**Android考核参考答案**

**1.** 编写和运行java程序需要三步

1)编写源程序，保存为.java为扩展名的源文件；

2)编译，生成.class为扩展名的字节码文件；

3)运行，可看到运行结果。

在JDK中，Java语言的编译命令是javac.exe，Java语言的执行命令是java.exe

**2.** Java运算符有以下几种:

1)算术运算符:+、-、\*、/、%

2)比较运算:>、>=、<、<=、==、!=

3)逻辑运算:!、&&、||

4)位运算符:~、^、&、|、<<、>>、>>>

5)三目运算:？、:

6)赋值运算符:=、+=、-=、\*=、/=、%=

**3.** 类型转换(cast)有以下两种:

1)自动类型转换:两种类型要兼容、目标类型大于源类型；

2)强制类型转换:两种类型要兼容、目标类型小于源类型。

**4.** &和&&都可以作为逻辑运算符”与”使用，但是&&是”短路与”，运算时先判断符号前面的表达式的值，如果能够确定整个表达式的值，则不进行符号后面的表达式的运算。另外，&也可作为位运算符使用。

**5.** pattern对象是一个正则表达式的编译表示。Pattern类没有公共构造方法。要创建一个Pattern对象，你必须首先调用其公共静态编译方法compile()，它返回一个Pattern对象。该方法接受一个正则表达式作为它的参数。

Matcher对象是对输入字符串进行解释和匹配操作的引擎。与Pattern类一样，Matcher也没有公共构造方法。你需要调用Pattern对象的matcher方法来获得一个Matcher对象

**6.** 两个进程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是两个进程都陷入了无限的等待中。

或

线程A和线程B都需要访问资源M和N。 （资源X均可替换为X的对象锁）

②线程A拥有资源M，在等待获取资源N，且在获取资源N之前，线程A不会释放资源M

③线程B拥有资源N，在等待获取资源M，且在获取资源M之前，线程B不会释放资源N

④线程A和B陷入互相等待对方释放资源的无限循环中,这种情况下线程A和B产生了死锁

**7.** URL是统一资源定位符，用于在Internet上指定资源的地址。

协议，主机(域名或IP)，端口号。

**8.** 相同点：

都可以存储多个元素。

都是有序列表。

都使用index来存取其元素。

都支持null值。

不同点：

array（数组）：

长度固定，一旦创建长度就不可变。

不能使用泛型，如果数组中元素类型不匹配，会抛出ArrayStoreException。

使用变量length表示数组长度。

可以存储基本数据类型和对象。

使用赋值操作符往数组中存储元素。

强制在创建时指定其长度。

不能使用Iterator迭代器遍历数组

ArrayList：

它是长度可变的collection类

可以使用泛型来保证类型安全。

提供size()方法来计算其长度。

不能存储基本数据类型，可以存储对象。

使用add()方法往ArrayList中存储元素。

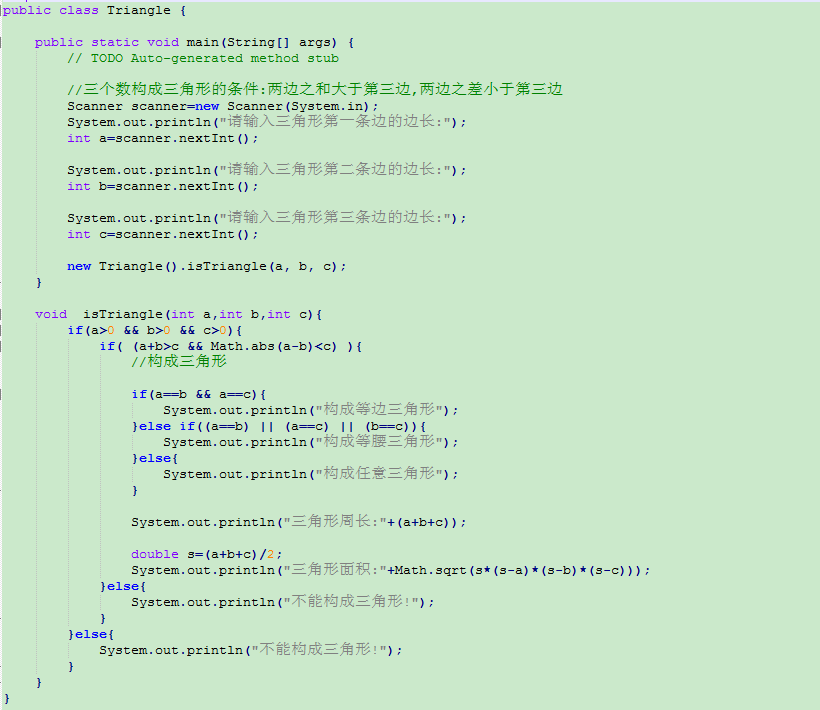
对象创建时不必指明其长度。

遍历ArrayList中可以使用Iterator迭代器

**9.**

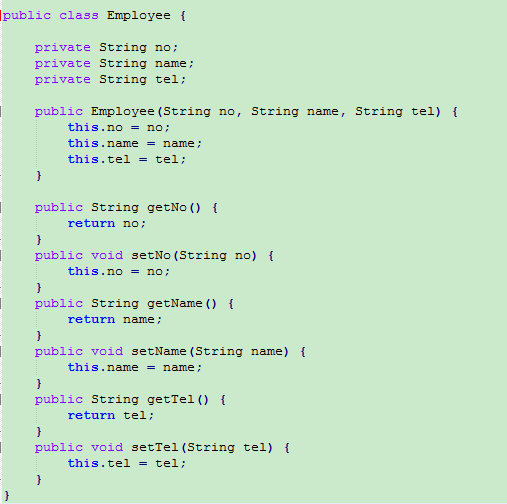
throws: throws语句列出方法可能抛出的异常类型，来通知调用方。除了Error和RuntimeException之外，方法内没有用try-catch捕获的异常都需要在throws语句中声明，表示由其上层方法处理这些问题。在方法定义时使用。

