June 2024

Android
期末复习讲义

林星辰

前言

本讲义主要功能在于考前学习与记忆，并不具备教材学习功能。学习本讲义内容可搭配讲解视频，视频发布在下方所述 B 站号。

特别鸣谢：兰总对讲义写作过程中技术指导

本讲义正式动笔于考试前，至完成仅用两天，时间之仓促，难免有些纰漏，如读者发现，可反馈到 B 站 UP《林鹤鸣 ID》（ID 号：102399253）。

版本：0.2

2024 年 6 月 23 日

目录

1 基础 **3**
1.1 基础知识 . . . 3
1.2 常见布局与控件属性 . . . 4
1.3 高级控件 . . . 6
1.4 Activity . . . 8
1.5 数据存储 . . . 9
1.6 内容提供者 . . . 10
1.7 服务 . . . 10

2 重点部分 **11**
2.1 常见控件与布局 . . . 11
2.2 生命周期 . . . 11
2.2.1 基本函数 . . . 11
2.2.2 生命周期分析 . . . 11
2.3 两种数据传输方式 . . . 13
2.4 SQLites 数据存储 . . . 14
2.4.1 SQLiteOpenHelper . . . 14
2.4.2 游标 . . . 15
2.4.3 游标适配器 . . . 16
2.5 广播 . . . 17
2.5.1 动态注册广播接收者 . . . 18
2.5.2 自定义广播 . . . 19
2.5.3 广播类型 . . . 19
2.6 JSON 数据解析 . . . 20
2.6.1 数据格式 . . . 20
2.6.2 JSON 解析 . . . 21

3 代码 **23**
3.1 常见布局代码 . . . 23
3.1.1 左图图书信息代码 . . . 23
3.1.2 右图登录界面代码 . . . 24
3.2 生命周期核心代码 . . . 25
3.3 数据传输方式代码 . . . 26
3.4 StarbuzzDatabaseHelper 代码 . . . 27
3.5 有序广播代码 . . . 28

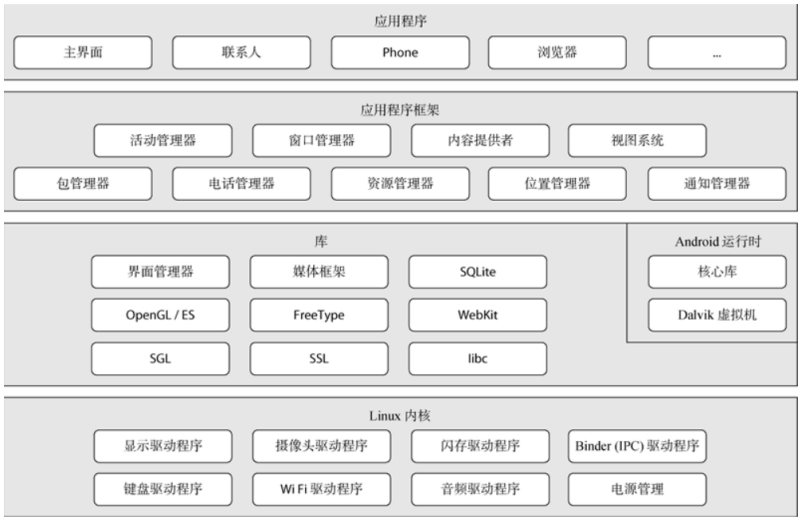
4 习题 **29**

5 答案 **31**

1 基础

1.1 基础知识

1. android 四层架构及特点



(a) 应用层: 表示用户直接接触和使用的部分。

(b) 应用框架层: 提供开发者用来构建应用程序的框架和工具。

(c) 库层和安卓运行时: 系统库和安卓运行时环境，提供基本功能支持。

(d) Linux 内核层: 安卓系统的基础，负责底层硬件的抽象和管理。

2. 项目程序结构

3. 资源管理与使用

- 通过 Java 代码调用图片资源

```
//调用mipmap文件夹中资源文件
getResources().getDrawable(R.mipmap.ic_launcher);
//调用以drawable开头的文件夹中的资源文件
getResources().getDrawable(R.drawable.icon);
```

- 通过 XML 代码调用图片资源

```
@mipmap/ic_launcher //调用mipmap文件夹中的资源文件
@drawable/icon    //调用以drawable开头的文件夹中的资源文件
```

- 通过 Java 代码调用布局资源文件

```
//在Activity的onCreate()方法中调用activity_main.xml布局资源
setContentView(R.layout.activity_main);
```

- 字符串资源
  - 定义

```
<resources>
    <string name="app_name">字符串</string>
</resources>
```

  - JAVA 调用

```
// 在Activity的onCreate()方法中调用名为app_name的字符串资源
getResources().getString(R.string.app_name);
```

  - XML 调用

```
// 在XML布局文件中调用名为app_name字符串资源
@string/app_name
```

4. Log 类，V：全部信息、D：调试信息、I 一般信息、W 警告信息、E：错误信息、Assert：断言失败后的错误信息

```
Log.v("MainActivity", "Verbose");
Log.d("MainActivity","Degug");
Log.i("MainActivity","Info");
Log.w("MainActivity", "Warning");
Log.e("MainActivity", "Error");
Log.wtf("MainActivity","Assert");
```

1.2 常见布局与控件属性

1. 了解常见布局与简单控件属性
2. Toast 显示在应用程序界面的最上层，显示一段时间后自动消失不会打断当前操作，也不获得焦点。
3. 实现页面

属性名称	功能描述
通用属性	
android:id	设置唯一标识
android:layout_width	设置宽度
android:layout_height	设置高度
android:background	设置背景
android:layout_margin	设置当前布局与屏幕边界或与周围控件的距离（外边距）
android:padding	设置当前布局与该布局中控件的距离（内边距）或控件与该控件中内容的距离
LinearLayout 线性布局	
android:orientation="vertical"	此属性控制控件排列方向：vertical(垂直)、horizontal(水平)
android:layout_gravity="center"	布局视图在其父容器中位置的属性
android:layout_weight="1"	权重
RelativeLayout 相对布局	
android:layout_centerInParent	设置当前控件位于父布局的中央位置
android:layout_centerVertical	设置当前控件位于父布局的垂直居中位置
android:layout_centerHorizontal	设置当前控件位于父控件的水平居中位置
android:layout_alignParentTop	设置当前控件是否与父控件顶端对齐
android:layout_alignParentLeft	设置当前控件是否与父控件左对齐
android:layout_alignParentRight	设置当前控件是否与父控件右对齐
android:layout_alignParentBottom	设置当前控件是否与父控件底端对齐
android:layout_above	设置当前控件位于某控件上方
android:layout_below	设置当前控件位于某控件下方
android:layout_toLeftOf	设置当前控件位于某控件左侧
android:layout_toRightOf	设置当前控件位于某控件右侧
android:layout_alignTop	设置当前控件的上边界与某控件的上边界对齐
android:layout_alignBottom	设置当前控件的下边界与某控件的下边界对齐
android:layout_alignLeft	设置当前控件的左边界与某控件的左边界对齐
android:layout_alignRight	设置当前控件的右边界与某控件的右边界对齐
TextView 控件	
android:text	设置文本内容
android:textColor	设置文字显示的颜色
android:textSize	设置文字大小，推荐单位为 sp
android:gravity	设置文本内容的位置（视图内容在该视图内部的对齐方式）
android:textStyle	设置文本样式，如 bold（粗体），italic（斜体），normal（正常）
EditText 编辑框（TextView 的子类）	
android:hint	控件中内容为空时显示的提示文本信息

