Universidade Federal de Ouro Preto
Departamento de Computação e Sistemas - DECSI

Computação Móvel

Persistência (Ref. Cap. 18)

Vicente Amorim
vicente.amorim.ufop@gmail.com
www.decom.ufop.br/vicente

Sumário

* SharedPreferences

* Trabalhando com Arquivos

* Memórias interna e externa

* Bancos de dados SQLite

* Backup na Nuvem

Introdução

Introdução

- Com a persistência é possível armazenar dados nos dispositivos móveis mesmo quando na falta de alimentação.

- Normalmente, banco de dados são utilizados para tal finalidade por serem flexíveis e de fácil manipulação.

- O Android possui integração com o SQLite, um banco de dados leve e poderoso.

- Como veremos, além do banco de dados, o Android permite ainda que dados sejam persistidos de outras maneiras.


SharedPreferences

SharedPreferences

- Introdução

✓ Internamente, na aplicação, é possível armazenar um conjunto de chaves/valores como uma uma tabela *hash*.

✓ A classe android.content.SharedPreferences salva automaticamente os dados em um banco de dados interno da aplicação.

✓ Deve ser utilizada somente para salvar dados pequenos: tipos primitivos (int, float, pequenas Strings, etc).

✓ A classe Prefs (android-utils) já implementa as principais operações relacionadas a SharedPreferences.

- Introdução

✓Exemplo AppNum60

✦ ActionBar foi descontinuada desde Android 5.0. Então, substituímos por Toolbar.

✦ Neste exemplo será utilizado a Toolbar com Fragments e ViewPager para demonstrar o SharedPreferences.

✦ Removendo o ActionBar através de um novo estilo para a app:

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="colorPrimary">#3F51B5</item>
        <item name="colorPrimaryDark">#303F9F</item>
        <item name="colorAccent">#FF4081</item>
    </style>
</resources>
```

SharedPreferences

- Introdução
  - ✓Exemplo AppNum60

0 - MODE_PRIVATE
1: MODE_WORLD_READABLE
2: MODE_WORLD_WRITABLE

```
public class Prefs {
    public static final String PREF_ID = "CEA436";

    public static void setBoolean(Context context, String chave, boolean valor) {
        SharedPreferences pref = context.getSharedPreferences(PREF_ID,0);
        SharedPreferences.Editor editor = pref.edit();
        editor.putBoolean(chave,valor);
        editor.commit();
    }

    public static boolean getBoolean(Context context, String chave) {
        SharedPreferences pref = context.getSharedPreferences(PREF_ID,0);
        boolean b = pref.getBoolean(chave, true);
        return b;
    }

...
```

SharedPreferences

- Introdução

✓Exemplo AppNum60: TabFragment1.java

```
public class TabFragment1 extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.tab_fragment_1, container, false);
    }
}
```

✓ tab_fragment_1.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView android:id="@+id/textView" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:layout_centerInParent="true"
    android:text="Tab 1" android:textAppearance="?android:attr/textAppearanceLarge"/>
</RelativeLayout>
```

SharedPreferences

- Introdução

✓Exemplo AppNum60: TabFragment2.java

```
public class TabFragment2 extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.tab_fragment_2, container, false);
    }
}
```

✓ tab_fragment_2.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView android:id="@+id/textView" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:layout_centerInParent="true"
    android:text="Tab 2" android:textAppearance="?android:attr/textAppearanceLarge"/>
</RelativeLayout>
```

SharedPreferences

- Introdução

✓Exemplo AppNum60: TabFragment3.java

```
public class TabFragment3 extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.tab_fragment_3, container, false);
    }
}
```

✓ tab_fragment_3.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView android:id="@+id/textView" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:layout_centerInParent="true"
    android:text="Tab 3" android:textAppearance="?android:attr/textAppearanceLarge"/>
</RelativeLayout>
```

SharedPreferences

- Introdução

✓Exemplo AppNum60: PagerAdapter.java

```
public class PagerAdapter extends FragmentStatePagerAdapter {
    int mNumOfTabs;

    public PagerAdapter(FragmentManager fm, int NumOfTabs) {
        super(fm);
        this.mNumOfTabs = NumOfTabs;
    }

    @Override
    public Fragment getItem(int position) {

        switch (position) {
            case 0:
                TabFragment1 tab1 = new TabFragment1();
                return tab1;
            case 1:
                TabFragment2 tab2 = new TabFragment2();
                return tab2;
            case 2:
                TabFragment3 tab3 = new TabFragment3();
                return tab3;
            default:
                return null;
        }
    }
...
```

```
...
    @Override
    public int getCount() {
        return mNumOfTabs;
    }
}
```