throw: 用于抛出一个明确的异常，在实际编码中常常用于抛出自定义异常。在方法内或者代码块内使用。

10。

**11.**

Employee类：

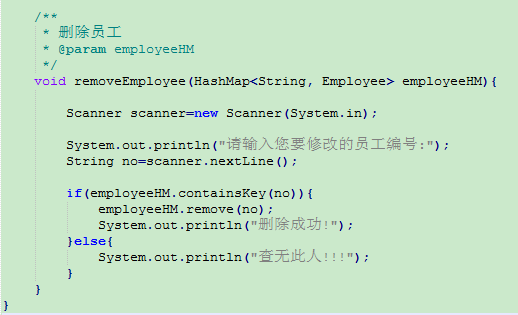


EmployeeManage类：

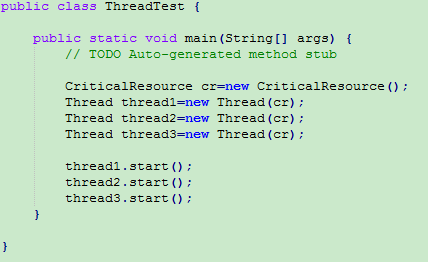








**12.**





**13.**

①线性布局LinearLayout：其中有多个View时，需要指明其排列方向为vertical（垂直的）还是horizontal(水平的)。默认排列方向为horizontal(水平的)。

②相对布局RelativeLayout：其中有多个view时，需要指明多个view之间的位置关系。

③帧布局FrameLayout：从屏幕的左上角开始布局，叠加显示。

④表格布局TableLayout：每一个TableLayout中都有表格行TableRow，里面可以具体定义每一个元素。

⑤绝对布局AbsoluteLayout：用x ,y坐标来指定元素的位置。

**14.**

①补间动画(Tween/View Animation)：

又分为alpha(透明度渐变)、rotate(旋转)、translate(位移)、scale(伸缩)

帧动画(Frame Animation)

③属性动画(Property Animation)

**15.**

①Activity(活动)

Activity是android应用程序的一个界面，可以通过这个界面查看联系人，打电话，玩游戏

一个应用程序通常包含多个Activity

Activity通过布局管理添加各种view组件

通过setContentView(int)方法可以将视图呈现出来

②Service(服务)

Service运行在后台，负责处理一些用户看不到，并且会有持续时间的事情。如下载数据。

Service不生成UI，是不可见的。

所有用户实现的Service必须继承系统的Service类，并且在配置文件中进行注册

③Content Provider(内容提供者)

数据在android当中是程序私有的

Content Provider提供了多个程序间数据交互的机制

Content Provider暴露出来一些标准的API来对数据进行操作

④Broadcast Receiver(广播接收器)

Broadcast Receiver负责对外部事件做出响应

Broadcast Receiver不生成UI，是不可见的

所有用户实现的Broadcast Receiver必须继承系统的Broadcast Receiver类，需要在配置文件中进行注册

应用程序可以通过Context.sendBroadcast()将自己的intent广播出去。其他程序通过自身的Broadcast Receiver截获后进行事件处理

**16.**

①设计布局时，使用相对布局，禁用绝对布局

②在开发中，我们一般用线性布局、相对布局和帧布局，不用绝对布局

设计控件的宽度和高度时，尽量用wrap\_content、match\_paren、weight

使用可以自动拉伸的位图，例如 .9图片

间距用单位用dp，字体大小单位用sp

提供备用位图，即对于同一张图片，提供几种不同分辨率的位图

**17.**

ListView的工作原理：ListView针对每个Item，要求Adapter“返回一个视图”(getView)，也就是说，ListView在开始绘制的时候，系统首先调用getCount()函数，根据他的返回值得到一个ListView的长度，然后根据这个长度，调用getView()一行行的绘制ListView的每一项。如果你的getCount（）返回值是0，列表一行都不会显示；如果返回1，就只显示一行。返回几行则显示几行。Android中有个1叫做Recycler的控件。

如果你有几千几万甚至更多的选项(item)时，其中只有可见的项目存在内存（所谓的优化就是在内存中的优化），其他的在Recycler中。

**18.**

android工程所有资源信息(组件、图片、字符等等)都是由HashMap<Integer,Object>来存储的  
key值就是R.java中的静态变量值  
value就是相对应的各种对象信息(组件、图片、字符等等)  
当工程被载入时，所有资源都被加载到HashMap中去

这样我们就可通过key值定位到相应的组件

该文件相当于项目的字典。项目中所涉及到的用户界面、字符串、图片、声音等资源都会在该文件中创建一个ID编号，这些编号为整形，以16进制自动生成。项目要使用这些资源时，会通过这个类得到资源的引用。

R.java是系统自动生成的，不能手动修改，否则会产生一些无法预料的错误

**19.**

会报错 图片命名不能有中文，且图片命名不能全为数字，若要使用数字，可以在文件名的中间部分或后面使用,eg：wechat2.jpg是可以的。



**20.**

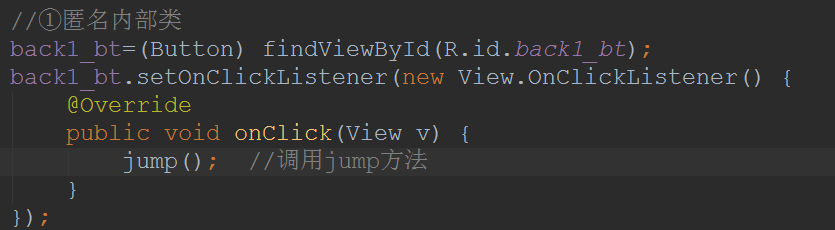
Activity7个生命周期:onCreate、onStart、onResume、onPause、onStop、onDestory、onRestart