下接下页

续前表

属性名称	功能描述
android:inputType	允许输入的信息类型
ImageView 图片 android:src="@drawable/图片名"	设置 ImageView 控件需要显示的图片资源
RadioButton 单选按钮 android:checked	指定是否选中的状态
RadioGroup 是单选组合框 RadioGroup.OnCheckedChangeListener	组里被选中的按钮发生改变时，该接口被自动调用
CheckBox 复选框 CompoundButton.OnCheckedChangeListener	当一个复选按钮的被选中的状态发生改变时，该接口被自动调用

```
// 当前组件Activity、提示的字符串信息、显示信息的时长 、显示信息
Toast.makeText(MainActivity.this,"WIFI已断开",Toast.LENGTH_SHORT).show();
```

1.3 高级控件

1. Button 点击事件实现

   (a) 给当前当前 Activity 实现 OnClickListener 接口

```
// 1. 当前Activity实现OnClickListener接口
      public class MainActivity extends AppCompatActivity implements View.
    onClickListener

// 2. 设置Button控件的点击监听事件
    btn.setOnClickListener(this); // this代表onClickListener的引用

// 3. 实现OnClickListener接口中的方法
  @Override
    public void onClick(View v) {
        Log.i("接口方式", "button  is clicked");
    }
```

   (b) 匿名内部类

```
    btn.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View v) {
           Log.i("匿名内部类方式", "button  is clicked");
       }
    });
```

   (c) 在布局文件中指定 Button 控件的 onClick 属性方式
   `android:onClick="click"`

2. 创建 AlertDialog 对话框步骤

   (a) 调用 AlertDialog 的静态内部类 Builder 创建 AlertDialog.Builder 的对象。

   (b) 调用 setTitle() 和 setIcon() 方法分别设置 AlertDialog 对话框的标题名称和图标。

(c) 调用 AlertDialog.Builder 的 setMessage()、setSingleChoiceItems() 或者 setMultiChoiceItems() 方法设置 AlertDialog 对话框的内容为简单提示对话框、单选对话框和多选对话框。

(d) 调用 AlertDialog.Builder 的 setPositiveButton() 和 setNegativeButton() 方法设置 AlertDialog 对话框的确定按钮和取消按钮。

(e) 调用 AlertDialog.Builder 的 create() 方法创建 AlertDialog 对象。

(f) 调用 AlertDialog 对象的 show() 方法显示该对话框。

(g) 调用 AlertDialog 对象的 dismiss() 方法取消该对话框。

```
AlertDialog dialog;
dialog = new AlertDialog.Builder(this)
                    .setTitle("普通对话框")
                   .setIcon(R.mipmap.ic_launcher)
                    .setMessage("是否退出应用？")
                    .setPositiveButton("确定",....)
                    .setNegativeButton("取消",.....)
                    .create();
dialog.show();
```

3. 单选对话框的内容区域显示为单选列表。单选列表通过 AlertDialog.Builder 对象调用 setSingleChoiceItem() 方法设置的

   - 单选列表中所有选项数据
   - 单选列表中的默认选项位置
   - 单选列表的点击事件监听器接口 DialogInterface.OnClickListener()

```
AlertDialog dialog;
       AlertDialog.Builder builder = new AlertDialog.Builder(this)
               .setTitle("设置字体大小")           //设置标题
               ......
               .setSingleChoiceItems(new String[]{"小号", "默认", "中号", "大号", "超大
   "}, textSize, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int which) {}})
               .setPositiveButton(“确定” ,....)//添加“确定”按钮
               .setNegativeButton(“取消” ,.....);
       dialog = builder.create();
       dialog.show();
```

4. 多选对话框的内容区域显示为多选列表。多选列表通过 AlertDialog.Builder 对象调用 setMultiChoiceItems() 方法设置的。

5. 常用菜单：选项菜单（OptionsMenu）、上下文菜单（ContextMenu）

   - 选项菜单（OptionsMenu）
     - 重写 `onCreateOptionsMenu()` 方法，在这个方法完成加载 Menu 资源
     - 重写 `onOptionsItemSelected()` 方法，实现选项监听
   - 上下文菜单（ContextMenu）
     - 重写 `onCreateContextMenu()` 方法，在这个方法完成加载 Menu 资源
     - 重写 `onContextItemSelected()` 方法，实现菜单项 id 监听

- 调用 `registerForContextMenu()` 方法, 注册上下文菜单

6. 适配器

- ArrayAdapter 和 SimpleAdapter 继承自 BaseAdapter
- BaseAdapter 顾名思义是基本的适配器。他实际上是一个抽象类，通常在自定义适配器时会继承 BaseAdapter

Adapter	**含义**
ArrayAdapter<T>	用来绑定一个数组与 ListView
SimpleAdapter	用来绑定在 xml 中定义的控件对应的数据
SimpleCursorAdapter	用来绑定游标得到的数据
BaseAdapter	通用的基础适配器，一般用自定义 Adapter 继承

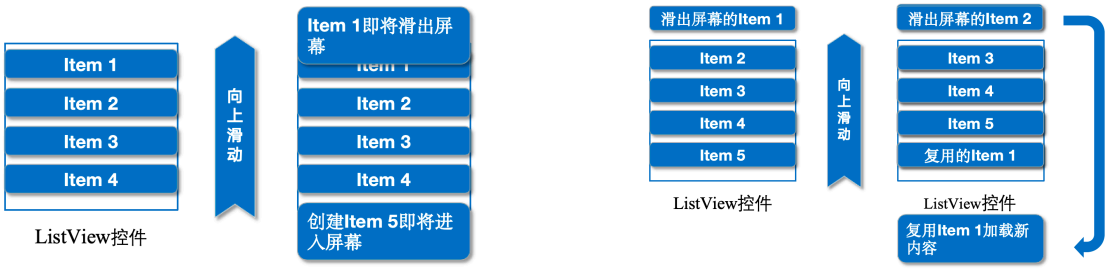
```
// 在布局文件中加入一个ListView控件，且 android:id="@+id/list_view"
ListView lv;

// 定义一个数组作为ListView中item的数据内容
private static final String[] data = {"apple","orange","pear","banana","peach","lemon","
    watermelon","strawbarry","grape","cherry"};

// 创建ArrayAdapter的实例对象
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.
    simple_list_item_1,data);
        lv = findViewById(R.id.list_view)
// 通过ListView的setAdapter()方法绑定ArrayAdapter
        lv.setAdapter(adapter);
```

