1. Android xml布局文件中尽量给每个控件指定layout\_weight和layout\_height,不然很可能会出现莫名其妙的错误。
2. CharSequence使用时注意，它可以用int赋值给它，但是还是会运行时失败。
3. SimpleAdapter是在具体的外部控件中调用setViewBinder方法将

**private** ViewBinder mViewBinder;成员变量赋值，这时在内部就可以调用了，但是外部的具体控件必须实现ViewBinder这个接口定义的方法，这样就可以使得SimpleAdapter自己按照自己的意愿行事，而具体的方法留给外面想使用自己的类去实现。

**这里有种很好的设计思路，就是自己定义内部接口，如果外部想使用自己，那么就请实现自己的借口，而自己可以按照自己的意愿假装一切都是已经实现的进行编写自己的更复杂的逻辑功能。切记，这里说的自己定义接口给别人去实现，并不只是说自己声明接口方法，让外部具体类去实现，还可能是自己定义一个内部接口类，而自己内部定义一个该接口类型的成员，提供一个setter方法，使得外部具体类不必显示实现该接口，只要提供一个该接口的具体实现的实例调用setter方法，就可以使得二者默契工作。这样做的好处一点是，外部实现类即使是已经定义好了，但是任然可以使用，这里的**

**public static interface ViewBinder {**

**boolean setViewValue(View view, Object data, String textRepresentation);**

**}**

**就是一个很好的例子。**

1. Android的布局管理器也是ViewGroup的子类。
2. Android 中android:gravity是设置子元素的对其方式，而androd:layout\_gravity是设置自己在父容器中的对齐方式。
3. 在Activity的onCreate方法中，如果想对读取控件上的用户输入，记得要写在对应 的事件回调函数中获取，这是显然的事情，但是有时大脑短路时也是“郁闷的”。
4. BroadCastReceiver类的onReceive方法不应该定义新线程，而应该是开启新的Service去执行特定的操作，以为BroadCastReceiver的生命周期很短，有可能定义的新线程任务还没有执行完，BroadCastReceiver的生命周期就结束了，导致新的线程也被迫提前终止。
5. 但是BroadCastReceiver类中的OnReceive方法怎么可以操作UI 线程中的控件的呢？
6. 多线程中，并不是同步每个方法就可以保证对变量的互斥访问了，因为可能有些业务逻辑，是有多个同步方法串联调用完成的，这时如果在同步方法调用期间暂停，则就会发生意想不到的错误。
7. 在问到线程同步时，就不单要回到锁相关的知识，还要回答ThreadLocal相关的东西。

但是要注意，多线程中两个问题，互斥访问和线程同步，互斥有加锁和线程本地化存储两种解决方式，但是线程同步，只能通过信号量同步。

总之，ThreadLocal不是用来解决对象共享访问问题的，而主要是提供了保持对象的方法和避免参数传递的方便的对象访问方式。归纳了两点：   
1。每个线程中都有一个自己的ThreadLocalMap类对象，可以将线程自己的对象保持到其中，各管各的，线程可以正确的访问到自己的对象。   
2。将一个共用的ThreadLocal静态实例作为key，将不同对象的引用保存到不同线程的ThreadLocalMap中，然后在线程执行的各处通过这个静态ThreadLocal实例的get()方法取得自己线程保存的那个对象，避免了将这个对象作为参数传递的麻烦。

**public** T get() {

Thread t = Thread.*currentThread*();

ThreadLocalMap map = getMap(t);

**if** (map != **null**) {

ThreadLocalMap.Entry e = map.getEntry(**this**);

**if** (e != **null**)

**return** (T)e.value;

}

**return** setInitialValue();

}

**public** **void** set(T value) {

Thread t = Thread.*currentThread*();

ThreadLocalMap map = getMap(t);

**if** (map != **null**)

map.set(**this**, value);

**else**

createMap(t, value);

}