# 多媒体技术实验

2018-2019-2

1. 课程介绍

课程实验目的：

多媒体应用编程实验是实践类课程，通过课堂讲授和工程实验交叉进行，以实际案例驱动教学等方法和手段，能使学生了解多媒体技术历史和发展现状，熟悉和掌握多媒体应用的关键技术，基于开源的Android平台，掌握多媒体应用开发框架，掌握多媒体应用的开发技术，使学生具备开发多媒体应用的能力。

课程学时为40。

开课时间：1-10周，周一，1-4节

地点：软件学院大楼西303

考核方法：案例分析报告30，APP设计与开发程序及报告。

工作实验室：主楼中412

联系方式：

电话：13981829632

Email：[chjiang@uestc.edu.cn](mailto:chjiang@uestc.edu.cn)

[chjohn@163.com](mailto:chjohn@163.com)

QQ/WX：569411087

课程内容：约有10章。

详细内容参考《多媒体应用编程实验》教学大纲

2019-2-25

1. 第一章多媒体技术介绍
2. 第二章Android开发平台简介
3. 应用程序框架层
4. Android应用的开发结构
5. Android开发环境搭建
6. Android应用程序编写.
7. 实验

2019-3-4

1. 第三章 音频系统框架及开发：格式和编解码器支持功能
2. 音频系统框架及开发技术
3. 游戏声效的实现
4. 简易音乐播放器的实现
5. 上机实践Android音频系统框架。

掌握常用音频应用接口类，掌握音频开发的基本方法。包含有SoundPool的用法，AudioManager的用法，MediaPlayer的用法。

1. 实验

2019-3-11

1. 第四章 多媒体框架及开发
2. 第五章 多媒体开发常用视图

重点：

1. 多媒体框架的工作原理
2. 开发的三种视图框架：View、SurfaceView、GLSurfaceView
3. 二维动画应用，掌握View和SurfaceView框架下的渲染方法
4. 实验

2019-3-18

1. 第六章 SurfaceView框架

重点：

1. 视图机制
2. 常用的Canvas方法
3. 实验

2019-3-25

1. 第七章 Android数据存储技术

重点：

1. SharePreference的实践
2. SQLite的实践
3. File的实践
4. Content Provider的实践
5. 实验
6. Android的数据存储技术
7. SharePreference、SQLite、File、Content Provider的数据存储方式