开启Activity，默认调用onCreate、onStart、onResume

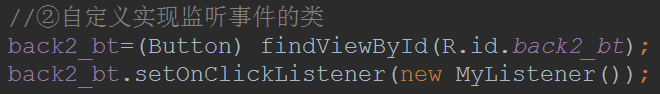
关闭Activity，默认调用onPause、onStop、onDestory

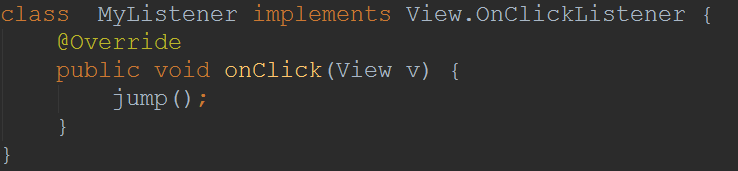
**21.**

方式一:匿名内部类



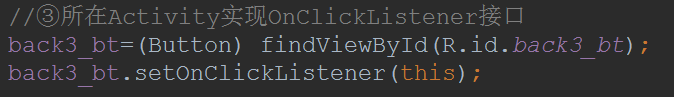
方式二:自定义实现监听事件的类

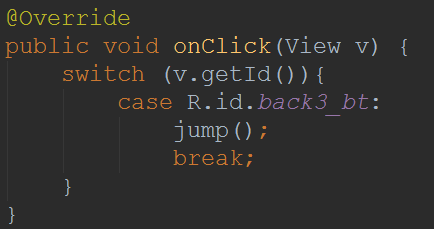




方式三:所在Activity实现onClickListener接口

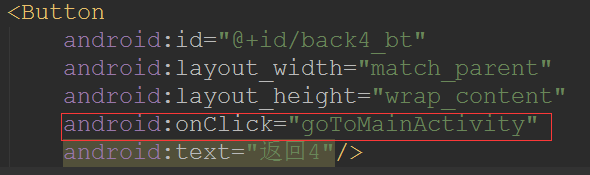


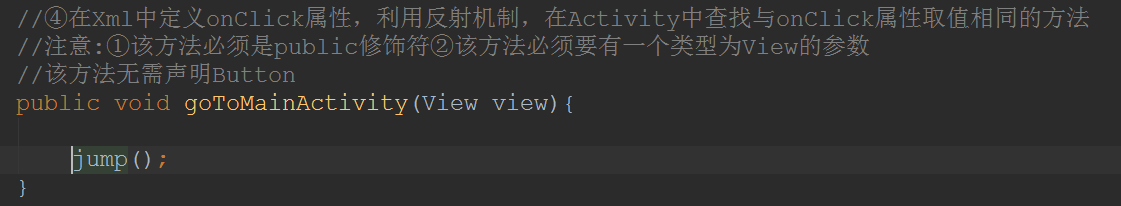




方式四:在Xml中定义onClick属性，利用反射机制，在Activity中查找与onClick属性取值相同的方法

Xml文件中:





**22.**

ANR：Application Not Responding应用程序无响应。

在Android上，如果你的应用程序有一段时间响应不够灵敏，系统会向用户显示一个对话框，这个对话框称作应用程序无响应（ANR：Application Not Responding）对话框。用户可以选择“等待”而让程序继续运行，也可以选择“强制关闭”。

在android中Activity的最长执行时间是5秒，BroadcastReceiver的最长执行时间则是10秒。

在Android里，应用程序的响应性是由Activity Manager和WindowManager系统服务监视的 。当它监测到以下情况中的一个时，Android就会针对特定的应用程序显示ANR：

1.在5秒内没有响应输入的事件（例如，按键按下，屏幕触摸）  
2.BroadcastReceiver在10秒内没有执行完毕

造成以上两点的原因有很多，比如在主线程中做了非常耗时的操作，比如说是下载，io异常等。

如何解决？

①运行在主线程里的任何方法都尽可能少做事情。特别是，Activity应该在它的关键生命周期方法（如onCreate()和onResume()）里尽可能少的去做创建操作。（可以采用重新开启子线程的方式）

②应用程序应该避免在BroadcastReceiver里做耗时的操作或计算

**23.**

分几种情况：  
     1.其他方法前是否加了synchronized关键字，如果没加，则能。  
     2.如果这个方法内部调用了wait，则可以进入其他synchronized方法。  
     3.如果其他个方法都加了synchronized关键字，并且内部没有调用wait，则不能。  
     4.如果其他方法是static，它用的同步锁是当前类的字节码，与非静态的方法不能同步，因为非静态的方法用的是this。即线程1调用非静态的synchronized方法，线程2调用静态的synchronized方法，两者互不影响。（若如果线程1和2，同为非静态或静态，则必须等一个线程执行完，再执行下一个。）

Synchronized（对象）{

需要同步的代码块

}

**24.**

AsyncTask的定义：

public abstract class AsyncTask<Params, Progress, Result>

三种泛型类型分别代表“启动任务执行的输入参数”、“后台任务执行的进度”、“后台计算结果的类型”。在特定场合下，并不是所有类型都被使用，如果没有被使用，可以用[Java](http://lib.csdn.net/base/javase).lang.Void类型代替。

一个异步任务的执行一般包括以下几个步骤：

①.execute(Params... params)，执行一个异步任务，需要我们在代码中调用此方法，触发异步任务的执行。

②.onPreExecute()，在execute(Params... params)被调用后立即执行，一般用来在执行后台任务前对UI做一些标记。

③.doInBackground(Params... params)，在onPreExecute()完成后立即执行，用于执行较为费时的操作，此方法将接收输入参数和返回计算结果。在执行过程中可以调用publishProgress(Progress... values)来更新进度信息。

④.onProgressUpdate(Progress... values)，在调用publishProgress(Progress... values)时，此方法被执行，直接将进度信息更新到UI组件上。

⑤.onPostExecute(Result result)，当后台操作结束时，此方法将会被调用，计算结果将做为参数传递到此方法中，直接将结果显示到UI组件上。

在使用的时候，有几点需要格外注意：

①.异步任务的实例必须在UI线程中创建。

②.execute(Params... params)方法必须在UI线程中调用。

③.不要手动调用onPreExecute()，doInBackground(Params... params)，onProgressUpdate(Progress... values)，onPostExecute(Result result)这几个方法。

