一、解释什么是面向对象编程（OOP），并讨论Java中封装、继承和多态的实现方式及其应用场景（百度AI）

> 这个问题，没有什么标准的答案，一般就是根据自己的技术储备和你项目中实际涉及到的点去聊。
>
> 其实面向对象的思想底层还是面向过程，只不过把一些复杂的操作封装成类，方便咱们操作。
>
> 就比如你们的项目中实际涉及到了什么事情，这个事情本质的行为逻辑市面上已经有很多成熟的方案。
>
> 比如我们的短信平台里，需要和运营商去交互，运行商对外提供了一套CMPP（中国移动）的协议，咱们需要根据他的协议去完成一套实现。但是市面上有很多开源框架都对接了CMPP2.0，还是CMPP3.0，我不需要自己再去造一套轮子，我可以拿过来稍微改一改直接应用到项目中。
>
> 当然，想要将一些封装好的工具，经过一些润色，从而更贴近咱们本身的业务方向，其实这个事情还是要究其底层的实现过程。
>
> ---
>
> **封装：** 封装其实非常常见，比如映射数据库表的ORM的实体类，或者是响应给前端数据时，封装一个VO，或者传输对象数据时，也会涉及到DTO之类的内容，这些都是封装的具体体现。而这些类内部都是将 **属性进行私有化** ，不允许直接修改，但是会对外提供 **set、get的公共方法** 去操作，而在方法中就可以直接很多限制，来 **保证数据的安全** 。
>
> **继承：** 这个更好办， **继承就是站在巨人的肩膀上** ，可以直接继承一个功能强大的类，直接使用他提供的各种功能。 这个强大的类，可以是前人种树留下来的，也可以是后续基于多个类似的类，向上抽取出来的一个公共的类。比如JUC包下的AQS，提供了响应的state属性，同步队列以及单向链表来给JUC包下的其他并发工具类来继承。。。。
>
> **多态：** 多态在平时写代码的时候，用的非常之频繁，同一个动作，可以有多种实现，代码最直观的体现，就是使用 **父类接口指向子类的实现** 。 比如Controller注入Service，都是声明Service接口，然后基于Spring注入一个实现类。 比如CacheService，是一个提供缓存的功能接口，可以给CacheService提供多种实现，比如MemcacheServiceImpl，比如RedisServiceImpl，再比如CoffeineServiceImpl，提供多种缓存策略的实现，可以基于不同的注入方式，使用不同的缓存实现。

二、数组和链表区别（国人通）

> 这个问题，约等于问了ArrayList和LinkedList的区别
>
> 数组和链表核心的区别就三个东西：
>
> - 查询效率
>   - 数组：可以通过索引直接访问数组上任意位置的元素，时间复杂度直接O1，效率嘎嘎快。
>   - 链表：只能顺序访问，要么从头开始找，要么从尾巴开始找到目标元素，时间复杂度On。
> - 增删效率：
>   - 数组：在数组中间插入或者删除元素，会导致需要移动其他元素，这个本身就是一个比较耗时的操作。
>   - 链表：插入和删除操作，只需要更改相邻的两个节点的指针就可以了，其他元素不需要动。

- 内存使用
    - 数组： 数组申请时，需要一片连续的内存空间，而且要提前指定好长度。或多或少可能会浪费一些空间，而且预计的大小不够，扩容还需要再构建一波新的数组。特别是如果你的JVM中有内存碎片的问题，在申请一个比较长的数组时，可能时间导致执行GC甚至是OOM。
    - 链表：不需要提前申请很大一片内存空间，但是链表的每个节点需要额外的存储空间来保存指针。链表他不需要扩容。

- 如果你需要快速的随机访问元素，并且可以大致预估出需要的数据个数，那么数组更合适。
- 如果你的业务是频繁的增删数据，很明显，链表更合适。

线程池 -- 阻塞队列 -- ArrayBlockingQueue， LinkedBlockingQueue，明显链表好，可以望这拐。

三、讨论Java集合框架的主要接口（如List、Set、Map）的特点，举例说明在什么场景下使用它们（美团地图、众安保险）

这种问题，就是会是应该的，不会就回家等通知。。

List：

- 特点：List是有序集合（存取有序），允许元素重复，并且可以存储null值。
- 适用场景：
    - 当你需要维护元素的插入顺序时，比如实现一个队列可以上List
    - 当你需要遍历整个元素时，可以采用List
- 扩展：List下有ArrayList，LinkedList，聊数组和链表的问题.....还有一个Vector，这哥们线程安全的，但是用的synchronized，效率不好.....

Set：

- 特点：Set不允许存储重复的元素，也不能保证元素的顺序（存取有序，除了LinkedHashSet），能存储Null，但是只能存一个
- 适用场景
    - 当你需要保证元素的唯一性时，比如用户手机号，用户的ID。
    - 当你需要做去重操作时，Set是一个很好的工具。
- 扩展：Set本质是基于HashMap的key去实现的。基于哈希表做到的去重效果。但是HashMap线程不安全。。。。。。。

Map：

- Map建议就直接点一嘴，他是双列集合。有key-value结构。底层就是哈希表，结构是数组 + 链表实现的，在JDK1.8之后，优化了一波红黑树来提升HashMap的查询效率。。。。

四、HashSet底层怎么实现的？（高御科技）Map和Set区别（江苏探路者国际物流）LinkedHashSet是怎么保证存储有序的（马士兵教育）

HashSet底层怎么实现的?