项目分析报告：

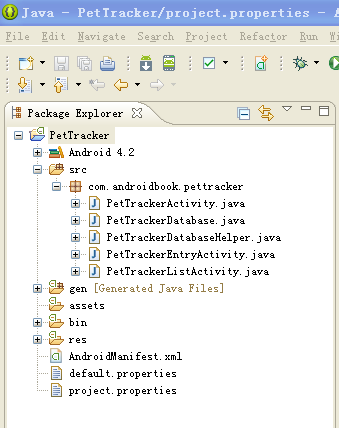
项目名：PetTracker

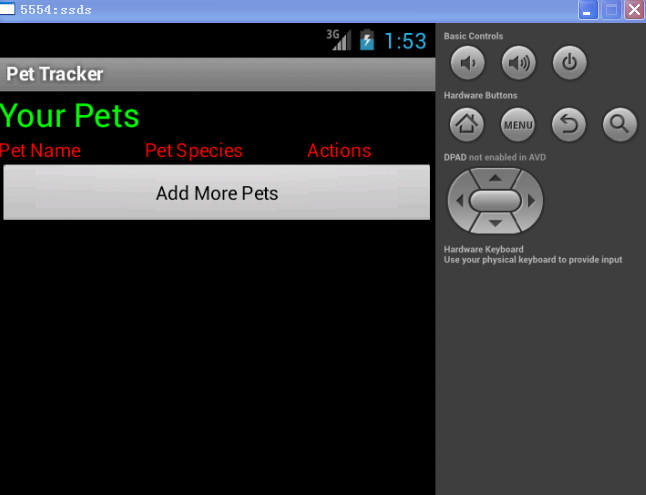
报告要求：

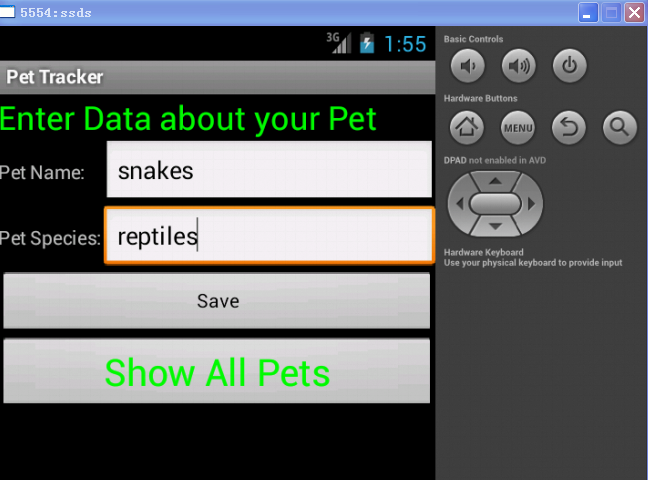
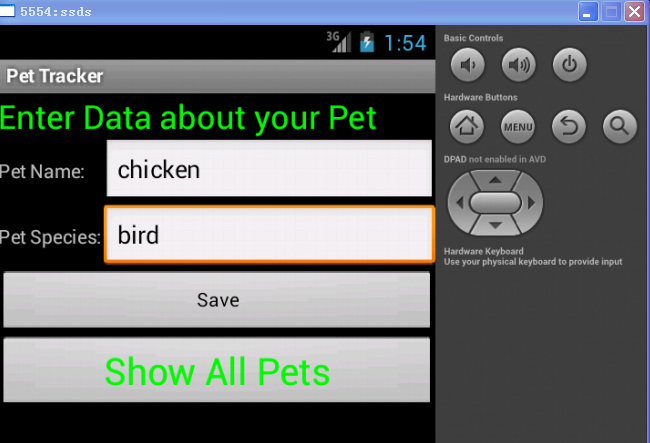
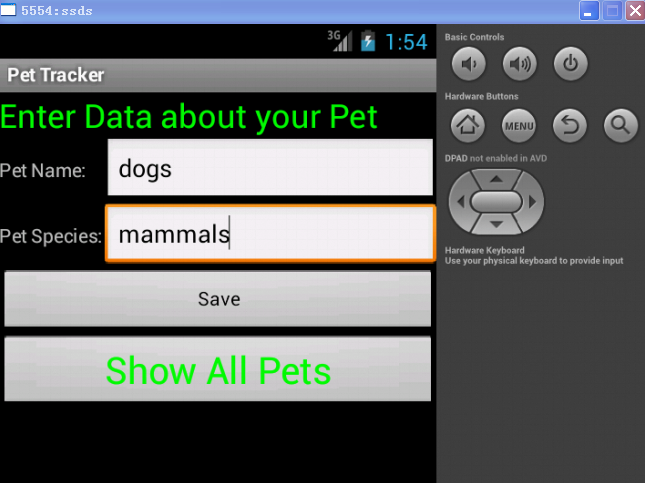
1. 分析项目功能；
2. 该项目实现的功能是实现数据的存储。
3. 在此基础之上进行数据的增删查功能
4. 功能实现的方法；

