Datenaufbewahrung in Android Apps

Prof. Dr.-Ing. V.Brovkov

Datenaufbewahrung in Android Apps

- Die Daten können aufbewahrt werden mit:
  - Preferences
  - Dateien (on SD Card oder im Speicher)
  - Datenbank SQLite

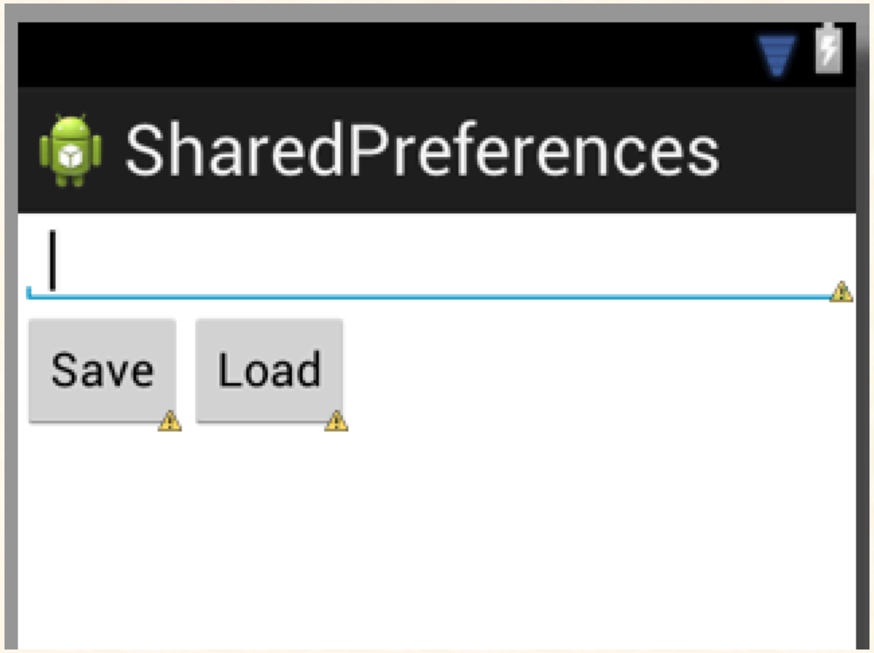
Preferences

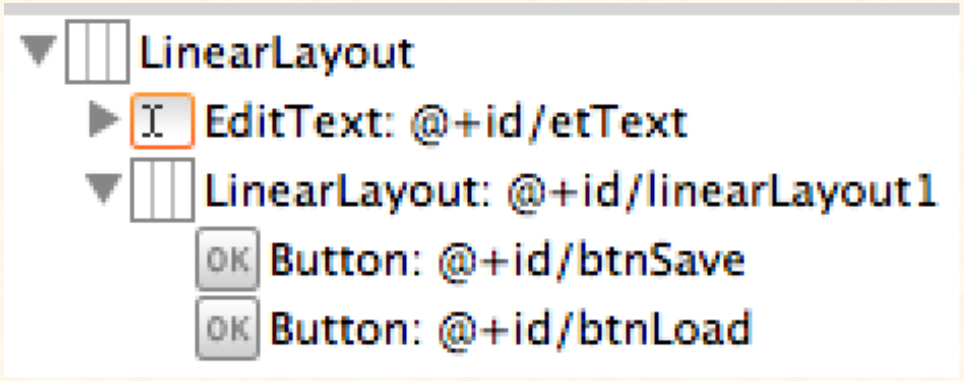
http://startandroid.ru/ru/uroki/vse-uroki-spiskom/73-urok-33-hranenie-dannyh-preferences.html

Projekt: P0331_SharedPreferences

Preferences

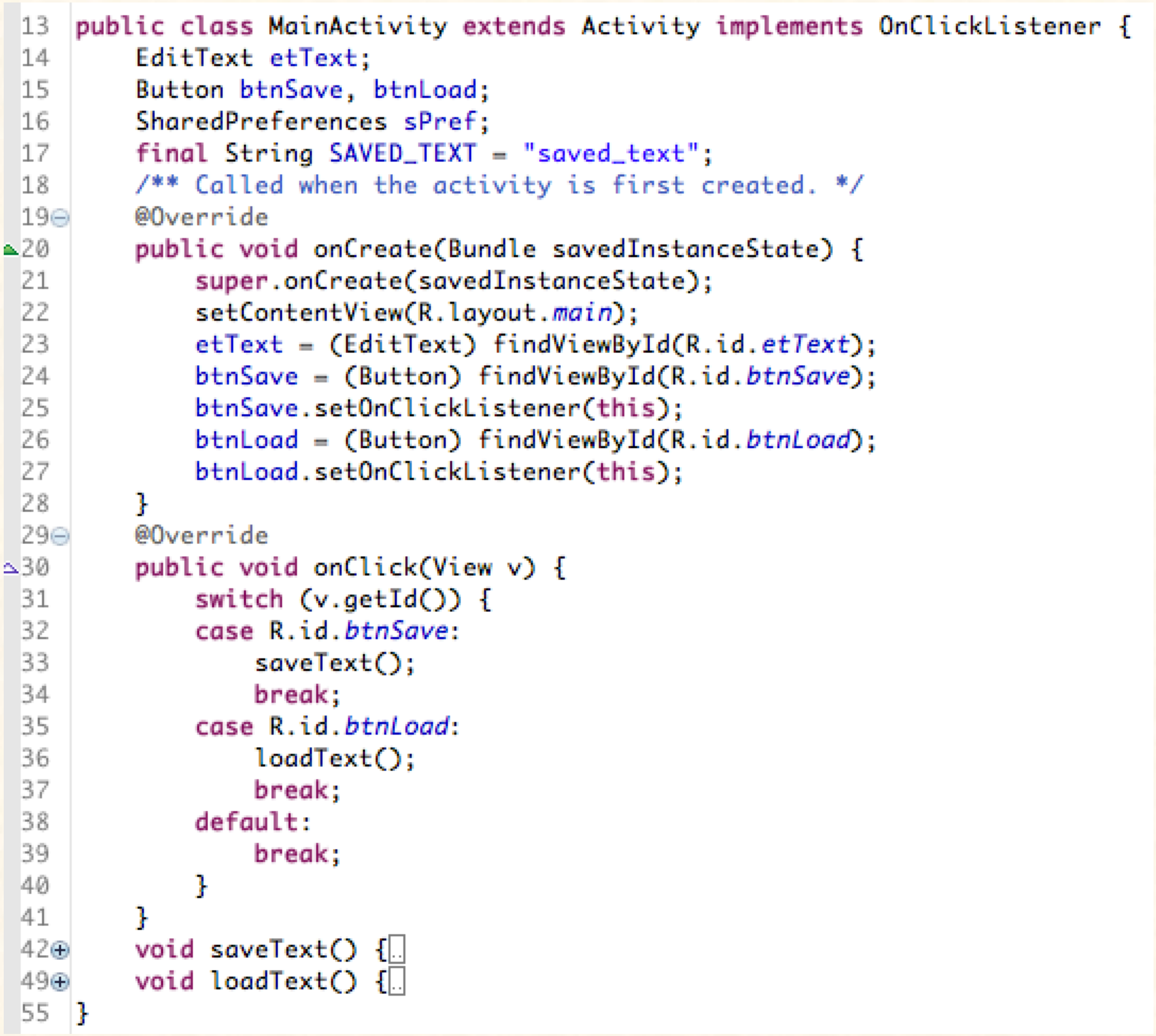
- Mit OnClick
  - speichern wir das Inhalt von etText in die Datei oder
  - loaden wir den gespeicherten Inhalt aus der Datei in etText
- Class SharedPreferences wird im Einsatz





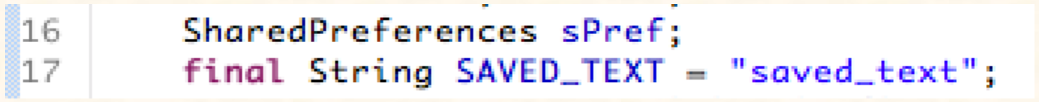
LinearLayout
EditText: @+id/etText
LinearLayout: @+id/linearLayout1
Button: @+id/btnSave
Button: @+id/btnLoad

