PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Persistência de dados com Android

Professor: Danilo Giacobo

OBJETIVOS DA AULA

- Apresentar novas técnicas de persistência em Android.
- Utilizar a classe *SharedPreferences*.
- Conhecer o modelo de persistência *PreferenceActivity*.

INTRODUÇÃO

- Hoje é muito difícil pensar na concepção de uma aplicativo, seja ele para a plataforma *Web*, *desktop* ou *mobile*, sem antes imaginar os procedimentos necessários para a persistência de seus dados.
- Persistência simples: pontuação de um jogo, configurações do aplicativo, volume do jogo, nível de dificuldade, etc.
- Persistência sofisticada: múltiplas tabelas relacionadas entre si, integração de diferentes fontes de informação, repositório remoto de dados, etc.
- O armazenamento de dados é um dos pontos principais a ser considerado no desenvolvimento de uma aplicação.

INTRODUÇÃO

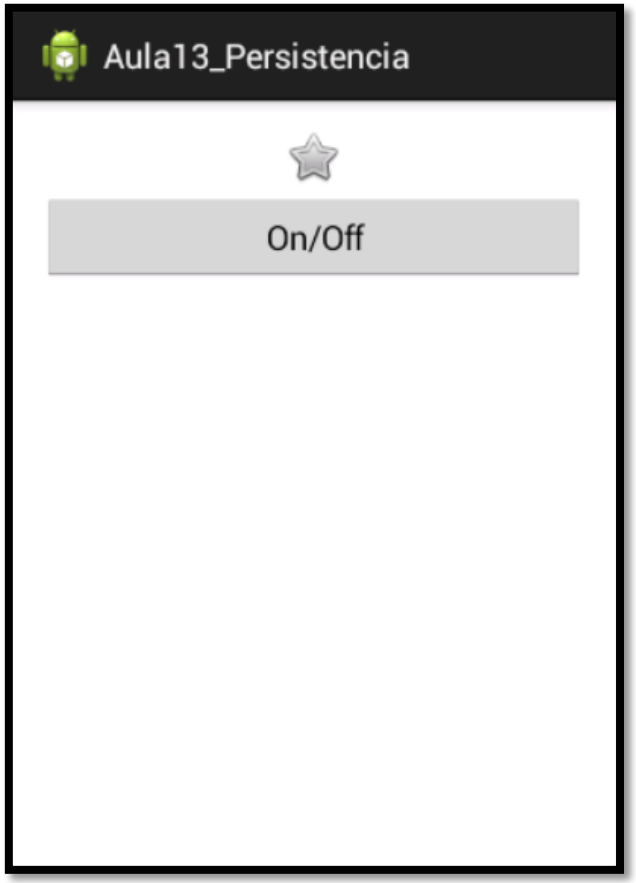
- O surgimento de novas plataformas para o desenvolvimento de aplicativos para a plataforma móvel já concebeu modelos mais sofisticados de persistência.
- Essas ferramentas facilitam o processo de persistência, dão várias opções ao usuário e podem ser adaptadas a um problema específico.
- O SQLite é muitas vezes utilizado para armazenar dados simples, como o conteúdo de campos, configurações do aplicativo e pequenas quantidades de dados, o que não é a situação ideal.
- Por este e outros motivos, o objetivo desta aula é apresentar novas técnicas de persistência, simples e ágeis, que permitem a persistência de pequenas quantidades de dados. São elas: SharedPreferences, PreferenceActivity, Internal Storage, Armazenamento em cache e External Storage.

UTILIZANDO SHARED PREFERENCES

- O Shared Preferences é um *framework* Android que permite armazenar dados do tipo primitivo utilizando o formato chave-valor.
- Ele é recomendado para armazenar as preferências e as configurações de uma aplicação.
- Para exemplificar o uso desse *framework*, será desenvolvido um aplicativo simples, onde uma única informação booleana, que corresponde a uma configuração do aplicativo, será armazenada.
- O código XML da interface gráfica do aplicativo é mostrada no próximo slide.
- O código Java da *Activity* que trata esta tela também será mostrada nos próximos slides.

