Android开发考核题目\_答案

谢澳

1、说说你对面相对象的理解。

从封装继承多态三个方面来进行分析，无固定答案，合理即可。

2、Overload和Override的区别。Overloaded的方法是否可以改变返回值的类型?   
方法的重写Overriding和重载Overloading是Java多态性的不同表现。重写Overriding是父类与子类之间多态性的一种表现，重载Overloading是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被”屏蔽”了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded的方法是可以改变返回值的类型。

3、String和StringBuffer，StringBuilder的区别   
String 字符串常量   
StringBuffer 字符串变量（线程安全）   
StringBuilder 字符串变量（非线程安全）   
String的长度是不可变的，StringBuffer，StringBuilder的长度是可变的。如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用StringBuffer，如果最后需要String，那么使用StringBuffer的toString()方法。

4、说出ArrayList,Vector, LinkedList的存储性能和特性   
ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector由于使用了synchronized方法（线程安全），通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

5.字符串“abcde”通过写一个函数不让调用第三方的字符串，实现一个字符串倒序，比如字符串“abcde”变成“edcba”   
String src = “abcde”;   
String dst = new StringBuffer(src).reverse().toString();

6、Collection 和 Collections的区别。   
Collection是集合类的上级接口，继承与他的接口主要有Set 和List.   
Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作

7、final, finally, finalize的区别。   
final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。   
finally是异常处理语句结构的一部分，表示总是执行。   
finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

8、同步和异步有何异同，在什么情况下分别使用他们？举例说明。   
如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。   
当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

9，抽象类与接口的区别（abstract与interface的区别）   
abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用abstract定义该类，即抽象类。   
用interface修饰的类，里面的方法都是抽象方法，因此在定义接口的时候，可以直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即public static final修饰的字段。

10、线程中wait，join，sleep，yield, notify，notifyall，synchronized，区别及联系   
1).sleep()方法   
在指定时间内让当前正在执行的线程暂停执行，但不会释放“锁标志”。不推荐使用。sleep()使当前线程进入阻塞状态，在指定时间内不会执行。   
2).wait()方法   
在其他线程调用对象的notify或notifyAll方法前，导致当前线程等待。线程会释放掉它所占有的“锁标志”，从而使别的线程有机会抢占该锁。   
唤醒当前对象锁的等待线程使用notify或notifyAll方法,waite() 和notify()必须在synchronized函数或synchronized　block中进行调用。   
yield方法暂停当前正在执行的线程对象。yield()只是使当前线程重新回到可执行状态，所以执行   
3)yield()的线程有可能在进入到可执行状态后马上又被执行。yield()只能使同优先级或更高优先级的线程有执行的机会。   
4).join方法   
等待该线程终止。等待调用join方法的线程结束，再继续执行。如：t.join();//主要用于等待t线程运行结束，若无此句，main则会执行完毕，导致结果不可预测。

11、接口是否可继承接口? 抽象类是否可实现(implements)接口? 抽象类是否可继承实体类(concrete class)?   
接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承实体类，但前提是实体类必须有明确的构造函数。

12、abstract的method是否可同时是static,是否可同时是native，是否可同时是synchronized?

13、是否可以继承String类?   
String类是final类故不可以继承。

14、[Java](http://lib.csdn.net/base/java)switch支持的数据类型：   
java支持的数据类型有五种   
他们分别是：   
byte、char、short、int、枚举   
以上是JDK1.6以前的版本。JDK1.7时，又增加了String，所以相对于JDK1.7而言就是六种了

15、什么是单例模式，请写出一个来：   
Singleton模式主要作用是保证在Java应用程序中，一个类Class只有一个实例存在。   
一般Singleton模式通常有几种种形式:   
第一种形式: 定义一个类，它的构造函数为private的，它有一个static的private的该类变量，在类初始化时实例话，通过一个public的getInstance方法获取对它的引用,继而调用其中的方法。

public class Singleton {

private Singleton(){}

//注意这是private 只供内部调用

private static Singleton instance = new Singleton();

//这里提供了一个供外部访问本class的静态方法，可以直接访问

public static Singleton getInstance() {

return instance;