Preferences

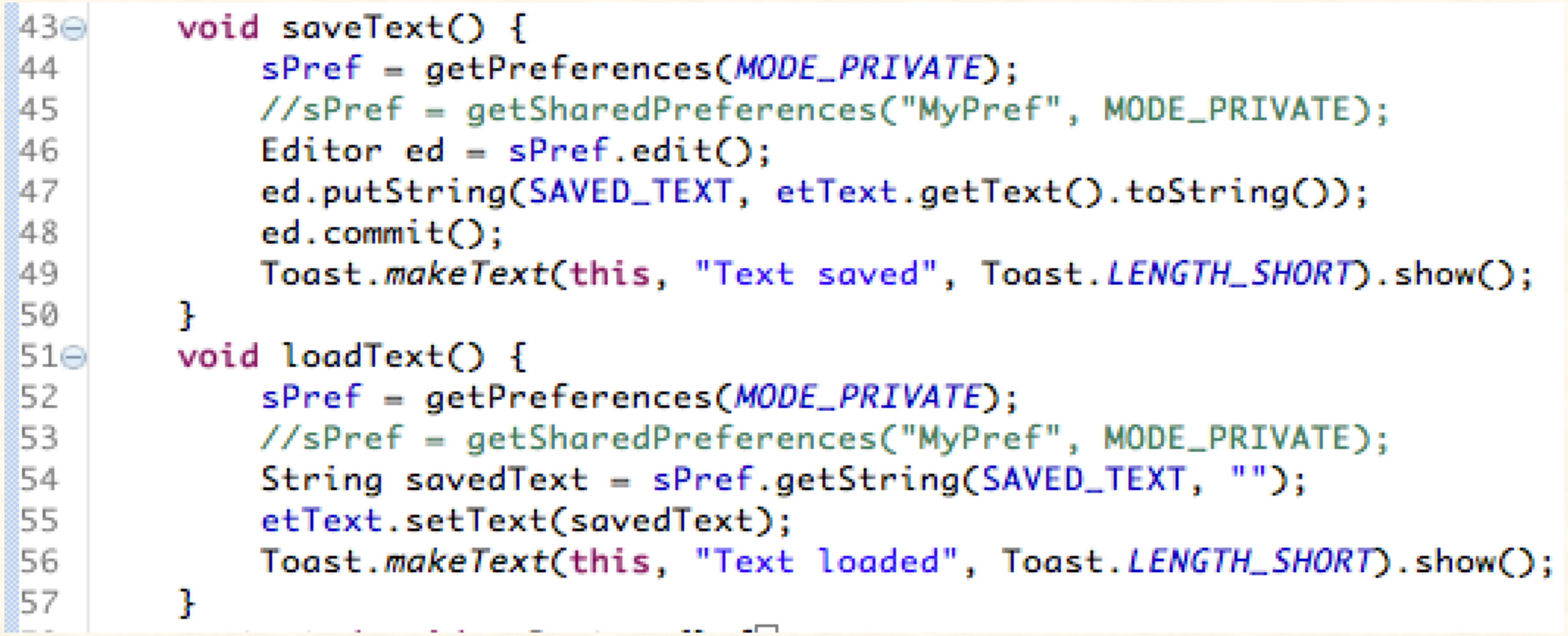


```
public class MainActivity extends Activity implements OnClickListener {
    EditText etText;
    Button btnSave, btnLoad;
    SharedPreferences sPref;
    final String SAVED_TEXT = "saved_text";
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        etText = (EditText) findViewById(R.id.etText);
        btnSave = (Button) findViewById(R.id.btnSave);
        btnSave.setOnClickListener(this);
        btnLoad = (Button) findViewById(R.id.btnLoad);
        btnLoad.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btnSave:
            saveText();
            break;
        case R.id.btnLoad:
            loadText();
            break;
        default:
            break;
        }
    }
    void saveText() {...
    void loadText() {...
}
```

Preferences



```
    SharedPreferences sPref;
    final String SAVED_TEXT = "saved_text";
```

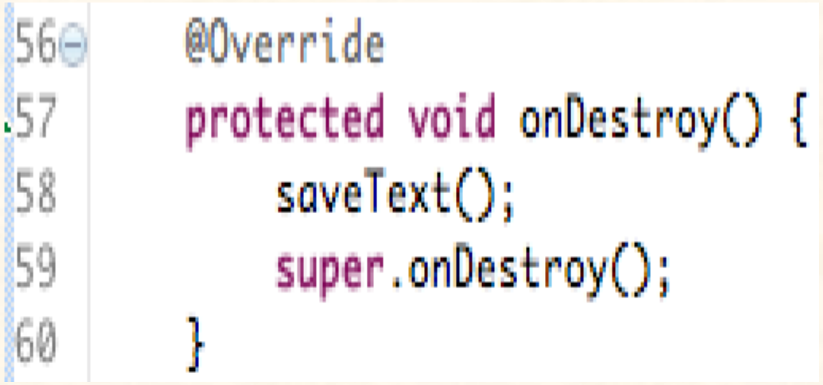


```
    void saveText() {
        sPref = getPreferences(MODE_PRIVATE);
        //sPref = getSharedPreferences("MyPref", MODE_PRIVATE);
        Editor ed = sPref.edit();
        ed.putString(SAVED_TEXT, etText.getText().toString());
        ed.commit();
        Toast.makeText(this, "Text saved", Toast.LENGTH_SHORT).show();
    }
    void loadText() {
        sPref = getPreferences(MODE_PRIVATE);
        //sPref = getSharedPreferences("MyPref", MODE_PRIVATE);
        String savedText = sPref.getString(SAVED_TEXT, "");
        etText.setText(savedText);
        Toast.makeText(this, "Text loaded", Toast.LENGTH_SHORT).show();
    }
```

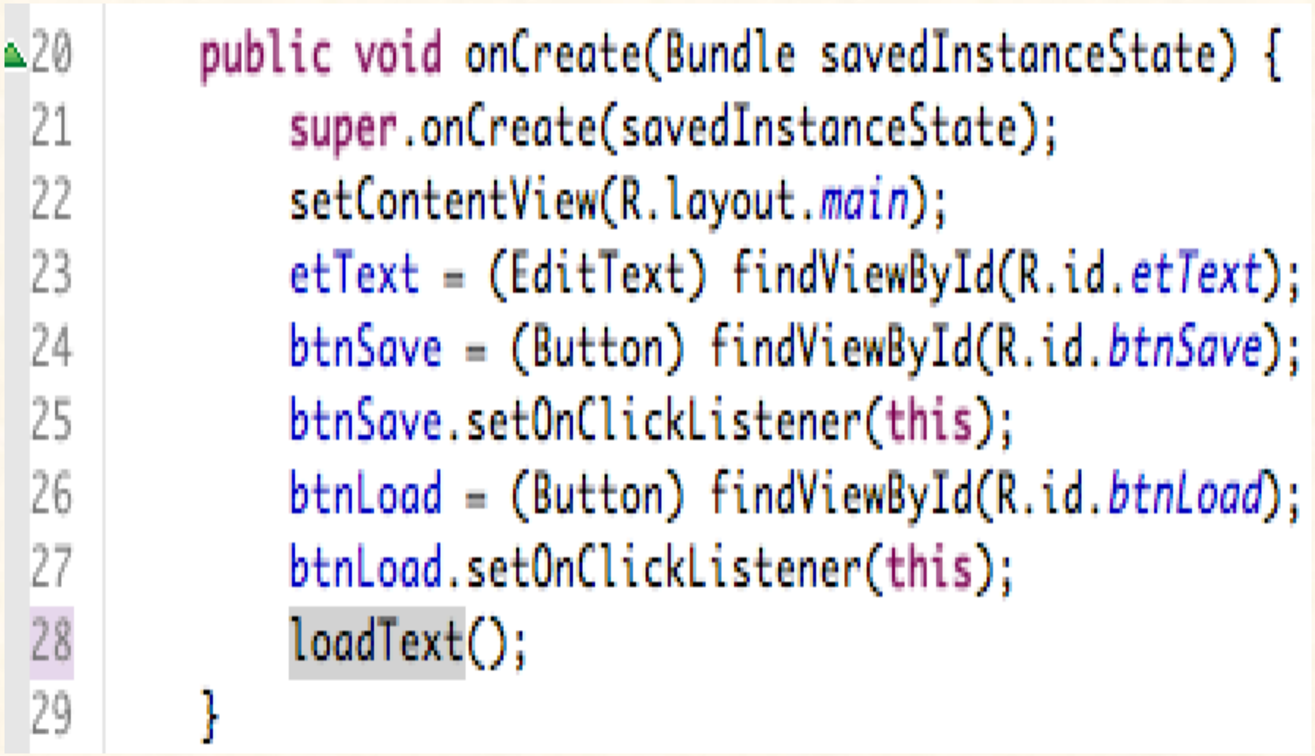
- public SharedPreferences **getPreferences (int mode)**
  - das Object von Class **SharedPreferences** verknüpft die App mit einer Datei, die speichert Daten in der Form ***Name – Value***
  - das Objekt von Class **Editor** lässt die Daten in der App Preferences Datei (sicher) zu ändern

Preferences

- Die Einstellungen können automatisch in onCreate() gelesen und in onDestroy() gespeichert werden



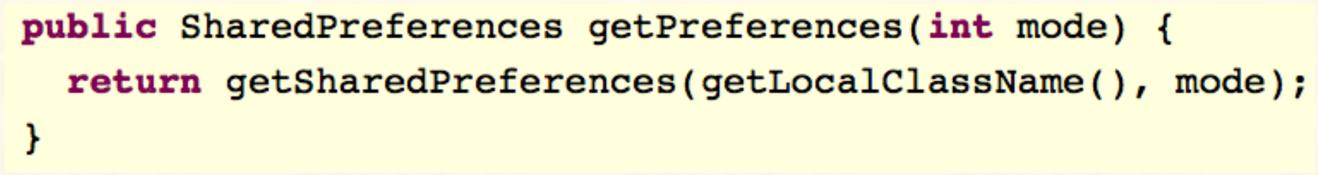
```
    @Override
    protected void onDestroy() {
        saveText();
        super.onDestroy();
    }
```



```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        etText = (EditText) findViewById(R.id.etText);
        btnSave = (Button) findViewById(R.id.btnSave);
        btnSave.setOnClickListener(this);
        btnLoad = (Button) findViewById(R.id.btnLoad);
        btnLoad.setOnClickListener(this);
        loadText();
    }
```

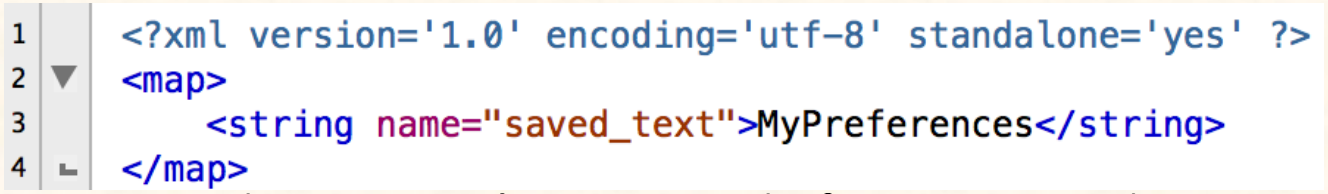
Preferences

- Die Name der Datei entspricht der Name der Activity. Das stammt aus:



```
public SharedPreferences getPreferences(int mode) {
  return getSharedPreferences(getLocalClassName(), mode);
}
```