④.不能在doInBackground(Params... params)中更改UI组件的信息。

⑤.一个任务实例只能执行一次，如果执行第二次将会抛出异常。

**25.**

package com.zxzq.javatest;

public class Test3 {

public static void main(String[] args) {

int[] arrays={80,2,30,27,38,19,100,20,19,40};

int minIndex=0;

for(int i=0;i<arrays.length;i++){

minIndex=i;

for(int j=i+1;j<arrays.length;j++){

if(arrays[minIndex]>arrays[j]){

minIndex=j;

}

}

if(minIndex!=i){

int temp=0;

temp=arrays[minIndex];

arrays[minIndex]=arrays[i];

arrays[i]=temp;

}

}

for(int i:arrays){

System.out.print(i+" ");

}

}

}

**26.**

①SharedPreferences存储

②文件存储

③SQLite数据库存储

④ContentProvider存储

⑤网络存储

**27.**

public class SqliteActivity extends Activity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_sqlite);  
  
 //打开或创建数据库 data/data/contacts.zxzq.com.android\_day09/userInfo.db  
 SQLiteDatabase db=openOrCreateDatabase("userInfo",MODE\_PRIVATE,null);  
  
 //存在表person，则删除该表；否则，不执行任何操作  
 db.execSQL("drop table if exists person");  
  
 //创建表  
 //创建表时，如果第一个字段指定为 id integer primary key，则默认为自增长  
 db.execSQL("create table person(id integer primary key autoincrement,name varchar,age smallint)");  
  
 //插入数据 ?:占位符 String[]类型的参数应和占位符的顺序一致  
 db.execSQL("insert into person values(null,?,?)",new String[]{"麦兜","30"});  
 db.execSQL("insert into person values(null,?,?),(null,?,?)",new String[]{"麦兜","30","麦兜2","31"});  
  
 ContentValues cv=new ContentValues();  
 cv.put("name","麦兜3");  
 cv.put("age",18);  
 //sqlite不允许插入空行  
 //第一个参数:表名  
 //第二个参数:只有在values为空的时候，才需要设置；为了防止向表中插入空行  
 //第三个参数:封装了列名和列取值的对象  
 db.insert("person",null,cv);  
  
 //更新数据  
 db.execSQL("update person set age=? where name=?",new String[]{"28","麦兜3"});  
  
 cv=new ContentValues();  
 cv.put("age",18);  
 //第一个参数:表名  
 //第二个参数:封装了列名以及其取值的ContentValues对象  
 //第三个参数:更新数据的条件  
 //第四个参数:条件的取值  
 //第 三+四 个参数的意义:where name="麦兜3"  
 db.update("person",cv,"name=?",new String[]{"麦兜3"});  
  
  
  
 //查询数据  
 //第一个参数:查询的select语句  
 //第二个参数:查询的条件  
 Cursor cursor=db.rawQuery("select \* from person where name=?",new String[]{"麦兜"});  
 //将游标指向下一条数据，有数据，则返回true;否则，返回false  
 while(cursor.moveToNext()){  
 //根据列名获得其索引  
 int id\_index=cursor.getColumnIndex("id");  
 //根据索引，获取该字段的取值  
 int id=cursor.getInt(id\_index);  
  
 //获取name  
 int name\_index=cursor.getColumnIndex("name");  
 String name=cursor.getString(name\_index);  
  
 //获取age  
 int age\_index=cursor.getColumnIndex("age");  
 int age=cursor.getInt(age\_index);  
  
 Log.e("db","id="+id+",name="+name+",age="+age);  
 }  
  
 //关闭游标  
 cursor.close();  
  
 //删除数据  
 db.execSQL("delete from person where name=?",new String[]{"麦兜3"});  
  
 //第一个参数:表名  
 //第二个参数:删除数据的条件  
 //第三个参数:条件的取值  
 //第 三+四 个参数的意义:where name="麦兜"  
 db.delete("person","name=?",new String[]{"麦兜"});  
 //不指明删除的条件时,删除数据，相当于清空表 与 sql 语句删除数据一致  
 db.delete("person",null,null);  
  
 //删除表  
 db.execSQL("drop table person");  
 //删除数据库  
 deleteDatabase("userInfo");  
 }  
}

**28.**

Service是Android四大组件之一，也是可执行的程序，有自己的生命周期。创建、配置Service和创建、配置Activity的过程相似。和Activity一样，都是从Context派生出来的。

服务的2种启动方式:

用start的方式开启服务

使用Service的步骤:

定义一个类继承Service

在清单文件Manifest.xml文件中配置该Service

使用Context的startService（Intent）方法启动该Service

不再使用时，调用stopService（Intent）方法停止该服务

使用这种start方式启动的Service的生命周期如下:

onCreate()🡪onStartCommand()(onStart（）方法已过时)🡪onDestory()

注:如果服务已经开启，不会重复的执行onCreate（），而是会调用onStart（）和onStartCommand（）。服务停止的时候调用onDestory（）。服务只会被停止一次。

特点:一旦服务开启跟调用者(开启者)就没有任何关系了。开启者退出了，开启者挂了，服务还在后台长期的运行。开启者不能调用服务里面的方法。

用bind的方式开启服务

使用Service的步骤

定义一个类继承Service

在清单文件Manifest.xml文件中配置该Service

使用Context的bindService(Intent，ServiceConnextion，int)方法启动该Service

不再使用时，调用unbindService(ServiceConnection)方法停止该服务

使用这种bind方式启动的Service的生命周期如下:

onCreate（）🡪onBind（）🡪onUnbind（）🡪onDestory()

注:绑定服务不会调用onStart（）或onStartCommand()方法

特点：bind的方式开启服务，绑定服务，调用者挂了，服务也会跟着挂掉。绑定者可以调用服务里面的方法。

**29.**

Android中View框架的工作机制中，主要有三个过程：

View树的测量(measure)