这还用问么，前面说了，HashSet底层就是HashMap的key。

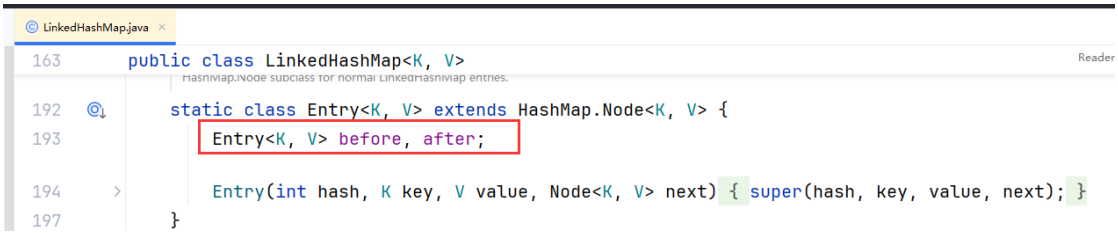
Map和Set区别?

没有所谓的区别，Set是基于Map的key实现的。

LinkedHashSet是怎么保证存储有序的?

你也可以理解为LinkedHashMap是怎么保证存取有序的。。所以他还是基于哈希表去存的。而哈希表有个特点，必然是基于key的hash运算跟数组做一些操作，得到要存储的索引位置，顺序必然是随机的，但是LinkedHashSet就是有序的，怎么保证的?

其实没啥难的，就是对HashMap里的Node又包装了一层，搞了个before和after，来记录存取顺序



当插入重复元素时，内部流程怎么走的?

将之前Node的value做了个替换。。

五、JDK8的新特性（众安保险，美团地图等等）

给方向

interface的新特性：（不需要说太细，点一嘴，他没啥扩展的。）

- default关键字：可以规避接口中必须要重写的这点，一些不必要的方法，不常用的方法，可以来个default给予默认的实现。
- static关键字：可以在接口中提供直接访问的方法，给予接口名.方法，去调用。

时间类新特性:

- 支持时区，ZonedDateTime类，允许你方便的处理不同时区的信息。不需要三方依赖库。
- 线程安全性，他一旦创建是不可变的，线程安全，可以在并发环境中用!
- 更丰富的API，之前Date需要配置Calender去做一些时间处理，而现在，java.time包下的时间类，API嘎嘎丰富。
- 但是注意一点，在使用time包下的时间时，注意指定好序列化JSON的序列化格式。。。。。

lanbda，Stream

- lambda

```
一个语法糖，让代码看着更舒服，比如匿名内部类，不需要再去new 接口，重新抽象方法了。
提高一些可读性，看着更舒服，并且利用它支持函数式编程和stream
```

- Stream

```
链式调用，舒服的一批，还提供了各种丰富的中间操作，map，filter等等，然后可以可以上一个
collect结束整个流操作。
而且操作结果时，过滤，映射，规约，排序，匹配等等，嘎嘎好用。
但是需要一定的学习成本。
```

hashMap的红黑树问题。。

六、使用Stream流的坑（美团地图）

> Stream最出名，最大的坑，就是他的parallel，并行流。
>
> 用了并行流，将相当于上了线程池，做并行操作，你会天真的以为，性能提升了。
>
> 首先parallel这个东西的核心问题，就是他默认所有的parallel都会用ForkJoinPool的线程池，默认的线程数是CPU内核 - 1（我记得是）。每个parallel任务都用他，如果自己没有主动的根据业务去设置一个参数合理的线程池的话，反而不会提升性能，会让处理速度变低。。。。。
>
> 比如你把大量的IO密集的操作用parallel并行流去玩，结果线程个数没有达到IO密集的要求，前几个任务还好，后几个任务就需要等待前面的任务完成，才有线程用，反而会让多个请求变成了一个串行处理的套路。甚至不如不用。。。
>
> 解决方案，就是必须自己设置一个合理的线程池，建议，及其建议，上ThreadPoolExecutor
>
> ---
>
> 把数据转成Map，因为获取的数据有重复，导致的错误（不算坑，完全因为不熟悉导致的）
>
> 在将List数据转成Map，方便基于get去查询数据时，如果转换时，List中数据作为key的如果有重复，会导致转换时出现错误。
>
> 其实不算是坑，就是单纯的stream不是所有API都熟练！！！

七、JDK1.8的数据结构的优化（美团地图）

> 问到这个，就是在HashMap。。。
>
> HashMap在JDK1.8的时候，织入了红黑树结构来提升HashMap的查询效率。。。
>
> 聊到这，还得在说一点，比如什么时候转红黑树？ 不啥不用别的树？？
>
> **数组 >= 64，链表 > 8（链表插入第9个元素时）**
>
> 在treeifyBin转红黑树的逻辑中，会先判断数组长度是否达到64，没达到会先扩容数组。
>
> 因为红黑树的目的是为了提升查询的性能，如果数组长度太短，导致出现了红黑树，那反而会导致写入的性能和查询的性能都收到影响。在数组长度太短时，会优先考虑扩容，而不是转红黑树。
>
> 链表大于8怎么来的，源码写的，数据添加到链表下后，会判断binCount是否达到了插入第9个元素的时候。
>
> **链表长度为啥是大于8。不是10，不是7，不是6？？凭啥。**
>
> 泊松分布，基于泊松分布，得出，在链表长度为9的时候，概率贼低。
>
> 但是也不是说非9不可，其实降一降，或者升一升也没啥毛病，概率依然很低，、
>
> 所以核心就是源码就这么写的。。。（不建议在面试的时候说。）

面试，是聊天，沟通，可以认为是相亲。