- Die Datei mit Einstellungen kann in der Datei-System des Android Gerätes gefunden werden:
  - Android Studio Menu: Tools --> Android --> Android Device Monitor
  - Window --> Show View --> Android --> File Explorer,
  - *data/data/ua.opu.brovkov. sharedpreferences/shared_prefs/*MainActivity.xml
  - die Datei zum Rechner kopieren und ansehen (Pull a file from the device):



```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="saved_text">MyPreferences</string>
</map>
```

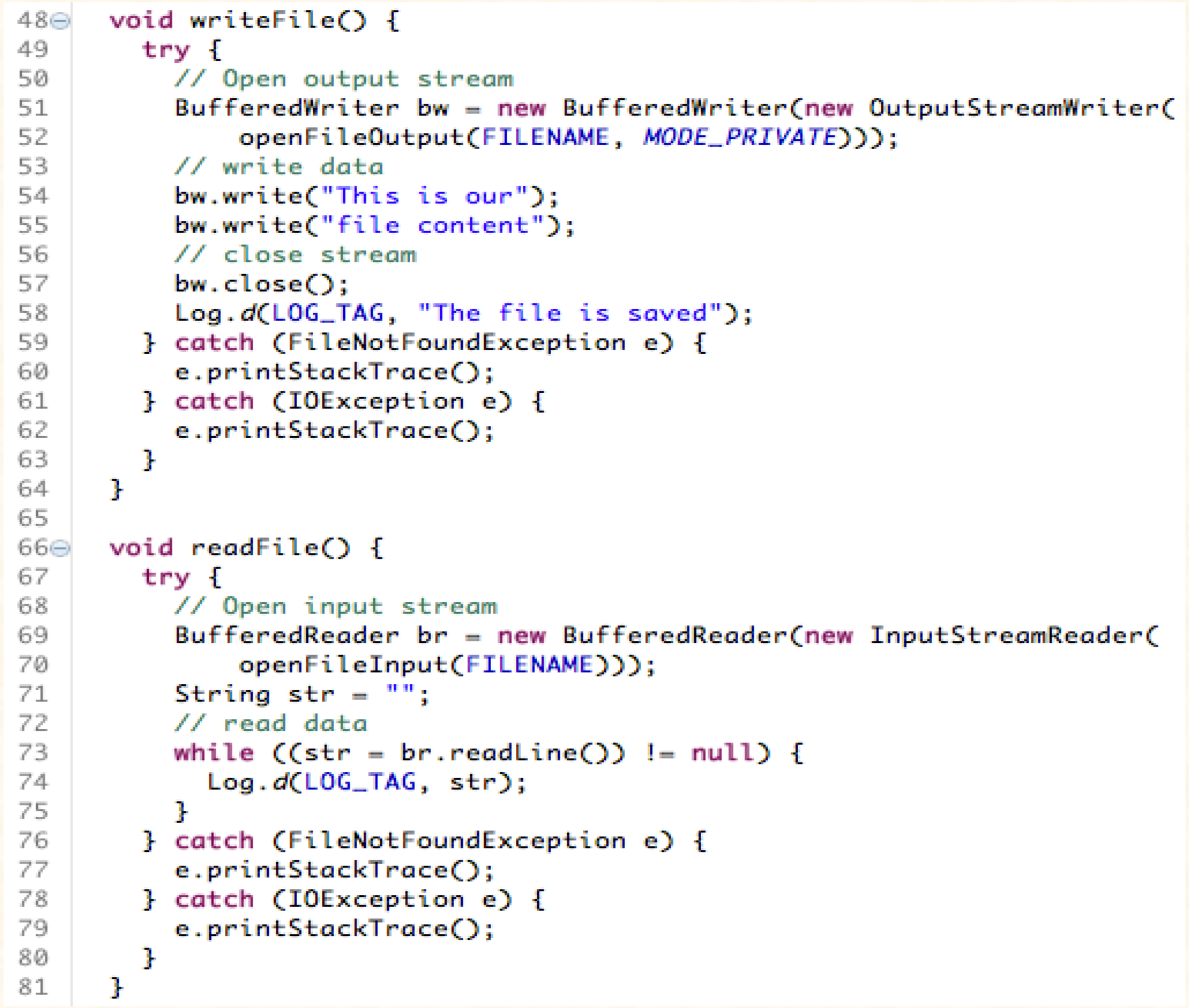
- Die Name der Datei kann vordefiniert werden:
sPref = getSharedPreferences("MyPref", MODE_PRIVATE);

Datei (on SD Card oder im Speicher)

http://startandroid.ru/ru/uroki/vse-uroki-spiskom/138-urok-75-hranenie-dannyh-rabota-s-fajlami.html