UTILIZANDO SHARED PREFERENCES

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <ImageView
        android:id="@+id/img"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@android:drawable/btn_star_big_off" />

    <Button
        android:id="@+id/btn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="On/Off"/>

</LinearLayout>
```



UTILIZANDO SHARED PREFERENCES

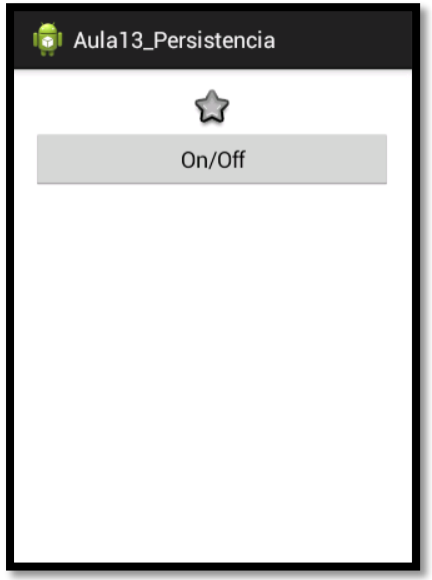
```java
package com.example.aula13_persistencia;

import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends Activity {
    public static final String PREFS_NAME = "prefs";
    private ImageView img;
    private Button btOnOff;
    private boolean online;

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

        img = (ImageView) findViewById(R.id.img);
        btOnOff = (Button) findViewById(R.id.btn);
        btOnOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onOff();
            }
        });
    }
```

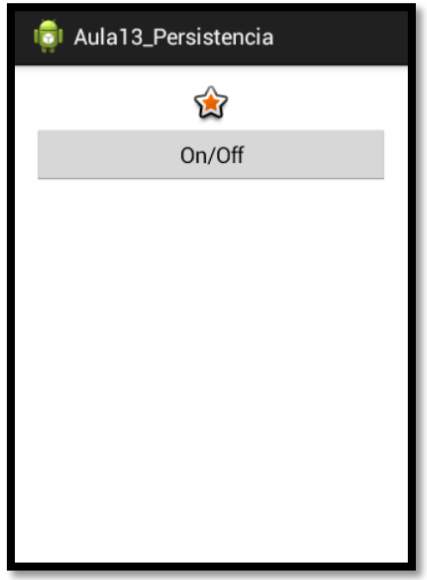
```java
    public void onOff() {
        online = !online;

        if(online) {
            img.setImageResource(android.R.drawable.star_big_on);
        } else {
            img.setImageResource(android.R.drawable.star_big_off);
        }

        setOnline(online);
    }

    protected void setOnline(boolean online) {
        SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
        SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("online", online);
        editor.commit();
    }
}
```

UTILIZANDO SHARED PREFERENCES

- O código anterior está salvando a informação, mas ainda não está a recuperando. Para mudar tal comportamento, insira o código abaixo ao final do método **onCreate();**

```
SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
online = settings.getBoolean("online", false);