}

}

第二种形式:

public class Singleton {

private static Singleton instance = null;

public static synchronized Singleton getInstance() {

//这个方法比上面有所改进，不用每次都进行生成对象，只是第一次

//使用时生成实例，提高了效率！

if (instance==null)

instance＝new Singleton();

return instance; }

}

其他形式:   
定义一个类，它的构造函数为private的，所有方法为static的。   
一般认为第一种形式要更加安全些

16、Java常用的设计模式？   
Java中的23种设计模式：   
Factory（工厂模式）， Builder（建造模式）， Factory Method（工厂方法模式），   
Prototype（原始模型模式），Singleton（单例模式）， Facade（门面模式），   
Adapter（适配器模式）， Bridge（桥梁模式）， Composite（合成模式），   
Decorator（装饰模式）， Flyweight（享元模式）， Proxy（代理模式），   
Command（命令模式）， Interpreter（解释器模式）， Visitor（访问者模式），   
Iterator（迭代子模式）， Mediator（调停者模式）， Memento（备忘录模式），   
Observer（观察者模式）， State（状态模式）， Strategy（策略模式），   
Template Method（模板方法模式）， Chain Of Responsibleity（责任链模式）   
工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某一个类的实例，   
通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。   
首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件   
生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

17、1、Android四大组件是什么？作用以及应用场景？   
Android 的四大组件分别是是Activity，Service，BroadcastReceiver和ContentProvider；

Activity：   
从字面上理解，Activity是活动的意思。一个Activity通常展现为一个可视化的用户界面，是Android程序与用户交互的窗口，也是Android组件中最基本也是最复杂的一个组件。从视觉效果来看，一个Activity占据当前的窗口，响应所有窗口事件，具备有控件，菜单等界面元素。从内部逻辑来看，Activity需要为了保持各个界面状态，需要做很多持久化的事情，还需要妥善管理生命周期，和一些转跳逻辑。

Service   
服务是运行在后台的一个组件，从某从意义上说，服务就像一个没有界面的Activity。它们在很多Android的概念方面比较接近，封装有一个完整的功能逻辑实现，接受上层指令，完成相关的事件，定义好需要接受的Intent提供同步和异步的接口。

BroadcastReceiver   
广播接收者，不执行任何任务，广播是一种广泛运用的在应用程序之间传输信息的机制 。而 BroadcastReceiver 是对发送出来的广播进行过滤接收并响应的一类组件。Broadcast Receiver 不包含任何用户界面。然而它们可以启动一个Activity以响应接受到的信息，或者通过NotificationManager通知用户。可以通过多种方式使用户知道有新的通知产生：闪动背景灯、震动设备、发出声音等等。通常程序会在状态栏上放置一个持久的图标，用户可以打开这个图标并读取通知信息。在Android中还有一个很重要的概念就是Intent，如果说Intent是一个对动作和行为的抽象描述，负责组件之间程序之间进行消息传递。那么Broadcast Receiver组件就提供了一种把Intent作为一个消息广播出去，由所有对其感兴趣的程序对其作出反应的机制。

Content Provider   
即内容提供者，作为应用程序之间唯一的共享数据的途径，Content Provider 主要的功能就是存储并检索数据以及向其他应用程序提供访问数据。   
对应用而言，也可以将底层数据封装成ContentProvider，这样可以有效的屏蔽底层操作的细节，并且使程序保持良好的扩展性和开放性。Android提供了一些主要数据类型的Contentprovider，比如音频、视频、图片和私人通讯录等。可在android.provider包下面找到一些android提供的Contentprovider。可以获得这些Contentprovider，查询它们包含的数据，当然前提是已获得适当的读取权限。如果我们想公开自己应用程序的数据，可以创建自己的 Content provider 的接口。

18、android中的动画有哪几类，它们的特点和区别是什么?   
两种，一种是Tween动画、还有一种是Frame动画。Tween动画，这种实现方式可以使视图组件移动、放大、缩小以及产生透明度的变化;另一种Frame动画，传统的动画方法，通过顺序的播放排列好的图片来实现，类似电影。   
   