View树的布局(layout)

View树的绘制(draw)

View框架的工作流程为:测量每个View的大小(measure)🡪把每个View放置在相应的位置(layout)🡪绘制每个View(draw)

自定义View时，一般需要重写onDraw()和onMeasure()方法。

onDraw()方法中绘制自定义的View，onMeasure()方法中测量该View应显示的宽度和高度。

**30.**

很多软件会有配置文件，里面存放这程序运行当中的各个属性值，由于其配置信息并不多，如果采用[数据库](http://lib.csdn.net/base/mysql)来存放并不划算，因为数据库连接跟操作等耗时大大影响了程序的效率，因此我们使用键值这种一一对应的关系来存放这些配置信息。SharedPreferences正是[Android](http://lib.csdn.net/base/android)中用于实现这种存储方式的技术

 使用SharedPreferences保存key-value对的步骤如下：

（1）使用Activity类的getSharedPreferences方法获得SharedPreferences对象，其中存储key-value的文件的名称由getSharedPreferences方法的第一个参数指定。

（2）使用SharedPreferences接口的edit获得SharedPreferences.Editor对象。

（3）通过SharedPreferences.Editor接口的putXxx方法保存key-value对。其中Xxx表示不同的数据类型。例如：字符串类型的value需要用putString方法。

（4）通过SharedPreferences.Editor接口的commit方法保存key-value对。commit方法相当于数据库事务中的提交（commit）

存储信息的SharedPreferences文件为xml文件。

关键代码:

A、存放数据信息

1、打开Preferences，名称为setting，如果存在则打开它，否则创建新的Preferences

SharedPreferences settings = getSharedPreferences(“setting”, MODE\_PRIVATE);

2、让setting处于编辑状态

SharedPreferences.Editor editor = settings.edit();

3、存放数据

editor.putString(“name”,”value1”);  
editor.putString(“URL”,”value2”);

4、完成提交

editor.commit();

B、读取数据信息

1、获取Preferences

SharedPreferences settings = getSharedPreferences(“setting”, MODE\_PRIVATE);

2、取出数据

String name = settings.getString(“name”,”默认值”);  
String url = setting.getString(“URL”,”default”);

**31.**

Android进程主要分为5种:前台进程、可见进程、服务进程、后台进程、空进程

前台进程:

进程中包含处于前台的正与用户交互的activity

进程中包含与前台activity绑定的service

进程中包含调用了startForeground()方法的service

进程中包含正在执行onCreate()、onStart()或onDestory()方法的service

进程中包含正在执行onReceive()方法的BroadcastReceiver

注:系统中前台进程的数量很少，前台进程几乎不会被杀死，只有当内存低到无法保证所有的前台进程同时运行时，才会选择杀死某个前台进程。

可视进程:

进程中包含未处于前台但仍然可见的activity(调用了activity的onPause()方法，但没有调用onStop()方法)。Eg:运行activity时弹出对话框，此时的activity虽然不是前台activity，但其仍然可见。

进程中包含与可见activity绑定的service

注：可视进程不会被系统杀死，除非为了保证前台进程的运行而不得已为之。

服务进程：进程中包含已启动的service

后台进程：进程中包含不可见的activity(onStop()方法调用后的activity)

空进程：不包含任何处于活动状态的进程是一个空进程。

注：系统经常杀死空进程，这不会造成任何影响。空进程存在的唯一理由是为了缓存一些启动数据，以便下次可以更快的启动。

级别越低越容易被系统清理掉。

**32.**

方法的重载(overload)：方法名相同，参数列表不同。参数列表不同，又分为①参数个数不同②参数类型不同

方法的重写(override)：又称为方法的覆盖，是指子类重写父类中的方法。要求子类中的方法名和参数要和父类中的完全一致。

重写方法不能使用比被重写方法更严格的访问权限。即子类访问权限>=父类。因为子类对象可以被当成父类对象使用。

相同点：方法名相同

**33.**

在JDK 1.2之后，为灵活的控制对象生命周期，把对象的引用分为四级，由高到低分别是:强引用、软引用、弱引用、虚引用

强引用：使用最普遍的引用。如果一个对象具有强引用，那垃圾回收器绝不会回收它。当内存空间不足，Java虚拟机宁愿抛出

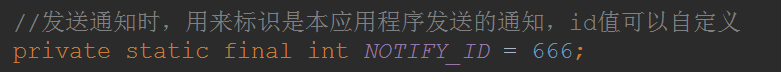
OutOfMemoryError错误，使程序异常终止，也不会靠随意回收具有强引用的对象来解决内存不足的问题。

