**JAVA基础**

**1.**haspMap与arraylist、linkedlist、hashTable的区别  
答:1、ArrayList是JDK1.2的集合类并且线程不安全，采用的是数组形式来保存对象的，这种方式将对象放在连续的位置中，所以最大的缺点就是插入删除时非常麻烦

2、LinkedList是链表结构的集合，采用的将对象存放在独立的空间中，而且在每个空间中还保存下一个链接的索引 但是缺点就是查找非常麻烦 要丛第一个索引开始

* LinkedList在中间或前面增加或删除数据时效率比ArrayList高。
* LinkedList在最后添加或删除数据时效率比ArrayList低。
* 遍历数据时ArrayList效率高于LinkedList。

3、HashMap不是线程安全的

           hastmap是一个接口 是map接口的子接口，是将键映射到值的对象，其中键和值都是对象，并且不能包含重复键，但可以包含重复值。HashMap允许null key和null value，而hashtable不允许。

 4   HashTable是线程安全的一个Collection。

 HashMap是Hashtable的轻量级实现（非线程安全的实现），他们都完成了Map接口，主要区别在于HashMap允许空（null）键值（key）,由于非线程安全，效率上可能高于Hashtable。 HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。 HashMap把Hashtable的contains方法去掉了，改成containsvalue和containsKey。因为contains方法容易让人引起误解。 Hashtable继承自Dictionary类，而HashMap是Java1.2引进的Map interface的一个实现。 最大的不同是，Hashtable的方法是Synchronize的，而HashMap不是，在多个线程访问Hashtable时，不需要自己为它的方法实现同步，而HashMap 就必须为之提供外同步。 Hashtable和HashMap采用的hash/rehash算法都大概一样，所以性能不会有很大的差

### 简述一下你了解的设计模式？

所谓设计模式，就是一套被反复使用的代码设计经验的总结（情境中一个问题经过证实的一个解决方案）。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。设计模式使人们可以更加简单方便的复用成功的设计和体系结构。将已证实的技术表述成设计模式也会使新系统开发者更加容易理解其设计思路。

Abstract Factory（抽象工厂模式），Builder（建造者模式），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 Responsibility（责任链模式）。

PS:工厂模式：工厂类可以根据条件生成不同的子类实例，这些子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（多态方法）。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。

-单例模式（Android必须要问）：

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:（Android单例模式较多）   
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; }

**3.**Java中有几种类型的流？以及常见的实现类都有哪些？  
答:首先应该从两个角度来看：(Android答出这些就行 )

 从输入输出方面来讲：       Java中有输入流和输出流

 从流的编码方式上来讲：    Java中有字节流和字符流

下面主要讲解从编码方式来看java几种常见的类型的流。

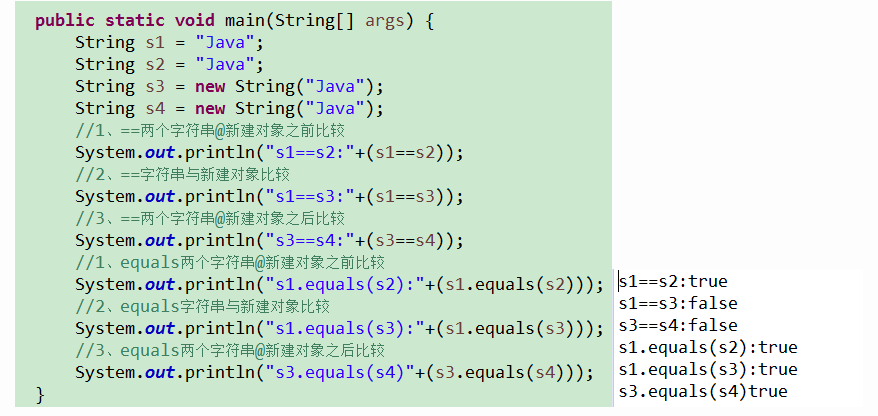
  对于字节流而言:主要继承的抽象类为 InputStream和OutputStream

  对于字符流而言:主要继承的抽象类为 InputStreamReader和OutputStreamReder

**4.**Java异常机制用到的几个关键字？并做简要介绍。

Java异常机制用到的几个关键字：**try、catch、finally、throw、throws。**  
• **try**        -- 用于监听。将要被监听的代码(可能抛出异常的代码)放在try语句块之内，当try语句块内发生异常时，异常就被抛出。  
• **catch**   -- 用于捕获异常。catch用来捕获try语句块中发生的异常。  
• **finally**  -- finally语句块总是会被执行。  
• **throw**   -- 用于抛出异常。  
• **throws** -- 用在方法签名中，用于声明该方法可能抛出的异常。