- BaseAdapter 中的方法
  - getCount(int position)：获取 Item 条目的总数
  - getItem(int position)：根据 position（位置）获取某个 Item 的对象
  - getItemId()：根据 position（位置）获取某个 Item 的 id
  - getView()：获取相应 position 对应的 Item 视图
  - 当系统开始绘制 ListView 的时候，首先调用 getCount() 方法。得到它的返回值，即 ListView 的长度。然后系统调用 getView() 方法，根据这个长度逐一绘制 ListView 的每一行
  - 而 getItem() 和 getItemId() 则在需要处理和取得 Adapter 中的数据时调用
- 优化 ListView 加载数据
  - convertView 用于复用已经创建的视图。
  - ViewHolder 用于存储视图项中子视图的引用，避免重复调用 findViewById，从而提高性能。

1.4 Activity

1. 生命周期见后文

2. Intent



- 显式意图可以直接通过名称开启指定的目标组件
- 隐式意图通过指定 action 和 category 等属性，系统根据这些信息进行分析后寻找满足条件的目标 Activity

3. 两种数据传递
   - 使用 Intent 的 putExtra() 方法传递数据
   - 使用 Bundle 类传递数据
4. IntentFilter（意图过滤）在 AndroidManifest.xml 设置主活动

```
<activity
    android:name=".TwoActivity"
    android:exported="false" />
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

1.5 数据存储

1. 数据存储方式: 文件存储、SharedPreferences、SQLite、ContentProvider、网络存储
2. SharedPreferences 存储
   - 使用 SharedPreferences 类存储数据时，按照以下步骤进行操作：
     (a) 调用 `getSharedPreferences(String name, int mode)` 方法获取实例对象。
     (b) 由于该对象本身只能获取数据，不能对数据进行存储和修改，因此需要调用 SharedPreferences 类的 `edit()` 方法获取可编辑的 Editor 对象。
     (c) 最后通过该对象的 `putXXX()` 方法存储数据，例如 `putString()`、`putInt()` 等。
   - Editor 对象是以 key/value 的形式保存数据的，并且根据数据类型的不同，调用不同的方法：
     (a) 例如，存储字符串使用 `putString(String key, String value)`。
     (b) 存储整数使用 `putInt(String key, int value)`。
     (c) 存储布尔值使用 `putBoolean(String key, boolean value)`。
   - 操作完数据后，一定要调用 `commit()` 方法进行数据提交，否则所有操作不生效：

(a) `editor.commit()` 用于同步提交数据。

(b) `editor.apply()` 用于异步提交数据。

3. SQLite(见下文重点)

   - SQLite 的主要数据类型: INTEGER（任何整数类型）、REAL（任何浮点数）、TEXT（任何字符类型）、NUMERIC（布尔、日期和日期时间）、BLOB（二进制大对象）

4. 游标（见下文重点）

5. 游标适配器（见下文重点）

```
// 获取SharedPreferences实例对象sp，参数data表示文件名，MODE_PRIVATE表示文件操作模式
SharedPreferences sp = getSharedPreferences("data",MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();      // 获取编辑器
editor.putString("name", "传智播客");              // 存入String类型数据
editor.putInt("age", 8);                          // 存入int类型数据
editor.commit();                                  // 提交修改

// 读取 SharedPreferences中的数据
SharedPreferences sp = getSharedPreferences("data",MODE_PRIVATE);
String data = sp.getString("name","")    // 获取用户名
```

1.6 内容提供者

内容提供者（ContentProvider）暴露数据，ContentResolver 接收数据

1.7 服务

1. Service 通常被称为“后台服务”，Activity 的生命周期结束时，Service 仍然可以继续运行

2. 服务的启动方式

   - 通过 `startService()` 启动,需要自身调用 stopSelf() 方法或者其他组件调用 stopService() 方法时服务才能停止。
   - 通过 `bindService()` 启动,，需要调用 onUnbind() 方法解除绑定之后服务才会被销毁。

2 重点部分

2.1 常见控件与布局

尝试实现以下界面

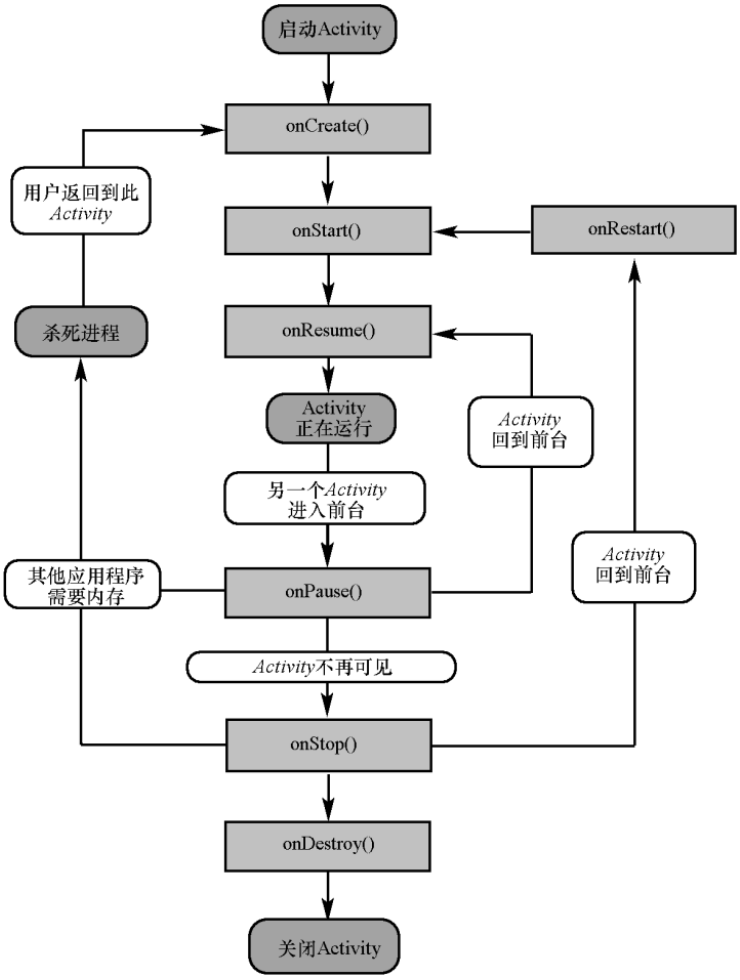
2.2 生命周期

2.2.1 基本函数

1. `onCreate()`: Activity 创建时调用，通常做一些初始化设置。
2. `onStart()`: Activity 即将可见时调用。
3. `onResume()`: 获取焦点时调用。
4. `onPause()`: （失去焦点）当前 Activity 被其他 Activity 覆盖或屏幕锁屏时调用。
5. `onStop()`: Activity 对用户不可见时调用。
6. `onRestart()`: （重新可见）Activity 从停止状态到再次启动时调用。
7. `onDestroy()`: Activity 销毁时调用。

2.2.2 生命周期分析

1. 启动 Activity 时
   (a) `onCreate`: 当 Activity 第一次被实例化时调用，用于初始化设置，如加载布局文件、绑定按钮监听器等。
   (b) `onStart`: Activity 变得可见但尚未获得用户焦点，不能与用户交互时调用。
   (c) `onResume`: Activity 可见且获得用户焦点，可以与用户交互时调用。



2. 按退出（Back）键

   (a) `onPause`: Activity 暂停，准备从前台返回至后台，用于保存数据和停止消耗 CPU 的操作。

   (b) `onStop`: Activity 不可见时调用。

   (c) `onDestroy`: Activity 即将被销毁，用于释放资源，如结束线程等。

3. 按 Home 键

   (a) ab 与上相同，没有 c，由于 Home 将程序挂到后台，故不调用 `onDestroy` 销毁进程。

4. 两个 Activity 的情况

   (a) 进入第一个 Activity 调用 `onCreate`、`onStart`、`onResume`。

   (b) 跳转第二个 Activity 时，将第一个 Activity 放到后台，调用 `onPause`。

   (c) 第一个 Activity 不可见，调用 `onStop`。

   (d) 点击返回时，返回到第一个 Activity 时，调用 `onRestart`。

   (e) 再调用 `onStart` 和 `onResume` 使第一个页面重新可见并获取焦点以交互。

   (f) 再次点击返回，调用 `onPause`、`onStop`、`onDestroy`。

5. 由 Activity1->Activity3->Back

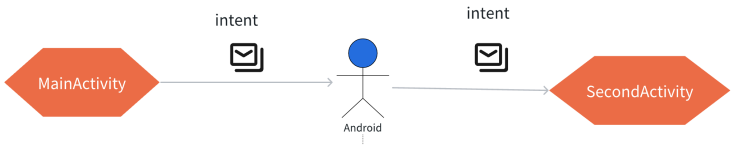
   (a) 进入第一个 Activity 调用 `onCreate`、`onStart`、`onResume`。

   (b) 跳转第三个 Activity 时，将第一个 Activity 放到后台，调用 `onPause`。

   (c) 由于第三个 Activity 不是新开而是弹窗形式，故不调用 `onStop`。

2.3 两种数据传输方式

1. Intent 被称为意图，是程序中各组件进行交互的一种重要方式，他不仅可以指定当前组件要执行的动作，还可以在不同组件之间进行数据传递。



2. Intent 完成 Activity 跳转