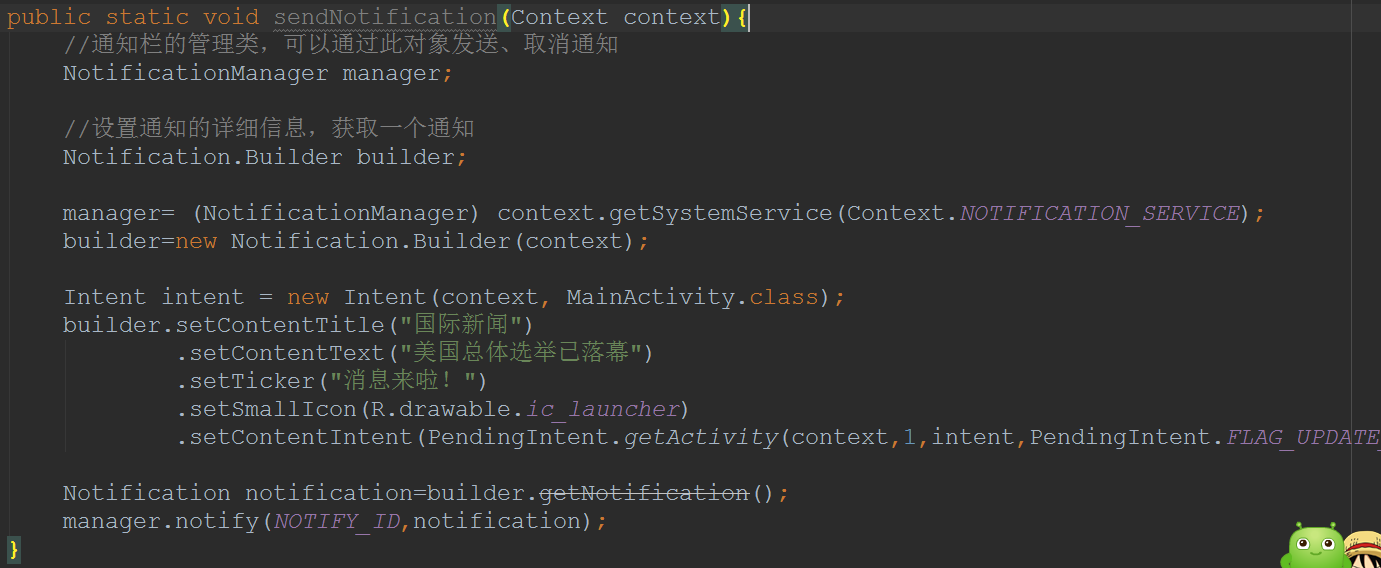
软引用：如果一个对象只具有软引用，则内存空间足够，垃圾回收器就不会回收它；如果内存空间不足了，就会回收这些对象的内存。只要垃圾回收器没有回收它，该对象就可以被程序使用。软引用可用来实现内存敏感的高速缓存。

弱引用：弱引用的对象拥有更短暂的生命周期。在垃圾回收器线程扫描它所管辖的内存区域的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。不过，由于垃圾回收器是一个优先级很低的线程，因此不一定会很快发现那些只具有弱引用的对象。

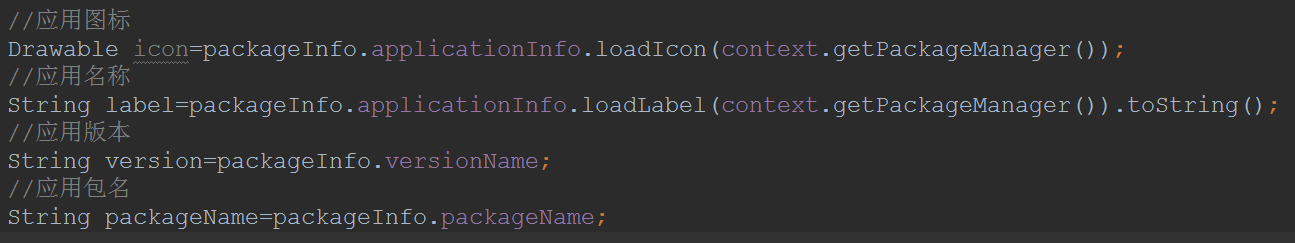
虚引用：“虚引用”顾名思义，就是形同虚设，虚引用不会决定对象的生命周期。如果一个对象仅持有虚引用，那么它就和没有任何引用一样，在任何时候都可能被垃圾回收器回收。

**34.**





**35.**



**36.**

广播机制的概念：

当某个事件产生时(如一条短信发来或一个电话打来)，[Android](http://lib.csdn.net/base/android)[操作系统](http://lib.csdn.net/base/operatingsystem)会把这个事件广播给所有注册的广播接收者，需要处理这个事件的广播接收者进行处理。其实这就是日常生活中的广播。发生一个新闻后，广播电台会广播这个新闻给打开收音机的人，对这个新闻感兴趣的人会关注，可能会拿笔记下。新闻就是事件，广播电台就是android系统，打开收音机的人就是广播接收者，感兴趣的人就是需要处理该事件的广播接收者，拿笔记下就是对该事件进行的操作。

广播的分类：普通广播和有序广播

①普通广播：完全异步，逻辑上可以被任何广播接收者接收到。优点是效率较高。缺点是一个接收者不能将处理结果传递给下一个接收者，并无法终止广播intent的传播。

②有序广播：按照被接收者的优先级顺序，在被接收者中一次传播。比如有三个广播接收者A，B，C，优先级是A > B > C。那这个消息先传给A，再传给B，最后传给C。每个接收者有权中终止广播，比如B终止广播，C就无法接收到。此外A接收到广播后可以对结果对象进行操作，当广播传给B时，B可以从结果对象中取得A存入的数据。如系统收到短信发出的广播就是有序广播。

**37.**

注册广播接收者的2种方式：

①在AndroidManifest.xml中注册

在配置文件中注册的接收者的特点是即使应用程序已被关闭，该接收者依然可接受它感兴趣的广播，比如手机电池电量的广播接收者，没有必要将某个程序开启。下面的例子1、2广播接收者会接收到拨打电话的广播。



②在Activity中注册

在Activity中绑定接收者必须依附该应用程序存在，或者一个BroadcastReceiver用于更新UI，就没有必要在程序关闭时接收者还运行，故无需在AndroidManifest.xml中注册而可以放在Activity中注册。



**38.**

一般情况下，在主线程中我们绑定了Handler，并在事件触发上面创建新的线程用于完成某些耗时的操作，当子线程中的工作完成之后，会对Handler发送一个完成的信号，而Handler接收到信号后，就进行主UI界面的更新操作。

andriod提供了 Handler 和 Looper 来满足线程间的通信。

　　Handler 先进先出原则。

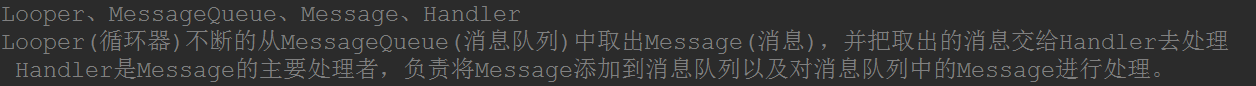
　　Looper类用来管理特定线程内对象之间的消息交换

1)Looper: 一个线程可以产生一个Looper对象，由它来管理此线程里的Message Queue(消息队列)。  
　　2)Handler: 你可以构造Handler对象来与Looper沟通，以便push新消息到Message Queue里;或者接收Looper从Message Queue取出)所送来的消息。  
　　3) Message Queue(消息队列):用来存放线程放入的消息。

