第一章——安卓简介

1.1 手机操作系统

手机操作系统主要包括：Android, IOS, Windows Phone 8, 黑莓 和 Linux

1.2 安卓的起源

记住两个虚拟机: Dalvik JIT以及现在主流的ART

1.3 Android特征

主要考填空题
安卓系统提供了访问硬件的API库函数
在内存和进程管理方面，安卓具有自己的运行时和虚拟机
安卓为了保证高优先级进程运行和正在与用户交互进程的访问速度，允许停止或终止正在运行的低优先级进程，以释放被占用的系统资源。Android进程的优先级并不是固定的，而是根据进程前台是否与用户交互而不断变化的
Android为组件定义了生命周期，并统一进行管理与控制
在界面设计上，Android提供了丰富的界面控件供使用者调用：
加快了用户开发速度，保证Android平台上的程序界面的一致性
Android将界面设计与程序逻辑分离，使用XML文件对界面布局进行描述，有利于界面的修改与维护
Android提供轻量级的进程间通讯机制Intent，使跨进程组件通信和发送系统级广播成为可能
Android提供了Service作为无用户界面、长时间后台运行的组件——Service无需用户干涉，可以长时间、稳定地运行，可为应用程序提供特定的后台功能。
Android支持高效、快速的数据存储方式：SharedPreferences、文件存储、轻量级关系数据库SQLite，为了便于进程共享数据，Android提供了通用的共享数据接口ContentProvider，可以无需了解数据源、路径的情况下，对共享数据进行查询、添加、删除和更新等操作。
Android支持位置服务和地图应用，并可以将Google地图嵌入到Android应用程序中。
Android支持Widget插件
Android NDK 支持使用本地代码开发应用程序的部分核心模块，提高了程序运行效率，有助于增加Android开发的灵活性

1.4 Android体系结构

体系分为四层（十分重要）：Linux内核、中间件层、应用程序框架层、应用程序层
Android运行时由核心库和ART虚拟机构成。

第二章——Android开发环境

2.3 Android目录结构

add-one存放Google提供的地图开发包
docs目录下是Android SDK帮助文档
extras\google目录下存放了Android手机的USB驱动程序
platforms用来存放SDK和AVD管理器下载的各种版本的SDK
platforms-tools保存了与平台调试的工具，如adb、appt和dx等
samples存放示例代码和程序
temp存放临时文件
tools存放了通用的Android开发调试工具和Android手机模拟器
SDK Manager.exe和ACD Manager.exe是SDK和AVD的管理器，SDK Readme.txt是Android SDK说明文档

第三章——Android应用程序（十分重要）

3.2 Android程序结构

5个子目录

src源代码目录
gen保存ADT自动生成的Java文件
assets存放原始格式的文件，例如音频、视频等二进制格式文件
bin保存了编译过程中产生的文件，以及最终产生的apk文件
res资源目录，包含了所有图像、颜色、风格、主题、界面布局和字符串等资源

2个库函数

android.jarAndroid程序所能引用的库函数文件，Android所支持的API都在这个文件中
android-support-v4.jar安卓兼容性包

3个工程文件

proguard.cfg供ProGuard工具进行代码优化和代码混淆使用的配置文件
project.properties记录了Android工程的相关设置
AndroidManifest.xml是XML格式的Android程序声明文件，包含了Android系统运行Android程序前所必须掌握的重要信息

2种资源引用方法

一种方法是在代码中引用资源，需要在代码中使用资源ID
一种方法是在资源中引用资源，一般的引用格式为@[package:]type:name

第四章——生命周期

概念：Android生命周期是从程序启动到程序终止的全过程。

进程优先级（十分重要）

高优先级：前台进程
中优先级：可见进程、服务进程
低优先级：后台进程、空进程

安卓的四大组件（必考）

安卓系统的4个重要的组件：Activity、Service、Broadcast receiver、Content provider
Activity是Android程序的呈现层，显示可视化的用户界面，并接收与用户交互所产生的界面事件。Android应用程序可以包含一个或多个Activity，一般需要指定一个程序启动时显示的Activity
Service一般用于没有用户界面，但需要长时间在后台运行的应用。公开Service的程序接口，供其他进程调用
Broadcast receiver是用来接收并相应广播消息的组件，不包含任何用户界面
Content provider是Android系统提供的一种标准的数据共享数据的机制，其他程序通过ContentProvider访问程序的私有数据

Android Activity的四种状态

变化关系图在书P68图4.2，要了解他们之间的转换关系
活动状态：完全能被用户看到，Activity在用户界面中处于最上层
暂停状态：Activity在界面上被部分遮挡，不再处于最上层并无法与用户进行交互
停止状态：Activity在界面上完全不能被用户看到，被全部遮挡
非活动状态：不在以上三个状态之内的状态是非活动状态

