**过往候选人java面试题汇总**

2、项目用到的技术点：Spring+SpringBoot+SpringMVC+SpringCloud

3、springboot自动配置

11、synchniztion原理

<https://blog.csdn.net/qq_43198052/article/details/119973666>

12、jvm虚拟机内存结构 可能导致内存溢出的区域 程序计数器会导致溢出吗 为什么

14、hashmap和concurrenthashmap的区别以及各自的结构和实现方式

<https://blog.csdn.net/qq_50227688/article/details/114301326>

16、java线程几种状态 有没有正在进行中这个状态(没有)

New ,RUNNABLE，BLOCKED，WAITING，TIMED\_WAITING，TERMINATED,

17、线程的创建方式几种

1：继承Thread，重写run方法，实例化，调用start()方法；

2：实例化继承Runnable接口类，或者lambda表达式，作为参数传入Thread类的对象中，调用Thread类对象的start()方法。

3：Future和Callable,

创建Callable接口的实现类，实现call方法，FutureTask类封装Callable实现类的实例，使用FutureTask对象作为Thread的参数创建并启动线程。FutureTask的get方法获得返回值

4．线程池

18、还有有数组和链表，链表判断是否有环

<https://blog.csdn.net/MAX_VALUE/article/details/117930686>

1：快慢指针，快的能追上慢的，就有环。

快慢指针。类似于数学中的追及问题，设置一个快指针，步长为2，设置一个慢指针，步长为1。如果不存在环，则快慢指针不会相遇。如果存在环，则快慢指针会相遇。[时间复杂度](https://so.csdn.net/so/search?q=%E6%97%B6%E9%97%B4%E5%A4%8D%E6%9D%82%E5%BA%A6&spm=1001.2101.3001.7020)为O(n)，空间复杂度为O(1)。**此为最优方法**

2：走过的放集合，有重复，就是环。

使用HashMap（或者HashSet），以[Node](https://so.csdn.net/so/search?q=Node&spm=1001.2101.3001.7020)节点作为key，遍历链表，如果node节点不存在则添加到集合中，如果节点存在则说明有环。时间复杂度为O(n)，空间复杂度为O(n)。

3: 双层循环，外层循环一个方向，内层循环另一个方向。

通过双循环[遍历](https://so.csdn.net/so/search?q=%E9%81%8D%E5%8E%86&spm=1001.2101.3001.7020)完成，外层循环从头结点开始遍历，对于遍历到一个新节点newNode，内层循环都要从头结点遍历到该新节点的上一个节点head。如果链表无环，则newNode和head永不相等，否则就会相等。时间复杂度为O(n^2)。

19、Threadlocal

20、JAVA的四种引用

强，软，弱，虚

21、线程池的拒绝策略

1：AbortPolicy：啥也不做，就抛错（默认），

2：CallerRunsPolicy：调用者处理

3：DiscardOldestPolicy：抛弃最早的

4：DiscardPolicy：啥也不做，也不抛错

【数据服务方向Java开发一面面试题】

1.自我介绍

2.项目介绍

10.hashmap介绍 完全没看，只能凭借以前的记忆了，回答了大部分内容

11.线程池如何管理线程，问到线程运行run方法后会被销毁，但是线程池中的核心线程怎么没有被销毁？（take方法阻塞） 不知道

12.一个特比大的文件，里面存储了很多url, 如何在有限内存里去重？ Hash分桶思想，将相同的url写入到同一个文件，但需要考虑热key问题

13.你有什么要问的吗？

【java 用户增长方向】

1. 介绍项目流程

2. HashMap结构、扩容，什么时候红黑树（放第9个的时候），为什么不选平衡二叉树（需要调整更多次数，浪费更多性能）

树化：放第9个的时候；

逆树化：<=6:拆分以后，有6个节点，就发生逆树化。

3. ConcurrentHashMap，1.7（数组+链表）、1.8（数组+链表/红黑树）分别结构

4. AQS原理

最重要的两个队列：竞争队列（线程的直接从该队列中获取任务）和等待队列（双向链表，await方法执行过后会进入该队列）

await:执行该方法后，将当前线程封装成节点，放入等待队列尾部。

signal：执行该方法，将等待队列中的首节点转移到竞争队列。

signalAll：执行该方法,将等待队列中的所有节点转移到竞争队列。

acquire：进入竞争队列并获取资源，直到成功或者中断。

线程池中：先满足核心线程池，在满足队列，最后满足最大线程池

5. ReentrantLock和sychronized结构、区别、原理

<https://www.cnblogs.com/chenchen127/p/11951445.html>

6. ReentrantReadWriteLock实现方式，state高16位和低16位

State:

高16位：共享锁数量；低16位：互斥锁数量；

7. jvm调优，g1垃圾回收

8. OOM，内存泄漏、内存溢出

<https://blog.csdn.net/KevinChen2019/article/details/125354491>

<https://blog.csdn.net/zhongyili_sohu/article/details/104728809>

【java 车后方向】

1. 项目做了什么，怎么实现的，项目实现过程的优化

2. 数组和链表的区别，性能和优缺点，面试官想问的是ArrayList和linkedlist

<https://www.cnblogs.com/g9420/articles/13767344.html>

<https://blog.csdn.net/TTTZZZTTTZZZ/article/details/84916281>

3. 多线程的线程安全问题，concurrentHashMap如何保证安全性的

<https://blog.csdn.net/gupaoedu_tom/article/details/124449788>

4.冒泡排序的思路，时间复杂度，优化

<https://blog.csdn.net/m0_56371721/article/details/121666241>

<https://blog.csdn.net/C_U_N_Z/article/details/89287602、>

优化思路：只要有一轮没有经过交换位置，说明已经排序完成，无须进行后边的比较。

时间复杂度O(N^2);

空间复杂度O(1);

5.单例模式的实现方法，双重检查加锁一定能保证单例吗？（不能，）

1：饿汉式

2：懒汉式

3：枚举

4：双重检查锁。

<https://blog.csdn.net/qq_36551991/article/details/106542350>