**二组模拟面试汇总**

**一、**

**·Android通信机制**

**二、**

**·EventBus**

**答: EventBus的原理，使用场景，优势。**

**☛EventBus的原理**

**EventBus底层采用的是注解和反射的方式来获取订阅方法信息。**

**☛使用场景:**

**a. 用于线程间的通讯代替Handler，或用于组件间的通信代替Intent。**

**b.广泛用于团购，商城，社交等应用。**

**C.实践证明已经有一亿多的App集成了EventBus。**

**☛优势:**

**◆1.简化了组件间的通讯，**

**◆2.分离了事件的发送者和接受者。**

**◆3.在Activity，Fragment和线程中表现良好，**

**◆4.避免了复杂和易错的依赖关系和生命周期的问题，**

**◆5.使代码更简洁，性能更好。**

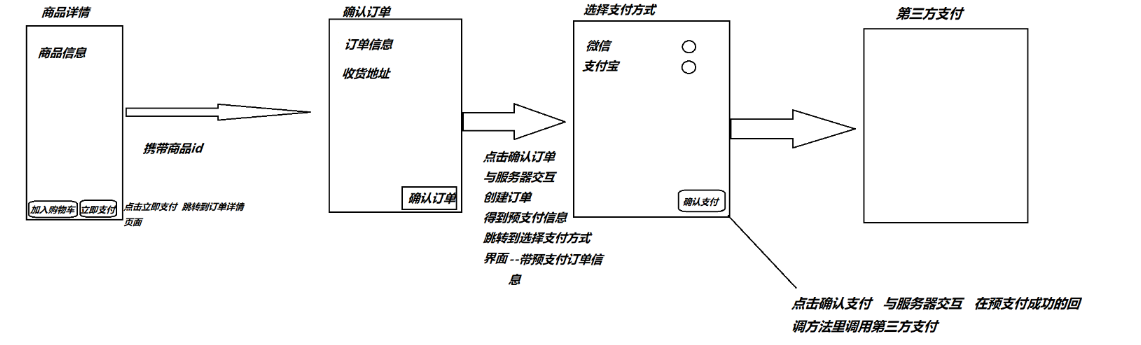
**◆6.更快，更小(约50Kjar包)**

**三、**

**·下单支付流程**

**答:集成第三方支付(支付宝，微信) 流程：1.注册账号 2.登录用户 3.创建应用 4.下载SDk,看文档集成。**

**从商品列表——>条目点击 跳转到 商品详情(拿到商品详情信息，点击立即购买，跳转传值到确认订单，携带商品信息)——>确认订单(拿到展示的是订单信息，点击提交订单，与服务器交互，创建订单，得到预支付订单信息)——>(携带预支付订单信息)选择支付方式(点击确认支付，与业务服务器交互，在预支付成功的方法里调用第三方支付)**



**四、**

**·泛型的理解及使用**

**五、**

**·注解及使用**

**六、**

**·huancuncelue**

**七、**

**·wait, notify , notifyAll 是Thread 还是Object**

**八、**

**·Picaso和 Glide 的区别**

**一、**

**·擅长的项目类型**

**二、**

**·Material Design新控件**

**答:**

1.Toolbar   
2.CollapsingToolbarLayout   
3.AppBarLayout   
4.CoordinatorLayout   
5.DrawerLayout   
6.NavigationView   
7.ActionBarDrawerToggle   
8.Recyclerview   
9.CardView

**三、**

**·动画分类**

**答:Android动画分为:帧动画(FrameAnimation)，补间动画(TweenAnimation)，属性动画(PropertyAnimation)。**

**★1.帧动画(FrameAnimation):类似于放电影，通过播放已经排列好的图片来实现。**

帧动画的实现方式有两种:

1. 在res/drawable文件夹下新建animation-list的XML实现帧动画
2. 通过代码实现帧动画

**使用帧动画实现动画的效果，可以减少所占内存的大小。**

**★2. 补间动画(TweenAnimation):分为 透明，旋转，位移，缩放。指通过指定View的初始状态、变化时间、方式，通过一系列的算法去进行图形变换，从而形成动画效果，带给人们视觉上的效果，只能作用在view上，只改变view的显示效果，不会改变view的属性，动画效果单一。**

