:10.639 2714-2714/com.example.wendell.passagemdedados E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
03-06 03:30:10.640 2714-2714/com.example.wendell.passagemdedados E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
03-06 03:30:10.710 2714-2714/com.example.wendell.passagemdedados E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
03-06 03:30:10.710 2714-2714/com.example.wendell.passagemdedados E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
03-06 03:30:13.548 2714-2714/com.example.wendell.passagemdedados I/Log do aplicativo: Iniciando Activity Resultado...
Terminal
6: Logcat
Android Profiler
0: Messages
4: Run
TODO
Event Log
Gradle Console
Gradle build finished in 1s 675ms (a minute ago)
35:1 LF UTF-8 Context: <no context>
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Flutter Inspector
Device File Explorer

Leitura complementar recomendada

- Capítulo 4 do livro **Google Android** de Ricardo Lecheta

CEFET-MG