

-

整理自互联网，仅供学习交流。



2018-10-30

[公司名称]

[公司地址]

目录

[一、 123 1](#_Toc3539499)

[1.1 1 1](#_Toc3539500)

[1.1.1 3 1](#_Toc3539501)

## 我从阿里面试回来，想和Java程序猿谈一谈

**第一个问题：阿里面试都问什么？**

很多人问我“去阿里面试都问了些什么”？，说实话，我只能隐约想起并发、JVM、分布式、TCP/IP协议这些个关键字，具体的问题真的是几乎都没记住。而且就算记住了，也告诉你了，你也背会了，但在面试中，你被问到一模一样问题的可能性依然很小。

甚至，就算你运气好被问到了，你也照着背下来了，也不一定就能对你的面试起到正面的作用，因为面试官万一多问一句，你可能就露馅了，那还不如干脆点说不会更好。

我当初参加的是阿里的社招面试，而社招不同于校招，问题的范围其实是很随机的。因为能参加一些比较知名的互联网公司社招的人，70%以上都会有个3-5年的经验。这倒不是说一两年经验的同学没有机会进这些公司，而是因为这种公司，大部分情况下只招一些比较资深的开发和应届生，而不招那些处于中间阶段的人。而1-2年经验的同学，往往就刚好处于这个尴尬的阶段。

对于能有3-5年经验的这部分人中，每个人的经历又都不同，所擅长的点也不一样，因此这就会导致每个人的问题和范围都不太一样。

很少说有哪个知名的互联网公司，比如BAT、京东、360、搜狐、网易等这些公司，其社招面试还有固定的问题和模式，让你可以像应届生面试一样，在面试前靠临时抱佛脚度过这一关。

大部分公司在社招的时候，不光是阿里，其它公司也都一样（因为我在一年多前也参加过很多其它知名互联网公司的面试，基本上都分为两个阶段的提问。

**第一个阶段**是主语言本身以及它的高级特性；

**第二个阶段**是讲述自己的项目，并在中间穿插着问题。

接下来我就针对这两个阶段，谈谈社招面试的准备，而不是去把阿里面试的过程背一遍。

**1、主语言本身以及它的高级特性**

对于Java来说，它的语言本身以及它的高级特性，都有哪些比较容易在面试中问到呢？

一般情况下，主要有以下知识点很容易被问到。（PS：以下所列举的，都是一些Java相对而言比较高级一点的知识点，因为这里谈的是社招，而不是校招）

1）Java的数据结构相关的类实现原理，比如LinkedList，ArrayList，HashMap，TreeMap这一类的。以下简单模拟一个数据结构的连环炮。

**比如，面试官先问你HashMap是不是有序的？**

你肯定回答说，不是有序的。那面试官就会继续问你，有没有有顺序的Map实现类？

你如果这个时候说不知道的话，那这个问题就到此结束了。如果你说有TreeMap和LinkedHashMap。

那么面试官接下来就可能会问你，TreeMap和LinkedHashMap是如何保证它的顺序的？

如果你回答不上来，那么到此为止。如果你依然回答上来了，那么面试官还会继续问你，你觉得它们两个哪个的有序实现比较好？

如果你依然可以回答的话，那么面试官会继续问你，你觉得还有没有比它更好或者更高效的实现方式？

如果你还能说出来的话，那么就你所说的实现方式肯定依然可以问你很多问题。

以上就是一个面试官一步一步提问的例子。所以，如果你了解的不多，千万不要敷衍，因为可能下一个问题你就暴露了，还不如直接说不会，把这个问题结束掉，赶紧切换到你熟悉的领域。

2）Java并发包当中的类，它们都有哪些作用，以及它们的实现原理，这些类就是java.concurrent包下面的。与上面一样，咱们也简单的模拟一个并发包的连环炮。

**比如，面试官可能会先问你，如果想实现所有的线程一起等待某个事件的发生，当某个事件发生时，所有线程一起开始往下执行的话，有什么好的办法吗？**

这个时候你可能会说可以用栅栏（Java的并发包中的CyclicBarrier），那么面试官就会继续问你，你知道它的实现原理吗？

如果你继续回答的话，面试官可能会继续问你，你还知道其它的实现方式吗？

如果你还能说出很多种实现方式的话，那么继续问你，你觉得这些方式里哪个方式更好？

如果你说出来某一个方式比较好的话，面试官依然可以继续问你，那如果让你来写的话，你觉得还有比它更好的实现方式吗？

如果你这个时候依然可以说出来你自己更好的实现方式，那么面试官肯定还会揪着这个继续问你。

为什么说面试的时候要引导面试官，原因就在这了。**因为面试官的提问很多时候都是有迹可循的，你如果抓住了他的轨迹，能够猜到他下面很可能会问什么，那你在回答的时候就可以往你想要谈的方向去说**。这样面试时就会显得更加从容，更加的游刃有余。

