**提示：**

**Java类的：**

进程是具有一定独立功能的程序。

进程要空间线程共享内存。

**线程池组成：**

1. 线程池管理器：创建线程池
2. 工作线程：线程池中的线程。
3. 任务接口：规定任务的入口
4. 任务队列：存放没有处理的线程

**Sleep和wait的区别：**

Sleep和wait都由锁控制同步访问，sleep没有释放锁，而wait释放了

Sleep要捕获异常wait不要

Sleep睡眠时保持对象锁，wait睡眠时释放对象锁。

Wait必须放在同步块中不然会报非法监控的异常

**Java引用方式：**

1. 强引用：把对象赋给变量
2. 软引用：实现内存敏感的高速缓存
3. 弱引用：描述非必须对象的
4. 虚引用：不影响对象的生命周期

**Java同步方法：**

Synchronized（同步）关键字

同步代码块

重入锁

局部变量

**Java内存模型：**

GC模型

并发模型

**Java设计模式：**

1. **单例模式**

意图：保证一个类仅有一个实例，并提供一个访问它的全局访问点。

**主要解决：一个全局使用的类频繁地创建与销毁。**

**何时使用：当您想控制实例数目，节省系统资源的时候。**

**如何解决：判断系统是否已经有这个单例，如果有则返回，如果没有则创建。**

关键代码：构造函数是私有的。

1. **原型模式：对象的拷贝**

意图：用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。

**主要解决：在运行期建立和删除原型。**

何时使用：当一个类的实例只能有几个不同状态组合中的一种时。

**如何解决：利用已有的一个原型对象，快速地生成和原型对象一样的实例。**

1. **建造者模式**

意图：将一个复杂的构建与其表示相分离，使得同样的构建过程可以创建不同的表示。

主要解决：一些基本部件不会变，而其组合经常变化的时候。

例子：肯德基的汉堡，可乐，薯条是不变的，但是他的组合是可变的，称为套餐。

1. **工厂模式：建立一个工厂方法来制造新的对象**

意图：定义一个创建对象的接口，让其子类自己决定实例化哪一个工厂类，工厂模式使其创建过程延迟到子类进行。

何时使用：我们明确地计划不同条件下创建不同实例时。

例子：可以举女朋友从话费的这个

**Android的提示**

**四大组件：**

1. 活动：显示层
2. 服务：应用程序中不可见的工人
3. 内容：提供共享数据存储
4. 广播的接收者

**项目**

**项目描述：**

我做的这个是移动校园商城App，这款App可以给在校大学生提供一个购买零食或者生活用品的一个平台。

然后我这个app主要用java语言开发的，整体是用RadioGroup和Fragment搭建的，并且使用了Okhttp联网请求数据，并且使用了Gson来解析传来的json数据，并且使用Glide用来加载网络图片。

**就是问你这个项目创新在哪：**

因为我们的后端是部署在服务器上的，再从服务器拿取后端的数据，然后把这些数据先传给**postman（网络请求调试插件）测试工具**，然后再由Okhttp获取postman里面的数据。因为获取的是Json相关的对象数据所以还需要用Gson进行解析成java的实体。当发送订单的时候再由利用Gson将对象转为json字符串传到postman。还使用了Glide框架加载网络的图片，我的登录界面用的是MVP这种开发框架减轻了代码的冗余。

**你当时遇到什么问题  怎么解决的：**

**问题1**：Fragment之间切换时里面的数据会重新加载，会消耗用户的流量和我们的资源。

如何解决：之前一般的都是用replace（）这个方法，这个方法只是上一个碎片不需要时采用的简单方法。这里我封装了一个方法：先定义两个fragment，一个是马上要被隐藏起来的Fragment称为：from，一个是马上要被切换到的Fragment称为：to。先判断from和to是否是一个Fragment，不一样再判断这个to是否被添加过了；如果to没被添加过那就将from隐藏起来再把to添加到那个地方；如果添加过了直接把from隐藏起来再把to个显示出来。这个样子碎片里面的数据就不会清除了。

**问题二：**编写代码的时候发现：activity里面有太多的事情要干了，比如要初始化界面啊还要进行逻辑的判断还要与后端进行交互，导致activity里面的代码量太多并且也不好修改。

如何解决：将之前的开发模式由MVC模式改为MVP模式。MVP由以前的单向通信改为双向通信的方式。如果界面的UI出现问题的话只需要找View层；如果业务逻辑出现问题的话直接找Model层；P层值需要进行交互的作用。**首先写三个抽象基类**：BaseActivity，BaseModel，BasePresenter。**BaseActivity用来初始化布局的实例化子类的P层**；**BasePresenter用来将View层和P层建立关系，并且实例化M层**；**BaseModel是用来M层和P层建立关系的**。写完抽象基类之后**再写实类去重写这些基类的方法**。写完实类之后**再写契约合同接口**，然后再要实课去继承这个接口，**约束具体业务需求的标准行为**。比如我这个契约合同是M层去接收传递来的数据，请求业务逻辑；V层要发送出去的是账号和密码到P层，收到的是P层传来的密码是否正确的消息（true，f）。

问题三：图片下载下来很耗费资源怎么解决：

使用Glide的框架把图片从后端进行网络加载。

**你觉得他还有什么地方是你还不够完善的**

我这个App目前也就登录界面做了MVP框架处理，app的占用的内存感觉也比较大等等。

**自定义view：**

先写个方法：先判断资源类型是否为bitmap（位图类型），不是这个类型的话就将资源图片转为bitmap类型，再重写onDraw再选取比较短的一条做为圆的半径。当边框宽度不为0的时候就画出边框，操作是先画个圆，填充的是边框的颜色，大小就是这个控件的大小，在边框的圆的基础上再画个圆，画的是图片， 半径 = 这个控件设置的大小 – 边框宽度 ， 就露出了边框。如果边框就是为0的话，直接画一个圆形图片即可。