```
// 创建一个Intent对象，第1个参数为当前的Activity对象，第2个参数为要启动的目标Activity。
Intent intent = new Intent(FirstActivity.this,SecondActivity.class);
// 调用Activity的startActivity方法启动目标组件
startActivity(intent);
```

3. 使用 Intent 的 putExtra() 方法传递数据

   - 发送方

```
// 注册intent
Intent intent = new Intent();
// 设置要跳转的 Activity
intent.setClass(SendMessageActivity.this,ReceiveMessageAtivity.class);
// 两个参数，参数1：数据名称，参数2：数据信息
intent.putExtra("message","lxc is great");
// 启用该intent
startActivity(intent);
```

   - 接收方

```
Intent intent = getIntent();
// 获取传递的信息
String receiveMessage = intent.getStringExtra("message");
```

4. 使用 Bundle 类传递数据

```
/*   使用Bundle接受多个数据     */
Intent intent = getIntent();
Bundle bundle = intent.getExtras(); // 获取 bundle 内容

String Name = bundle.getString("name");
String receiveMessage = bundle.getString("message");

textView.setText(Name+" "+receiveMessage+"");
```

2.4 SQLites 数据存储

SQLite 是 Android 自带的一个轻量级的数据库，具有运算速度快，占用资源少的特点，支持基本 SQL 语法。可以存储大量数据。

2.4.1 SQLiteOpenHelper

1. Android 为了更加方便管理数据库，提供了一个 SQLiteOpenHelper 帮助类，这个类可以对数据库进行创建和升级。SQLiteOpenHelper 是一个抽象类，因此需要创建一个自定义的帮助类去继承它。

2. SQLiteOpenHelper 中有两个抽象方法，分别是 `onCreate()` 和 `onUpgrade()`，必须在自己的帮助类里面重写这两个方法，分别实现创建和升级数据库的逻辑。

3. 另外，SQLiteOpenHelper 还有两个非常重要的实例方法：

   - `getReadableDatabase()` 得到可读的数据库
   - `getWritableDatabase()` 得到可写的数据库

4. 创建数据库

```java
public class StarbuzzDatabaseHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "starbuzz.db";
    private static final int DB_VERSION = 2;

    public StarbuzzDatabaseHelper(Context context){
        // 调用SQLiteOpenHelper超类的构造方法，并传入数据库名和版本号
        super(context,DB_NAME,null,DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db){
        db.execSQL("CREATE TABLE DRINK(" +
                    "_id INTEGER PRIMARY KEY AUTOINCREMENT,"+
                    "NAME TEXT,"+
                    "DESCRIPTION TEXT,"+    // 字段之间逗号不要忘记
                    "IMAGE_RESOURCE_ID INTEGER);");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

5. 添加数据：`ContentValues values = new ContentValues();`

   - 使用 `ContentValues` 对象
   - 每个数据行都需要创建一个新的 `ContentValues` 对象
   - `values.put("NAME", data);` 使用键值对，“NAME”是列名，data 是数据
   - *null 是指当数据调用 insertDrink 却没有给键值对中 data 添加数据时，*

```java
public static void insertDrink(SQLiteDatabase db,String name,String description,int
    imageResourceId){
    ContentValues values = new ContentValues(); // 封装数据集
```

```
    values.put("NAME",name);
    values.put("DESCRIPTION",description);
    values.put("IMAGE_RESOURCE_ID",imageResourceId);
    db.insert("DRINK",null,values);
}
```

```
insertDrink(db,"拿铁","意大利浓缩咖啡与牛奶的混合",R.drawable.latte);
insertDrink(db,"卡布奇诺","同量的意大利特浓咖啡和蒸汽泡沫牛奶混合",R.drawable.cappuccino
    );
insertDrink(db,"美式咖啡","意式浓缩中加入大量的水",R.drawable.filter);
```