**★3. 属性动画(PropertyAnimation): 从Android3.0，API11开始的。具有补间动画所有的动画效果，并实现它不能实现的效果。带给人们真的视觉感受。**

**作用对象:**任意java对象。

**原理**:在一定时间间隔内，通过不断对值进行改变，并不断将值赋值给对象的属性，从而实现该对象在该属性上的动画效果。

**属性动画常用的API**

**四、**

**·Activity启动模式**

**答:四种启动模式:**

**★Standard:标准模式 （默认模式）**

**★SingleTop:栈顶模式:**如果在任务的栈顶存在该Activity的实例，就重用该实例，否则就会创建新的实例并放入栈顶，即使栈中已经存在Activity实例，只要不在栈顶，都会创建实例。

**★SingleTask如果栈中已经有该Activity的实例，就重用该实例（会调用实例的onNewIntent()方法），重用时，会让该实例回到栈顶，因此在它上面的实例将会被移除栈。如果栈中不存在该实例，就会创建新的实例放入栈中。**

**★SingleInstance:创建Activity实例，并让多个应用共享该Activity中的实例，任何应用再激活该Activity都会重用该Activity实例，相当于多个应用程序共享一个应用，不管谁激活Activity都会进入同一个应用。**

**五、**

**·为什么要使用Retrofit+RxJava**

**Retrofit**

1. **可以直接拿到对象，支持Gson解析**
2. **支持Rxjava**
3. **链式调用**
4. **带有封装的OKHttp**

**RxJava优势**

**解决嵌套问题，迷之缩进。**

**观察者，被观察者直接拿到数据，**

**完成之后 解绑，处理内存泄漏问题**

**六、**

**·多线程的实现方式**

**一：**

**·擅长的项目类型“”**

**二：**

**·购物车数据同步的实现**

**三：**

**·Okhttp切换成Retrofit+rxJava的原因**

**四：**

**·Retrofit1.0和Retrofit2.0的区别**

**五：**

**·什么是背压**

RxJava背压（Backpressure）机制

在RxJava中，可以通过对Observable连续调用多个Operator组成一个调用链，其中数据从上游向下游传递。当上游发送数据的速度大于下游处理数据的速度时，就需要进行Flow Control了。如果不进行Flow Control，就会抛出MissingBackpressureException异常。

这就像小学做的那道数学题：一个水池，有一个进水管和一个出水管。如果进水管水流更大，过一段时间水池就会满（溢出）。这就是没有Flow Control导致的结果。

再举个例子，在 RxJava1.x 中的 observeOn， 因为是切换了消费者的线程，因此内部实现用队列存储事件。在 Android 中默认的 buffersize 大小是16，因此当消费比生产慢时， 队列中的数目积累到超过16个，就会抛出MissingBackpressureException。

如果你想了解更多关于背压的知识，请参考：

<http://blog.csdn.net/jdsjlzx/article/details/52717636>   
<http://www.jianshu.com/p/2c4799fa91a4>

**六：**

**·RxJava使用完对象之后怎么处理**

**（内存泄漏相关问题）**

**软引用**

**七：**

**·RxJava操作符及使用**

[1. RxJava使用介绍](http://blog.csdn.net/jdsjlzx/article/details/51485778) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/360)   
 [2. RxJava操作符](http://www.stay4it.com/course/27)   
  [• Creating Observables(Observable的创建操作符)](http://blog.csdn.net/jdsjlzx/article/details/51485861) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/359)   
  [• Transforming Observables(Observable的转换操作符)](http://blog.csdn.net/jdsjlzx/article/details/51493772) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/369)   
  [• Filtering Observables(Observable的过滤操作符)](http://blog.csdn.net/jdsjlzx/article/details/51489061) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/371)   
  [• Combining Observables(Observable的组合操作符)](http://blog.csdn.net/jdsjlzx/article/details/52415615) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/370)   
  [• Error Handling Operators(Observable的错误处理操作符)](http://blog.csdn.net/jdsjlzx/article/details/52448173) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/372)   
  [• Observable Utility Operators(Observable的辅助性操作符)](http://blog.csdn.net/jdsjlzx/article/details/54768360) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/364)   
  [• Conditional and Boolean Operators(Observable的条件和布尔操作符)](http://blog.csdn.net/jdsjlzx/article/details/54767751) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/374)   
  [• Mathematical and Aggregate Operators(Observable数学运算及聚合操作符)](http://blog.csdn.net/jdsjlzx/article/details/51489793) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/368)   
  [• 其他如observable.toList()、observable.connect()、observable.publish()等等；](http://blog.csdn.net/jdsjlzx/article/details/53365839) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/366)   
[3. RxJava Observer与Subcriber的关系](http://blog.csdn.net/jdsjlzx/article/details/51534504) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/367)   
[4. RxJava线程控制（Scheduler）](http://blog.csdn.net/jdsjlzx/article/details/51685769) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/365)   
[5. RxJava 并发之数据流发射太快如何办（背压（Backpressure））](http://blog.csdn.net/jdsjlzx/article/details/51868640) [【视频教程】](http://www.stay4it.com/course/27/learn#lesson/373)

**八：**

**·简述观察者模式**

**（Java中类是被观察者，接口是观察者）**

**九：**

**·接口回调如何进行+使用场景**

**（MVP,RecyclerView的条目点击事件,Fragment通信）**

**十：**

**·String | StringBuffer | StringBuilder 区别**

**十一：**

**·异常的分类：**