19、如何将一个Activity设置成窗口的样式。   
在AndroidManifest.xml 中定义Activity的地方一句话android:theme=”@android:style/Theme.Dialog”或android:theme=”@android:style/Theme.Translucent”就变成半透明的

20、.注册广播有几种方式，有何优缺点？   
有两种，一种是代码动态注册：   
//生成广播处理   
smsBroadCastReceiver = new SmsBroadCastReceiver();   
//实例化过滤器并设置要过滤的广播

IntentFilter intentFilter = new IntentFilter(“android.provider.Telephony.SMS\_RECEIVED”);

//注册广播   
BroadCastReceiverActivity.this.registerReceiver(smsBroadCastReceiver, intentFilter);   
一种是在AndroidManifest.xml中配置广播

<!--广播注册-->

<receiver android:name=".SmsBroadCastReceiver">

<intent-filter android:priority="20">

<action android:name="android.provider.Telephony.SMS\_RECEIVED"/>

</intent-filter>

</receiver>

两种注册类型的区别是：   
1)第一种不是常驻型广播，也就是说广播跟随程序的生命周期   
2)第二种是常驻型，也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。   
注册的方法有两种，一种是静态注册，一种是动态注册。   
动态注册优点： 在 Android 的广播机制中， 动态注册的优先级是要高于静态注册优先级的 ， 因此在必要的情况下，我们是需要动态注册广播接收器的。   
静态注册优点：动态注册广播接收器还有一个特点，就是当用来注册的 Activity 关掉后 ，广播也就失效了。同时反映了静态注册的一个优势，就是无需担忧广播接收器是否被关闭 ， 只要设备是开启状态，广播接收器就是打开着的。

21、IntentService有何优点?   
普通的service ,默认运行在ui main 主线程   
这是带有异步处理的service类,   
异步处理的方法 OnHandleIntent()   
OnHandleIntent() 处理耗时的操作   
Android的进程处理器现在会尽可能的不kill掉你

22、横竖屏切换时候activity的生命周期?   
1、不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次   
　　2、设置Activity的android:configChanges=”orientation”时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次   
　　3、设置Activity的android:configChanges=”orientation|keyboardHidden”时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法   
　　   
23、如何将SQLite[数据库](http://lib.csdn.net/base/mysql)(dictionary.db文件)与apk文件一起发布? 如何将打开res raw目录中的数据库文件?   
　　解答：以将dictionary.db文件复制到res raw目录中 ，在Android中不能直接打开res raw目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或SD卡的某个目录中，然后再打开该数据库文件。复制的基本方法是使用getResources().openRawResource方法获得res raw目录中资源的 InputStream对象，然后将该InputStream对象中的数据写入其他的目录中相应文件中。在Android SDK中可以使用SQLiteDatabase.openOrCreateDatabase方法来打开任意目录中的SQLite ；   
　　   
24、.AndroidManifest.xml文件中主要包含哪些信息  
说明应用程序的[Java](http://lib.csdn.net/base/javase) 包，该包名是应用程序的唯一标识；   
描述应用程序的组件：该应用程序由哪些activity，service，broadcast receiver和content provider组成；   
声明应用程序所必须具备的权限，用以访问受保护的部分API，以及与其他应用程序的交互；   
声明应用程序所需要的Android API的最低版本级别，比如1.0，1.1，1.5；   
manifest：根节点，描述了 package 中所有的内容。   
uses-permission：请求你的 package 正常运作所需赋予的安全许可。   
permission： 声明了安全许可来限制哪些程序能你 package 中的组件和功能。   
instrumentation：声明了用来[测试](http://lib.csdn.net/base/softwaretest)此 package 或其他 package 指令组件的代码。   
application：包含 package 中 application 级别组件声明的根节点。   
activity：Activity 是用来与用户交互的主要工具。   
receiver：IntentReceiver 能使的 application 获得数据的改变或者发生的操作，即使它当 前不在运行。   
service：Service 是能在后台运行任意时间的组件。   
provider：ContentProvider 是用来管理持久化数据并发布给其他应用程序使用的组件。