if(online) {
    img.setImageResource(android.R.drawable.star_big_on);
} else {
    img.setImageResource(android.R.drawable.star_big_off);
}
```





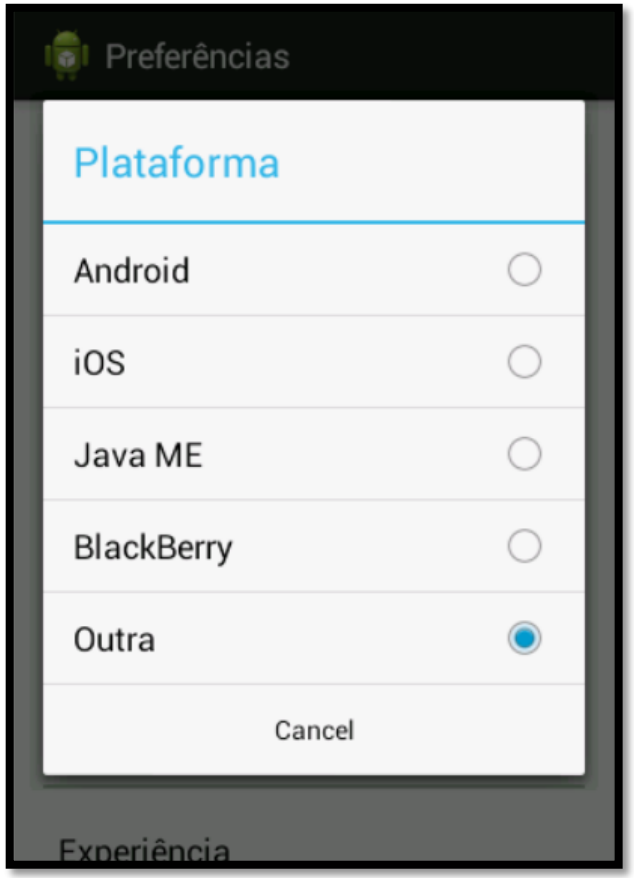
UTILIZANDO PREFERENCE ACTIVITY

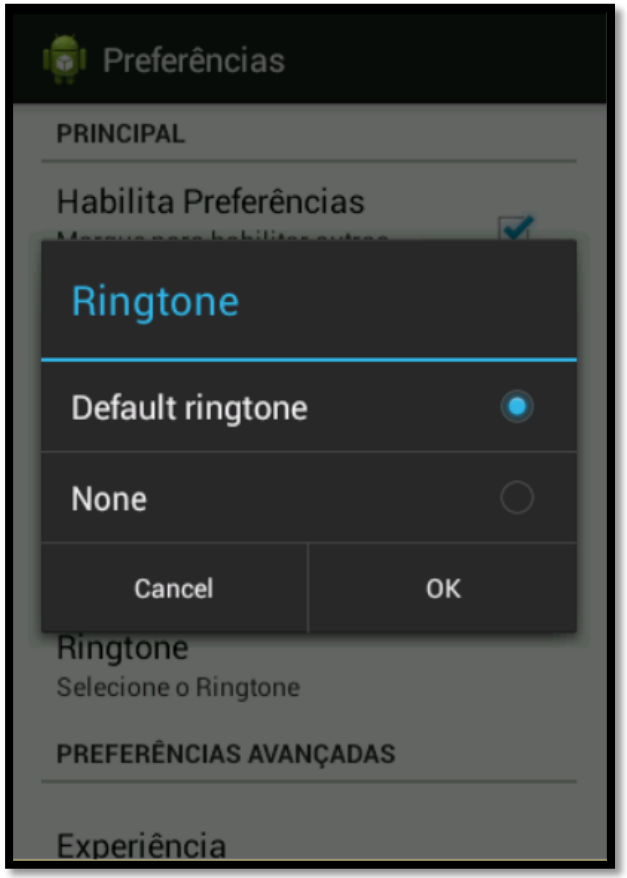
- *PreferenceActivity* é um sistema de persistência bastante interessante, utilizando principalmente nas telas de configurações dos aplicativos Android.
- Ela é uma especialização da classe *Activity* e pode ser utilizada para montar um grupo de configurações, como, por exemplo, o *ringtone* de um aplicativo, se o aplicativo deve executar o áudio ou não, o grau de dificuldade de um jogo, armazenamento local de usuário e senha, entre outros.
- A grande vantagem de *PreferenceActivity* está na facilidade de montar a tela de configuração e a persistência automática dos dados dessa tela, utilizando a *SharedPreferences* mostrada anteriormente.
- A classe *PreferenceActivity* permite mostrar uma hierarquia de *Preferences* definidas em um arquivo XML, ou ainda, por classes que herdem a referida classe.

UTILIZANDO PREFERENCE ACTIVITY

- Dentro da hierarquia de classes de *Preference*, podemos utilizar uma das seguintes classes que correspondem aos componentes visuais de configuração:
  - CheckBoxPreference: é uma caixa de seleção simples, que pode retornar *true* ou *false*;
  - ListPreference: mostra uma caixa de seleção *popup*, onde apenas um item pode ser selecionado. A persistência é realizada utilizando o elemento selecionado na lista;
  - EditTextPreference: mostra uma caixa de diálogo para a digitação de um texto. Retorna uma *string*;
  - RingtonePreference: mostra um *popup* com todos os ringtones existentes no dispositivo;
  - PreferenceScreen: conduz o usuário para uma nova tela de preferências;
  - PreferenceCategory: categoria as preferências.
- Para o exemplo, iremos desenvolver uma tela de configuração, conforme apresentado no próximo slide.

UTILIZANDO PREFERENCE ACTIVITY





UTILIZANDO PREFERENCE ACTIVITY

