### 1、理解Activity，View,Window三者关系

### 2、SparseArray和HashMap区别

### 3、String a = "chenssy"; String b = "chenssy"; String c = new String("chenssy");说下a,b,c的关系

### 4、谈一下Android 8.0行为变更

### 5、浅谈下IPC，IPC有哪些方式

### 6、现有多个线程任务正在执行，所有任务执行完成后需要回调，你会怎么做

### 7、ViewGroup事件分发

### 8、View绘制机制的流程

### 9、是否使用过IntentService，作用是什么

### 10、如何做SQLite升级？

### 11、Java和JS的相互调用怎么实现

### 12、Socket建立网络连接的步骤

### 13、TCP和UDP的区别

### 14、你了解的排序算法

### 15、谈一下DVM和ART

### 16、了解反编译吗

### 17、Activity的onNewIntent调用机制

### 18、adb常用命令行了解吗

### 19、了解反射吗

### 20、Bitmap内存占用怎么计算的

### 21、SurfaceView 和 View 区别

### 22、android在子线程更新UI主线程有哪些方法

### 23、了解git吗，常用命令有哪些

### 24、进程和线程的区别

### 25、Serializable 和Parcelable 的区别

### 26、Android中性能优化

### 附答案：

1、

Activity就像工匠，Window就像是窗户，View就像是窗花，LayoutInflater像剪刀，Xml配置像窗花图纸。  
Android根据他们不同的职能让他们各斯其活，同时也相互配合展示给我们灵活、精致的界面。

一个应用程序里所有的界面展示都来自于Activity

View如何跟Activity关联起来的？  
其实View并不是直接跟Activity关联起来的，而是通过Window这个中间人。如前面所说，View只是窗花，Window才是直接关联到Activity上的。那么：  
View如何跟Window关联起来呢？

下面先了解一下Window，就可以理解这个问题了

Window如何跟Activity关联？  
每一个Activity都包含了唯一一个PhoneWindow，这个就是Activity根Window（之所以是说根Window是因为在它上面可以增加更多其他的Window，例如：弹出框（dialog））

那么，PhoneWindow如何跟Activity关联起来的呢？  
来个最简单的，setContentView其实就让View与Window关联，Window跟Activity关联起来了。

2、

HashMap内部存储结构是使用哈希表的拉链结构（数组+链表）

ArrayMap是一个<key,value>映射的数据结构，它设计上更多的是考虑内存的优化，内部是使用两个数组进行数据存储，一个数组记录key的hash值，另外一个数组记录Value值，它和SparseArray一样，也会对key使用二分法进行从小到大排序，在添加、删除、查找数据的时候都是先使用二分查找法得到相应的index，然后通过index来进行添加、查找、删除等操作，所以，应用场景和SparseArray的一样，如果在数据量比较大的情况下，那么它的性能将退化至少50%。

3、

a、b和字面上的chenssy都是指向JVM字符串常量池中的"chenssy"对象，他们指向同一个对象。

所以a==b

String c = new String("chenssy");

new关键字一定会产生一个对象chenssy（注意这个chenssy和上面的chenssy不同），同时这个对象是存储在堆中。所以上面应该产生了两个对象：保存在栈中的c和保存堆中chenssy。但是在Java中根本就不存在两个完全一模一样的字符串对象。故堆中的chenssy应该是引用字符串常量池中chenssy。所以c、chenssy、池chenssy的关系应该是：c--->chenssy--->池chenssy

所以a==b!=c

4、

后台执行限制，如应用在后台使用 [startService()](https://developer.android.com/reference/android/content/Context.html?hl=zh-cn" \l "startService(android.content.Intent)) 函数，则该函数将引发一个 [IllegalStateException](https://developer.android.com/reference/java/lang/IllegalStateException.html?hl=zh-cn)。

提醒窗口只能使用 [TYPE\_APPLICATION\_OVERLAY](https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html?hl=zh-cn#TYPE_APPLICATION_OVERLAY) 窗口类型

findViewById() 签名变更

等等自由发挥，主要考察对android官方文档的掌握

5、

通过Intent在Activity、Service或BroadcastReceiver间进行进程间通信，可通过Intent传递数据

AIDL方式

Messenger方式

利用ContentProvider

Socket方式

基于文件共享的方式

6、

CountDownLatch与栅栏CyclicBarrier

7、

按顺序：

**public boolean dispatchTouchEvent(MotionEvent event)**

**public boolean onInterceptTouchEvent(MotionEvent event)**

**public boolean onTouchEvent(MotionEvent event)**

8、

主要三大流程：

onMeasure 测量组件本身的大小

onLayout 确定组件在视图中的位置

onDraw 根据位置和大小，将组件画出来

9、

生成一个默认的且与主线程互相独立的工作者线程来执行所有传送至 onStartCommand() 方法的Intetnt。

生成一个工作队列来传送Intent对象给你的onHandleIntent()方法，同一时刻只传送一个Intent对象，这样一来，你就不必担心多线程的问题。在所有的请求(Intent)都被执行完以后会自动停止服务，所以，你不需要自己去调用stopSelf()方法来停止。

