**1、请描述一下Activity的启动模式都有哪些以及各自的特点.**

**Activity的启动模式决定了Activity被激活时，如何添加到任务栈。**

**Activity 一共有以下四种 launchMode（启动模式）：**

1. **standard：标准模式（默认），**

**standard模式的Activity可以被反复激活且每次激活都是新的Activity**

**对象。**

1. **singleTop：栈顶唯一**

**如果Activity已经位于栈顶，再次被激活时，不会创建新的Activity对象，反之如果被激活的****Activity没有处于栈顶，则还是会创建新的****Activity对象。**

1. **singleTask：栈内唯一**

**被激活的Activity 定位于栈顶，且在栈内只有一个，即如果当前任务栈没有该Activity，则直接创建并且压栈，当被激活的Activity自然处于栈顶位置，显示在前台，如果当前任务栈中已经存在该Activity，则在栈中该Activity之上的所有Activity被强制出栈（被销毁），而被激活的Activity自然弹栈，获得栈顶位置，显示在前台。**

1. **singleInstance：单例**

**被激活的Activity在任务栈绝对有且仅有一个，处于该模式下的Activity将独立存在于一个专属的任务栈中。**

**通过 AndroidManifest.xml中的<activity>节点的 android:launchMode 属性可以配置以上四种启动模式。**

**2.描述一下Service的生命周期。**

**Service 有绑定模式和非绑定模式，以及这两种模式的混合使用方式。不同的使用方法生命周期方法也不同。**

**非绑定模式： 当第一次调用 startService 的时候执行的方法依次为 onCreate()、 onStartCommand()， （onStart()）**

**当 Service 关闭的时候调用 onDestory 方法。**

**绑定模式：第一次 bindService（）的时候，执行的方法为 onCreate()、onBind(）解除绑定的时候会执行**

**onUnbind()、onDestory()。**

**上面的两种生命周期是在相对单纯的模式下的情形。我们在开发的过程中还必须注意 Service 实例只会有一个，**

**也就是说如果当前要启动的 Service 已经存在了那么就不会再次创建该 Service 当然也不会调用 onCreate（）方法。**

**3.Service的onStartCommand方法有几种返回值？各代表什么意思？**

**有四种返回值，不同值代表的意思如下：**

**START\_STICKY：如果 service 进程被 kill 掉，保留 service 的状态为开始状态，但不保留递送的 intent 对象。随**

**后 系 统 会 尝 试 重 新 创 建 service ， 由 于 服 务 状 态 为 开 始 状 态 ， 所 以 创 建 服 务 后 一 定 会 调 用**

**onStartCommand(Intent,int,int)方法。 如果在此期间没有任何启动命令被传递到service， 那么参数Intent将为null。**

**START\_NOT\_STICKY：“非粘性的”。使用这个返回值时，如果在执行完 onStartCommand 后，服务被异常 kill**

**掉，系统不会自动重启该服务。**

**START\_REDELIVER\_INTENT：重传 Intent。使用这个返回值时，如果在执行完 onStartCommand 后，服务被异**

**常 kill 掉，系统会自动重启该服务，并将 Intent 的值传入。**

**START\_STICKY\_COMPATIBILITY：START\_STICKY 的兼容版本，但不保证服务被 kill 后一定能重启。**

**4什么是ANR 如何避免它？**

**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。用户可以选择让程序继续运行，但是，他们在使用你**

**的应用程序时，并不希望每次都要处理这个对话框。因此，在程序里对响应性能的设计很重要，这样，系统不会显示**

**ANR 给用户。**

**不同的组件发生 ANR 的时间不一样，主线程（Activity、Service）是 5 秒，BroadCastReceiver 是 10 秒。**

**解决方案：**

**将所有耗时操作，比如访问网络，Socket 通信，查询大量 SQL 语句，复杂逻辑计算等都放在子线程中去，然后**

**通过 handler.sendMessage、runonUITread、AsyncTask 等方式更新 UI。无论如何都要确保用户界面操作的流畅度。**

**如果耗时操作需要让用户等待，那么可以在界面上显示进度条。**