```xml
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="Principal">
        <CheckBoxPreference
            android:title="Habilita Preferências"
            android:key="EnablePreferences"
            android:summary="Marque para habilitar outras opções" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Plataformas">
        <ListPreference
            android:title="Plataforma"
            android:key="plataforma"
            android:dependency="EnablePreferences"
            android:summary="Selecione a plataforma"
            android:entries="@array/plataformas"
            android:entryValues="@array/plataformasValues" />
        <EditTextPreference
            android:title="Opinião"
            android:key="opiniao"
            android:dependency="EnablePreferences"
            android:summary="Diga sua opinião"
            android:dialogTitle="Diga sua opinião"
            android:defaultValue="" />
        <RingtonePreference
            android:title="Ringtone"
            android:key="Ringtone"
            android:dependency="EnablePreferences"
            android:summary="Selecione o Ringtone"
            android:ringtoneType="all" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Preferências Avançadas">
        <PreferenceScreen android:title="Experiência">
            <EditTextPreference
                android:title="Informe sua experiência"
                android:key="experiencia" />
        </PreferenceScreen>
    </PreferenceCategory>
</PreferenceScreen>
```

UTILIZANDO PREFERENCE ACTIVITY

- Para preencher os valores do campo *Plataforma* um arquivo XML de recurso é utilizado. O seu conteúdo é exibido abaixo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="plataformas">
        <item>Android</item>
        <item>iOS</item>
        <item>Java ME</item>
        <item>BlackBerry</item>
        <item>Outra</item>
    </string-array>
    <string-array name="plataformasValues">
        <item>1</item>
        <item>2</item>
        <item>3</item>
        <item>4</item>
        <item>5</item>
    </string-array>
</resources>
```

arrays.xml

UTILIZANDO PREFERENCE ACTIVITY

- Após o desenvolvimento do XML que representa a interface da tela de configuração, é necessário codificar a classe Java que apresenta essa interface na tela. Seu código poderia ser simplificado da maneira como é apresentado abaixo:

```
1  package com.example.aula13_persistencia;
2
3  import android.os.Bundle;
4  import android.preference.PreferenceActivity;
5
6  public class ExemploPreferenceActivity extends PreferenceActivity {
7
8      @SuppressWarnings("deprecation")
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         //setContentView(R.layout.activity_exemplo_preference);
13         addPreferencesFromResource(R.layout.activity_exemplo_preference);
14     }
15 }
```