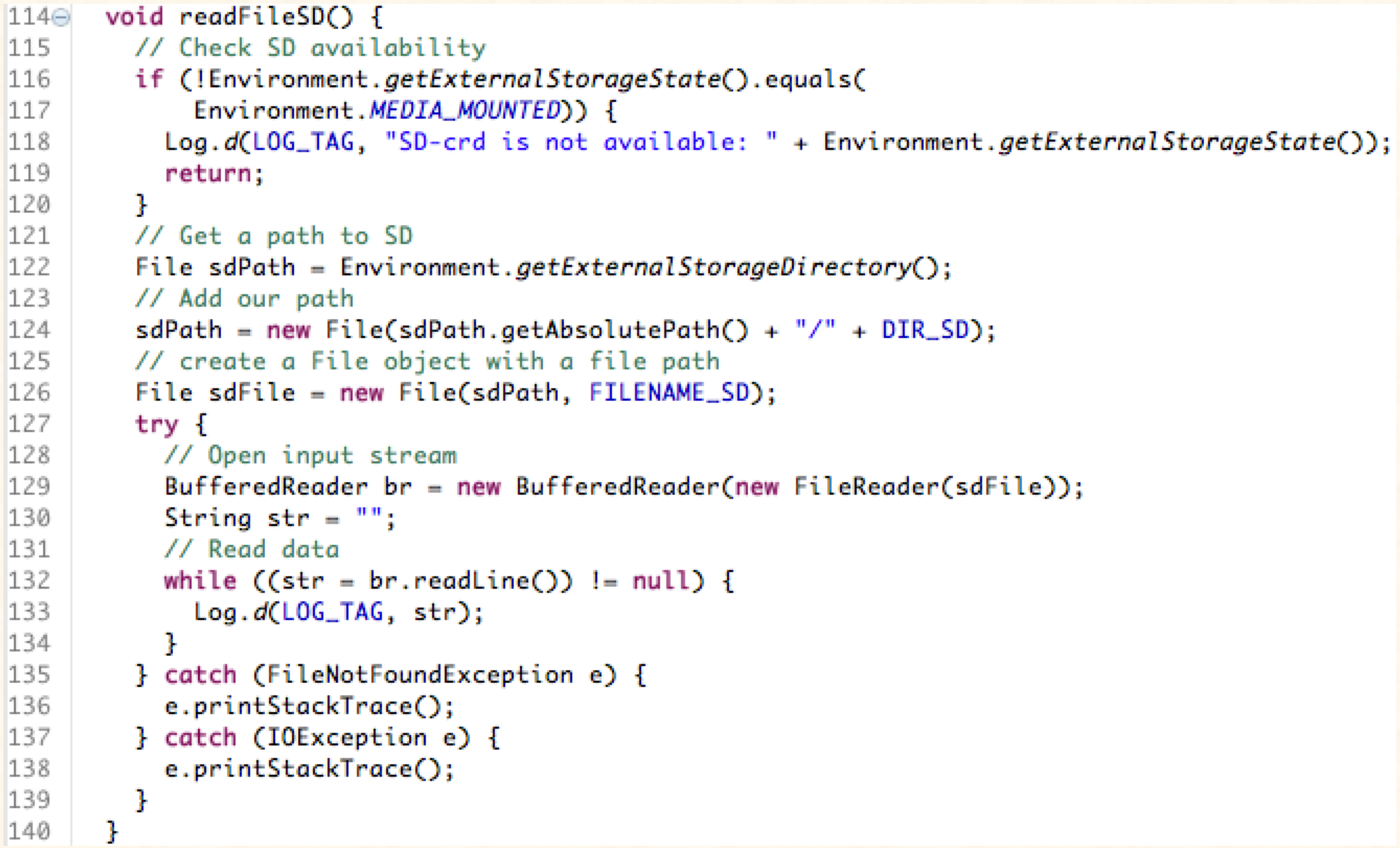
Projekt: P0751_Files

Speicher: Datei lesen und schreiben



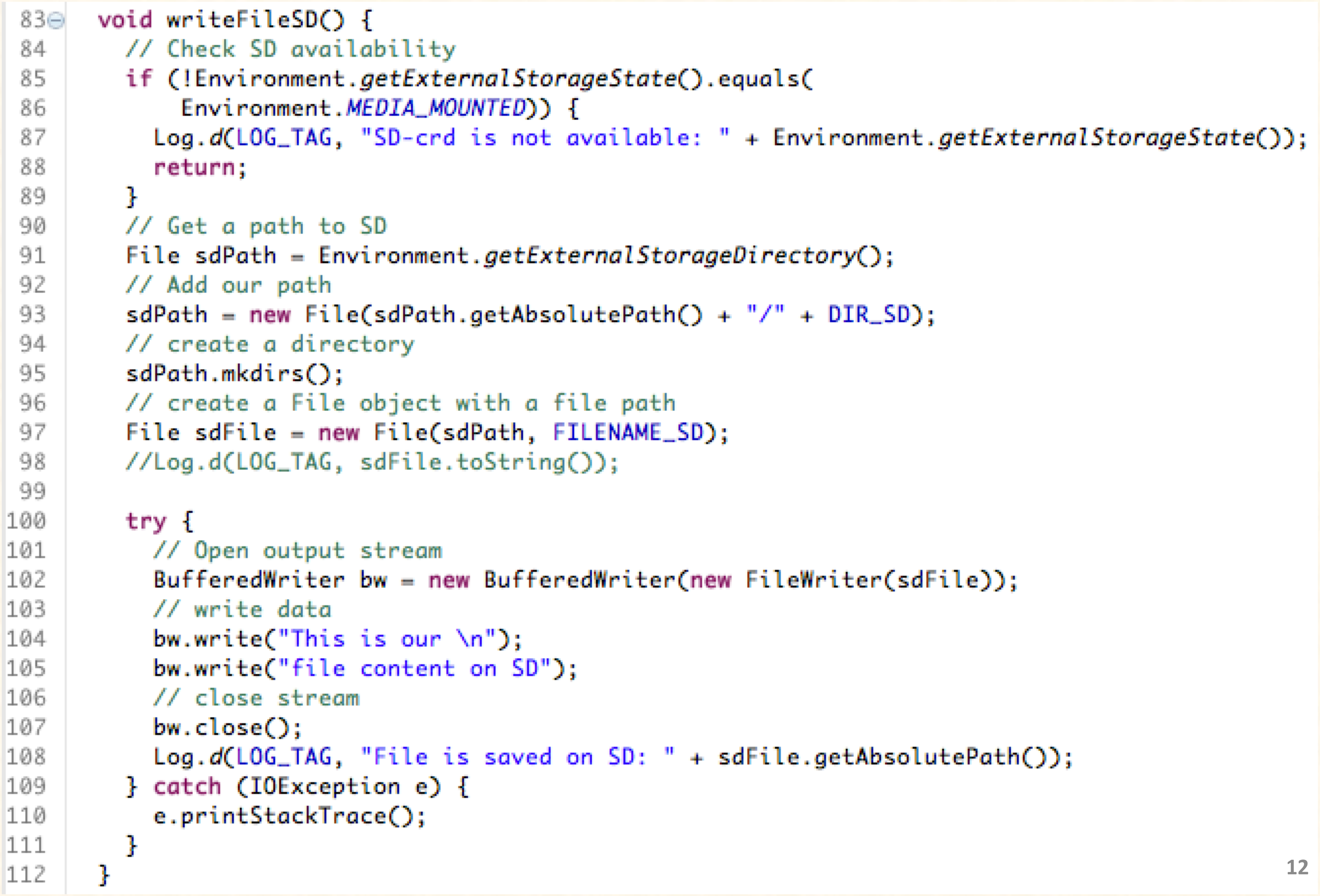
```java
void writeFile() {
  try {
    // Open output stream
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
        openFileOutput(FILENAME, MODE_PRIVATE)));
    // write data
    bw.write("This is our");
    bw.write("file content");
    // close stream
    bw.close();
    Log.d(LOG_TAG, "The file is saved");
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}

void readFile() {
  try {
    // Open input stream
    BufferedReader br = new BufferedReader(new InputStreamReader(
        openFileInput(FILENAME)));
    String str = "";
    // read data
    while ((str = br.readLine()) != null) {
      Log.d(LOG_TAG, str);
    }
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

SD-Card: Datei lesen



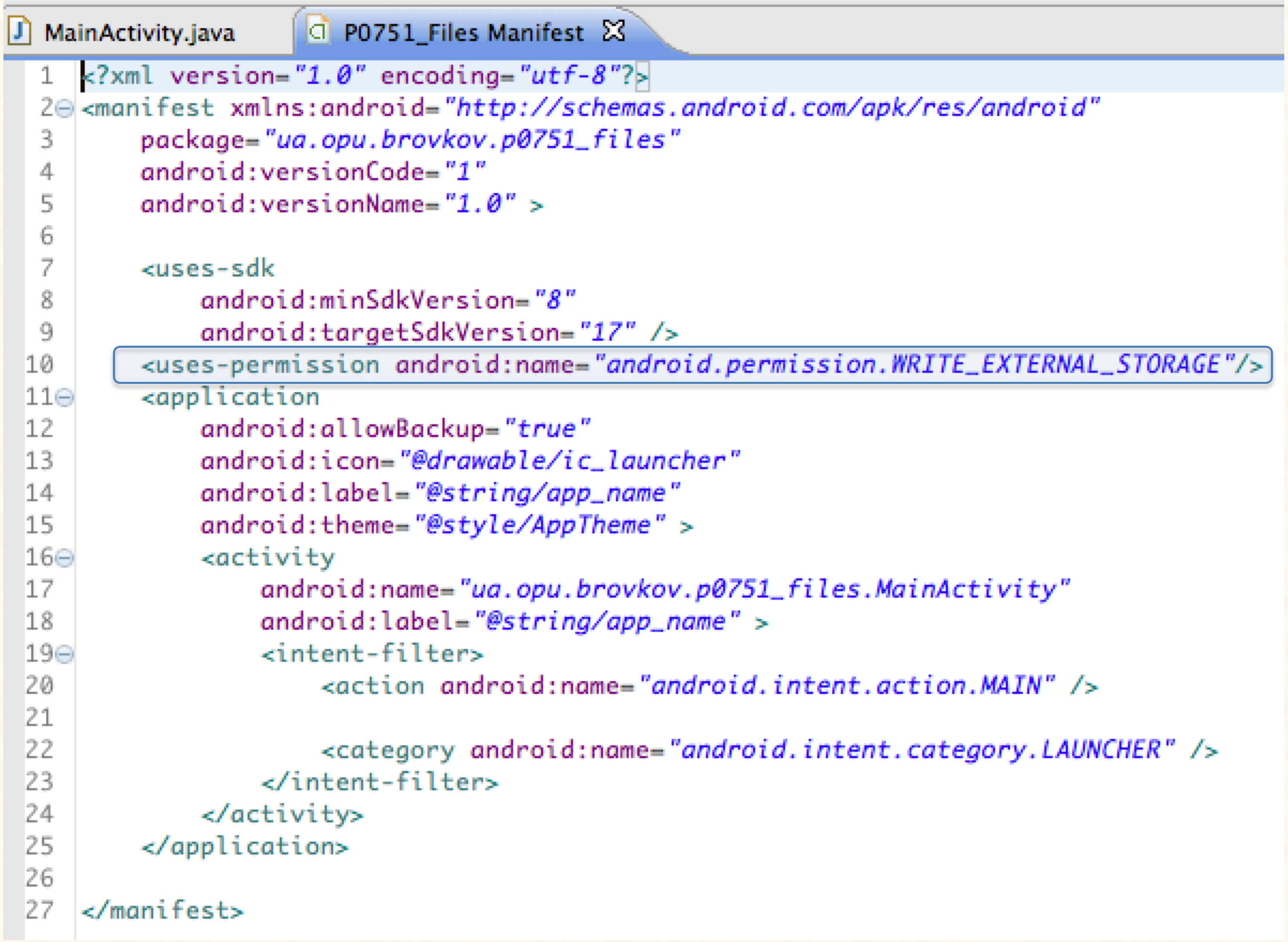
```java
void readFileSD() {
  // Check SD availability
  if (!Environment.getExternalStorageState().equals(
      Environment.MEDIA_MOUNTED)) {
    Log.d(LOG_TAG, "SD-crd is not available: " + Environment.getExternalStorageState());
    return;
  }
  // Get a path to SD
  File sdPath = Environment.getExternalStorageDirectory();
  // Add our path
  sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
  // create a File object with a file path
  File sdFile = new File(sdPath, FILENAME_SD);
  try {
    // Open input stream
    BufferedReader br = new BufferedReader(new FileReader(sdFile));
    String str = "";
    // Read data
    while ((str = br.readLine()) != null) {
      Log.d(LOG_TAG, str);
    }
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

SD-Card: Datei schreiben



```java
void writeFileSD() {
  // Check SD availability
  if (!Environment.getExternalStorageState().equals(
      Environment.MEDIA_MOUNTED)) {
    Log.d(LOG_TAG, "SD-crd is not available: " + Environment.getExternalStorageState());
    return;
  }
  // Get a path to SD
  File sdPath = Environment.getExternalStorageDirectory();
  // Add our path
  sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
  // create a directory
  sdPath.mkdirs();
  // create a File object with a file path
  File sdFile = new File(sdPath, FILENAME_SD);
  //Log.d(LOG_TAG, sdFile.toString());

  try {
    // Open output stream
    BufferedWriter bw = new BufferedWriter(new FileWriter(sdFile));
    // write data
    bw.write("This is our \n");
    bw.write("file content on SD");
    // close stream
    bw.close();
    Log.d(LOG_TAG, "File is saved on SD: " + sdFile.getAbsolutePath());
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

Achtung! Permissions!



MainActivity.java | P0751_Files Manifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ua.opu.brovkov.p0751_files"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="ua.opu.brovkov.p0751_files.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>
    </application>

</manifest>
```