SharedPreferences

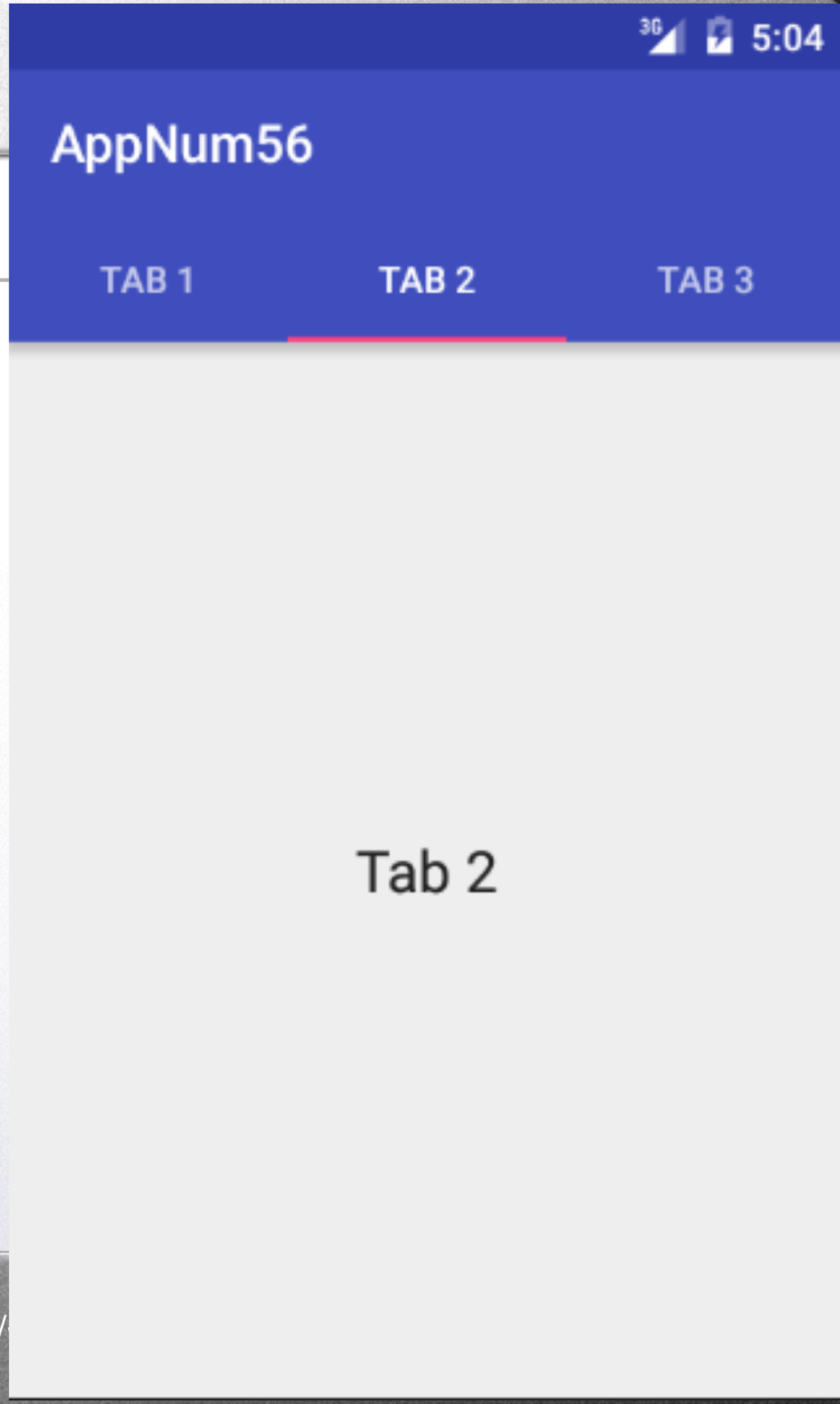
- Introdução

  ✓Exemplo AppNum60: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

...

final ViewPager viewPager = (ViewPager) findViewById(R.id.pager);
final PagerAdapter adapter = new PagerAdapter
(getSupportFragmentManager(), tabLayout.getTabCount());
viewPager.setAdapter(adapter);

int tabId = Prefs.getInteger(this, "tabId");
viewPager.setCurrentItem(tabId);
viewPager.addOnPageChangeListener(new TabLayout.TabLayoutOnPageChangeListener(tabLayout));
tabLayout.setOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        viewPager.setCurrentItem(tab.getPosition());
        Prefs.setInteger(MainActivity.this, "tabId", viewPager.getCurrentItem());
    }
```

SharedPreferences

- Introdução
  - ✓Exemplo AppNum60:




SharedPreferences

- Activity de Configurações

  ✓ É comum que aplicativos comerciais possuam módulos de gerência de configurações.

  ✓ É possível definir operações/opções mais comuns e torná-las permanentes nos aplicativos.

  ✓ Android possui uma maneira automatizada de se fazer o controle de preferências/configurações.

  ✓ Dados são salvos de forma “automática”.


SharedPreferences

- Activity de Configurações
  - ✓Exemplo AppNum61: PrefsFragment.java