6. 升级数据库：版本号高于当前数据库的版本，则调用 SQLite 帮助器的 `onUpgrade()` 方法。

2.4.2 游标

- 查询方法 query()

```
public Cursor query (
    String table,            // 查询的表名
    String[] columns,        // 要查询的列名数组。如果想查询所有列，可以传 null
    String selection,        // 查询条件（WHERE 子句）。使用 ? 作为占位符来防止 SQL 注
    入。
    String[] selectionArgs,  // 查询条件的参数值数组。每个值将按顺序替换 selection 中的
    ? 占位符。
    String groupBy,          // 分组条件（GROUP BY 子句）如果不需要分组可以传 null
    String having,           // 过滤条件（HAVING 子句），通常与 groupBy 一起使用。如果不
    需要过滤，可以传 null。
    String orderBy           // 排序条件（ORDER BY 子句）。如果不需要排序，可以传 null
)
```

- 获取游标值:cursor 的 getXXX() 方法：getString()、getInt();
- 最后关闭游标及数据库

```
SQLiteOpenHelper helper = new StarbuzzDatabaseHelper(this);
try {
    SQLiteDatabase db = helper.getReadableDatabase();
    Cursor cursor = db.query("DRINK",new String[]{"NAME","DESCRIPTION","
    IMAGE_RESOURCE_ID"},"_id=?",new String[]{Integer.toString(1)},null,null,null);
    if(cursor.moveToFirst()){   // 导航至游标第一条记录
        // 取出记录的 1-3 列
        String nameText = cursor.getString(0);       // 获取第0列，即"NAME"列的值，类型
    为TEXT，用getString
        String descriptionText = cursor.getString(1);// 获取第1列，即"DESCRIPTION"列的
    值，类型为TEXT，用getString
        int photoId = cursor.getInt(2);              // 获取第2列，即"IMAGE_RESOURCE_ID"
    列的值，类型为INTEGER，用getInt

        // 显示 name
        TextView name = findViewById(R.id.tv_name);
        name.setText(nameText);
        // 显示description
        TextView description = findViewById(R.id.tv_description);
        description.setText(descriptionText);
        // 显示图片
        ImageView photo = findViewById(R.id.image_photo);
        photo.setImageResource(photoId);
        photo.setContentDescription(nameText);
```

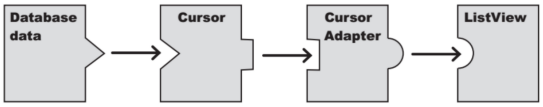
```
21
22      }
23      cursor.close();
24      db.close();
25  }catch (SQLException e){
26      Toast.makeText(this, "数据库不可用", Toast.LENGTH_SHORT).show();
27  }
```

- 使用 `moveToFirst` 的原因

  1. 移动游标：`cursor.moveToFirst()` 将游标移动到查询结果的第一条记录。
  2. 检查查询结果：通过 `if (cursor.moveToFirst())` 可以确保查询结果不为空。
  3. 读取数据：只有在游标成功移动到第一条记录后，才能安全地读取数据。

2.4.3 游标适配器

1. 定义：使用游标适配器 cursor adapter 适配 ListView，将数据显示到列表中。

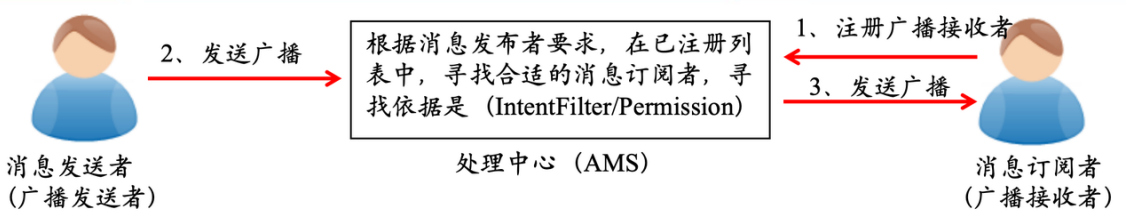


2. ListView 向 SimpleCursorAdapter 请求数据（CursorAdapter 的子类）。
3. 适配器向游标（源自数据库）请求数据。
4. 适配器向 ListView 返回数据。
5. OnItemClickListener 监听列表项的单击事件 AdapterView。

```
1  /* 在该 Activity 得到一个数据库的引用，先实例化 helper */
2  SQLiteOpenHelper helper = new StarbuzzDatabaseHelper(this);
3  try {
4      /* 再获取数据库的引用,并妥善处理SQLiteException异常 */
5      SQLiteDatabase db = helper.getReadableDatabase();
6
7      /* 创建一个游标从数据库读取饮料数据，检索出DRINK表中所有记录的 _id 和 NAME。 */
8      Cursor cursor = db.query("DRINK",new String[]{"_id","NAME"},null,null,null,null,null
       );
9
10     /* 使用游标适配器cursor adapter的子类适配器（SimpleCursorAdapter）适配ListView，将数
       据显示到列表中 */
11     SimpleCursorAdapter listAdapter = new SimpleCursorAdapter(this,
12             // Android 内置布局：每行显示一个文本框
13              android.R.layout.simple_list_item_1,
14             // 数据源
15             cursor,
16             // 在 ListView 的文本框
17             new String[]{"NAME"},
18             // simple_list_item_1 的布局
19             new int[]{android.R.id.text1},
20             0);
21     listDrinks.setAdapter(listAdapter);
22     AdapterView.OnItemClickListener itemClickListener = new AdapterView.
       OnItemClickListener() {
23         // 某一项被点击时，会调用此方法。
```

```
24          @Override
25          public void onItemClick(AdapterView<?> parent, View view, int position, long id)
        {
26              Intent intent = new Intent(DrinkCategoryActivity.this,DrinkActivity.class);
27              // 将被点击项的行 ID 作为附加数据传递
28              intent.putExtra(DrinkActivity.EXTRA_DRINK_ID,(int)id);
29              startActivity(intent);
30          }
31      };
32      listDrinks.setOnItemClickListener(itemClickListener);
33      }catch (SQLException e){
34      Toast.makeText(this, "数据库不可用", Toast.LENGTH_SHORT).show();
35  }
```

2.5 广播



2.5.1 动态注册广播接收者

1. 动态注册方法：
   - 创建一个类继承 BroadcastReceiver。
   - 并重写 onReceive() 方法来实现。
2. 创建完广播接收者之后，还需要对广播接收者进行注册才可以接收广播。
3. 在 MainActivity 销毁时，取消注册 MyReceiver，以确保不会在活动销毁后继续接收广播，避免内存泄漏。

```
// 发送方
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    myReceiver = new MyReceiver();
    String action = "android.net.conn.CONNECTIVITY_CHANGE";

    // 实例化过滤器并设置要过滤的广播
    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction(action);
    // 这表示 MyReceiver 对网络连接变化的广播感兴趣。

    // 注册广播
    registerReceiver(myReceiver,intentFilter);
}

protected void onDestroy() {
    super.onDestroy();
    // 当Activity销毁时，取消注册
    unregisterReceiver(myReceiver);
}
```

```
// 接收方
public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // 返回当前活动的网络连接信息。
        ConnectivityManager connectivityManager = (ConnectivityManager) context.
    getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();

        if(networkInfo != null && networkInfo.isAvailable()){
            Toast.makeText(context,"网络连接成功",Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(context,"网络连接失败",Toast.LENGTH_LONG).show();
        }
    }
}
```