SQLite

Projekt: P0341_SimpleSQLite

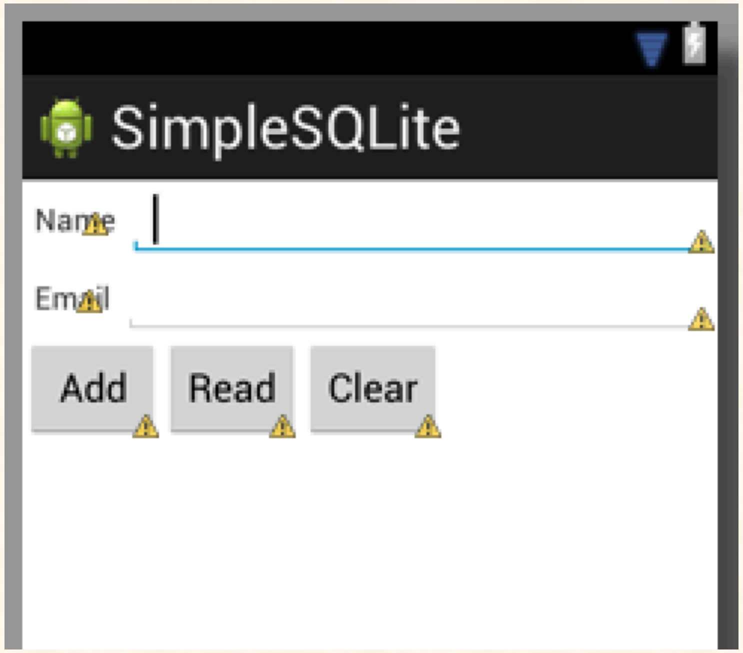
http://startandroid.ru/ru/uroki/vse-uroki-spiskom/74-urok-34-hranenie-dannyh-sqlite.html

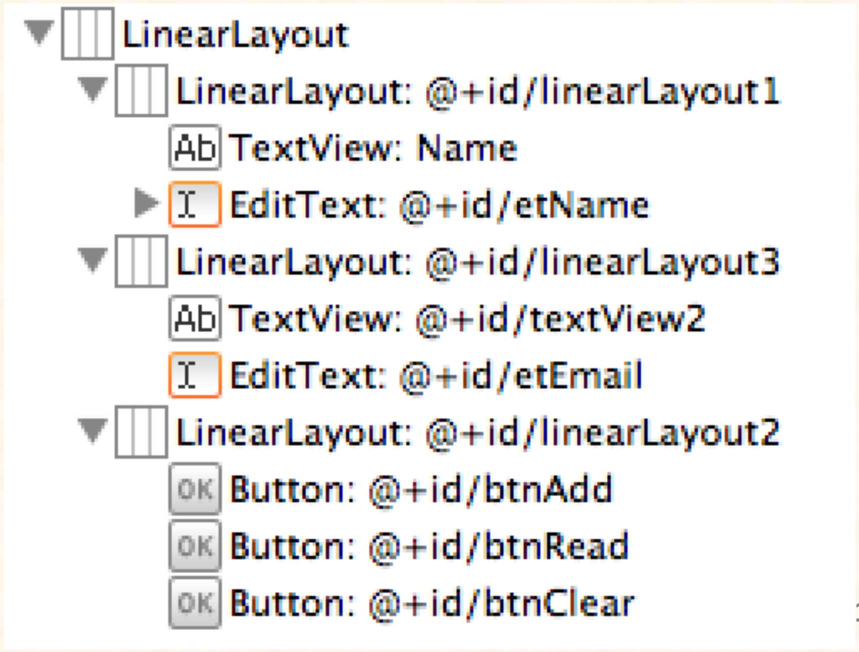
SQLite

- Eine Datenbank ist viel mehr flexibel
- Wenn eine App mit einer Datenbank verknüpft ist, dann können folgende Varianten entstehen:
  - es gibt keine Datenbank; muss generiert werden
  - es gibt eine alte Version von Datenbank (z.B. müssen zusätzliche Tabellen erstellt werden); ein Update ist erforderlich
  - es gibt eine aktuelle Version von Datenbank; die Daten können gelesen oder gespeichert werden.
- Eine Klasse SQLiteOpenHelper Erweiterung kann helfen. Es gibt Methoden **onCreate(), onUpgrade(), onOpen(), getWritableDatabase()** ...

SQLite

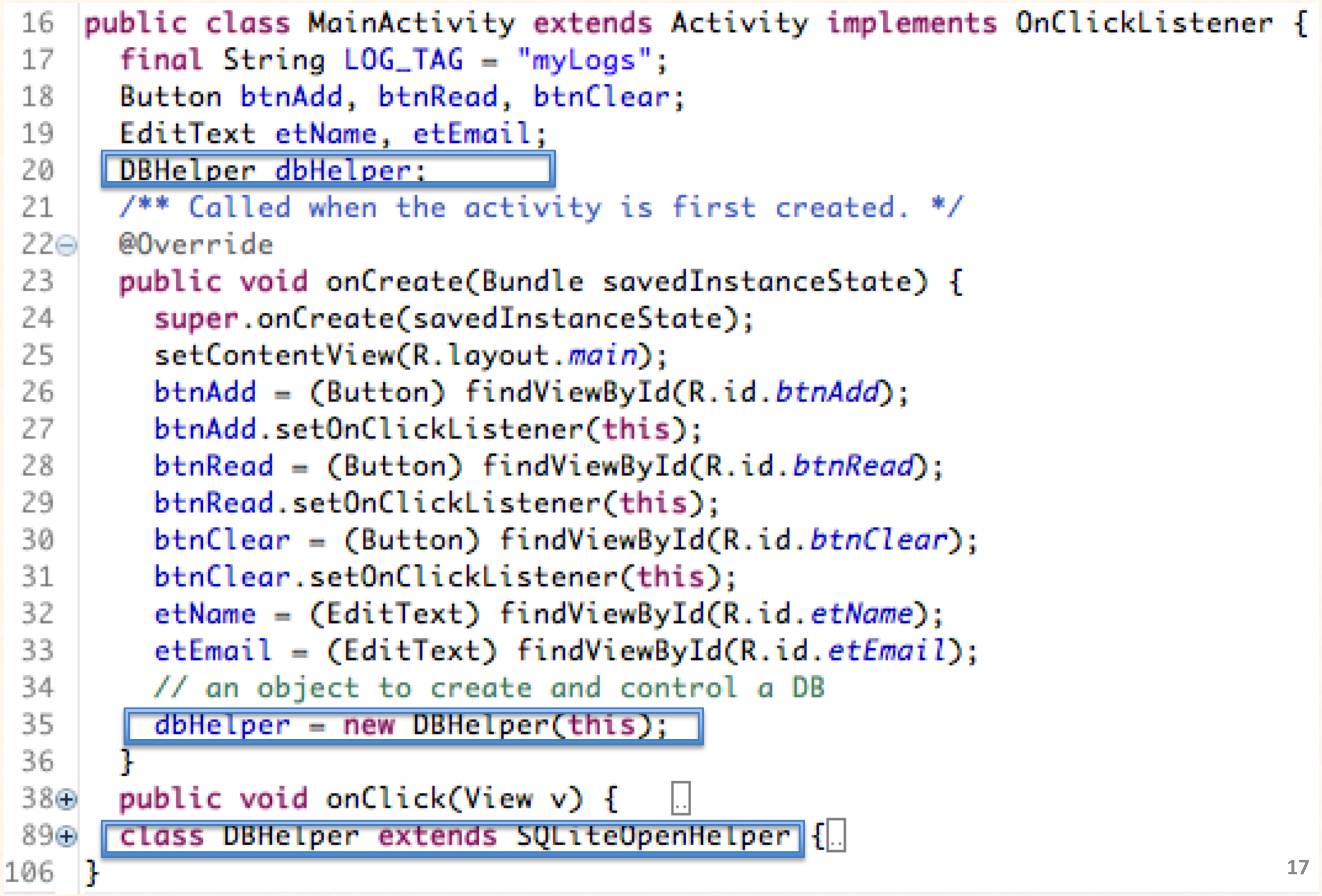
- Eine Activity arbeitet mit einer Tabelle; die Daten werden eingetragen / gelesen / gelöscht
- die Felder :
  - ID (automatisch generiert)
  - Name
  - Email
- User Interface:





```
LinearLayout
  LinearLayout: @+id/linearLayout1
    TextView: Name
    EditText: @+id/etName
  LinearLayout: @+id/linearLayout3
    TextView: @+id/textView2
    EditText: @+id/etEmail
  LinearLayout: @+id/linearLayout2
    Button: @+id/btnAdd
    Button: @+id/btnRead
    Button: @+id/btnClear
```