## **5.**[equals和==的区别小结](https://www.cnblogs.com/Eason-S/p/5524837.html)。

* == 比较的是变量(栈)内存中存放的对象的(堆)内存地址，用来判断两个对象的地址是否相同，即是否是指相同一个对象。比较的是真正意义上的指针操作。
* equals用来比较的是两个对象的内容是否相等，由于所有的类都是继承自java.lang.Object类的，所以适用于所有对象，如果没有对该方法进行覆盖的话，调用的仍然是Object类中的方法，而Object中的equals方法返回的却是==的判断  
  

#### Android面试题

##### Rxjava相关面试题。

##### Rxjava主要作用就是用来处理异步，当你的业务需要访问数据库，访问网络，或者任何耗时的操作，都可以借助Rxjava来实现，****通过观察者模式实现的****。

为什么Rxjava要使用观察者模式：

模块之间划定了界限，降低模块耦合性、提高代码的可维护性和重用性。

##### RxJava 有四个基本概念：

##### ****Observable (可观察者，即被观察者)****、 ****Observer (观察者)****、 ****subscribe (订阅)****、****事件。****

##### Rxjava常用的的回调方法有三种：  - onNext：完成队列中的一个事件  - onComplete：完成队列中所有的事件  - onError：事件发生错误时，并且后续的事件终止。

### **Rxjava基本实现方法：**

**1、创建Observer**

**Observer是观察者，当被观察者状态发生变化的时候，他会收到相应的事件，使用者可以根据不同的事件进行不同的处理。**

其实，除了使用Observer以外，Rxjava还有个Subscriber。这个是实现了Observer的抽象类，里面对Observer进行了一些扩展。

**2、创建Observable**

**使用create方法创建Observable（被观察者），然后call方法会被自动调用，在call方法内部定义事件的回调的行为。**

**其实这段代码中，当执行了onComplete方法之后，就不会在往下执行了，也就是说onError方法不会被调用，因为事件已经完全执行完成，就会停止执行之后的事件。**

**3、**Subscribe(订阅)

**执行回调操作**

##### Android 消息机制相关问题。

1. Handle机制 主要涉及的类有哪些

**Looper**

负责从MessageQueue中获取消息及将消息分发到对应的Handler.

**Thread**

任务执行的线程环境

**ThreadLocal**

负责保存Thread关联的Looper.

**ThreadLocalMap**

ThreadLocal中具体保存数据的类.内部使用Entry数组.来保存数据.

**MessageQueue**

保存Message的容器,内部实现为链表.

**Handler**

向MessageQueue中插入消息,以及对收到的Looper分发的消息的处理

**Message**

消息载体.主要如下属性

**callback**

任务的runnable对象

**when**

message需要被dispatch的执行时间

**obj**

message中传递的对象

1. Looper、MessageQueue、Handler三者关系 自己的理解。
2. Looper是如何分发message的?
3. 为什么一个线程只有一个Looper、只有一个MessageQueue？。
4. 如何获取当前线程的Looper？是怎么实现的？（理解ThreadLocal）

###### 部分解决答案：

3、当looper.loop方法中获取到message时,会调用message.target.dispatch方法.

其中message.target属性是在Handler内部调用Handler.enqueueMessage方法时,将当前调用方法的handler对象设置到Message中的.

通过message.target.dispatchMessage方法,

将该message的执行环境切换到了该handler对应的thread中.

handler可以实现handleMessage方法来处理消息.

调用完handler.dispatchMessage方法后,则会将该消息通过recycleUnchecked方法,对message进行回收.

在Message.recycleUnchecked方法中会重置该message对象(将message对象相关属性置空),

并将该对象添加到Message缓存区中.

加入缓存去的过程也就是将该对象加入到缓存区链表中.加入缓冲区的message后面可通过Message.obtain方法来获取

5.Looper在消息机制里扮演着消息循环的角色，它不停的从消息队列里查看是否有新消息，如果有，则立即处理新消息，没有则一直阻塞在那里。当需要为一个线程创建一个Looper的时候，需要调用Looper的两个静态方法就可以给这个线程创建一个Looper对象了，这两个方法是Looper.prepare();和Looper.loop();