```
public class PrefsFragment extends PreferenceFragment {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Load the preferences from an XML resource
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

SharedPreferences

- Activity de Configurações

✓Exemplo AppNum61: MainActivity.java

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Display the fragment as the main content.
        FragmentManager mFragmentManager = getFragmentManager();
        FragmentTransaction mFragmentTransaction = mFragmentManager.beginTransaction();
        PrefsFragment mPrefsFragment = new PrefsFragment();
        mFragmentTransaction.replace(android.R.id.content, mPrefsFragment);
        mFragmentTransaction.commit();

        SharedPreferences sp = PreferenceManager.getDefaultSharedPreferences(this);
        Log.i("CEA436", "MSG = "+sp.getString("edittext_preference",""));
    }
}
```

SharedPreferences

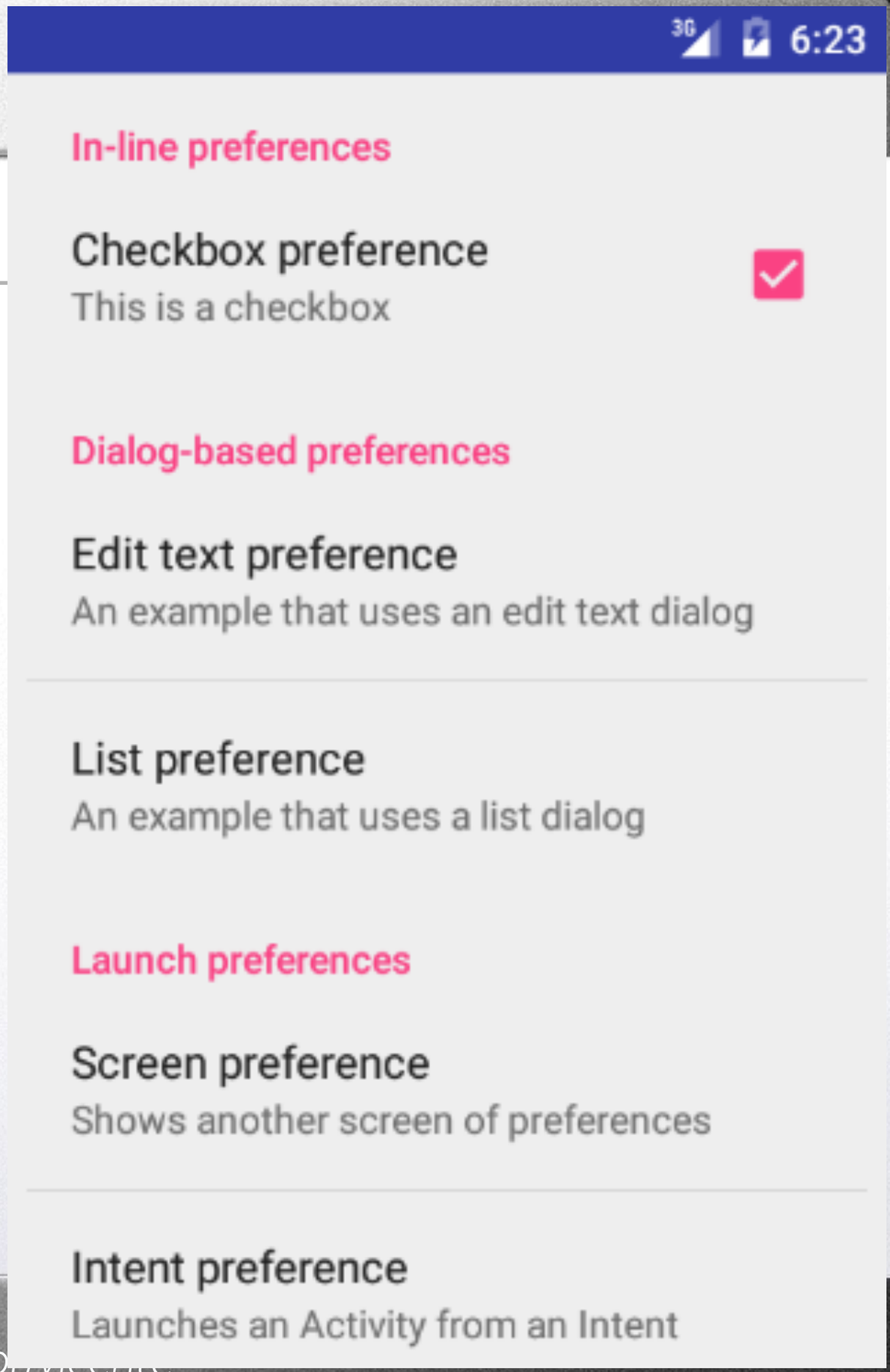
- Activity de Configurações

✓Exemplo AppNum61: preferences.xml

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <PreferenceCategory android:title="@string/inline_preferences">

        <CheckBoxPreference
        android:key="checkbox_preference"
        android:title="@string/title_checkbox_preference"
        android:summary="@string/summary_checkbox_preference" />

    </PreferenceCategory>

    <PreferenceCategory android:title="@string/dialog_based_preferences">

        <EditTextPreference android:key="edittext_preference"
        android:title="@string/title_edittext_preference"
        android:summary="@string/summary_edittext_preference"
        android:dialogTitle="@string/dialog_title_edittext_preference" />

        <ListPreference android:key="list_preference"
        android:title="@string/title_list_preference"
        android:summary="@string/summary_list_preference"
        android:entries="@array/entries_list_preference"
        android:entryValues="@array/entryvalues_list_preference"
        android:dialogTitle="@string/dialog_title_list_preference" />

    </PreferenceCategory>
...
```