Activity事件回调函数的调用顺序（最重要的地方，必须掌握）

图在书P71页图4.4——**（必考）** 7 + 2

7个生命周期事件回调函数

onCreate() 通常用来进行Activity的初始化
onStart() 当Activity显示在屏幕上后，该函数被调用
onRestart() 当Activity从停止状态进入活动状态后，该函数被调用
onResume() 当Activity可以接收用户输入时，该函数被调用。此时Activity位于Activity栈的栈顶
onPause() 当Activity处于暂停状态时，该函数被调用。其主要是用来保存持久数据、关闭动画、释放CPU资源等
onStop() 当Activity不对用户可见后，该函数被调用，Activity进入停止状态
onDestroy() 在Activity被终止之前，该函数被调用。该函数有两种情况会被调用：1、程序主动调用finish()函数。2、程序被Android系统终结

2个Activity状态保存/恢复的事件回调函数

onSaveInstanceState() 暂停或停止Activity前调用该函数，用以保存Activity的临时状态信息
onRestoreInstanceState() 恢复onSaveInstanceState()保存的Activity状态信息

程序调试

6种LogCat

概念：LogCat是用来获取系统日志信息的工具，可以显示在Eclipse集成开发环境中
级别从上到下依次增高：
[V]:详情(Verbose)信息

[D]:调试(Debug)信息
[I]:通告(Info)信息
[W]:警告(Warn)信息
[E]:错误(Error)信息
[A]:断言(Assert) -> (这是ppt里面家出来的)

第五章——Android用户界面

MVC模型

图在书P87页 图5.1 -> 会出填选或者简答题
MVC模型(Model-View-Controller):
控制器(Controller):处理用户输入
视图(View):显示用户界面和图像
模型(Model):保存数据和代码

5.2 Android界面控件

常见的系统控件有(会出选择题，需要背诵): TextView, EditText, Button, ImageButton, Checkbox, RadioButton, Spinner, ListView, (TabHost)

TextView和EditText——会考代码题

代码位置——课本的P88、P89页，5.2.1章节看完

```
<!-- XML 代码 -->
<!--注意id内部的格式问题，有 @ + id-->
<TextView android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView01">
</TextView>
```

```
<EditText android:id="@+id/EditText01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="EditText01">
</EditText>
```

这里一行代码注意和之前的TextView01相呼应

```
// Java 代码
TextView textView = (TextView)findViewById(R.id.TextView01); // 这里TextView01
EditView editView = (EditView) findViewById(R.id.EditView01); // 这里EditView01
textView.setText("用户名:");
editView.setText("Rajan");
```

Button和ImageButton

按钮响应点击事件：添加点击事件的监听器
button对象通过调用setOnClickListener()函数注册一个点击(Click)事件的监听器View.onClickListener()
代码在书P90、91页

```
// Java 代码
final TextView textView = (TextView) findViewById(R.id.textView01);
button.setOnClickListener(new View.onClickListener(){ // 点击事件，最重要的一行
    public void onClick(View view) {
        textView.setText("Button按钮");
    }
});
```

Spinner和ListView两者二选一考

Spinner概念：从多个选项中选择一个选项的控件
Spinner重点考点：代码题，代码在书P94页

```
Spinner spinner = (Spinner) findViewById(R.id.Spinner01);
List<String> list = new ArrayList<String>(); // String是泛型，意味着之后加入的元素都是String
list.add("Spinner 子项1"); // 添加子项的代码
list.add("Spinner 子项2");
list.add("Spinner 子项3");
ArrayAdapter<String> adapter = new ArrayAdapter<String>
   (this, android.R.layout.simple_spinner_item, list);
     // 注意这里的list要和前面对应，重点是this和list
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter); // 这里挺重要的
```

ListView概念: 用于垂直显示的列表控件，如果显示内容过多，则会出现垂直滚动条。ListView能够通过适配器将数据和显示控件绑定，在有限的屏幕上提供大量内容供用户选择；而且支持点击事件，可以用少量的代码实现复杂的选择功能。

ListView重点考点：代码题，代码在书P96页

由于代码和spinner高度相似，详见书上代码即可，这里懒得打了

5.3 界面布局

概念：界面布局(Layout)是用户界面结构的描述，定义了界面中所有的元素，结构和互相关系
两种声明方法：第一种是使用XML文件描述界面布局，另一种是在程序运行时动态添加或者修改界面布局
一般情况下，会选择在XML文件中描述用户页面中的基本元素，而在代码中动态修改需要更新状态的界面元素
使用XML文件声明界面布局，能够更好地将程序的表现层和控制层分离，在修改界面时将不再需要更改程序的源代码
常用的6种界面布局(会在填选中出现——背出来):
线性布局、框架布局、表格布局、相对布局、绝对布局、网格布局
可能会出一些代码的填选