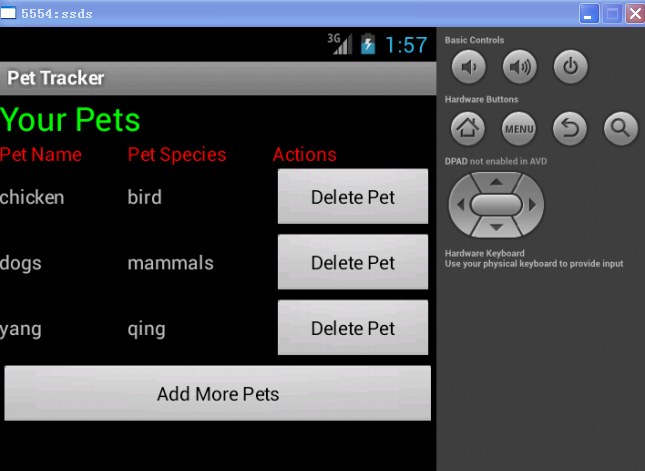
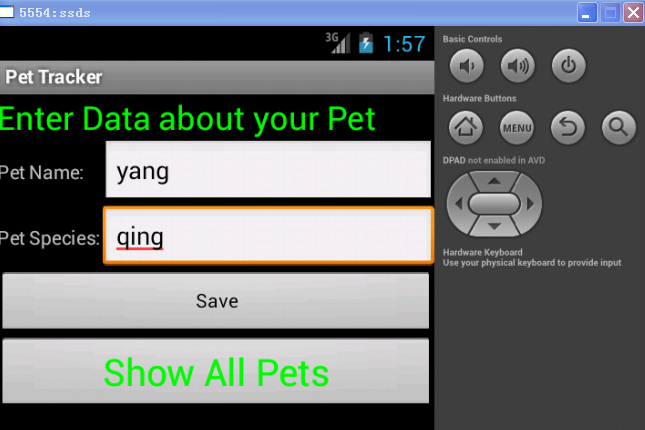
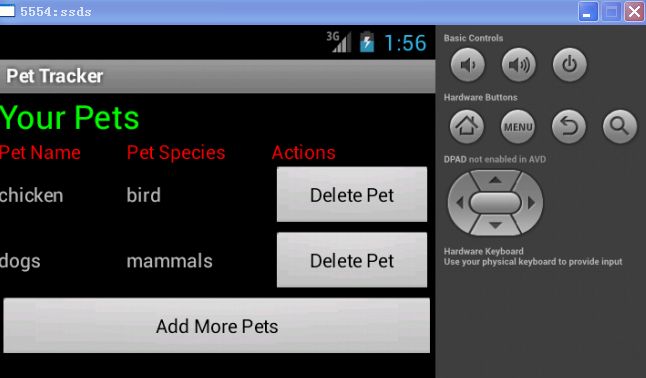
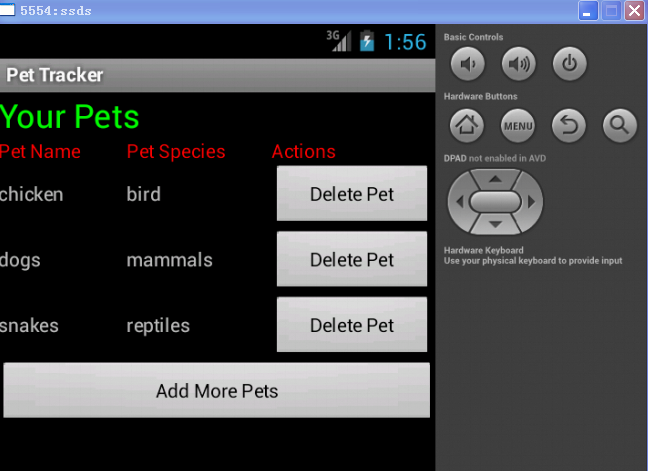
该项目主要包括5个方法：

1. PetTrackerDatabaseHelper()
2. PetTrackerDatabase()
3. onCreate()
4. onUpgrade()
5. onOpen()
6. PetTrackerEntryActivity()
7. onClick()
8. fillAutoCompleteFromDatabase()
9. PetTrackerListActivity()
10. fillPetList()
11. deletePet()
12. 运行结果描述。
13. 首先输入关于宠物的数据，包括宠物的名称和种类
14. 点击保存
15. 查看所有输入的宠物
16. 删除特定的宠物
17. 实现数据的增删查功能









报告提交时间和方式：

1. 提交时间：第7周星期五之前；
2. 方式：[发邮件到569411087@qq.com](mailto:发邮件到569411087@qq.com)

文件名：学号+姓名+ PetTracker分析报告.doc

**（1）PetTrackerActivity.java：**

**package** com.androidbook.pettracker;

**import** android.app.Activity;

**import** android.os.Bundle;

**public** **class** PetTrackerActivity **extends** Activity {

// Our application database

**protected** PetTrackerDatabaseHelper mDatabase = **null**;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

mDatabase = **new** PetTrackerDatabaseHelper(**this**.getApplicationContext());

}

@Override

**protected** **void** onDestroy() {

**super**.onDestroy();

**if**(mDatabase != **null**)

{

mDatabase.close();

}

}

}

**（2）PetTrackerDatabase.java**

**package** com.androidbook.pettracker;

**import** android.provider.BaseColumns;

// We use this class to keep track of database schema information like table and column names