- MyReceiver 继承自 BroadcastReceiver，用于接收和处理广播消息。
- onReceive 方法是 BroadcastReceiver 类的抽象方法，当广播接收器接收到广播时，系统会调用此方法。

IntentFilter

1. 作用

   (a) `IntentFilter` 用于定义广播接收器、活动或服务感兴趣的 `Intent` 类型。当系统或其他应用程序发送一个 `Intent` 时，只有符合 `IntentFilter` 的组件才会接收并处理该 `Intent`。

2. `IntentFilter` 的常用方法

   (a) 添加动作（Action）：

   - `addAction(String action)`：添加一个动作。一个 `IntentFilter` 可以包含多个动作。

   (b) 设置优先级（Priority）

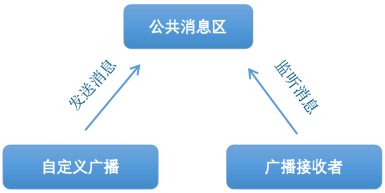
   - `setPriority(int priority)` 方法用于设置广播接收器的优先级，以控制接收器接收广播的顺序。

网络状态检查方法

- 直接检查网络状态：适用于应用启动时或用户触发操作时检查网络状态。优点是简单直接，但无法实时响应网络状态变化。
- 使用广播接收器：适用于需要实时监控网络状态变化的场景。优点是可以立即响应网络状态变化，适合需要后台运行时监控网络状态的应用。

2.5.2 自定义广播

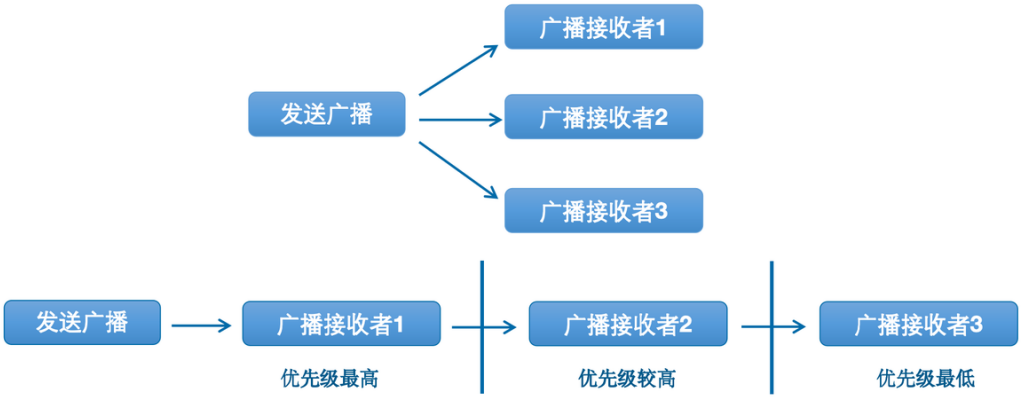
当自定义广播发送消息时，会储存到公共消息区中，而公共消息区中如果存在对应的广播接收者，就会及时的接收这条信息。




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

    registerReceiver(); // 注册广播接收者
    findViewById(R.id.btn_send).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent();
            intent.setAction("HELP");
            // 发送一个名为“HELP”的有序广播
            sendOrderedBroadcast(intent, null);
        }
    });
```

2.5.3 广播类型

1. 无序广播：完全异步执行，发送广播时所有监听这个广播的广播接收者都会接收到此消息，但接收的顺序不确定。



2. 有序广播：按照接收者的优先级接收，只有一个广播接收者能接收消息，在此广播接收者中逻辑执行完毕后，才会继续传递。

   - 广播接收器的优先级范围为 -1000 到 1000。默认优先级为 0。
   - 两个广播接收者的优先级相同，则先注册的广播接收者优先级高。

```
// 动态注册 MyReceiverOne 广播
private void registerReceiver() {
    one = new MyReceiverOne();
    IntentFilter filter1 = new IntentFilter();
    filter1.setPriority(1000);  // 设置广播优先级
    filter1.addAction("HELP");
    registerReceiver(one, filter1);
}
```

   - 广播拦截

```
public class MyReceiverTwo extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i("MyReceiverTwo","自定义广播接收者 Two 接收到广播事件");
        abortBroadcast();
        Log.i("MyReceiverTwo","我是广播接收者 Two 广播被我拦截");
    }
}
```

2.6 JSON 数据解析

2.6.1 数据格式

- 对象结构：以“{”开始，以“}”结束，间部分由 0 个或多个以“,”分隔的 `key:value` 对构成
- 数组结构：以“[”开始，以“]”结束。中间部分由 0 个或多个以“,”分隔的值

```
{
  key1:value1,
  key2:value2,
  ……
}
```

```
[
  value1,
  value2,
  ……
]
```

```
// 对象结构
{"city":"Beijing","street":"Xisanqi","postcode":100096}
// 数组结构
["abc",12345,false,null]
```

```
// 数组结构JSON包含两个对象结构的数据
[
  {
    "name":"LiLi",
    "city":"Beijing"
  },
  {
    "name":"LiLei",
    "city":"Shanghai"
  }
]
```

```
// 对象包含对象
 {
  "name": "zhangsan",
  "address":{
    "city":"Beijing",
    "street":"Xisanqi",
    "postcode":100096
  }
 }
```

```
[
  {
    "name":"LiLi",
    "city":"Beijing"
    "hobby":["篮球","乒乓球","听音乐"]
  },
  {
    "name":"LiLei",
    "city":"Shanghai"
    "hobby":["看电影","羽毛球","游泳"]
  }
]
```

2.6.2 JSON 解析

- 要解析的 JSON 数据如下

```
{ "name":"zhangsan", "age":27, "married":true }      // json1 一个json对象
[{"name": "lisi","age": 25},{"name": "Jason","age": 20}] //json2 一个json数组
```

- 使用 JSONObject 解析 JSON 对象:

```
    JSONObject jsonObj = new JSONObject(json1);
    String name = jsonObj.optString("name");
    int age = jsonObj.optInt("age");
    boolean married = jsonObj.optBoolean("married");
```

- 使用 JSONArray 解析 JSON 数组:

```
JSONArray jsonArray = new JSONArray(json2);
for(int i = 0; i < jsonArray.length(); i++) {
  JSONObject jsonObj = jsonArray.optJSONObject(i);
  String name = jsonObj.optString("name");
  int age = jsonObj.optInt("age");
}
```

- 使用 GSON 解析 JSON 对象:

```
    Gson gson = new Gson();
    Person person1 = gson.fromJson(json1, Person1.class);
```

- 使用 GSON 解析 JSON 数组:

```
1 Gson gson = new Gson();
2 Type listType = new TypeToken<List<Person2>>(){}.getType();
3 List<Person2> person2 = gson.fromJson(json2, listType);
```

  - 定义 Person2 类来存储 name 和 age。
  - 使用 Gson 库将 JSON 字符串 json2 转换为 List<Person2>。
  - 第 2 行代码的作用是创建一个 Type 对象，用于描述 List<Person2> 类型
  - gson.fromJson(json2, listType) 使用 Gson 将 json2 字符串解析为 List<Person2> 类型的对象。
  - 解析后的结果是一个 List<Person2>，每个 Person2 对象对应于 JSON 数组中的一个 JSON 对象。

3 代码

3.1 常见布局代码

3.1.1 左图图书信息代码

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="10dp"

        android:src="@drawable/shu" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/imageView"
        android:layout_toRightOf="@id/imageView"
        android:text="《Android移动应用基础教程》" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textView"
        android:layout_alignLeft="@id/textView"
        android:paddingLeft="5dp"
        android:text="这是一本Android入门书籍....."
        android:textColor="#ff0000" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/textView"

        android:layout_alignParentRight="true"
        android:layout_marginRight="15dp"
        android:text="ISBN" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="5dp"
        android:layout_below="@id/imageView"
        android:layout_marginTop="10dp"
```

```
        android:background="@color/black" />

</RelativeLayout>
```

3.1.2 右图登录界面代码

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="10dp"

        android:src="@drawable/shu" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/imageView"
        android:layout_toRightOf="@id/imageView"
        android:text="《Android移动应用基础教程》" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textView"
        android:layout_alignLeft="@id/textView"
        android:paddingLeft="5dp"
        android:text="这是一本Android入门书籍....."
        android:textColor="#ff0000" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/textView"

        android:layout_alignParentRight="true"
        android:layout_marginRight="15dp"
        android:text="ISBN" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="5dp"
        android:layout_below="@id/imageView"
        android:layout_marginTop="10dp"
```

```
        android:background="@color/black" />

</RelativeLayout>
```

3.2 生命周期核心代码

```
public class FirstActivity extends AppCompatActivity {

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

        Log.d("ActivityLife","onCreat");
        findViewById(R.id.btn_1to2).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(FirstActivity.this,SecondActivity.class);
                startActivity(intent);
            }
        });
        findViewById(R.id.btn_1to3).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(FirstActivity.this,ThirdActivity.class);
                startActivity(intent);
            }
        });
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("ActivityLife","onDestroy");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d("ActivityLife","onStart");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d("ActivityLife","onStop");
    }

    @Override
    protected void onRestart() {
        Log.d("ActivityLife","onRestart");
        super.onRestart();
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d("ActivityLife","onPause");
    }
```

```
53
54      @Override
55      protected void onResume() {
56          super.onResume();
57          Log.d("ActivityLife","onResume");
58      }
59  }
```

3.3 数据传输方式代码

```
1   public class ReceiveMessageAtivity extends AppCompatActivity {
2
3       @Override
4       protected void onCreate(Bundle savedInstanceState) {
5           super.onCreate(savedInstanceState);
6           setContentView(R.layout.activity_receive_message_ativity);
7           TextView textView = findViewById(R.id.tv_receive);
8
9           Intent intent = getIntent();
10          /*   使用putExtra()接受单个数据  */
11          String receiveMessage = intent.getStringExtra("message");
12          textView.setText(receiveMessage);
13
14          /*   使用Bundle接受多个数据     */
15  //        Bundle bundle = intent.getExtras(); // 获取 bundle 内容
16  //
17  //        String Name = bundle.getString("name");
18  //        String receiveMessage = bundle.getString("message");
19  //
20  //        textView.setText(Name+" "+receiveMessage+"");
21      }
22  }
```

```
1   public class SendMessageActivity extends AppCompatActivity {
2
3       EditText etMessage;
4
5       @Override
6       protected void onCreate(Bundle savedInstanceState) {
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.activity_send_message);
9
10          etMessage = findViewById(R.id.et_message);// 获取输入内容
11          findViewById(R.id.btn_send).setOnClickListener(new View.OnClickListener() {
12              @Override
13              public void onClick(View v) {
14                  Intent intent = new Intent(SendMessageActivity.this, ReceiveMessageAtivity.
        class);
15                  String strMessage = etMessage.getText().toString();
16
17                  /* 使用putExtra()传递单个数据   */
18                  intent.putExtra("message", strMessage);
19                  startActivity(intent);
20
21                  /* 使用Bundle类传递多个数据     */
22  //                Bundle bundle = new Bundle();   // 创建Bundle对象。
23  //                bundle.putString("name", "林星辰"); // 将用户名封装到Bundle对象中
24  //                bundle.putString("message",strMessage);
```

```
//                intent.putExtras(bundle);     // 将Bundle对象封装到Intent对象中
//                startActivity(intent);
            }
        });
    }
}
```

3.4 StarbuzzDatabaseHelper 代码

```
public class StarbuzzDatabaseHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "starbuzz.db";
    private static final int DB_VERSION = 2;

    public StarbuzzDatabaseHelper(Context context){
        // 调用SQLiteOpenHelper超类的构造方法，并传入数据库名和版本号
        super(context,DB_NAME,null,DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db){updateDatabase(db,0,DB_VERSION);}

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        updateDatabase(db, oldVersion, newVersion);
    }

    public static void insertDrink(SQLiteDatabase db,String name,String description,int
    imageResourceId){
        ContentValues values = new ContentValues(); // 封装数据集
        values.put("NAME",name);
        values.put("DESCRIPTION",description);
        values.put("IMAGE_RESOURCE_ID",imageResourceId);
        db.insert("DRINK",null,values);
    }

    public void updateDatabase(SQLiteDatabase db,int oldVersion,int newVersion){
        if(oldVersion < 1){
            db.execSQL("CREATE TABLE DRINK(" +
                    "_id INTEGER PRIMARY KEY AUTOINCREMENT,"+
                    "NAME TEXT,"+
                    "DESCRIPTION TEXT,"+    // 字段之间逗号不要忘记
                    "IMAGE_RESOURCE_ID INTEGER);");
            insertDrink(db,"拿铁","铁咖啡(Coffee Latte)是花式咖啡的一种,是咖啡与牛奶交融的极至
    之作。",R.drawable.latte);
            insertDrink(db,"卡布奇诺","卡布奇诺的口感丰富，结合了浓缩咖啡的苦涩和深沉、蒸汽牛
    奶的甜味，以及顶部细腻奶泡的轻盈口感，形成了一种平衡的味道。",R.drawable.cappuccino);
            insertDrink(db,"手冲咖啡","手冲咖啡，一种味觉的旅行。从淡淡的草本香到果香，最后是
    浓郁的巧克力味，它像一场精心编排的交响乐，让你的味蕾经历一次难忘的旅程。",R.drawable.
    filter);