SharedPreferences

- Activity de Configurações
  - ✓Exemplo AppNum61



Trabalhando com Arquivos

Trabalhando com Arquivos

- Introdução

✓ Leitura e escrita de arquivos em Android utilizam as conhecidas interfaces java.io.InputStream e java.io.OutputStream.

✓ Desenvolvedor precisa se preocupar somente em/de qual pasta os arquivos serão escritos/lidos.

✓ Tipos de memória:

✦ Interna: Utilizada para salvar arquivos internos a aplicação; e

✦ Externa: Permite salvar arquivos de forma pública.

Memórias Interna e Externa

Memórias Interna e Externa

- Introdução

  ✓ Operações memória Interna:

  ✦ FileInputStream openFileInput(nome): Abre um arquivo para leitura. Retorna um FileInputStream ou a exceção "FileNotFoundException".

  ✦ FileOutputStream openFileOutput(nome, modo): Abre um arquivo para escrita. Retorna um FileOutputStream ou a exceção "FileNotFoundException":

  - Context.MODE_PRIVATE: Arquivo criado só pode ser acessado pela aplicação que o criou.
  - Context.MODE_APPEND: Escritas são adicionadas ao final do arquivo.

Memórias Interna e Externa

- Introdução

  ✓ Operações memória Interna:

  ✦ boolean deleteFile(String nome): Exclui o arquivo (nome) e retorna um booleano indicando se a exclusão foi bem sucedida.

  ✦ String getFilesDir(): Retorna o caminho atual de salvamento dos dados.

Memórias Interna e Externa

- Introdução

  ✓ Operações memória Interna: Exemplo AppNum62

```
public class MainActivity extends AppCompatActivity {

    private final String LOG_TAG = "CEA436";
    private OutputStreamWriter osw = null;
    private InputStreamReader isr = null;
    private FileOutputStream fos = null;
    private FileInputStream fis = null;
    private final String TEST_FILENAME = "arquivo_exemplo.txt";

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

        Log.i(LOG_TAG, "getExternalStorageDirectory() = " + Environment.getExternalStorageDirectory());
        Log.i(LOG_TAG, "getExternalDataDirectory() = " + Environment.getDataDirectory());
        Log.i(LOG_TAG, "getDownloadCacheDirectory() = " + Environment.getDownloadCacheDirectory());
        Log.i(LOG_TAG, "getFilesDir() = " + this.getFilesDir());
```

Memórias Interna e Externa

- Introdução
  - ✓ Operações memória Interna: Exemplo AppNum62

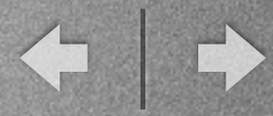
```
...
            try {
                fos = this.openFileOutput(TEST_FILENAME, Context.MODE_PRIVATE);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }

            if (fos != null) {
                osw = new OutputStreamWriter(fos);
                try {
                    osw.append("CEA436");
                    osw.flush();
                    fos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }

                osw = null;
            }
        ...
```

Memórias Interna e Externa

- Introdução

  ✓ Operações memória Interna: Exemplo AppNum62

```
...
        try {
            fis = this.openFileInput(TEST_FILENAME);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        if (fis != null) {
            isr = new InputStreamReader(fis);
            BufferedReader br = new BufferedReader(isr);
            String line;
            try {
                while ((line = br.readLine()) != null) {
                    Log.i(LOG_TAG,"Lido: "+line+"\n");
                }
                fis.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
...
```

Memórias Interna e Externa

- Introdução

✓ Operações memória Externa:

✦ Memória externa (cartões de memória) são muito utilizados para armazenamento de grandes arquivos.