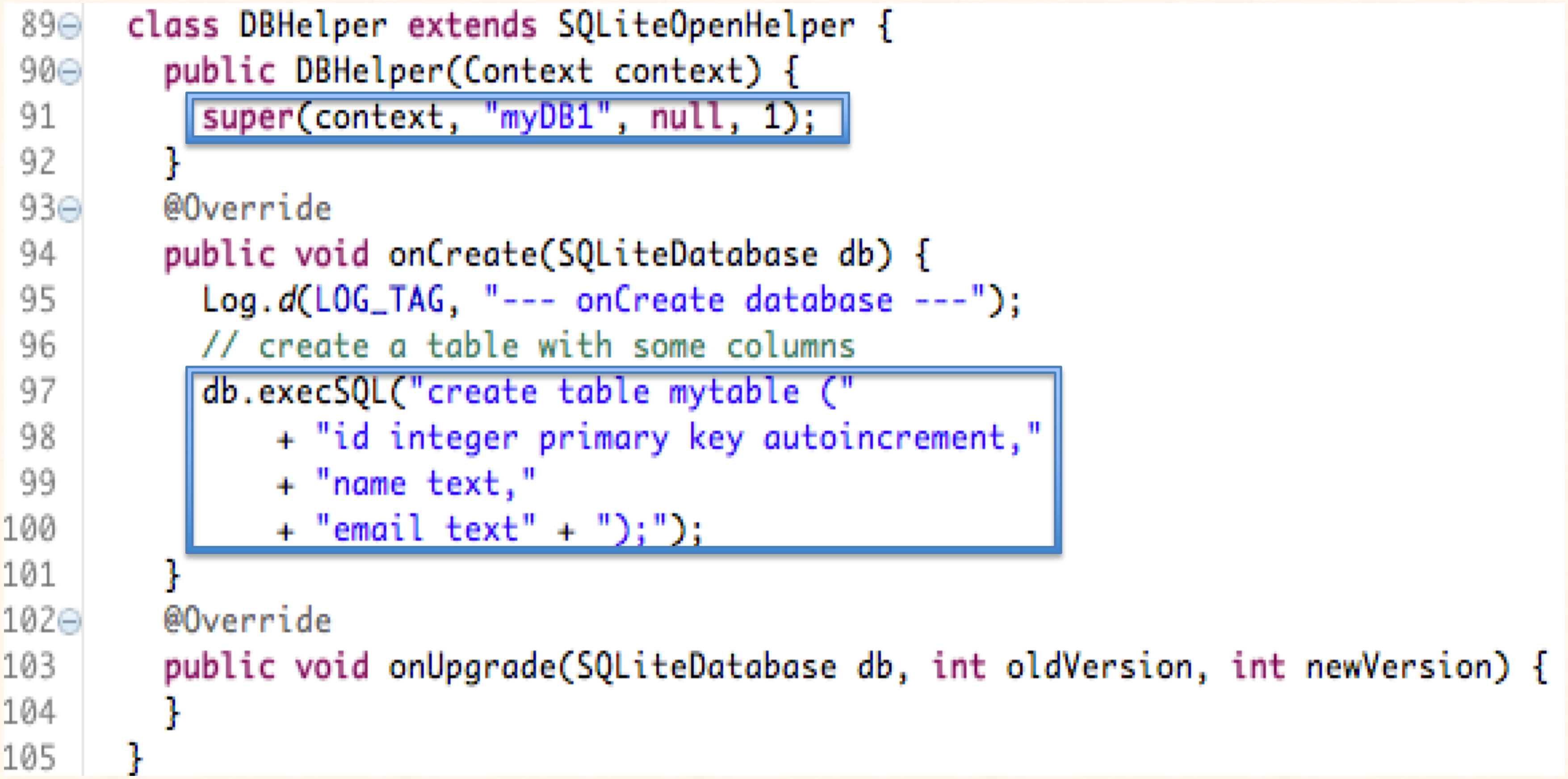
SQLite



```java
public class MainActivity extends Activity implements OnClickListener {
  final String LOG_TAG = "myLogs";
  Button btnAdd, btnRead, btnClear;
  EditText etName, etEmail;
  DBHelper dbHelper;
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    btnAdd = (Button) findViewById(R.id.btnAdd);
    btnAdd.setOnClickListener(this);
    btnRead = (Button) findViewById(R.id.btnRead);
    btnRead.setOnClickListener(this);
    btnClear = (Button) findViewById(R.id.btnClear);
    btnClear.setOnClickListener(this);
    etName = (EditText) findViewById(R.id.etName);
    etEmail = (EditText) findViewById(R.id.etEmail);
    // an object to create and control a DB
    dbHelper = new DBHelper(this);
  }
  public void onClick(View v) {
  class DBHelper extends SQLiteOpenHelper {
}
```

SQLiteOpenHelper

- SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)
  - Create a helper object to create, open, and/or manage a database.



```java
class DBHelper extends SQLiteOpenHelper {
  public DBHelper(Context context) {
    super(context, "myDB1", null, 1);
  }
  @Override
  public void onCreate(SQLiteDatabase db) {
    Log.d(LOG_TAG, "--- onCreate database ---");
    // create a table with some columns
    db.execSQL("create table mytable ("
        + "id integer primary key autoincrement,"
        + "name text,"
        + "email text" + ");");
  }
  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  }
}
```

onClick: DB Access

```
@Override
public void onClick(View v) {
  // an object for data
  ContentValues cv = new ContentValues();
  // get data from user
  String name = etName.getText().toString();
  String email = etEmail.getText().toString();
  // connect to DB
  SQLiteDatabase db = dbHelper.getWritableDatabase();
  switch (v.getId()) {
  case R.id.btnAdd:
    //..
    break;
  case R.id.btnRead:
    //..
    break;
  case R.id.btnClear:
    //..
    break;
  }
  // close DB
  dbHelper.close();
}
```

onClick: DB Access

- Am Start wird das Objekt dbHelper die Verbindung mit dem Datenbank erstellen:

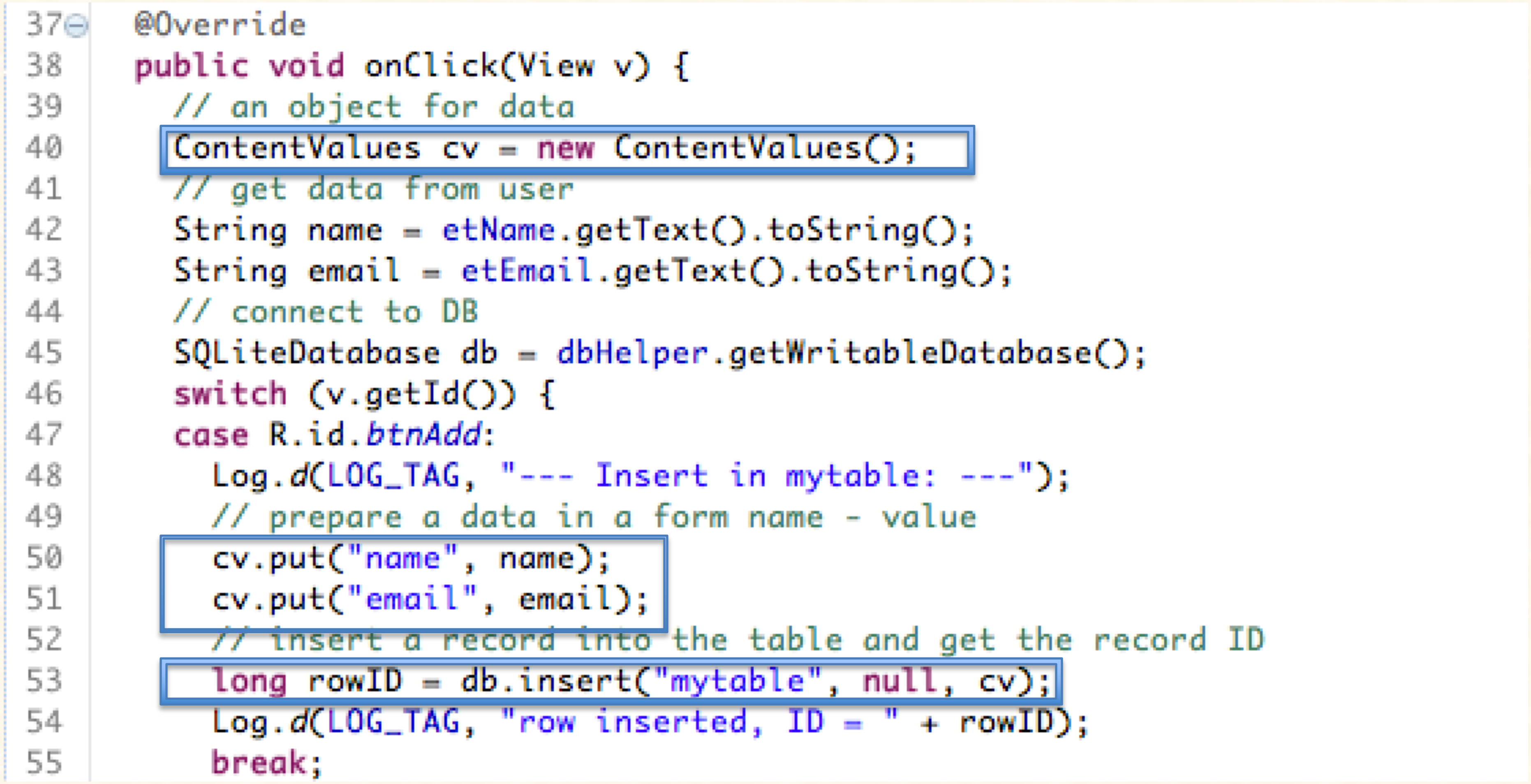
  SQLiteDatabase db = dbHelper.getWritableDatabase();

  - Create and/or open a database that will be used for reading and writing. The first time this is called, the database will be opened and onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and/or onOpen(SQLiteDatabase) will be called.
  - Returns a read/write database object valid until close() is called

- Am Ende wird das Objekt dbHelper die Verbindung mit dem Datenbank schließen:

  dbHelper.close()

btnAdd:

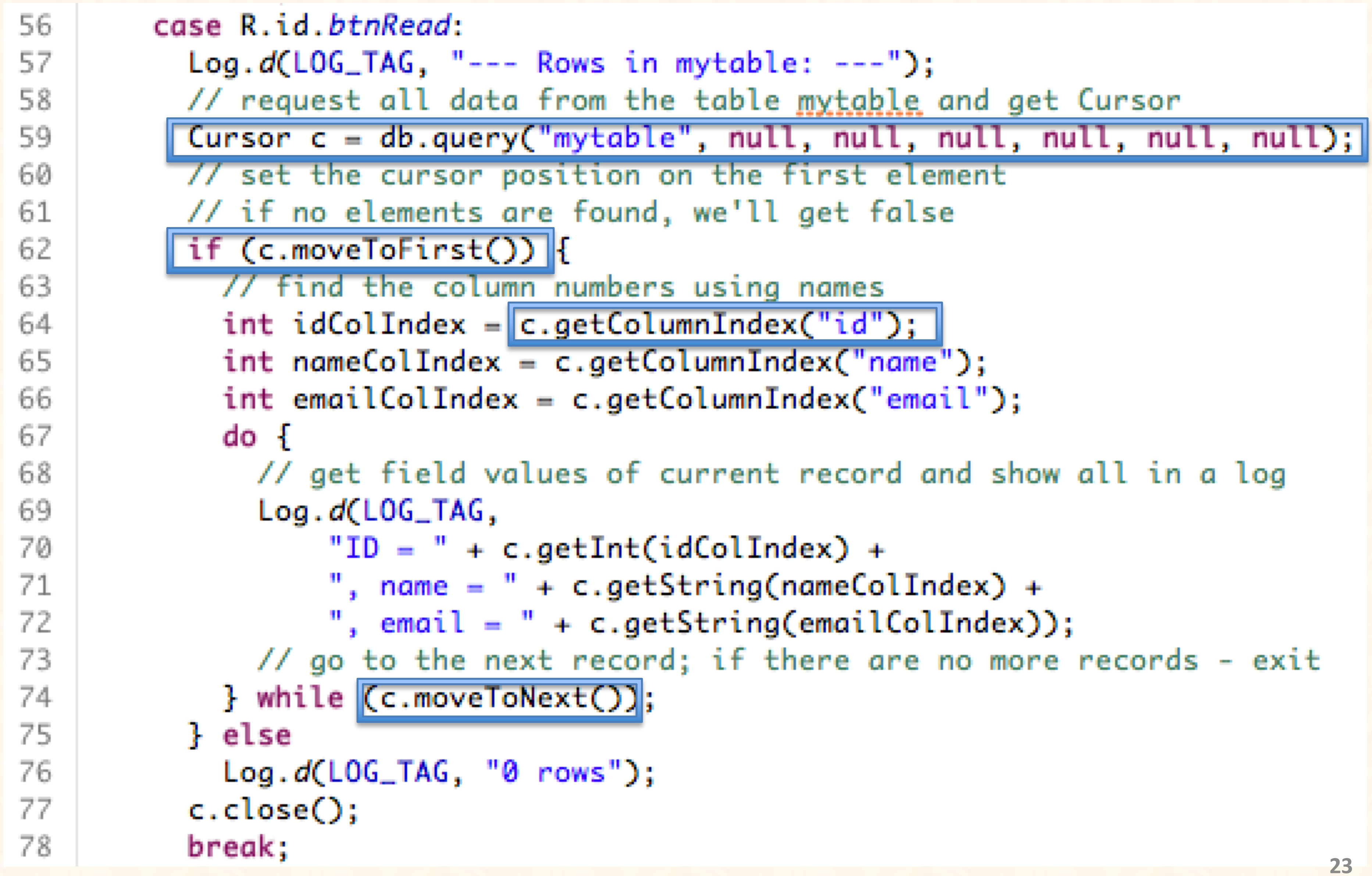


```
@Override
public void onClick(View v) {
  // an object for data
  ContentValues cv = new ContentValues();
  // get data from user
  String name = etName.getText().toString();
  String email = etEmail.getText().toString();
  // connect to DB
  SQLiteDatabase db = dbHelper.getWritableDatabase();
  switch (v.getId()) {
  case R.id.btnAdd:
    Log.d(LOG_TAG, "--- Insert in mytable: ---");
    // prepare a data in a form name - value
    cv.put("name", name);
    cv.put("email", email);
    // insert a record into the table and get the record ID
    long rowID = db.insert("mytable", null, cv);
    Log.d(LOG_TAG, "row inserted, ID = " + rowID);
    break;
```

btnAdd:

- **ContentValues cv = new ContentValues()**
  - Creates an empty set of values using the default initial size
- **cv.put("name", name);**
  cv.put("email", email);
  long rowID = db.insert("mytable", null, cv);
  - long insert(String table, String nullColumnHack, ContentValues values)
    Convenience method for inserting a row into the database.
- Das Object cv wird am onClick() Ende zerstreut!

btnRead:



```java
case R.id.btnRead:
  Log.d(LOG_TAG, "--- Rows in mytable: ---");
  // request all data from the table mytable and get Cursor
  Cursor c = db.query("mytable", null, null, null, null, null, null);
  // set the cursor position on the first element
  // if no elements are found, we'll get false
  if (c.moveToFirst()) {
    // find the column numbers using names
    int idColIndex = c.getColumnIndex("id");
    int nameColIndex = c.getColumnIndex("name");
    int emailColIndex = c.getColumnIndex("email");
    do {
      // get field values of current record and show all in a log
      Log.d(LOG_TAG,
          "ID = " + c.getInt(idColIndex) +
          ", name = " + c.getString(nameColIndex) +
          ", email = " + c.getString(emailColIndex));
      // go to the next record; if there are no more records - exit
    } while (c.moveToNext());
  } else
    Log.d(LOG_TAG, "0 rows");
  c.close();
  break;
```

btnRead:

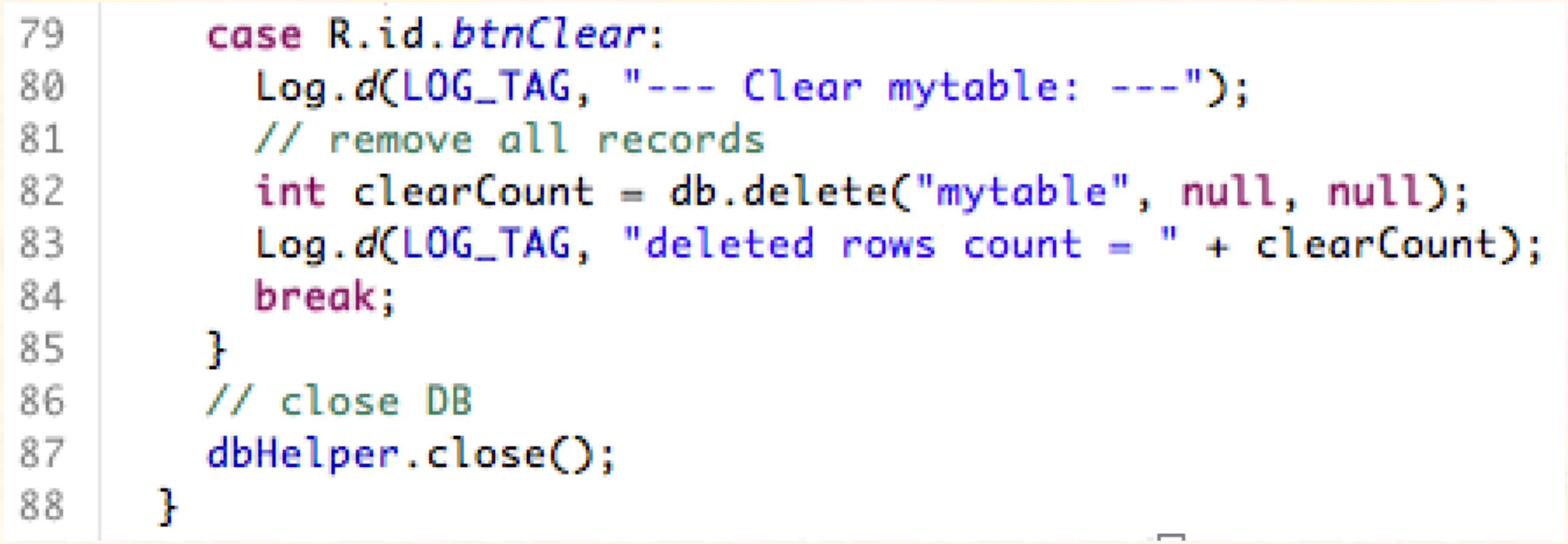
- Ein Result Set kann aus DB erstellt werden:
  public Cursor query (
  ```
  String table, String[] columns,
  String selection, String[] selectionArgs,
  String groupBy, String having,
  String orderBy)
  ```
- Das Objekt Cursor hilft den Result Set bearbeiten
- Es werden alle Records aus der Tabelle "mytable" gelesen:
  ```
  Cursor c = db.query("mytable",
                    null, null, null, null, null, null);
  ```

Object Cursor

- Object Cursor
  - Navigation:
    boolean moveToFirst (),
    boolean moveNext ()
  - Column Index/Name Access:
    int getColumnIndex (String columnName)
    - Returns the zero-based index for the given column name, or -1 if the column doesn't exist

    String getColumnName (int columnIndex)
  - Data Access:
    int getInt(int columnIndex)
    double getDouble (int columnIndex)
    String getString(int columnIndex)
    ...