25、.Android数据储存方式   
一.SharedPreferences方式：它是 Android提供的用来存储一些简单配置信息的一种机制，采用了 XML 格式将数据存储到设备中。只能在同一个包内使用，不能在不同的包之间使用。   
二.文件存储方式 /data/data/包名/files 内存里面   
/ context.openFileInput(name) 默认是私有的访问权限   
三.SQLite数据库方式：SQLite 是 Android 所带的一个标准的数据库， 它支持 SQL语句，它是一个轻量级的[嵌入式](http://lib.csdn.net/base/embeddeddevelopment)数据库。   
四.内容提供器（Content provider）方式：主要用于应用程序之间进行数据交换，从而能够让其他的应用保存或读取此 Content Provider 的各种数据类型。   
五. 网络存储方式： 通过网络上提供给我们的存储空间来上传(存储)和下载(获取)我们存储 在网络空间中的数据信息。

26、.Android中常见五种布局介绍   
FrameLayout（帧布局）： 从屏幕的左上角开始布局,叠加显示, 实际应用 播放器的暂停按钮.   
LinearLayout （线性布局）：可分为垂直布局，水平布局；   
AbsoluteLayout（绝对布局）：用X,Y坐标来指定元素的位置；   
RelativeLayout（相对布局）： 相对布局可以理解为某一个元素为参照物，来定位的布局方式。   
TableLayout（表格布局）：表格布局类似Html里面的Table。每一个TableLayout里面有表格行TableRow，TableRow里面可以具体定义每一个元素，设定他的对齐方式 android:gravity=”” 。   
每一个布局都有自己适合的方式，另外，这五个布局元素可以相互嵌套应用，做出美观的界面。

27、Activity的四种启动模式？,activity与task的启动方式有哪些，她们的含义具体说明   
standard: 标准模式，一调用 startActivity()方法就会产生一个新的实例。   
singleTop: 如果已经有一个实例位于 Activity 栈的顶部时， 就不产生新的实例， 而只是调用Activity 中的 newInstance()方法。如果不位于栈顶，会产生一个新的实例。   
singleTask: 会在一个新的 task 中产生这个实例，以后每次调用都会使用这个，不会去产生 新的实例了。   
singleInstance: 这个跟 singleTask 基本上是一样， 只有一个区别： 在这个模式下的 Activity实例所处的 task 中，只能有这个 activity实例，不能有其他的实例。

28、.Android 中的长度单位详解   
如果设置表示长度 、 高度等属性时可以使用 dp 或 sp。但如果设置字体，需要使用 sp。dp 是与密度无关，sp 除了与密度无关外，还与 scale 无关。如果屏幕密度为160，这时 dp 和 sp 和 px 是一样   
的。1dp=1sp=1px，   
也就是说，如果使用 dp 和 sp，系统会根据屏幕密度的变化自动进行转换   
px ：表示屏幕实际的象素   
in：表示英寸，是屏幕的物理尺寸。每英寸等于2.54 厘米。

29、android 析 中有哪几种解析 l xml 的类，官方推荐哪种？以及它们的原 理和区别   
DOM 解析   
优点:   
1.XML 树在内存中完整存储,因此可以直接修改其数据和结构.   
2.可以通过该解析器随时访问 XML 树中的任何一个节点.   
3.DOM 解析器的 API 在使用上也相对比较简单.   
缺点:如果 XML 文档体积比较大时,将文档读入内存是非常消耗系统资源的   
Ø SAX 解析   
优点:   
SAX 对内存的要求比较低,因为它让开发人员自己来决定所要处理的标签.特别是当开发人 员只需要处理文档中所包含的部分数据时,SAX 这种扩展能力得到了更好的体现.   
缺点:   
用 SAX 方式进行 XML 解析时,需要顺序执行,所以很难访问到同一文档中的不同数据.此外,在基于该方式的解析编码过程也相对复杂   
Ø Xmlpull 解析（官方推荐使用）   
android SDK 提供了 xmlpull api,xmlpull 和 sax 类似,是基于流（stream）操作文件,然后根据节点事件回调开发者编写的处理程序.因为是基于流的处理,因此 xmlpull 和 sax 都比较节   
约内存资源,不会象 dom 那样要把所有节点以对橡树的形式展现在内存中.xmlpull 比 sax 更 简明,而且不需要扫描完整个流.