✦ Permitem salvar arquivos temporários da aplicação que não precisam ser privados.

✦ Pastas podem ser criadas na raiz do SD card.

✦ Para não poluir a raiz, indica-se salvar os arquivos dentro de diretórios específicos.

Memórias Interna e Externa

- Introdução

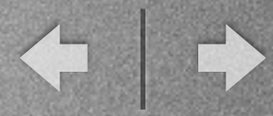
✓ Operações memória Externa:

✦ Environment.getExternalStorageDirectory(tipo): Retorna o diretório externo de armazenamento.

- O tipo pode ser dado por uma das constantes: DIRECTORY_MUSIC, DIRECTORY_PODCASTS, DIRECTORY_RINGTONES, DIRECTORY_ALARMS, DIRECTORY_NOTIFICATIONS, DIRECTORY_PICTURES, DIRECTORY_MOVIES, DIRECTORY_DOWNLOADS ou DIRECTORY_DCIM.

✦ Environment.getExternalFilesDir(tipo): Retorna o diretório privado do armazenamento externo.

- Vantagem: Diretório é excluído quando da remoção do aplicativo do dispositivo.

Memórias Interna e Externa

- Introdução

  ✓ Operações memória Externa:

  ✦ Environment.getDataDirectory(): Retorna o diretório de dados do usuário.

  ✦ Environment.getDownloadCacheDirectory(): Retorna o diretório de download/cache do usuário.

  ✦ Environment.getCacheDir(): Retorna o diretório de cache do usuário. Caso precise de espaço tal diretório será automaticamente apagado pelo Android.

Memórias Interna e Externa

- Introdução

  ✓ Operações memória Externa: Exemplo AppNum63

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:orientation="vertical">

    ...
    </EditText>

    <LinearLayout
        android:layout_width="match_parent" android:layout_height="wrap_content"
        android:orientation="horizontal" android:weightSum="1.0" android:layout_marginTop="20dp">

        <Button android:id="@+id/saveExternalStorage" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="@string/escrever"
        android:layout_weight="0.5"/>

        <Button android:id="@+id/getExternalStorage" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:layout_weight="0.5"
        android:text="@string/ler" />

    </LinearLayout>

   ...

</LinearLayout>
```


Memórias Interna e Externa

- Introdução

  ✓ Operações memória Externa: Exemplo AppNum63

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

    inputText = (EditText) findViewById(R.id.myInputText);
    response = (TextView) findViewById(R.id.response);
...
saveButton = (Button) findViewById(R.id.saveExternalStorage);
saveButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            FileOutputStream fos = new FileOutputStream(myExternalFile);
            fos.write(inputText.getText().toString().getBytes());
            fos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        inputText.setText("");
        response.setText(R.string.writing_data);
    }
});
```

Memórias Interna e Externa

- Introdução

✓ Operações memória Externa: Exemplo AppNum63

```
readButton = (Button) findViewById(R.id.getExternalStorage);
readButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            FileInputStream fis = new FileInputStream(myExternalFile);
            DataInputStream in = new DataInputStream(fis);
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
            String strLine;
            while ((strLine = br.readLine()) != null) {
                myData = myData + strLine;
            }
            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        inputText.setText(myData);
        response.setText(R.string.reading_data);
    }
});

if (!isExternalStorageAvailable() || isExternalStorageReadOnly()) {
    saveButton.setEnabled(false);
}
else {
    myExternalFile = new File(getExternalFilesDir(filepath), filename);
}
```