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- 这里交代了是线性布局 -->
<!-- 注意代码最下面的vertical，交代了朝向，垂直的， 如果是水平的就是 Horizontal-->
<LinearLayout
    xmlns:android="http://www.w3.org/2001/XMLSchema-hasFacetAndProperty"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
</LinearLayout>
```

5.4 菜单

概念：菜单是应用程序中非常重要的组成部分；菜单在不占用界面空间的前提下，为应用程序提供了统一的功能和设置界面；菜单为程序开发人员提供了易于使用的编程接口
Android支持的三种菜单模式(背出来):选项菜单(Option Menu)、子菜单(Submenu)、快捷菜单(Context Menu)
菜单代码P119可能会出一个选择题——inflater.inflate(R.menu.main_menu, menu);

5.6 界面事件

概念(可能会出填选): 在Android系统中，存在多种界面事件，如点击事件、触摸事件、焦点事件和菜单事件等。在这些事件发生时，Android界面框架调用界面控件的事件处理函数对事件进行处理

第六章——组件与广播消息

Intent

Intent前两段仔细看，可能会出填空题，在书的P145
Intent是一个动作的完整描述，包含了(这里可能会出选择题)产生组件、接收组件和传递的数据信息
显式启动代码

```
Intent intent = new Intent(IntentDemo.this, ActivityToStart.class); // class 表示被启动的
startActivity(intent); // 启动了
```

获取Activity返回值(可能出填空题)

获取Activity返回值，一般分为以下三个步骤：

1.以Sub-Activity的方式启动子Activity；
— 使用的函数是 startActivityForResult(Intent, requestCode) 函数
2.设置子Activity的返回值；
— 使用的函数是 setResult() 函数
3.在父Activity中获取返回值。
— 使用的函数是 onActivityResult(int requestCode, int resultCode, Intent data) 函数

6.2 Intent过滤器

这里可能出填选题——Intent过滤器是一种根据Intent中的动作(action)、 类别(category)、 数据(data)等内容，对合适接收该Intent的组件进行匹配和筛选的机制。可以匹配数据类型、路径和协议，还可以匹配多个匹配项顺序的优先级(Priority)
注册Intent过滤器的组件——Activity、Service、 BroadcastReceiver

P157页会出代码大题，代码比较多这里不打了，细看书

6.3 广播消息

这里也是代码大题，在P158，细看

第七章——后台服务

7.1 Service生命周期

生命周期图在P162页，细看。本次考试就考两个生命周期
onCreate(): Service的生命周期开始，完成Service的初始化工作
onStart(): 启动线程
onDestroy: Service的生命周期结束，释放Service所有占有的资源
Service的两种使用方式：启动方式和绑定方式 —— 可能出简答题
启动方式：调用Context.startService()启动；调用Context.stopService()或Context.stopSelf()停止

绑定方式：通过Context.bindService()建立服务链接，通过Context.unbindService()停止服务链接
注册Service的代码在P164，如下：

```
<service android:name=".RandomService"/>
```

7.2.2 启动线程，在P169页，之后的代码必须全部细看，一直到P178页

第八章——数据存储与访问

概念: Android系统提供多种数据存储方式，包括SharedPreferences、文件存储以及SQLite数据库

8.1 SharedPreferences

SharedPreferences代码在P196，细看

8.2 文件存储

两个函数(可能出填选): openFileOutput() 和 openFileInput()

8.2.3 资源文件

开发人员除了可以在内部和外部存储设备上读写文件以外，还可以访问在/res/raw和res/xml目录中的原始格式文件和XML文件(可能出填选)

8.3 数据库存储(重点)

请细看增删改查代码部分P218

SQLLiteDatabase类的共有函数insert()、delete()、update()和query()封装了执行添加、删除、修改、查询的SQL命令

8.4 数据共享

8.4.1 ContentProvider

概念图建议看看，帮助理解，图片在P222页

8.4.2 创建数据提供者

三个步骤(可能考):

(1)继承 ContentProvider，并重载6个函数，这6个函数分别是:
delete()、getType()、insert()、onCreate()、query()、update() (增删改查再加俩函数)
(2)声明 CONTENT_URI，实现 UriMatcher，用于匹配单条/多条
(3)使用<Provider>标签来注册 ContentProvider

第九章——位置服务与地图应用

位置服务

(可能会出填空题)Android平台支持位置服务的API，在开发过程中主要使用LocationManager和LocationProviders对象。在获取LocationManager后，还需要指定LocationManager的定位方法，然后才能够调用LocationManager.getLastKnowLocation()方法获取当前位置。
目前LocationManager中有两种定位方法: GPS定位和网络定位