该服务提供了一个onBind()方法的默认实现，它返回null。

提供了一个onStartCommand()方法的默认实现，它将Intent先传送至工作队列，然后从工作队列中每次取出一个传送至onHandleIntent()方法，在该方法中对Intent对相应的处理

10、

数据库升级增加表和删除表都不涉及数据迁移，但是修改表涉及到对原有数据进行迁移。升级的方法如下所示：

将现有表命名为临时表。

创建新表。

将临时表的数据导入新表。

删除临时表。

如果是跨版本数据库升级，可以由两种方式，如下所示：

逐级升级，确定相邻版本与现在版本的差别，V1升级到V2,V2升级到V3，依次类推。

跨级升级，确定每个版本与现在数据库的差别，为每个case编写专门升级大代码。

11、

Java调用Js主要方法loadUrl

Js调java主要js注入，在java里面注册好接口方法，js直接调用就行。

12、

建立Socket连接至少需要一对套接字，其中一个运行与客户端--ClientSocket，一个运行于服务端--ServiceSocket

1、服务器监听：服务器端套接字并不定位具体的客户端套接字，而是处于等待连接的状态，实时监控网络状态，等待客户端的连接请求。

2、客户端请求：指客户端的套接字提出连接请求，要连接的目标是服务器端的套接字。注意：客户端的套接字必须描述他要连接的服务器的套接字，

指出服务器套接字的地址和端口号，然后就像服务器端套接字提出连接请求。

3、连接确认：当服务器端套接字监听到客户端套接字的连接请求时，就响应客户端套接字的请求，建立一个新的线程，把服务器端套接字的描述

发给客户端，一旦客户端确认了此描述，双方就正式建立连接。而服务端套接字则继续处于监听状态，继续接收其他客户端套接字的连接请求。

13、

tcp是面向连接的，由于tcp连接需要三次握手，所以能够最低限度的降低风险，保证连接的可靠性。

udp 不是面向连接的，udp建立连接前不需要与对象建立连接，无论是发送还是接收，都没有发送确认信号。所以说udp是不可靠的。

由于udp不需要进行确认连接，使得UDP的开销更小，传输速率更高，所以实时行更好。

14、

冒泡，快排等等。

15、

ART能够把应用程序的字节码转为机器码，是Android所使用的一种新的虚拟机，与Dalvik不同在于：Dalvik采用的是JIT技术而ART采用Ahead-of-time(AOT)技术。ART同时也改善了性能，垃圾回收机制，应用程序出错以及性能分析

ART引入了AOT这种预编译技术，在应用程序安装的过程中，ART就已经将所有的字节码重新编译成机器码。应用程序运行过程中无需进行实时的编译工作，只需要进行直接调用，因此ART极大的提高了应用程序的运行效率，同时也减少了手机的电量消耗，在垃圾回收机制等机制上也有了较大的提升。

相对于Dalvik虚拟机模式。ART模式下Android应用程序的安装需要消耗更多的时间，同时也会占用更大的存储空间（指内部存储，用于存储编译后的代码），但节省了很多dalvik虚拟机用于实时编译的时间

16、

Dex2jar，ApkTool，jd-gui

17、

该方法被启动模式设置为“singleTop、singleTask”的Activity回调，或者当通过设置Intent.FLAG\_ACTIVITY\_SINGLE\_TOP、Intent.FLAG\_ACTIVITY\_SINGLE\_TASK的Intent启动Activity时被回调。也就是说，只要栈顶Activity被重新启动时没有重建一个新的Activity实例，而是依然使用栈顶的Activity实例，那么onNewIntent方法就会被回调。

18、

adb，即Android Debug Bridge，可以用来管理或者手机模拟器的状态。

查看当前连接的设别

adb devices

安装和卸载apk程序

安装apk：adb install apk\_name

卸载apk：adb uninstall

上传和下载文件

上传文件：adb push <本地路径> <远程路径>

下载文件：adb pull <远程路径>

19****、****

反射就是动态加载对象，并对对象进行剖析。在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法，这种动态获取信息以及动态调用对象方法的功能成为Java反射机制。