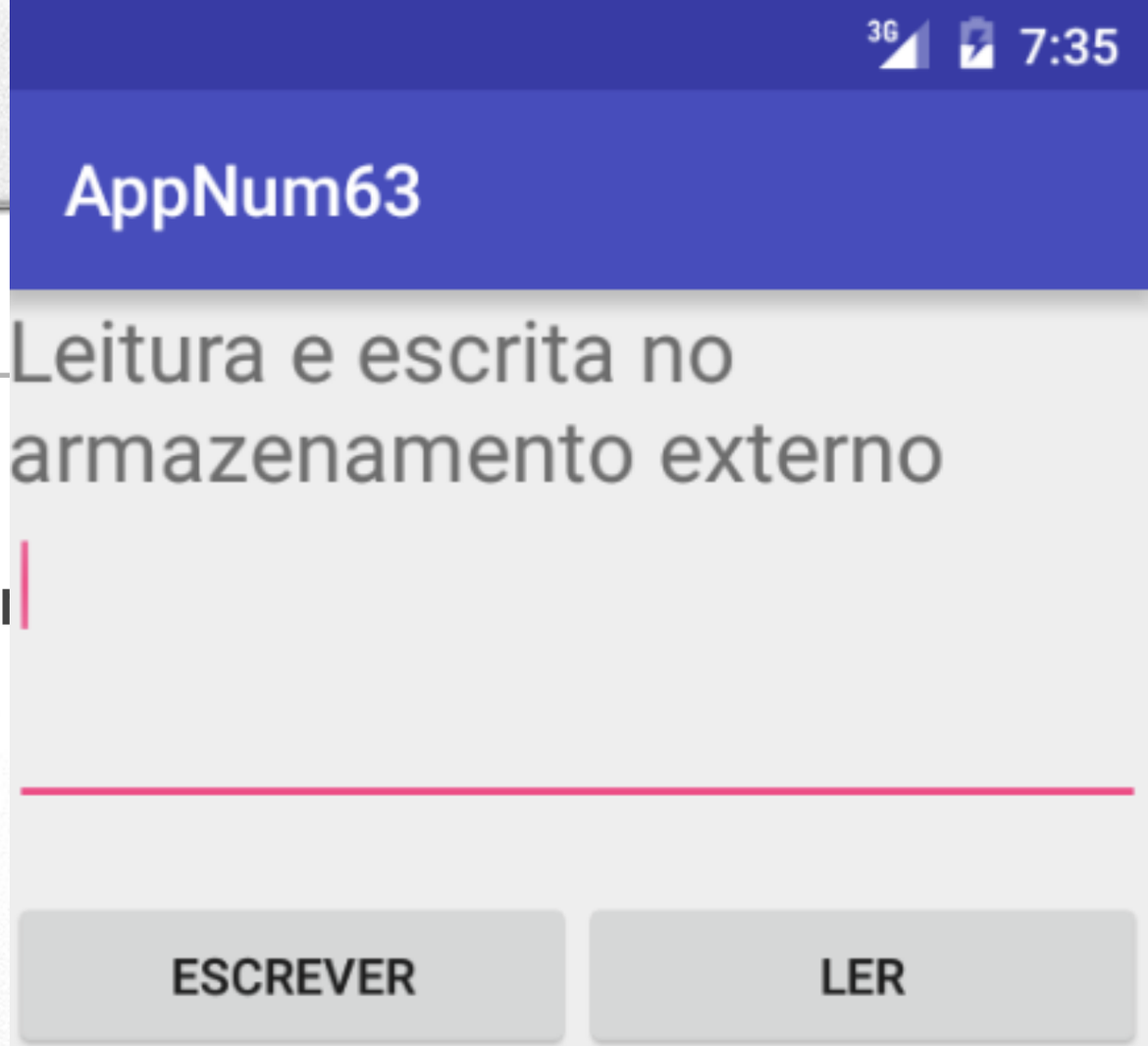

Memórias Interna e Externa

- Introdução
  - ✓ Operações memória Externa: Exemplo AppNum63

```
private static boolean isExternalStorageReadOnly() {
        String extStorageState = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(extStorageState)) {
            return true;
        }
        return false;
}

private static boolean isExternalStorageAvailable() {
    String extStorageState = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(extStorageState)) {
        return true;
    }
    return false;
}
```

Memórias Interna e Externa

- Introdução
  - ✓ Operações memória Externa: Exer



Banco de dados SQLite

Banco de dados SQLite

- Introdução

✓ O Android possui suporte nativo ao SQLite.

✓ Cada aplicativo Android pode criar um ou mais bancos de dados para uso próprio.

✓ Banco de dados criados pelos aplicativos ficam na pasta:

✦ `/data/data/<pacote_do_aplicativo>/databases`

✓ Um banco de dados somente é visível ao aplicativo que o criou.

Banco de dados SQLite

- Criação do BD

✓ Pode-se criar um banco de dados de diversas formas:

✦ Utilizando a API SQLite do Android (logo após a criação é possível executar um script para criar as tabelas e preencher os dados).

✦ Utilizando um cliente SQLite como o "**SQLite Expert Personal**" (http://www.sqliteexpert.com/).

✦ Utilizando o aplicativo **sqlite3** pelo console do emulador.

