1. activity的生命周期（概率80%）

onCreate，onStart，onResume，onPause，onStop，onDestroy；onRestart;(6+1)

10.framgment生命周期？（概率30%）

MyFragment onAttach() 粘贴到activity上

MyFragment onCreate() fragment创建

MyFragment onCreateView() fragment创建自己的视图

MainActivity onCreate()

MyFragment onActivityCreated() 可以处理fragment数据的初始化

MainActivity onStart()

MyFragment onStart()

MainActivity onResume()

MyFragment onResume()

按后退键

MyFragment onPause()

MainActivity onPause()

MyFragment onStop()

MainActivity onStop()

MyFragment onDestoryView() 销毁掉自己的视图

MyFragment onDestory()

MyFragment onDetach() 解除和activity的关系

MainActivity onDestory()

1.常用的存储方式有哪些？（概率50%）

1: Shared Preferences(**SP存储**)(以键值对的形式存储私有的,简单数据类型)

2: Internal Storage(**内存存储**)(私有的数据存储在设备的内存里面)

3: External Storage(**外部存储**)(将公共的数据存储在SD卡里面)

4: SQLite Databases(**数据库存储**)(将数据存储在私有的数据库里面)

(数据库存储包括SQLite和ContentProvider,其实ContentProvider的底层就是在操作数据库)

5: Network Connection(**网络存储**)(存储数据在web端,在你自己的数据服务器上)

2.安卓中动画有几类，他们的特点和区别？（概率80%）

1: Property Animation(**属性动画**别加入在3.0)

2: View Animation (**补间动画**分为:透明,旋转,位移,缩放)

注意:补间动画是假动画,点击区域不变.

3: Drawable Animation(**帧动画**帧动画类似于放电影，通过播放已经排列放好的图片来实现。)

3.handler机制原理（概率100%）

为什么引进？UI控件不是线程安全的,子线程不能更新UI,所以安卓引入了Handler机制,

为什么系统不对UI控件的访问加锁机制,缺点有两个

1、加上锁机制会让UI访问的逻辑变的复杂,

2、其次锁机制会降低UI访问的效率,因为锁机制会阻塞某些线程的执行,

Handler主要用于线程间的通信。

一个Handler允许发送和处理Message和Runable对象，

UI主线程会自动分配一个**Looper**（消息轮询器），每个Looper中封装着**MessageQueue**（消息队列），遵循先进先出原则。Looper负责不断的从自己的消息队列里取出队头的任务或消息执行。

一般是在子线程执行完耗时操作之后，通过Handler的sendMessage或post方法将Message和Runable对象传递给MessageQueue，而且在这些对象离开MessageQueue时，Handler负责执行他们（用到handleMessage方法，主要执行刷新UI的代码）。

其中Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象提供额外的两个int域和一个Object域。

4除了handler和子线程，还有一个处理线程的是什么，主要方法是什么？（概率60%）

**AsynTask**

doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如访问网络。该方法必须重写。

onPostExecute(Result) 这个方法运行于UI主线程，在doInBackground(Params…)方法执行后调用，该方法用于接收后台任务执行后返回的结果，刷新UI显示

6线程之间的通信方式（如何避免ANR）（概率70%）

（1）. AsyncTask

（2.）子线程 + handler

5.tcp和udp区别.（概率80%）

**TCP---(长链接)(管发管收,数据安全)**传输控制协议,提供的是面向连接、可靠的字节流服务,传输数据前经过“**三次握手**”建立连接，保证数据传输的可靠性，但效率比较低。一般用于对于**数据传输安全性较高**的场合。

**UDP---(短连接,只管发,不管收)**用户数据报协议，是一个简单的面向数据报的运输层协议，面向无连接。UDP不提供可靠性，数据传输可能发生错序，丢包，但效率较高。一般用于对于**实时性（聊天）要求较高**的场合。

11·图片异步加载怎么做？（概率80%）

三级缓存:

1:首先从强引用中获取(Lrucache),如果强引用中没有,再去软引用中获取(SoftReference),如果软引用再去SD卡获取(SD卡被Dis Lrucache替代掉),如果这三级缓存中都没有,就开启网络去请求图片,请求回来的图片,在加入到强引用中(Lrucache),如果强引用已经满了,这时候,强引用通过Lrucache算法,删除掉一些最近不常用的图片,这时候咱们把这些图片放到软引用里面,(当内存不足到时候软引用就会被被垃圾回收机制回收),如果垃圾回收机制没有,软引用也满了,将不常用的图片移除掉,同时加入到SD卡,(现在被Dis Lrucache替代)