30、.ListView 的 优化方案   
（1），如果自定义适配器，那么在 getView 方法中要考虑方法传进来的参数 contentView 是否 为 null，如果为 null 就创建 contentView 并返回，如果不为 null 则直接使用。在这个方法中，尽可能少创建 view。   
（2），给 contentView 设置 tag(setTag()),传入一个 viewHolder 对象，用于缓存要显示的数 据，可以达到图像数据异步加载的效果   
（3），如果 listview 需要显示的 item 很多，就要考虑分页加载。比如一共要显示100条或者更多的时候，我们可以考虑先加载20条，等用户拉到列表底部的时候，再去加载接下来的20 条。   
（4） 快速滑动时不加载图片   
（5） 如果自定义的item中有图片，需要处理图片（减少图片所占内存）   
1.对图片进行边界压缩 2.用option类来保存图片大小 3.避免图片的实时缩放，最好预先缩放到视图大小   
（6）尽量避免在listview适配器中使用线程，因为线程是产生内存泄露的主要原因在于线程的生命周期不可控。

31、Android中intent的作用   
在一个 Android 应用中，主要是由一些组件组成（Activity,Service,ContentProvider,etc.) 在这些组件之间的通讯中，由 Intent 协助完成。   
Intent 负责对应用中一次操作的动作、动作涉及数据、附加数据   
进行描述，Android 则根据此 Intent 的描述，负责找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用。Intent 在这里起着实现调用者与被调用者之间的解耦作用。   
有两种方式，一种是显示匹配，一种是隐示匹配；   
显示：明确的指定要跳转的Activity，或者Service等；   
隐示：隐式匹配，首先要匹配 Intent 的几项值：Action,Category, Data/Type,Component如果填写了 Componet 就是这就形成了显示匹配。   
1.如果你填写了 Action， 如果有一个程序的 Manifest.xml 中的某一个 Activity 的 IntentFilter 段中定义了包含了相同的 Action 那么这个 Intent 就与这个目标 Action 匹配， 如果这个 Filter段中没有定义 Type,Category，那么这个 Activity 就匹配了。但是如果手机中有两个以上的程序匹配，那么就会弹出一个对话可框来提示说明。   
2,data/type，你可以用 Uri 来做为 data,比如 Uri uri = Uri.parse([http://www.google.com](http://www.google.com/) );   
Intent i = new Intent(Intent.ACTION\_VIEW,uri);手机的 Intent 分发过程中，会根据 [http://www.google.com](http://www.google.com/) 的 scheme 判断出数据类型 type手机的 Brower 则能匹配它，在 Brower 的 Manifest.xml 中的 IntenFilter中首先有ACTION\_VIEW Action,也能处理 http:的 type；   
3，至于分类 Category，一般不要去在 Intent 中设置它，如果你写 Intent 的接收者，就在Manifest.xml 的 Activity 的 IntentFilter 中包含 android.category.DEFAULT,这样所有不设置Category（Intent.addCategory(String c);）的 Intent 都会与这个 Category 匹配。   
4,extras（附加信息） ，是其它所有附加信息的集合。使用 extras 可以为组件提供扩展信息 ，比如，如果要执行“发送电子邮件”这个动作，可以将电子邮件的标题、正文等保存在 extras里，传给电子邮件发送组件。

32、如何退出 Activity ？如何安全退出？   
对于单一 Activity 的应用来说，退出很简单，直接 finish()即可。   
当然，也可以用 killProcess()和 System.exit()这样的方法。   
为了编程方便，最好定义一个 Activity 基类，处理这些共通问题。   
1、记录打开的 Activity：   
每打开一个 Activity，就记录下来。在需要退出时，关闭每一个 Activity 即可。   
2、发送特定广播：   
在需要结束应用时，发送一个特定的广播，每个 Activity 收到广播后，关闭即可。