优点：可以动态的创建对象和编译，最大限度发挥了java的灵活性。

缺点：对性能有影响。使用反射基本上一种解释操作，告诉JVM我们要做什么并且满足我们的要求，这类操作总是慢于直接执行java代码。

****20、****

 Bitmap图片在加载到内存的时候是按照：宽\*高\*像素点位数来计算的。你可以把图片看成是由width行、height列的矩阵组成，每一个矩阵元素代表一个像素点，每一个像素点都是1byte整数倍的数据，这个数据越大，表示的色彩就越丰富,图片的显示质量就越高。Bitmap中有一个枚举类Config用来配置图片的压缩格式，代表每个像素是用多大的数据来存储的，数值越大能够存储的颜色信息就越多，也就越丰富，显示效果也就越好。Config.ALPHA\_8是1 byte，Config.RGB\_565和Config.ARGB\_4444都是2 bytes，Config.RGB\_565没有Alpha值所以多用来配置没有透明度的图片，Config.ARGB\_8888是4 bytes，一般图片都是按照这个来配置的

21、

surfaceView是在一个新起的单独线程中可以重新绘制画面，而View必须在UI的主线程中更新画面。

在UI的主线程中更新动画，时间一旦太长就会出现问题，surfaceView 在新的线程中更新画面所以不会阻塞你的UI主线程。

View 主要适用于主动更新的情况，而 surfaceView 主要适用于被动更新，例如频繁的刷新。

View 在主线程中对画面进行刷新，而 surfaceView 通常会通过一个子线程来进行页面的刷新

View 在绘图时没有使用双缓冲机制，而 surfaceView 在底层实现机制上就已经实现了双缓冲机制。

总结就是，如果你的自定义 View 需要频繁刷新，或者刷新时数据处理量很大，考虑用 SurfaceView 来替代 View。

22、

方法一：通过handler.sendMessage，更新UI主线程

方法二：通过handler.post，handler.postDelayed更新UI主线程

方法三：使用runOnUiThread，更新UI主线程

方法四：通过view.post，更新UI主线程

方法五：通过AsyncTask,更新UI主线程

23、

git init //初始化本地git环境

git clone XXX//克隆一份代码到本地仓库

git pull //把远程库的代码更新到工作台

git pull --rebase origin master //强制把远程库的代码跟新到当前分支上面

git add . //把本地的修改加到stage中

git commit -m 'comments here' //把stage中的修改提交到本地库

git push //把本地库的修改提交到远程库中

git branch -r/-a //查看远程分支/全部分支

git checkout master/branch //切换到某个分支

git checkout -b test //新建test分支

24、

进程是cpu资源分配的最小单位，线程是cpu调度的最小单位。  
进程之间不能共享资源，而线程共享所在进程的地址空间和其它资源。  
一个进程内可拥有多个线程，进程可开启进程，也可开启线程。  
一个线程只能属于一个进程，线程可直接使用同进程的资源,线程依赖于进程而存在。

25、

Serializable Java 序列化接口 在硬盘上读写 读写过程中有大量临时变量的生成，内部执行大量的i/o操作，效率很低。  
Parcelable Android 序列化接口 效率高 使用麻烦 在内存中读写（AS有相关插件 一键生成所需方法） ，对象不能保存到磁盘中

26、

由于手机硬件的限制，内存和CPU都无法像pc一样具有超大的内存，Android手机上，过多的使用内存，会容易导致oom，过多的使用CPU资源，会导致手机卡顿，甚至导致anr。我主要是从一下几部分进行优化：

布局优化，绘制优化，内存泄漏优化，响应速度优化，listview优化，bitmap优化，线程优化。

布局优化：

工具 hierarchyviewer，解决方式：

1、删除无用的空间和层级。

2、选择性能较低的viewgroup，如Relativelayout，如果可以选择Relativelayout也可以使用LinearLayout,就优先使用LinearLayout，因为相对来说Relativelayout功能较为复杂，会占用更多的CPU资源。

3、使用标签<include/>重用布局，<Merge/>减少层级，<viewStub/>进行预加载，使用的时候才加载。

绘制优化：

绘制优化指view在ondraw方法中避免大量的耗时操作，由于ondraw方法可能会被频繁的调用。

1、ondraw方法中不要创建新的局部变量，ondraw方法被频繁的调用，很容易引起GC。

2、ondraw方法不要做耗时操作。

内存优化：

参考内存泄漏。

响应优化：

主线程不能做耗时操作，触摸事件5s,广播10s，service20s。

listview优化：

1、getview方法中避免耗时操作。

2、view的复用和viewholder的使用。

3、滑动不适合开启异步加载。

4、分页处理数据。

5、图片使用三级缓存。

Bitmap优化：

1、等比例压缩图片。

2、不用的图片，及时recycler掉

线程优化：

线程优化的思想是使用线程池来管理和复用线程，避免程序中有大量的Thread，同时可以控制线程的并发数，避免相互抢占资源而导致线程阻塞。

其他优化：

1、少用枚举，枚举占用空间大。

2、使用Android特有的数据结构，如SparseArray来代替hashMap。

3、适当的使用软引用和弱引用。

注：此题目答案比较开放，说出上面的数条即可。