##### 3、Android 中的动画有哪几类，它们的特点和区别是什么。（这个是必须要会的 协同APP里大量动画）（那自己手机里特效问他）

视图动画，或者说补间动画。只是视觉上的一个效果，实际view属性没有变化，性能好，但是支持方式少。

属性动画，通过变化属性来达到动画的效果，性能略差，支持点击等事件。android 3.0

帧动画，通过drawable一帧帧画出来。

Gif动画，原理同上，canvas画出来。

##### 4、子线程发消息到主线程进行更新 UI 有哪些方法。（知道要回答3个以上）

1、AsyncTask 2 、runinUiThread 3、EventBus 4、handle.sendMessge()等操作 5、广播 6、view.post

##### 5、AsyncTask的工作原理（你对AsyncTask有所了解吗？）

AsyncTask是Android本身提供的一种轻量级的异步任务类。它可以在线程池中执行后台任务，然后把执行的进度和最终的结果传递给主线程更新UI。

实际上，AsyncTask内部是封装了Thread和Handler。虽然AsyncTask很方便的执行后台任务，以及在主线程上更新UI，

但是，AsyncTask并不合适进行特别耗时的后台操作，对于特别耗时的任务，个人还是建议使用线程池。

AsyncTask提供有4个核心方法：

1、onPreExecute():该方法在主线程中执行，在执行异步任务之前会被调用，一般用于一些准备工作。

2、doInBackground(String... params):这个方法是在线程池中执行，此方法用于执行异步任务。在这个方法中可以通过publishProgress方法来更新任务的进度，publishProgress方法会调用onProgressUpdate方法，另外，任务的结果返回给onPostExecute方法。

3、onProgressUpdate(Object... values):该方法在主线程中执行，主要用于任务进度更新的时候，该方法会被调用。

4、onPostExecute(Long aLong)：在主线程中执行，在异步任务执行完毕之后，该方法会被调用，该方法的参数及为后台的返回结果。

除了这几个方法之外还有一些不太常用的方法，如onCancelled(),在异步任务取消的情况下，该方法会被调用。

备选面试题：

1. 请介绍下Android的数据存储方式。
2. activity的启动模式有哪些

“standard” (默认) “singleTop” “singleTask”“ singleInstance”

1. activity的启动模式有哪些

“standard” (默认) “singleTop” “singleTask”“ singleInstance”

1. 面试项目里 框架 如何封装问答
2. BItmap的三级缓存的理解 加载 Bitmap 的方法（不借助第三方）：

首次加载的时候通过网络加载，获取图片，然后保存到内存和 SD 卡中。

之后运行 APP 时，优先访问内存中的图片缓存。

如果内存没有，则加载本地 SD 卡中的图片。

内存缓存、本地缓存（磁盘缓存）、网络缓存。

•decodeFile：从文件系统加载 Bitmap 对象

•decodeResource：从资源文件中加载 Bitmap

•decodeStream：从输入流加载 Bitmap

•decodeByteArray：从字节数组中加载 Bitmap

decodeFile 和 decodeResource 间接调用了 decodeStream 方法。

6、性能优化、内存优化

7、从那些角度可以减少APK体积的（自己理解）

1.尽量少的依赖第三方sdk

2.删除不需要的代码跟没有被引用的图片

3.压缩图片体积

4.混淆apk

5.优化代码逻辑，减少Assets文件夹内容

6.使用progard进行代码混淆，减少一些类和方法名的长度。同时progard也可以删除一些无用的方法，（这点加分）。

8、MVC模式和MVP模式的区别（自己理解）

1. MVC的所有通信都是单向的。

　　　　2. view传送指令到controller(用户也可以直接将指令传到controller)。

　　　　3. controller完成业务逻辑后要求model改变状态。

　　　　4. model将新的数据发送到view，用户得到反馈。

**MVP模式**将Controller改名为Presenter，同时改变了通信方向。

1. 各部分之间的通信都是双向的。

　　　　2. View与Model不发生联系，都通过Presenter传递

　　　　3.  View非常薄，不部署任何业务逻辑，称为“被动视图”，即没有任何主动性，而Presenter非常厚，所有逻辑都

　　　　　　部署在这里。

8、Android 屏幕适配（自己的理解）

屏幕适配的方式：xxxdpi， wrap\_content,match\_parent. 获取屏幕大小，做处理。

dp来适配屏幕，sp来确定字体大小

drawable-xxdpi, values-1280\*1920等 这些就是资源的适配。

wrap\_content,match\_parent, 这些是view的自适应

weight，这是权重的适配。

8、Android有哪些Context的相关类？（自己的理解）

1. Context抽象类
2. ContextImpl：Context的实现类
3. ContextWrapper：Context的包装类（内部是ContextImpl），Applicaiton、Activity、Service都间接或者直接继承自ContextWrapper