UTILIZANDO PREFERENCE ACTIVITY

```java
package com.example.aula13_persistencia;

import android.annotation.TargetApi;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceFragment;

public class ExemploPreferenceActivity extends PreferenceActivity {
    public static int prefs = R.layout.activity_exemplo_preference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            getClass().getMethod("getFragmentManager");
            adiciona11OuSuperior();
        } catch (NoSuchMethodException e) { // API < 11
            adicionaAte11();
        }
    }

    @SuppressWarnings("deprecation")
    protected void adicionaAte11() {
        addPreferencesFromResource(prefs);
    }
```

```java
    @TargetApi(11)
    protected void adiciona11OuSuperior() {
        getFragmentManager().beginTransaction().replace(android.R.id.content, new PF()).commit();
    }

    @TargetApi(11)
    public static class PF extends PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(ExemploPreferenceActivity.prefs);
        }
    }
}
```

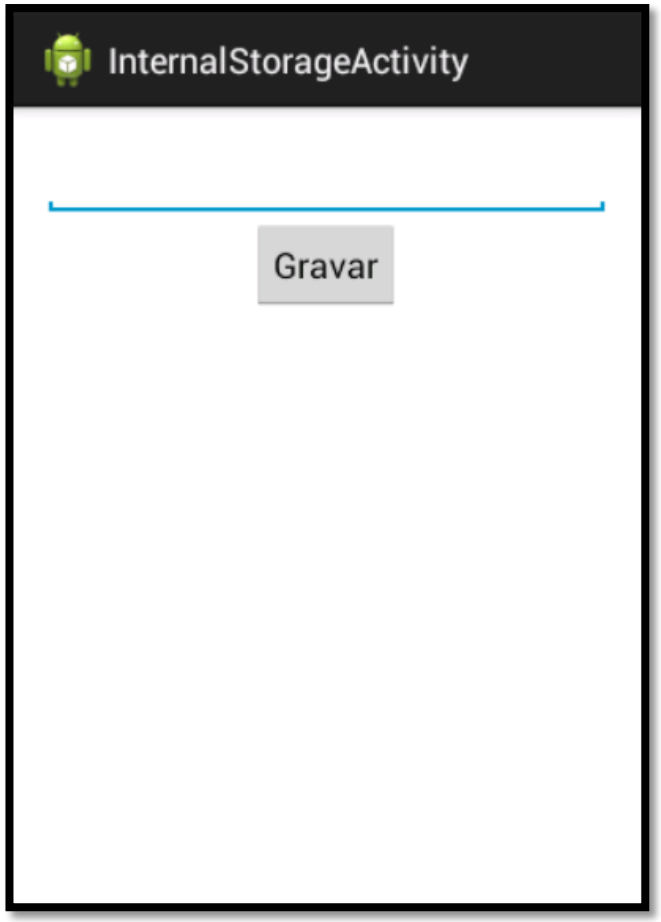
Realize as alterações abaixo para que a tela de preferências seja compatível com todas as versões do Android.

UTILIZANDO O ARMAZENAMENTO INTERNO

- Uma alternativa para o armazenamento de dados na plataforma é o uso do Internal Storage (armazenamento interno), que permite armazenar os dados no sistema de arquivos da memória interna do Android.
- Para exemplificar seu uso, iremos desenvolver uma interface gráfica simples, composta de uma caixa de texto e um botão Gravar.
- O objetivo dessa interface é gravar o texto digitado na caixa de texto em um arquivo e, ao iniciar o aplicativo, recuperar o conteúdo e apresentar na própria caixa de texto.
- O código dessa interface gráfica é apresentado no próximo slide.

UTILIZANDO O ARMAZENAMENTO INTERNO

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".InternalStorageActivity" >

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:id="@+id/edt"
        android:maxLength="20" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btn"
        android:text="Gravar"
        android:layout_gravity="center_horizontal" />

</LinearLayout>
```



UTILIZANDO O ARMAZENAMENTO INTERNO

```java
package com.example.aula13_persistencia;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class InternalStorageActivity extends Activity {
    private EditText edt;
    private Button btn;
    String FILENAME = "hello_file";