33、谈谈你对Android 的安全问题了解。

①错误导出组件   
② 参数校验不严   
③WebView引入各种安全问题,webview中的js注入   
④不混淆、不防二次打包   
⑤明文存储关键信息   
⑦ 错误使用HTTPS   
⑧山寨加密方法   
⑨滥用权限、内存泄露、使用debug签名

34.、如何缩减APK包大小？   
代码   
保持良好的编程习惯，不要重复或者不用的代码，谨慎添加libs，移除使用不到的libs。   
使用proguard混淆代码，它会对不用的代码做优化，并且混淆后也能够减少安装包的大小。   
native code的部分，大多数情况下只需要支持armabi与x86的[架构](http://lib.csdn.net/base/architecture)即可。如果非必须，可以考虑拿掉x86的部分。   
资源   
使用Lint工具查找没有使用到的资源。去除不使用的图片，String，XML等等。 assets目录下的资源请确保没有用不上的文件。   
生成APK的时候，aapt工具本身会对png做优化，但是在此之前还可以使用其他工具如tinypng对图片进行进一步的压缩预处理。   
jpeg还是png，根据需要做选择，在某些时候jpeg可以减少图片的体积。 对于9.png的图片，可拉伸区域尽量切小，另外可以通过使用9.png拉伸达到大图效果的时候尽量不要使用整张大图。   
策略   
有选择性的提供hdpi，xhdpi，xxhdpi的图片资源。建议优先提供xhdpi的图片，对于mdpi，ldpi与xxxhdpi根据需要提供有差异的部分即可。   
尽可能的重用已有的图片资源。例如对称的图片，只需要提供一张，另外一张图片可以通过代码旋转的方式实现。   
能用代码绘制实现的功能，尽量不要使用大量的图片。例如减少使用多张图片组成animate-list的AnimationDrawable，这种方式提供了多张图片很占空间。

4.Android与服务器交互的方式中的对称加密和非对称加密是什么?   
对称加密，就是加密和解密数据都是使用同一个key，这方面的[算法](http://lib.csdn.net/base/datastructure)有DES。   
非对称加密，加密和解密是使用不同的key。发送数据之前要先和服务端约定生成公钥和私钥，使用公钥加密的数据可以用私钥解密，反之。这方面的算法有RSA。ssh 和 ssl都是典型的非对称加密。

35、设备横竖屏切换的时候，接下来会发生什么？   
1、不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次   
2、设置Activity的android:configChanges=”orientation”时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次   
3、设置Activity的android:configChanges=”orientation|keyboardHidden”时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法

36、Android启动Service的两种方式是什么? 它们的适用情况是什么?   
如果后台服务开始后基本可以独立运行的话，可以用startService。音乐播放器就可以这样用。它们会一直运行直到你调用 stopSelf或者stopService。你可以通过发送Intent或者接收Intent来与正在运行的后台服务通信，但大部分时间，你只是启动服务并让它独立运行。如果你需要与后台服务通过一个持续的连接来比较频繁地通信，建议使用bind()。比如你需要定位服务不停地把更新后的地理位置传给UI。Binder比Intent开发起来复杂一些，但如果真的需要，你也只能使用它。   
startService：生命周期与调用者不同。启动后若调用者未调用stopService而直接退出，Service仍会运行   
bindService：生命周期与调用者绑定，调用者一旦退出，Service就会调用unBind->onDestroy

37、谈谈你对Android中Context的理解?   
Context:包含上下文信息(外部值) 的一个参数. Android 中的 Context 分三种,Application Context ,Activity Context ,Service Context.   
它描述的是一个应用程序环境的信息，通过它我们可以获取应用程序的资源和类，也包括一些应用级别操作，例如：启动一个Activity，发送广播，接受Intent信息等

38、Service的onCreate回调在UI线程中吗？   
Service生命周期的各个回调和其他的应用组件一样，是跑在主线程中，会影响到你的UI操作或者阻塞主线程中的其他事情