**public** **final** **class** PetTrackerDatabase {

**private** PetTrackerDatabase() {}

// Pets table

**public** **static** **final** **class** Pets **implements** BaseColumns {

**private** Pets() {}

**public** **static** **final** String *PETS\_TABLE\_NAME* = "table\_pets";

**public** **static** **final** String *PET\_NAME* = "pet\_name";

**public** **static** **final** String *PET\_TYPE\_ID* = "pet\_type\_id";

**public** **static** **final** String *DEFAULT\_SORT\_ORDER* = "pet\_name ASC";

}

// Pet Type table

**public** **static** **final** **class** PetType **implements** BaseColumns {

**private** PetType() {}

**public** **static** **final** String *PETTYPE\_TABLE\_NAME* = "table\_pettypes";

**public** **static** **final** String *PET\_TYPE\_NAME* = "pet\_type";

**public** **static** **final** String *DEFAULT\_SORT\_ORDER* = "pet\_type ASC";

}

}

**(3)** **PetTrackerDatabaseHelper.java**

**package** com.androidbook.pettracker;

**import** com.androidbook.pettracker.PetTrackerDatabase.PetType;

**import** com.androidbook.pettracker.PetTrackerDatabase.Pets;

**import** android.content.Context;

**import** android.database.sqlite.SQLiteDatabase;

**import** android.database.sqlite.SQLiteOpenHelper;

// This class handles the creation and versioning of the application database

**class** PetTrackerDatabaseHelper **extends** SQLiteOpenHelper {

**private** **static** **final** String *DATABASE\_NAME* = "pet\_tracker.db";

**private** **static** **final** **int** *DATABASE\_VERSION* = 1;

PetTrackerDatabaseHelper(Context context) {

**super**(context, *DATABASE\_NAME*, **null**, *DATABASE\_VERSION*);

}

@Override

**public** **void** onCreate(SQLiteDatabase db) {

// Create the PetType table

db.execSQL("CREATE TABLE " + PetType.*PETTYPE\_TABLE\_NAME*+ " ("

+ PetType.*\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT ,"

+ PetType.*PET\_TYPE\_NAME* + " TEXT"

+ ");");

// Create the Pets table

db.execSQL("CREATE TABLE " + Pets.*PETS\_TABLE\_NAME* + " ("

+ Pets.*\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT ,"

+ Pets.*PET\_NAME* + " TEXT,"

+ Pets.*PET\_TYPE\_ID* + " INTEGER" // this is a foreign key to the pet type table

+ ");");

}

@Override

**public** **void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {

// Housekeeping here.

// Implement how "move" your application data during an upgrade of schema versions

// There is no ALTER TABLE command in SQLite, so this generally involves

// CREATING a new table, moving data if possible, or deleting the old data and starting fresh

// Your call.

}

@Override

**public** **void** onOpen(SQLiteDatabase db) {

**super**.onOpen(db);

}

}

**（4）PetTrackerEntryActivity**

**package** com.androidbook.pettracker;

**import** android.content.ContentValues;

**import** android.content.Intent;

**import** android.database.Cursor;

**import** android.database.sqlite.SQLiteDatabase;

**import** android.database.sqlite.SQLiteQueryBuilder;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.ArrayAdapter;

**import** android.widget.AutoCompleteTextView;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** com.androidbook.pettracker.PetTrackerDatabase.PetType;

**import** com.androidbook.pettracker.PetTrackerDatabase.Pets;

// Pet Entry Screen

**public** **class** PetTrackerEntryActivity **extends** PetTrackerActivity {

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*petentry*);

// Fill AutoComplete word list from database

fillAutoCompleteFromDatabase();

// Handle Save Button

**final** Button savePet = (Button) findViewById(R.id.*ButtonSave*);

savePet.setOnClickListener(**new** View.OnClickListener() {

**public** **void** onClick(View v) {

**final** EditText petName = (EditText) findViewById(R.id.*EditTextName*);

**final** EditText petType = (EditText) findViewById(R.id.*EditTextSpecies*);

// Save new records

SQLiteDatabase db = mDatabase.getWritableDatabase();

db.beginTransaction();

**try** {

// check if species type exists already

**long** rowId = 0;

String strPetType = petType.getText().toString()