btnClear

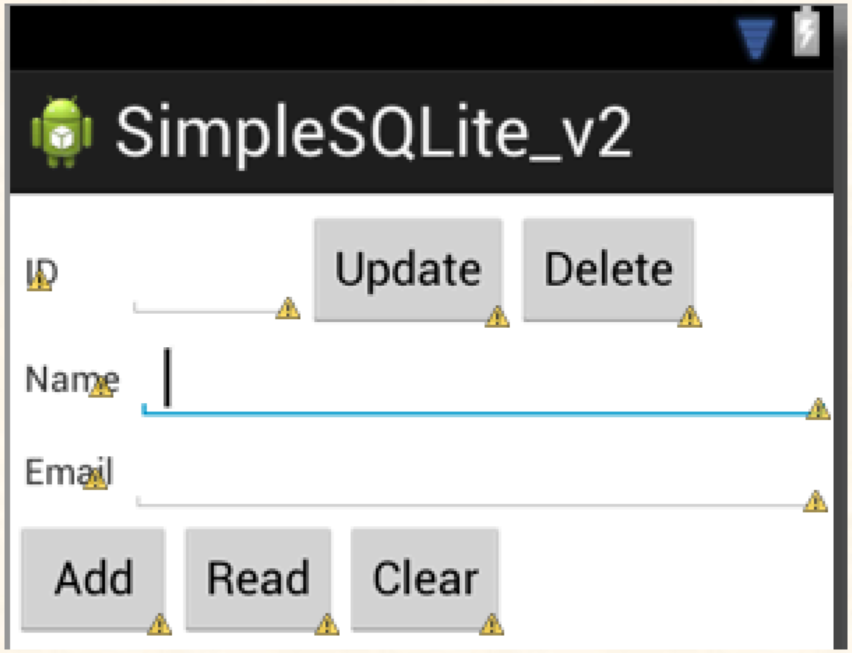


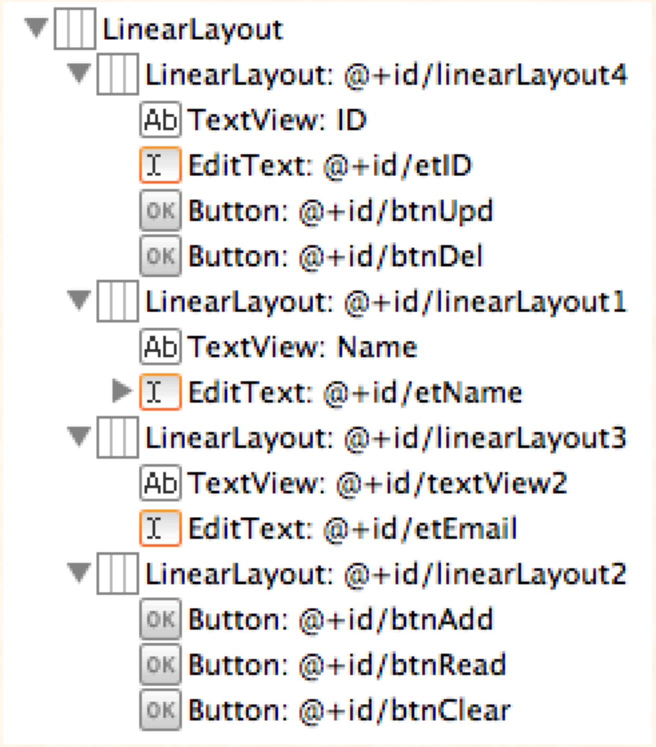
```
    case R.id.btnClear:
      Log.d(LOG_TAG, "--- Clear mytable: ---");
      // remove all records
      int clearCount = db.delete("mytable", null, null);
      Log.d(LOG_TAG, "deleted rows count = " + clearCount);
      break;
    }
    // close DB
    dbHelper.close();
  }
```

- die Records in einer Tabelle können entfernt werden:
  public int delete (String table, String whereClause,
  String[] whereArgs)
- Löchen wir alle Records in der Tabelle **mytable**:
  int clearCount = db.delete("mytable", null, null);

SimpleSQLite_v2

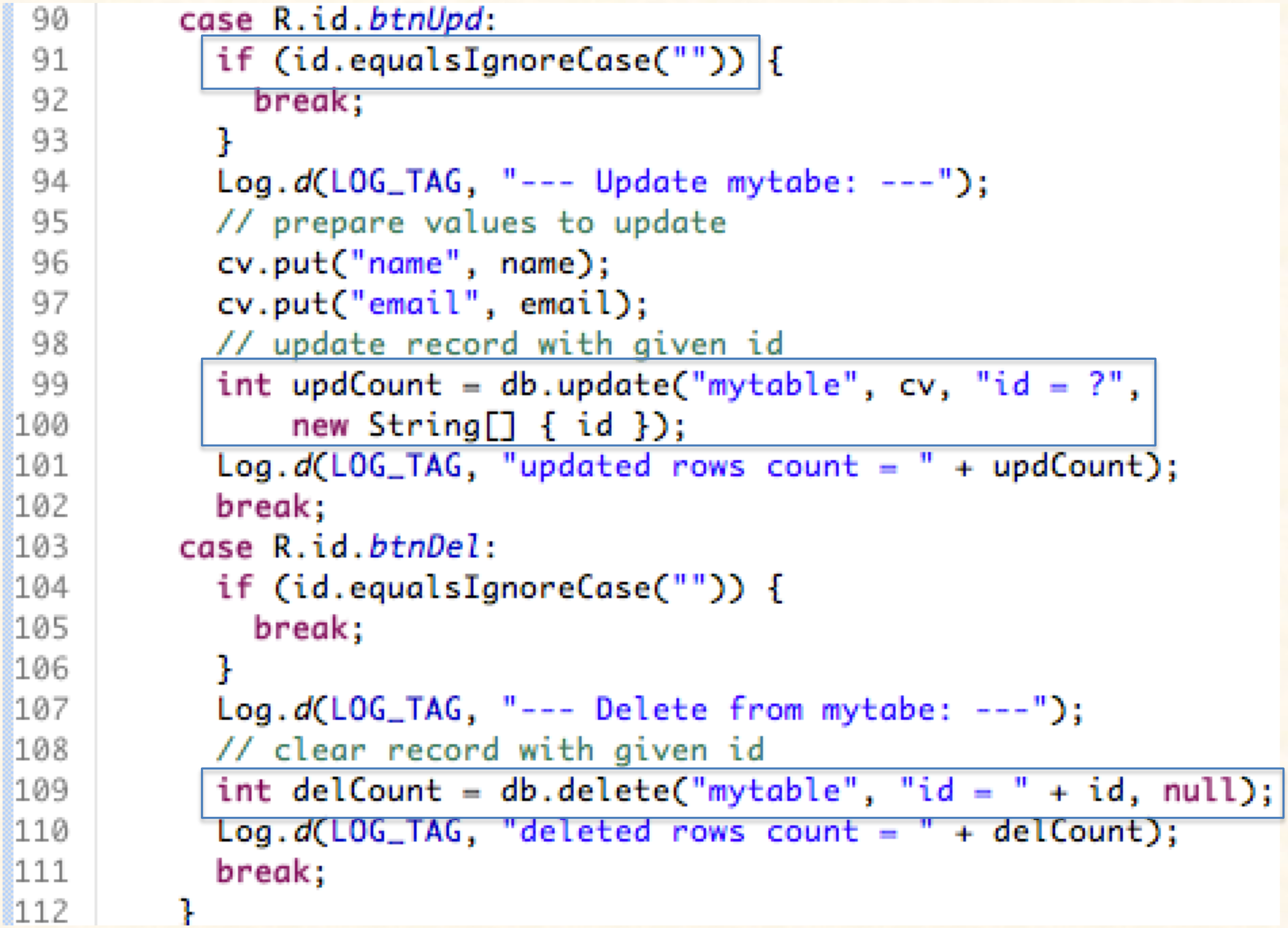
- Erweitern wir unseren Projekt. Geben wir ein TextEdit Feld für ID und zwei Knöpfe:
- Das modifizierte Projekt: P0342_SimpleSQLite_v2
http://startandroid.ru/ru/uroki/vse-uroki-spiskom/75-urok-35-metody-query-i-delete-s-ukazaniem-uslovija.html





```
LinearLayout
  LinearLayout: @+id/linearLayout4
    TextView: ID
    EditText: @+id/etID
    Button: @+id/btnUpd
    Button: @+id/btnDel
  LinearLayout: @+id/linearLayout1
    TextView: Name
    EditText: @+id/etName
  LinearLayout: @+id/linearLayout3
    TextView: @+id/textView2
    EditText: @+id/etEmail
  LinearLayout: @+id/linearLayout2
    Button: @+id/btnAdd
    Button: @+id/btnRead
    Button: @+id/btnClear
```

btnUpd



```
    case R.id.btnUpd:
      if (id.equalsIgnoreCase("")) {
        break;
      }
      Log.d(LOG_TAG, "--- Update mytabe: ---");
      // prepare values to update
      cv.put("name", name);
      cv.put("email", email);
      // update record with given id
      int updCount = db.update("mytable", cv, "id = ?",
          new String[] { id });
      Log.d(LOG_TAG, "updated rows count = " + updCount);
      break;
    case R.id.btnDel:
      if (id.equalsIgnoreCase("")) {
        break;
      }
      Log.d(LOG_TAG, "--- Delete from mytabe: ---");
      // clear record with given id
      int delCount = db.delete("mytable", "id = " + id, null);
      Log.d(LOG_TAG, "deleted rows count = " + delCount);
      break;
    }
```

btnUpd

```
int updCount = db.update("mytable", cv, "id = ?",
        new String[] { id });
```

- public int update (String table, ContentValues values, String whereClause, String[] whereArgs)
  - Convenience method for updating rows in the database.
  - Parameters

• table	the table to update in
• values	a map from column names to new column values. null is a valid value that will be translated to NULL.
• whereClause	the optional WHERE clause to apply when updating. Passing null will update all rows.
• whereArgs	You may include ?s in the where clause, which will be replaced by the values from whereArgs. The values will be bound as Strings.

  - Returns the number of rows affected

btnUpd

```
int delCount = db.delete("mytable", "id = " + id,
    null);
```

- public int delete (String table, String whereClause, String[] whereArgs)
  - Convenience method for deleting rows in the database.
  - Parameters

whereClause	the optional WHERE clause to apply when deleting. Passing null will delete all rows.
whereArgs	You may include ?s in the where clause, which will be replaced by the values from whereArgs. The values will be bound as Strings.

  - Returns the number of rows affected if a whereClause is passed in, 0 otherwise. To remove all rows and get a count pass "1" as the whereClause.

Datenaufbewahrung
in Android Apps

Fragen?