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

        edt = (EditText) findViewById(R.id.edt);
        btn = (Button) findViewById(R.id.btn);
```

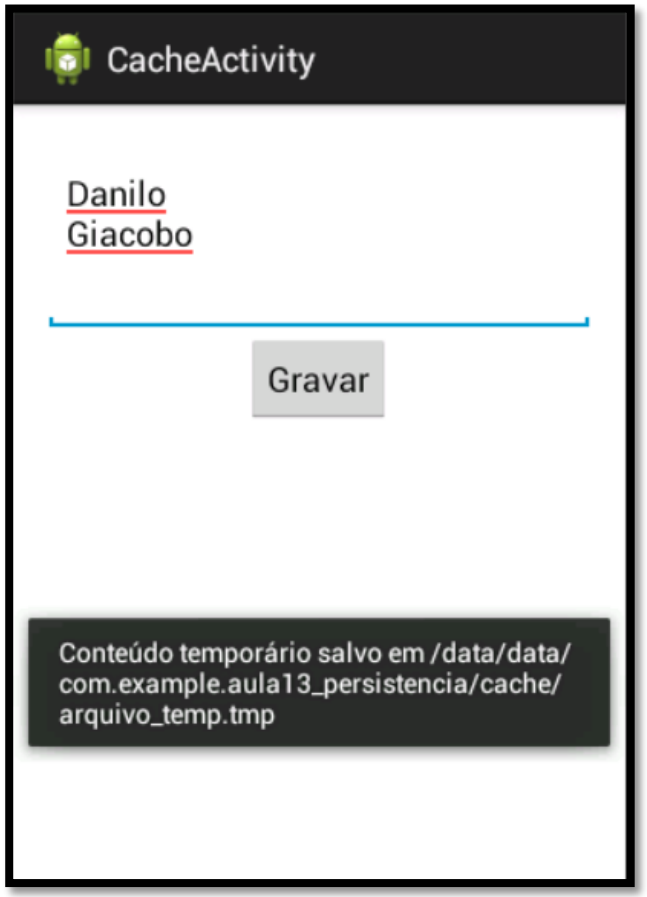
```java
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    FileOutputStream fos = openFileOutput(FILENAME, MODE_PRIVATE);
                    fos.write(edt.getText().toString().getBytes());
                    fos.close();
                }
                catch (FileNotFoundException e) { }
                catch (IOException e) {}
            }
        });

        try {
            byte[] dados = new byte[20];
            FileInputStream fis = openFileInput(FILENAME);
            fis.read(dados);
            edt.setText(new String(dados));
            fis.close();
        }
        catch (FileNotFoundException e) { }
        catch (IOException e) {}
    }
}
```

ARMAZENANDO DADOS TEMPORÁRIOS

- Em algumas situações, o programador necessita guardar os dados de forma temporária em um aplicativo. A forma mostrada anteriormente armazena os arquivos de forma permanente no dispositivo.
- Para utilizar o armazenamento temporário, deve-se utilizar o caminho referenciado pelo método **getCacheDir()**.
- Esse método de persistência tem uma propriedade muito interessante. Quando dispositivo está com pouco espaço de armazenamento interno livre, o Android pode excluir automaticamente esses arquivos de *cache* para liberar mais espaço. No entanto não podemos confiar no sistema para limpar arquivos automaticamente.
- É aconselhável manter uma política de pouca utilização do *cache* (entre 512 KB e 1MB).

ARMAZENANDO DADOS TEMPORÁRIOS

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".CacheActivity" >

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="textMultiLine"
        android:id="@+id/edtDados"
        android:minLines="4"
        android:maxLines="4" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnGravar"
        android:text="Gravar"
        android:onClick="btnGravarOnClick"
        android:layout_gravity="center_horizontal" />

</LinearLayout>
```