.toLowerCase();

// SQL Query

SQLiteQueryBuilder queryBuilder = **new** SQLiteQueryBuilder();

queryBuilder.setTables(PetType.*PETTYPE\_TABLE\_NAME*);

queryBuilder.appendWhere(PetType.*PET\_TYPE\_NAME* + "='"

+ strPetType + "'");

// run the query

Cursor c = queryBuilder.query(db, **null**, **null**, **null**, **null**,

**null**, **null**);

**if** (c.getCount() == 0) {

// add the new type to our list

ContentValues typeRecordToAdd = **new** ContentValues();

typeRecordToAdd.put(PetType.*PET\_TYPE\_NAME*, strPetType);

rowId = db.insert(PetType.*PETTYPE\_TABLE\_NAME*,

PetType.*PET\_TYPE\_NAME*, typeRecordToAdd);

} **else** {

c.moveToFirst();

rowId = c.getLong(c.getColumnIndex(PetType.*\_ID*));

}

c.close();

// Always insert new pet records, even if the names clash

ContentValues petRecordToAdd = **new** ContentValues();

petRecordToAdd.put(Pets.*PET\_NAME*, petName.getText()

.toString());

petRecordToAdd.put(Pets.*PET\_TYPE\_ID*, rowId);

db.insert(Pets.*PETS\_TABLE\_NAME*, Pets.*PET\_NAME*,

petRecordToAdd);

db.setTransactionSuccessful();

} **finally** {

db.endTransaction();

}

// reset form

petName.setText(**null**);

petType.setText(**null**);

db.close();

}

});

// Handle Go to List button

**final** Button gotoList = (Button) findViewById(R.id.*ButtonShowPets*);

gotoList.setOnClickListener(**new** View.OnClickListener() {

**public** **void** onClick(View v) {

// Go to other activity that displays pet list

Intent intent = **new** Intent(PetTrackerEntryActivity.**this**, PetTrackerListActivity.**class**);

startActivity(intent);

}

});

}

// This method fills the AutoCompleteTextView data adapter with pet types from the database

// In this case, we are using a manual method of mapping Cursor data to

// an ArrayAdapter which we then use in the AutoCompelteTextView

// We show you this method to illustrate how you can use a simple array to seed the AutoText.

// Normally for ListViews and TextViews, you'd just use a SimpleCursorAdapter or a CursorAdapter.

// Unfortunately with AutoCompletTextView, those adapter/view pairings enforce the "value" to the chosen auto-complete string

// to be the id of the item, instead of the string itself, so it doesn't work quite as one would like by default.

// The more appropriate way to handle this is using data-binding, which involves implementing:

// SimpleCursorAdapter.CursorToStringConverter and a FilterQueryProvider is shown in the sample app called MediaPetTracker

**void** fillAutoCompleteFromDatabase()

{

SQLiteDatabase db = mDatabase.getReadableDatabase();

Cursor c = db.query(PetType.*PETTYPE\_TABLE\_NAME*, **new** String[] {PetType.*PET\_TYPE\_NAME*}, **null**, **null**,

**null**, **null**, PetType.*DEFAULT\_SORT\_ORDER*);

**int** iNumberOfSpeciesTypes = c.getCount();

String astrAutoTextOptions[] = **new** String[iNumberOfSpeciesTypes];

**if**((iNumberOfSpeciesTypes > 0) && (c.moveToFirst()))

{

**for**(**int** i = 0; i < iNumberOfSpeciesTypes; i++)

{

astrAutoTextOptions[i] = c.getString(c.getColumnIndex(PetType.*PET\_TYPE\_NAME*));

c.moveToNext();

}

ArrayAdapter<String> adapter =

**new** ArrayAdapter<String>(

**this**,

android.R.layout.*simple\_dropdown\_item\_1line*,

astrAutoTextOptions);

AutoCompleteTextView text = (AutoCompleteTextView) findViewById(R.id.*EditTextSpecies*);

text.setAdapter(adapter);

}

c.close();

db.close();

}

}

**（5）PetTrackerListActivity.java**

**package** com.androidbook.pettracker;

**import** android.database.Cursor;

**import** android.database.sqlite.SQLiteDatabase;

**import** android.database.sqlite.SQLiteQueryBuilder;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**import** android.widget.TableLayout;