39、请介绍下AsyncTask的内部实现，适用的场景是？   
AsyncTask内部也是Handler机制来完成的，只不过Android提供了执行框架来提供线程池来执行相应地任务，因为线程池的大小问题，所以AsyncTask只应该用来执行耗时时间较短的任务，比如HTTP请求，大规模的下载和[数据库](http://lib.csdn.net/base/mysql)的更改不适用于AsyncTask，因为会导致线程池堵塞，没有线程来执行其他的任务，导致的情形是会发生AsyncTask根本执行不了的问题。

40、谈谈你对binder机制的理解?   
binder是一种IPC机制,进程间通讯的一种工具.   
[Java](http://lib.csdn.net/base/javase)层可以利用aidl工具来实现相应的接口.

41、Android中进程间通信有哪些实现方式？   
Intent，Binder（AIDL），Messenger，BroadcastReceiver

42、介绍下实现一个自定义view的基本流程   
1、自定义View的属性 编写attr.xml文件   
2、在layout布局文件中引用，同时引用命名空间   
3、在View的构造方法中获得我们自定义的属性 ，在自定义控件中进行读取（构造方法拿到attr.xml文件值）   
4、重写onMesure   
5、重写onDraw

43、Android中touch事件的传递机制是怎样的?   
1.Touch事件传递的相关API有dispatchTouchEvent、onTouchEvent、onInterceptTouchEvent   
2.Touch事件相关的类有View、ViewGroup、Activity   
3.Touch事件会被封装成MotionEvent对象，该对象封装了手势按下、移动、松开等动作   
4.Touch事件通常从Activity#dispatchTouchEvent发出，只要没有被消费，会一直往下传递，到最底层的View。   
5.如果Touch事件传递到的每个View都不消费事件，那么Touch事件会反向向上传递,最终交由Activity#onTouchEvent处理.   
6.onInterceptTouchEvent为ViewGroup特有，可以拦截事件.   
7.Down事件到来时，如果一个View没有消费该事件，那么后续的MOVE/UP事件都不会再给它

44、.Android开发中何时使用多进程？使用多进程的好处是什么？   
要想知道如何使用多进程，先要知道Android里的多进程概念。一般情况下，一个应用程序就是一个进程，这个进程名称就是应用程序包名。我们知道进程是系统分配资源和调度的基本单位，所以每个进程都有自己独立的资源和内存空间，别的进程是不能任意访问其他进程的内存和资源的。   
45、如何让自己的应用拥有多个进程？

我们的四大组件在AndroidManifest文件中注册的时候，有个属性是android:process，1.这里可以指定组件的所处的进程。默认就是应用的主进程。指定为别的进程之后，系统在启动这个组件的时候，就先创建（如果还没创建的话）这个进程，然后再创建该组件。你可以重载Application类的onCreate方法，打印出它的进程名称，就可以清楚的看见了。再设置android:process属性时候，有个地方需要注意：如果是android:process=”:deamon”，以:开头的名字，则表示这是一个应用程序的私有进程，否则它是一个全局进程。私有进程的进程名称是会在冒号前自动加上包名，而全局进程则不会。一般我们都是有私有进程，很少使用全局进程。他们的具体区别不知道有没有谁能补充一下。

2.使用多进程显而易见的好处就是分担主进程的内存压力。我们的应用越做越大，内存越来越多，将一些独立的组件放到不同的进程，它就不占用主进程的内存空间了。当然还有其他好处，有心人会发现Android后台进程里有很多应用是多个进程的，因为它们要常驻后台，特别是即时通讯或者社交应用，不过现在多进程已经被用烂了。典型用法是在启动一个不可见的轻量级私有进程，在后台收发消息，或者做一些耗时的事情，或者开机启动这个进程，然后做监听等。还有就是防止主进程被杀守护进程，守护进程和主进程之间相互监视，有一方被杀就重新启动它。应该还有还有其他好处，这里就不多说了。

3.坏处的话，多占用了系统的空间，大家都这么用的话系统内存很容易占满而导致卡顿。消耗用户的电量。应用程序架构会变复杂，应为要处理多进程之间的通信。这里又是另外一个问题了。