4)线程：UI thread 通常就是main thread，而Android启动程序时会替它建立一个Message Queue。

Android程序启动后会起一个进程，所有的组件都在这个进程里面运行。开始这个进程只包含一个线程， 叫做UI主线程，负责处理UI界面的显示更新。对于一些费时的操作需要单独启动一个子线程去处理。子线程处理完毕将结果通知给UI主线程，主线程得到结果

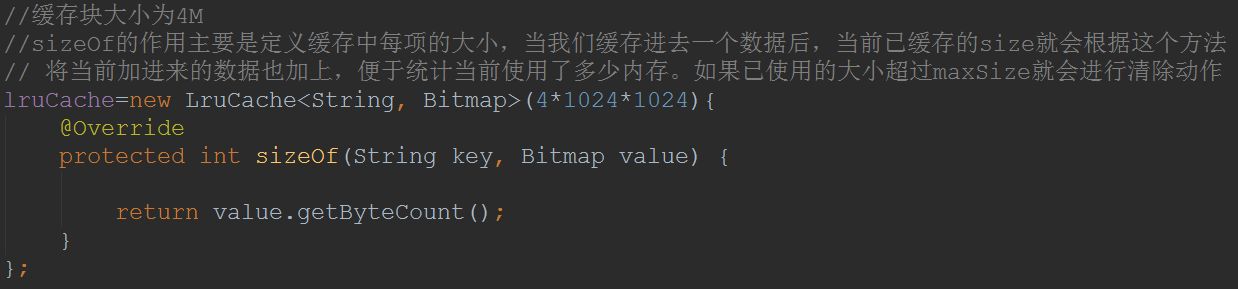
后更新UI界面。



**39.**

LruCache 是Android在android-support-v4包引入的。LruCache的主要功能是能够可控的释放缓存内容，并且LruCache缓存的数据全部采用的是强引用，缓存内容更加可靠。LruCache的内部原理是通过实现了LRU算法来完成对缓存的管理，其内部是维护了一个HashMap。





向LruCache中缓存对象:lruCache.put(k,v)

从LruCache中取出缓存对象:V v=lruCache.get(k)

**40.**

优点：

开放性、开源、免费、可定制

挣脱运营商束缚

丰富的硬件选择

不受任何限制的开发商

无缝结合的Google应用

缺点：

安全问题、隐私问题

同质化严重

运营商对Android手机仍然有影响

山寨化严重

过分依赖开发商，缺乏标准配置

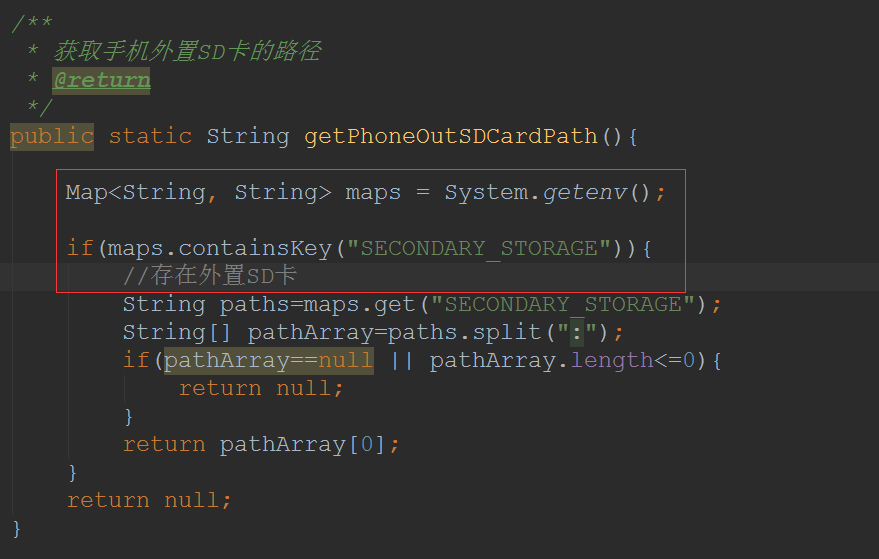
**41.**

内置SD卡通过如下方法：

Environment.getExternalStorageState().equals(Environment.MEDIA\_MOUNTED)

如果返回true就是有sdcard，如果返回false则没有

外置SD卡通过如下方法：



**42.**

Android是一个完整的移动设备操作系统，由Linux核心层，系统运行库层，应用程序框架层和应用程序层组成。

Java是一种编程语言。

编写应用在Android系统的手机程序主要就是使用Java,Android有自己的API，不少API Android直接使用的就是Java API，只是运行环境不一样了。Android应用程序不能在JavaME环境下运行，JavaME程序也不能直接在Android里运行，它们都有自己的API。

**43.**

①．Android是基于Linux内核的，因此Linux对文件权限的控制同样适用于Android

在Android 中每个应用都有自己的/data/data/包名 文件夹，该文件夹只能该应用访问，而其他应用则无权访问。

②．Android的权限机制保护了用户的合法权益

如果我们的代码想拨打电话、发送短信、访问通信录、定位、访问sdcard等所有可能侵犯用于权益的行为都是必须要在AndroidManifest.xml中进行声明的，这样就给了用户一个知情权。