3）IO包和NIO包中的内容。这部分里面NIO会是重点，IO包大部分都会比较熟悉，因此可能会直接略过，直接问你NIO的内容。

IO包和NIO包的内容相对来说不是很多，首先NIO模型要熟悉，特别是其中的selector一定要非常清楚它的职责和实现原理。其实NIO的核心是IO线程池，一定要记住这个关键点。有的时候，面试官可能也会问你IO包的设计模式（装饰器模式），为什么要这样设计？

有的面试官还会问你有没有更好的设计，这个时候如果你不知道请果断说自己现在的水平有限，想不出来更好的设计，千万不要信口开河，随意YY。

4）Java的虚拟机的内容。这部分主要包括三部分，GC、类加载机制，以及内存。

一个GC部分简单的连环炮。

面试官可以先问你什么时候一个对象会被GC？

接着继续问你为什么要在这种时候对象才会被GC？

接着继续问你GC策略都有哪些分类？

你如果说出来了，继续问你这些策略分别都有什么优劣势？都适用于什么场景？

你继续说出来了以后，给你举个实际的场景，让你选择一个GC策略？

你如果选出来了，继续问你，为什么要选择这个策略？

下面是关于类加载机制的简单连环炮。

首先肯定是先问你Java的类加载器都有哪些？

回答了这些以后，可能会问你每个类加载器都加载哪些类？

说完以后，可能会问你这些类加载之间的父子关系是怎样的？

你在回答的时候可能会提到双亲委派模型，那么可以继续问你什么是双亲委派模型？

你解释完了以后，可能会继续问你，为什么Java的类加载器要使用双亲委派模型？

你回答完以后，可能会继续问你如何自定义自己的类加载器，自己的类加载器和Java自带的类加载器关系如何处理？

再来一个关于内存的连环炮。

首先肯定就是问你内存分为哪几部分，这些部分分别都存储哪些数据？

然后继续问你一个对象从创建到销毁都是怎么在这些部分里存活和转移的？

接着可能会问你，内存的哪些部分会参与GC的回收？

完事以后，可能还会问你Java的内存模型是怎么设计的？

你回答了以后，还会继续问你为什么要这么设计？

问完以后，还可能会让你结合内存模型的设计谈谈volatile关键字的作用？

你在谈的时候，肯定会提到可见性，那么接着可见性这三个字，还可以继续问你并发的内容。

基本上Java语言本身以及语言稍微高级点的内容就是以上部分，如果你能把以上四部分了解的非常透彻，那基本上Java这部分就没啥问题了，因为光以上的内容就够你跟面试官聊很久了。你聊这些聊得久了，自然问你其它问题的时间就会短点。

你从以上这些问题的过程也应该能感受出来，很多时候，面试官都是顺着一条线一路问下去的，如果你觉得这条线你不熟悉的话，就要及时拐弯，引导面试官去问其它方面的问题。千万不要一直往下深入，直到自己跳不出来为止，那就尴了个尬了。

**2、讲述自己的项目，并在中间穿插着问题**

这一部分是面试过程中必问，也是聊得最久的一个阶段。除非你前面的语言部分非常扎实，扎实到面试官问了一两个小时，依旧没有探出你对语言本身的了解到底有多深。否则的话，你一定逃不过自己的项目这一关，而且一般情况下聊得时间不会太短。

这一部分内容，一般的模式就是你自己去讲你做过的项目，然后面试官会冷不丁的让你去解释其中某一部分，比如让你解释当时为什么要这么做，或者问你现在觉得有没有更好的办法。而这些穿插的问题，大部分与你的项目所用到的技术有关。**而你需要做的，就是充分、再充分的去总结自己做过的项目（尤其是最近的一两个项目），挖掘出一个甚至N个亮点，以备于到时候可以让面试官产生眼前一亮的感觉。**如果你能达到这种效果的话，基本上离你成功就不远了。

**3、额外的加分项**

上面两个阶段基本上是必问的，还有一些加分项。这些加分项中，有些内容面试官也会问你（比如TCP/IP协议、算法），但更多的是会先问你了解不了解，你了解的话再继续聊，不了解的话就直接略过了，不至于因为这种问题而直接把你打入地狱。

下面列举一下这些加分项，如果可以的话，这些加分项还是要争取一下的。

1、计算机系统原理。

2、网络通信协议（TCP/IP，HTTP等）。

3、数据结构与算法。

4、著名开源项目的源码。

5、你自己有很棒的开源项目。

6、你的个人博客。

7、待评论区补充。

这几项当中，对于前1-3项，如果你之前就比较了解，只是由于时间问题忘记了的话，还是可以临时抱佛脚一下的。至于后面4-6项，就需要你日常的积累了，不是一时半会儿能做到的。如果你平日里没有积累，那么后面这三个加分项只能抛弃了。

**4、与你职位相关的内容**

其实这最后一项是对前面三项的补充，你应该尽量去主攻和你面试的职位相关的内容。比如你面试一个实时计算的职位，那么你的算法最好要厉害，对于著名的实时计算开源项目要熟悉，最好阅读过源码，而且还要对分布式系统有一定的见解。

因此，这个第4部分没有具体的内容，只是提醒你，如果你很明确自己的面试职位，最好在面试前准备的时候，尽量朝职位的需求方向靠拢，这样成功的可能性更大。