**import** android.widget.TableRow;

**import** android.widget.TextView;

**import** com.androidbook.pettracker.PetTrackerDatabase.PetType;

**import** com.androidbook.pettracker.PetTrackerDatabase.Pets;

// Pet Listing Screen

**public** **class** PetTrackerListActivity **extends** PetTrackerActivity {

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*showpets*);

// Fill TableLayout with database results

fillPetList();

// Handle Go to List button

**final** Button gotoEntry = (Button) findViewById(R.id.*ButtonEnterMorePets*);

gotoEntry.setOnClickListener(**new** View.OnClickListener() {

**public** **void** onClick(View v) {

// Go to other activity that displays pet list

finish();

}

});

}

**public** **void** fillPetList()

{

// TableLayout where we want to Display list

**final** TableLayout petTable = (TableLayout) findViewById(R.id.*TableLayout\_PetList*);

// SQL Query

SQLiteQueryBuilder queryBuilder = **new** SQLiteQueryBuilder();

queryBuilder.setTables(Pets.*PETS\_TABLE\_NAME* +", " +PetType.*PETTYPE\_TABLE\_NAME*);

queryBuilder.appendWhere(Pets.*PETS\_TABLE\_NAME* + "." + Pets.*PET\_TYPE\_ID* + "=" + PetType.*PETTYPE\_TABLE\_NAME* + "." + PetType.*\_ID*);

// Get the database and run the query

SQLiteDatabase db = mDatabase.getReadableDatabase();

String asColumnsToReturn[] = { Pets.*PETS\_TABLE\_NAME* + "." + Pets.*PET\_NAME*, Pets.*PETS\_TABLE\_NAME* + "." + Pets.*\_ID*, PetType.*PETTYPE\_TABLE\_NAME* + "." +PetType.*PET\_TYPE\_NAME* };

Cursor c = queryBuilder.query(db, asColumnsToReturn, **null**, **null**, **null**, **null**, Pets.*DEFAULT\_SORT\_ORDER*);

// Display the results by adding some TableRows to the existing table layout

**if**(c.moveToFirst())

{

**for**(**int** i = 0; i< c.getCount(); i++)

{

TableRow newRow = **new** TableRow(**this**);

TextView nameCol = **new** TextView(**this**);

TextView typeCol = **new** TextView(**this**);

Button deleteButton = **new** Button(**this**);

newRow.setTag(c.getInt(c.getColumnIndex(Pets.*\_ID*))); // set the tag field on the TableRow view so we know which row to delete

nameCol.setText(c.getString(c.getColumnIndex(Pets.*PET\_NAME*)));

typeCol.setText(c.getString(c.getColumnIndex(PetType.*PET\_TYPE\_NAME*)));

deleteButton.setText("Delete Pet");

deleteButton.setTag(c.getInt(c.getColumnIndex(Pets.*\_ID*))); // set the tag field on the button so we know which ID to delete

deleteButton.setOnClickListener(**new** View.OnClickListener() {

**public** **void** onClick(View v) {

// Delete that pet

Integer id = (Integer) v.getTag();

deletePet(id);

// Find and destroy the row tagged with the deleted pet id in the Table

**final** TableLayout petTable = (TableLayout) findViewById(R.id.*TableLayout\_PetList*);

// This should return the TableRow as the first tagged view in the layout but it would be nice if it returned an array of views with that tag...

View viewToDelete = petTable.findViewWithTag(id);

petTable.removeView(viewToDelete);

}

});

newRow.addView(nameCol);

newRow.addView(typeCol);

newRow.addView(deleteButton);

petTable.addView(newRow);

c.moveToNext();

}

}

**else**

{

TableRow newRow = **new** TableRow(**this**);

TextView noResults = **new** TextView(**this**);

noResults.setText("No results to show.");

newRow.addView(noResults);

petTable.addView(newRow);

}

c.close();

db.close();

}

**public** **void** deletePet(Integer id)

{

SQLiteDatabase db = mDatabase.getWritableDatabase();

String astrArgs[] = { id.toString() };

db.delete(Pets.*PETS\_TABLE\_NAME*, Pets.*\_ID* + "=?",astrArgs );

db.close();

}

}