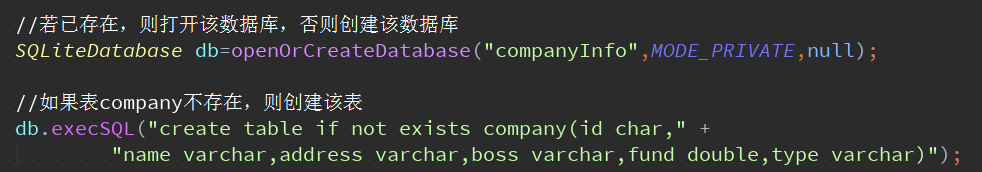
③．Android的代码混淆保护了开发者的劳动成果

**44.**

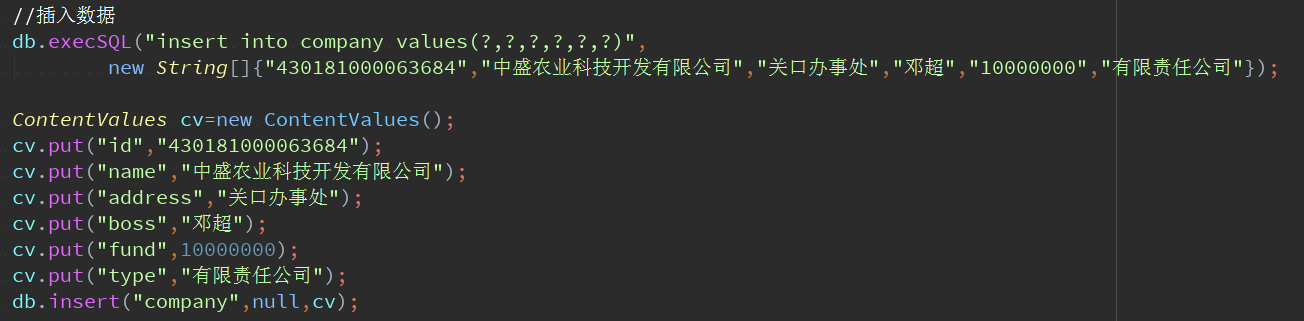
Activity 、 Service 、 ContentProvider 如 果 要 使 用 则 必 须 在 AndroidManifest.xml 中 进 行 注 册 ， 而

BroadcastReceiver则有两种注册方式，静态注册和动态注册。其中静态注册就是指在AndroidManifest.xml中进行注册，而动态注册是通过代码注册。

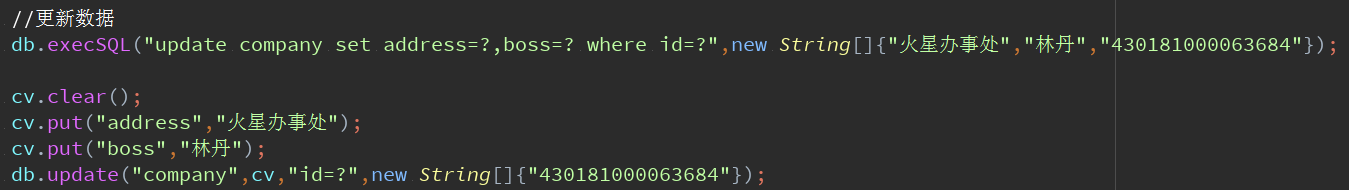
**45.**



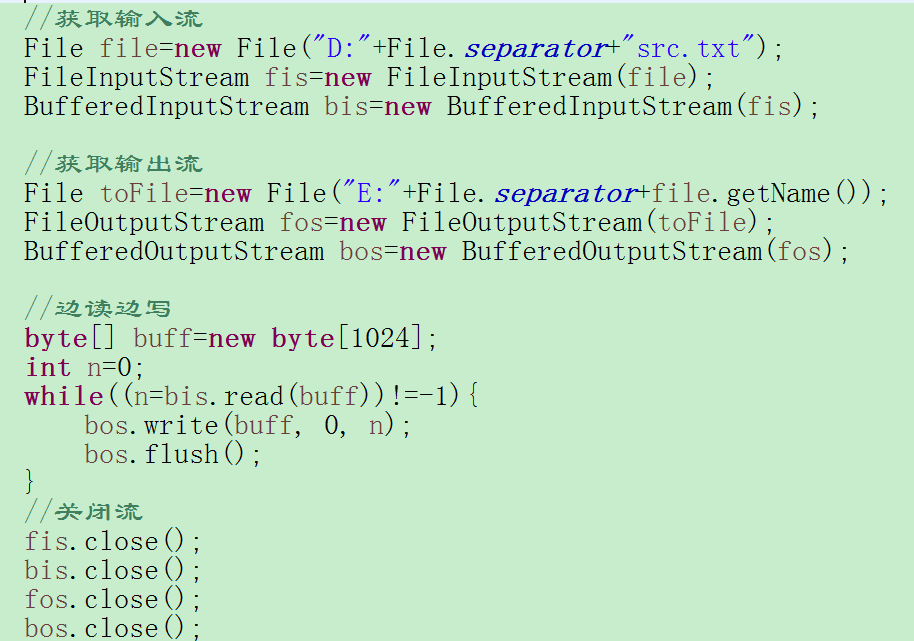
**46.**



**47.**

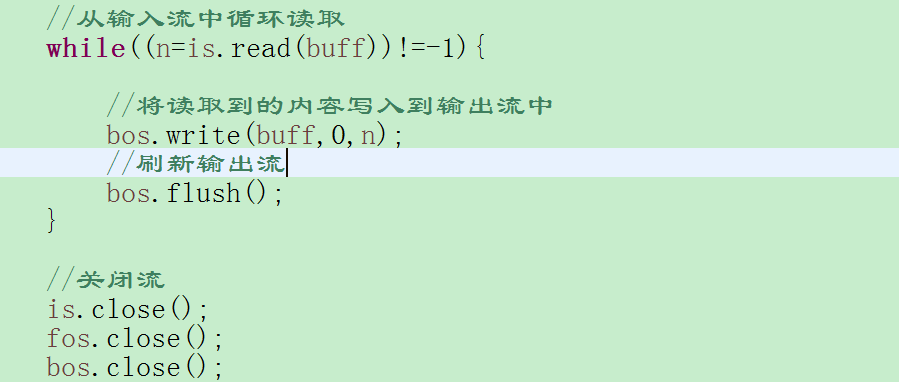


**48.**



**49.**





**50.**