46、ANR是什么？怎样避免和解决ANR?   
ANR:Application Not Responding，即应用无响应   
ANR一般有三种类型：   
1：KeyDispatchTimeout(5 seconds) –主要类型   
按键或触摸事件在特定时间内无响应

2：BroadcastTimeout(10 seconds)   
BroadcastReceiver在特定时间内无法处理完成

3：ServiceTimeout(20 seconds) –小概率类型   
Service在特定的时间内无法处理完成

超时的原因一般有两种：   
(1)当前的事件没有机会得到处理（UI线程正在处理前一个事件没有及时完成或者looper被某种原因阻塞住）   
(2)当前的事件正在处理，但没有及时完成

UI线程尽量只做跟UI相关的工作，耗时的工作（数据库操作，I/O，连接网络或者其他可能阻碍UI线程的操作）放入单独的线程处理，尽量用Handler来处理UI thread和thread之间的交互。

UI线程主要包括如下：   
Activity:onCreate(), onResume(), onDestroy(), onKeyDown(), onClick()   
AsyncTask: onPreExecute(), onProgressUpdate(), onPostExecute(), onCancel()   
Mainthread handler: handleMessage(), post(runnable r)   
other

47、Android下解决滑动冲突的常见思路是什么?   
相关的滑动组件 重写onInterceptTouchEvent，然后判断根据xy值，来决定是否要拦截当前操作

48、如何把一个应用设置为系统应用？   
成为系统应用，首先要在 对应设备的 Android 源码 SDK 下编译，编译好之后：   
此 Android 设备是 Debug 版本，并且已经 root，直接将此 apk 用 adb 工具 push 到 system/app 或 system/priv-app 下即可。   
如果非 root 设备，需要编译后重新烧写设备镜像即可。

有些权限(如 WRITE\_SECURE\_SETTINGS )，是不开放给第三方应用的，只能在对应设备源码中编译然后作为系统 app 使用。

49、Android内存泄露研究   
Android内存泄漏指的是进程中某些对象（垃圾对象）已经没有使用价值了，但是它们却可以直接或间接地引用到gc roots导致无法被GC回收。无用的对象占据着内存空间，使得实际可使用内存变小，形象地说法就是内存泄漏了。   
场景   
类的静态变量持有[大数据](http://lib.csdn.net/base/hadoop)对象   
静态变量长期维持到大数据对象的引用，阻止垃圾回收。   
非静态内部类的静态实例   
非静态内部类会维持一个到外部类实例的引用，如果非静态内部类的实例是静态的，就会间接长期维持着外部类的引用，阻止被回收掉。   
资源对象未关闭   
资源性对象如Cursor、File、Socket，应该在使用后及时关闭。未在finally中关闭，会导致异常情况下资源对象未被释放的隐患。   
注册对象未反注册   
未反注册会导致观察者列表里维持着对象的引用，阻止垃圾回收。   
Handler临时性内存泄露   
Handler通过发送Message与主线程交互，Message发出之后是存储在MessageQueue中的，有些Message也不是马上就被处理的。在Message中存在一个 target，是Handler的一个引用，如果Message在Queue中存在的时间越长，就会导致Handler无法被回收。如果Handler是非静态的，则会导致Activity或者Service不会被回收。   
由于AsyncTask内部也是Handler机制，同样存在内存泄漏的风险。   
此种内存泄露，一般是临时性的。

50、内存泄露检测有什么好方法？   
检测：   
1、DDMS Heap发现内存泄露   
dataObject totalSize的大小，是否稳定在一个范围内，如果操作程序，不断增加，说明内存泄露   
2、使用Heap Tool进行内存快照前后对比   
BlankActivity手动触发GC进行前后对比，对象是否被及时回收

定位：   
1、MAT插件打开.hprof具体定位内存泄露：   
查看histogram项，选中某一个对象，查看它的GC引用链，因为存在GC引用链的，说明无法回收   
2、AndroidStudio的Allocation Tracker：   
观测到期间的内存分配，哪些对象被创建，什么时候创建，从而准确定位