Banco de dados SQLite

- Criação e operações no BD SQLite

  ✓ Exemplo AppNum64 - Book.java

```
public class Book {

    ...
    public int getId() {
        return this.id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTitle() {
        return this.title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
    ...
    ...
    public String getAuthor() {
        return this.author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    @Override
    public String toString() {
        return "Book [id=" + id + ", title=" + title +
", author=" + author
            + "]";
    }
}
```

Banco de dados SQLite

- Criação e operações no BD SQLite

✓ Exemplo AppNum64 - MySQLiteHelper.java

```
public class MySQLiteHelper extends SQLiteOpenHelper {

    //Log tag
    private final String LOG_TAG = "CEA436";
    //Database version
    private static final int DATABASE_VERSION = 1;
    //Database name
    private static final String DATABASE_NAME = "CEA436_DB";

    public MySQLiteHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        // SQL statement to create book table
        String CREATE_BOOK_TABLE = "CREATE TABLE books ( " +
        "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
        "title TEXT, "+
        "author TEXT )";

        // create books table
        sqLiteDatabase.execSQL(CREATE_BOOK_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        // Drop older books table if existed
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS books");

        // create fresh books table
        this.onCreate(sqLiteDatabase);
    }
```

Banco de dados SQLite

- Criação e operações no BD SQLite

✓ Exemplo AppNum64 - MySQLiteHelper.java

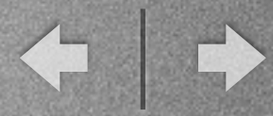
```
/* CRUD (create/add, read/get, update, delete) */
    // Books table name
    private static final String TABLE_BOOKS = "books";

    // Books Table Columns names
    private static final String KEY_ID = "id";
    private static final String KEY_TITLE = "title";
    private static final String KEY_AUTHOR = "author";

    private static final String[] COLUMNS = {KEY_ID,KEY_TITLE,KEY_AUTHOR};

    public void addBook(Book book){
        Log.d(LOG_TAG,"addBook = " + book.toString());
        // 1. get reference to writable DB
        SQLiteDatabase db = this.getWritableDatabase();

        // 2. create ContentValues to add key "column"/value
        ContentValues values = new ContentValues();
        values.put(KEY_TITLE, book.getTitle()); // get title
        values.put(KEY_AUTHOR, book.getAuthor()); // get author

        // 3. insert
        db.insert(TABLE_BOOKS, // table
                  null, //nullColumnHack
                  values); // key/value -> keys = column names/ values = column values

        // 4. close
        db.close();
    }
```


Banco de dados SQLite

- Criação e operações no BD SQLite

  ✓ Exemplo AppNum64 - MainActivity.java

```
public class MainActivity extends AppCompatActivity {

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

        MySQLiteHelper db = new MySQLiteHelper(this);

        // add Books
        db.addBook(new Book("Android Application Development Cookbook", "Wei Meng Lee"));
        db.addBook(new Book("Android Programming: The Big Nerd Ranch Guide", "Bill Phillips and Brian Hardy"));
        db.addBook(new Book("Learn Android App Development", "Wallace Jackson"));

        // get all books
        List<Book> list = db.getAllBooks();

        // delete one book
        db.deleteBook(list.get(0));

        // get all books
        db.getAllBooks();
    }
}
```

Banco de dados SQLite

- Criação e operações no BD SQLite

✓ Exercício 16

✦ Utilizando os conceitos de banco de dados vistos anteriormente, faça as seguintes melhorias:

1) Crie uma app funcional para o Exemplo 64: Você deve prover maneiras ao usuário de inserir, remover, listar e atualizar os dados dos livros.

2) Utilize Fragments em pelo menos uma das funcionalidades para melhorar a usabilidade de seu sistema.

Backup na Nuvem

Backup na Nuvem

- Introdução

✓ O Android disponibiliza uma API de backup de dados na nuvem do Google.

✓ Em caso de erro, reinstalação do aplicativo, troca do aparelho, etc., os dados podem ser recuperados remotamente.

✓ Backup dos dados é feito através da classe BackupAgent.

✓ A implementação do backup é facilitada pelo uso da classe BackupAgentHelper.

Backup na Nuvem

- Introdução

✓ Para ser utilizado, o backup na nuvem do Google precisa ser inicialmente cadastrado.

✓ Acessando o site https://developer.android.com/google/backup/signup.html é possível a geração de uma chave de acesso para cada aplicativo.

✓ Nome do pacote deve ser informado para gerar a chave.