(key 一般使用的是网络请求图片的url,来保证唯一性,一般情况下我会做一下MD5加密,这样做的目的是为了保证不会出现非法字符)

现在SoftReference已经被替代掉,现在我们只使用两级缓存,在加网络, Lrucache, Dis Lrucache

LRC算法:最近最少使用的(内部如何实现的)

强引用: Lrucache

软引用: SoftReference

弱引用:weak(随时被垃圾回收机制回收)

虚引用:phantomreference

可定义图片异步加载工具类，核心方式实现思路如下：

1.先从内存缓存（Map<String,SoftReference<Bitmap>>中获取图片显示

2.获取不到的话从本地SD卡里获取并显示

3.都获取不到的话通过子线程从网络加载图片并保存到内存及SD卡中并通过handler显示

9.安卓内存的优化？（概率90%）

[http://blog.csdn.net/awangyunke/article/details/20380719](http://blog.csdn.net/awangyunke/article/details/20380719" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

handler内存泄露（thread， AsyncTask)

[http://www.linuxidc.com/Linux/2013-12/94065.htm](http://www.linuxidc.com/Linux/2013-12/94065.htm" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

context内存泄露

[http://blog.sina.com.cn/s/blog\_5da93c8f0102w86x.html](http://blog.sina.com.cn/s/blog_5da93c8f0102w86x.html" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

Bitmap内存泄露

1）静态变量引起内存泄露(内存泄露多了就变成溢出)

在代码优化的过程中，我们需要对代码中的静态变量特别留意。静态变量是类相关的变量，它的生命周期是从这个类被声明，到这个类彻底被垃圾回收器回收才会被销毁。所以，一般情况下，静态变量从所在的类被使用开始就要一直占用着内存空间，直到程序退出。如果不注意，静态变量引用了占用大量内存的资源，造成垃圾回收器无法对内存进行回收，就可能造成内存的浪费

2）使用Application(单例模式)的Context

在Android中，Application Context的生命周期和应用的生命周期一样长，而不是取决于某个Activity的生命周期。如果想保持一个长期生命的对象，并且这个对象需要一个 Context，就可以使用Application对象。可以通过调用Context.getApplicationContext()方法或者 Activity.getApplication()方法来获得Application对象。

3）及时关闭资源

Cursor是Android查询数据后得到的一个管理数据集合的类。正常情况下，如 果我们没有关闭它，系统会在回收它时进行关闭，但是这样的效率特别低。如果查询得到的数据量较小时还好，如果Cursor的数据量非常大，特别是如果里面 有Blob信息时，就可能出现内存问题。所以一定要及时关闭Cursor。

4）使用Bitmap及时调用recycle() // 把 重复循环的方法设置 为 null，释放内存

前面的章节讲过，在不使用Bitmap对象时，需要调用recycle()释放内存，然后将它设置为null。虽然调用recycle()并不能保证立即释放占用的内存，但是可以加速Bitmap的内存的释放。

在代码优化的过程中，如果发现某个Activity用到了Bitmap对象，却没有显式的调用recycle()释放内存，则需要分析代码逻辑，增加相关代码，在不再使用Bitmap以后调用recycle()释放内存。

5）对Adapter进行优化

下面以构造ListView的BaseAdapter为例说明如何对Adapter进行优化。

@软引用和弱引用。

如果一个对象只具有软引用，那么如果内存空间足够，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。软引用可以和一个引用队列（ReferenceQueue）联合使用，如果软引用所引用的对象被垃圾回收，Java虚拟机就会把这个软引用加入到与之关联的引用队列中。

如果一个对象只具有弱引用，那么在垃圾回收器线程扫描的过程中，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。弱引用也可以和一个引用队列（ReferenceQueue）联合使用，如果弱引用所引用的对象被垃圾回收，Java虚拟机就会把这个弱引用加入到与之关联 的引用队列中。

弱引用与软引用的根本区别在于：只具有弱引用的对象拥有更短暂的生命周期，可能随时被回收。而只具有软引用的对象只有当内存不够的时候才被回收，在内存足够的时候，通常不被回收。

6）UI优化

在Android应用开发过程中，屏幕上控件的布局代码和程序的逻辑代码通常是分开 的。界面的布局代码是放在一个独立的xml文件中的，这个文件里面是树型组织的，控制着页面的布局。通常，在这个页面中会用到很多控件，控件会用到很多的 资源。Android系统本身有很多的资源，包括各种各样的字符串、图片、动画、样式和布局等等，这些都可以在应用程序中直接使用。这样做的好处很多，既 可以减少内存的使用，又可以减少部分工作量，也可以缩减程序安装包的大小。