ARMAZENANDO DADOS TEMPORÁRIOS

```
1  package com.example.aula13_persistencia;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.io.FileReader;
7  import java.io.FileWriter;
8  import java.io.IOException;
9
10 import android.app.Activity;
11 import android.os.Bundle;
12 import android.view.View;
13 import android.widget.EditText;
14 import android.widget.Toast;
15
16 public class CacheActivity extends Activity {
17     private EditText edtDados;
18     private final String NOME_ARQUIVO_TEMP = "arquivo_temp.tmp";
19     File fArquivoTemp;
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_cache);
25
26         edtDados = (EditText) findViewById(R.id.edtDados);
27         File fDir = getBaseContext().getCacheDir();
28         fArquivoTemp = new File(fDir.getPath() + "/" + NOME_ARQUIVO_TEMP) ;
```

ARMAZENANDO DADOS TEMPORÁRIOS

```
30          String strLinha = "";
31          StringBuilder sbTexto = new StringBuilder();
32
33          /** Lendo o conteúdo do arquivo temporário se ele já existe */
34          try {
35              FileReader fReader = new FileReader(fArquivoTemp);
36              BufferedReader bReader = new BufferedReader(fReader);
37
38              /** Lendo conteúdo do arquivo, linha a linha */
39              while( (strLinha = bReader.readLine()) != null  ){
40                  sbTexto.append(strLinha + "\n");
41              }
42
43              bReader.close();
44
45          } catch (FileNotFoundException e) {
46              e.printStackTrace();
47          }catch(IOException e){
48              e.printStackTrace();
49          }
50
51          /** Colocando conteúdo do arquivo no campo texto */
52          edtDados.setText(sbTexto);
53      }
```

ARMAZENANDO DADOS TEMPORÁRIOS

```
55    public void btnGravarOnClick(View v) {
56        FileWriter writer = null;
57
58        try {
59            writer = new FileWriter(fArquivoTemp);
60
61            /** Salvando os dados no arquivo */
62            writer.write(edtDados.getText().toString());
63
64            /** Fechando o objeto de escrita de arquivos */
65            writer.close();
66
67            Toast.makeText(getBaseContext(), "Conteúdo temporário salvo em " + fArquivoTemp.getPath(), Toast.LENGTH_LONG).show();
68
69        } catch (IOException e) {
70            e.printStackTrace();
71        }
72    }
73 }
```

UTILIZANDO O ARMAZENAMENTO EXTERNO

- Do ponto de vista da programação, o uso do armazenamento externo é muito parecido com o armazenamento interno, porém, tecnicamente esses dois modos de armazenamento são bastante diferentes.
- Enquanto o primeiro armazena informações internamente, usando os recursos de persistência interno do aparelho, o segundo grava informações nos dispositivos externos de persistência, como cartões SD, MMC, repositórios USB, entre outros.
- Como se trata de um armazenamento externo, alguns cuidados devem ser tomados na utilização deste recurso:
  - Os arquivos salvos em dispositivos de armazenamento externo não são de propriedade exclusiva da aplicação, ou seja, a segurança desses dados deve ser levada em consideração na gora de sua utilização.
  - Nessa situação, é sempre necessário verificar se a mídia de armazenamento externo está disponível antes de tentarmos utilizá-la.

UTILIZANDO O ARMAZENAMENTO EXTERNO

- Para apresentar a utilização de External Storage, utilizaremos a mesma interface gráfica apresentada no slide 17, porém, modificando o código da *Activity*.
- O código deste exemplo (apresentado no slide seguinte) é muito parecido com a utilização de Internal Storage, diferenciando apenas no código para a escrita e a leitura dos arquivos.
- Para ler/escrever arquivos em um armazenamento externo, sua aplicação deve possuir permissão para realizar estas operações. Exemplo:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

UTILIZANDO O ARMAZENAMENTO EXTERNO

- Será necessário também verificar se a mídia externa está disponível. Como exemplo, usaremos os métodos abaixo para verificar a disponibilidade da mídia externa.

```
/* Verifica se um dispositivo externo está disponível para leitura e escrita */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Verifica se um dispositivo externo está ao menis disponível para leitura */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

UTILIZANDO O ARMAZENAMENTO EXTERNO