Backup na Nuvem

- Introdução

Developers | Develop > Google Services > Register for Android Backup Service | Developer Console

- Google Play In-app Billing
- Filters on Google Play
- Google Play Developer API
- Multiple APK Support
- APK Expansion Files
- Application Licensing
- Android Backup Service
  - Register

4.7 You agree that your use of the Service will be in compliance with any documentation guidelines provided by Google and that failure to comply with the documentation guidelines may result in the disabling of the Backup Service Key(s) for your Application(s).

4.8 Unless you have been specifically permitted to do so in a separate agreement with Google, you agree that you will not reproduce, duplicate, copy, sell, trade or resell (a) use of the Service, or (b) access to the Service.

4.9 You agree that you are solely responsible for (and that Google has no responsibility to you or to any third party for) your and your Application's use of the Service, any breach of your obligations under the Terms, and for the consequences (including any loss or damage which Google may suffer) of any such breach.

4.10 You agree that in your use of the Service, you and your Applications will protect the privacy and legal rights of users. You must provide legally adequate privacy notice and protection for users whose data your Applications back up to the Service. Further, your Application may only use that information for the limited

☑ I have read and agree with the Android Backup Service Terms of Service

Application package name: com.example.android.app

REGISTER WITH ANDROID BACKUP SERVICE

- Introdução

Android Backup Service Key

Agradecemos seu registro para obter uma chave do Android Backup Service!

Sua chave é:

```
AEdPqrEAAAAIS-ib3Uaz40zTnZhdEMTw-x22UFaBqF0ftICekw
```

Esta chave é válida para o app com nome de pacote:

```
com.cea436.appnum65
```

Inclua esta chave em seu arquivo AndroidManifest.xml, com o elemento `<meta-data>` a seguir, dentro do elemento `<application>`:

```
<meta-data android:name="com.google.android.backup.api_key" android:value="AEdPqrEAAAAIS-ib3Uaz40zTnZhdEMTw-x22UFaBqF0ftICekw" />
```

Para mais informações, consulte o Guia de backup para desenvolvedores ou volte para o registro no Android Backup Service.

Backup na Nuvem

- Introdução

  ✓ Exemplo AppNum65: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cea436.appnum65">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:backupAgent=".MyBackupAgent">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data android:name="com.google.android.backup.api_key" android:value="AEdPqrEAAAAIS-
ib3Uaz40zTnZhdEMTw-x22UFaBqF0ftICekw" />
    </application>

</manifest>
```


Backup na Nuvem

- Introdução

  ✓ Exemplo AppNum65: MyBackupAgent.java

```
public class MyBackupAgent extends BackupAgentHelper {
    // The names of the SharedPreferences groups that the application maintains.  These
    // are the same strings that are passed to getSharedPreferences(String, int).
    static final String PREFS_DISPLAY = "CEA436";
    static final String PREFS_SCORES = "CEA436";

    // An arbitrary string used within the BackupAgentHelper implementation to
    // identify the SharedPreferenceBackupHelper's data.
    static final String MY_PREFS_BACKUP_KEY = "CEA436_BACK_KEY";

    // Simply allocate a helper and install it
    public void onCreate() {
        SharedPreferencesBackupHelper helper = new SharedPreferencesBackupHelper(this, PREFS_DISPLAY);
        addHelper(MY_PREFS_BACKUP_KEY, helper);
    }
}
```

Backup na Nuvem

- Introdução

✓ Exemplo AppNum65: MainActivity.java

```
@Override
   protected void onCreate(Bundle savedInstanceState) {
       ...

       backupManager = new BackupManager(this);

       final TextView textView1 = (TextView)findViewById(R.id.textView1);
       final EditText editText1 = (EditText)findViewById(R.id.editText1);
       final Button button1 = (Button)findViewById(R.id.bt1);
       final Button button2 = (Button)findViewById(R.id.bt2);

       myData = Prefs.getString(this,"MyData");
       textView1.setText("Old data: "+myData);

       button1.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
               Prefs.setString(MainActivity.this, "MyData", editText1.getText().toString());
               textView1.setText("Data copied sent to prefs! = " + editText1.getText().toString());
               backupManager.dataChanged();
           }
       });

       button2.setOnClickListener(new View.OnClickListener(){
           @Override
           public void onClick(View view) {
               textView1.setText(Prefs.getString(MainActivity.this, "MyData").toString());
           }
       });
   }
}
```

Backup na Nuvem

- Introdução

✓ Como este é um ambiente de desenvolvimento (aplicativo ainda não está na PlayStore), é necessário fazer o *backup* manual dos dados:

```
$ adb shell bmgr enable true
Backup Manager now enabled
$ adb shell bmgr backup com.cea436.appnum65
$ adb shell bmgr run
```

Backup na Nuvem

- Introdução
  - ✓ Exemplo AppNum65