        }

        if (oldVersion < 2){
            db.execSQL("ALTER TABLE DRINK ADD COLUMN FAVERT");
        }
    }
}
```

3.5 有序广播代码

```
public class MainActivity extends AppCompatActivity {

    MyReceiverOne one;
    MyReceiverTwo two;
    MyReceiverThree three;

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

        registerReceiver(); // 注册广播接收者
        findViewById(R.id.btn_send).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.setAction("HELP");
                // 发送一个名为“HELP”的广播
                sendOrderedBroadcast(intent, null);
            }
        });
    }

    private void registerReceiver() {

        // 动态注册 MyReceiverTwo 广播
        two = new MyReceiverTwo();
        IntentFilter filter2 = new IntentFilter();
        filter2.setPriority(900);
        filter2.addAction("HELP");
        registerReceiver(two, filter2);

        // 动态注册 MyReceiverOne 广播
        one = new MyReceiverOne();
        IntentFilter filter1 = new IntentFilter();
        filter1.setPriority(1000);  // 设置广播优先级
        filter1.addAction("HELP");
        registerReceiver(one, filter1);


        // 动态注册 MyReceiverThree 广播
        three = new MyReceiverThree();
        IntentFilter filter3 = new IntentFilter();
        filter3.setPriority(600);
        filter3.addAction("HELP");
        registerReceiver(three, filter3);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(one);
        unregisterReceiver(two);
        unregisterReceiver(three);
    }
}
```

4 习题

【例 1】 在 Android 中，短信、联系人管理和浏览器等属于 Android 系统架构中的（____）。

a. Linux 内核层

b. 核心类库层

c. 应用程序层

d. 应用程序框架层

【例 2】 Android 项目中，用于存放程序资源的文件夹是（____）。

a. src/main/java

b. src/main/res

c. src/AndroidManifest.xml

d. src/androidTest

【例 3】 Android 中，线性布局是 ViewGroup 的子类，用 _____ 类表示。在布局文件中，指定线性布局的方向属性用 android:_____ 表示，可以取 horizontal 和 vertical 两个值。

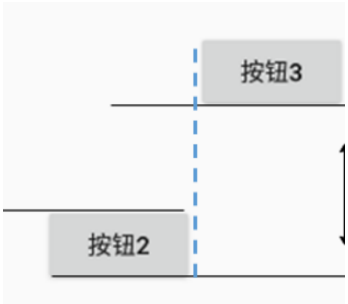
【例 4】 如右图所示，在相对布局中，将按钮 3 设置到按钮 2 的右侧，使用（_____ ）属性。

(a) android:layout_toRightOf=”@+id/btn_two”

(b) android:layout_alignParentRight=”true”

(c) android:layout_toRightOf =”true”

(d) android:layout_alignParentRight =”@+id/btn_two”



【例 5】 Button 控件被点击时，可以通过（_____ ）接口监听此事件的发生。

(a) View.OnLongClickListener

(b) View.OnClickListener

(c) View.onFocusChangeListener

(d) View.OnTouchListener

【例 6】 把 RadioButton 控件需要放在 _____ 里，才不会出现多个 RadioButton 同时被选中的情况。RadioButton“选中”和“未选中”两种状态通过 android:_____ 属性指定的。

【例 7】 监听单选列表对话框的监听器接口是（_____ ）

(a) RadioGroup.OnCheckedChangeListener

(b) View.OnClickListener

(c) DialogInterface.OnClickListener

(d) CompoundButton.OnCheckedChangeListener

【例 8】 创建选项菜单，需要在所在的类中重写（_____ ）方法，在方法中加载选项菜单。

(a) getMenuInflater()

(b) onContextItemSelected()

(c) onOptionsItemSelected()

(d) onCreateOptionsMenu()

【例 9】 ArrayAdapter 一般用来绑定数组数据源，它直接继承自 _____ 。将适配器绑定到 ListView，使用的方法是 _____ 。

【例 10】 需要自己定义适配器类在 ListView 中展示数据时，应该继承 _____ 类，并重写这个类中的四个方法，系统在绘制每行 Item 时会自动调用 _____ 方法。

【例 11】 Activity 生命周期中系统调用 _____ 方法后，Activity 创建了但并不可见，一般在这个方法中完成 Activity 的初始化。调用 _____ 方法后，Activity 可见，且获得焦点，处于运行状态。当用户按 Home 键时，系统会调用 _____ 方法，Activity 变为不可见。

【例 12】 在 MainAcitivity 中要跳转到 SecondActivity，使用以下代码可以实现开启 SecondActivity。

```
Intent intent = new Intent(MainActivity.this, _______ );
startActivity(_______ );
```

【例 13】 SharedPreferences 是 Android 平台上一个轻量级的存储类，用于程序中一些少量数据持久化存储。通过 _____ 方法获取实例对象。该对象本身只能获取数据。需调用它的 _____ 方法获取可编辑的 Editor 对象进行数据的存储和修改。

【例 14】 Android 为了更加方便管理数据库，提供了一个帮助类，类名为 _____ ，这个类可以对数据库进行创建和升级。数据库帮助类中有两个抽象方法，分别实现创建和升级数据库的逻辑。其中 _____ 方法当数据库第一次被建立的时候被自动调用，一般进行创建表，初始化表数据等工作；_____ 方法当数据库需要被更新的时候被自动调用，一般进行删除旧表，创建新表等改变数据库结构等工作。

【例 15】 请写出使用 `JSONArray` 类解析 JSON 数据的主要逻辑代码。JSON 数据如下所示:

```
[{"name": "LiLi", "score":"95"}, {"name":"LiLei", "score":"99"},
{"name":" 王小明","score":"100"},{"name":"LiLei","score":"89"}]
```

5 答案

【答案 1】 C

【答案 2】 B

【答案 3】 LinearLayout、orientation

【答案 4】 A

【答案 5】 B

【答案 6】 RadioGroup、checked

【答案 7】 C

【答案 8】 D

【答案 9】 BaseAdapter、setAdapter()

【答案 10】 BaseAdapter、getView()

【答案 11】 onCreate()、onResume()、onStop()

【答案 12】 SecondActivity.class、intent

【答案 13】 getSharedPreferences()、edit()

【答案 14】 SQLiteOpenHelper、onCreate()、onUpgrade()

【答案 15】 答案如下

```
String json2 = "[{\"name\": \"LiLi\",\"score\": 95},{\"name\": \"LiLei\",\"score\":
  99},{\"name\": \"王小明\",\"score\": 100},{\"name\": \"LiLei\",\"score\": 89}]";

// 创建 JSON 数组
  JSONArray jsonArray = new JSONArray(json2);
  for (int i = 0; i < jsonArray.length(); i++) {
      // 获取每个 JSON 对象
      JSONObject jsonObj = jsonArray.getJSONObject(i);
      // 获取 name 和 score 字段的值
      String name = jsonObj.optString("name");
      String score = jsonObj.optString("score");
   }
```