面试题总结

> 这个主要是我被问到的一些题目,仅做参考

java基础

JVM,JRE,JDK的解答

jvm:Java 虚拟机（JVM）是运行 Java 字节码的虚拟机。JVM 有针对不同系统的特定实现（Windows，Linux，macOS），目的是使用相同的字节码，它们都会给出相同的结果。

jre:是 Java 运行时环境。它是运行已编译 Java 程序所需的所有内容的集合，包括 Java 虚拟机（JVM），Java 类库，java 命令和其他的一些基础构件。但是，它不能用于创建新程序

JDK: 是 Java Development Kit，它是功能齐全的 Java SDK,它拥有 JRE 所拥有的一切，还有编译器（javac）和工具（如 javadoc 和 jdb）。它能够创建和编译程序。

java数据基本类型及它的初始化的值

```
boolean
byte
short
int
long
float
double
char
```

==和equals的区别

== : 它的作用是判断两个对象的地址是不是相等。即，判断两个对象是不是同一个对象(基本数据类型==比较的是值，引用数据类型==比较的是内存地址)。

equals() : 它的作用也是判断两个对象是否相等。但它一般有两种使用情况：

情况 1：类没有覆盖 equals() 方法。则通过 equals() 比较该类的两个对象时，等价于通过“==”比较这两个对象。
情况 2：类覆盖了 equals() 方法。一般，我们都覆盖 equals() 方法来比较两个对象的内容是否相等；若它们的内容相等，则返回 true (即，认为这两个对象相等)。

BIO,AIO,NIO有什么区别

BIO (Blocking I/O): 同步阻塞 I/O 模式，数据的读取写入必须阻塞在一个线程内等待其完成。在活动连接数不是特别高（小于单机 1000）的情况下，这种模型是比较不错的，可以让每一个连接专注于自己的 I/O 并且编程模型简单，也不用过多考虑系统的过载、限流等问题。线程池本身就是一个天然的漏斗，可以缓冲一些系统处理不了的连接或请求。但是，当面对十万甚至百万级连接的时候，传统的 BIO 模型是无能为力的。因此，我们需要一种更高效的 I/O 处理模型来应对更高的并发量。

NIO (Non-blocking/New I/O): NIO 是一种同步非阻塞的 I/O 模型，在 Java 1.4 中引入了 NIO 框架，对应 java.nio 包，提供了 Channel , Selector，Buffer 等抽象。NIO 中的 N 可以理解为 Non-blocking，不单纯是 New。它支持面向缓冲的，基于通道的 I/O 操作方法。 NIO 提供了与传统 BIO 模型中的 Socket 和 ServerSocket 相对应的 SocketChannel 和 ServerSocketChannel 两种不同的套接字通道实现,两种通道都支持阻塞和非阻塞两种模式。阻塞模式使用就像传统中的支持一样，比较简单，但是性能和可靠性都不好；非阻塞模式正好与之相反。对于低负载、低并发的应用程序，可以使用同步阻塞 I/O 来提升开发速率和更好的维护性；对于高负载、高并发的（网络）应用，应使用 NIO 的非阻塞模式来开发

AIO (Asynchronous I/O): AIO 也就是 NIO 2。在 Java 7 中引入了 NIO 的改进版 NIO 2,它是异步非阻塞的 IO 模型。异步 IO 是基于事件和回调机制实现的，也就是应用操作之后会直接返回，不会堵塞在那里，当后台处理完成，操作系统会通知相应的线程进行后续的操作。AIO 是异步 IO 的缩写，虽然 NIO 在网络操作中，提供了非阻塞的方法，但是 NIO 的 IO 行为还是同步的。对于 NIO 来说，我们的业务线程是在 IO 操作准备好时，得到通知，接着就由这个线程自行进行 IO 操作，IO 操作本身是同步的。查阅网上相关资料，我发现就目前来说 AIO 的应用还不是很广泛，Netty 之前也尝试使用过 AIO，不过又放弃了

jdk 1.8的新特性

Lambda表达式

函数式接口

方法引用和构造器调用

Stream API

接口中的默认方法和静态方法

新时间日期API

泛型的使用,以及泛型的上下边界

泛型，即“参数化类型”

泛型的本质是为了参数化类型（在不创建新的类型的情况下，通过泛型指定的不同类型来控制形参具体限制的类型）。也就是说在泛型使用过程中，操作的数据类型被指定为一个参数，这种参数类型可以用在类、接口和方法中，分别被称为泛型类、泛型接口、泛型方法

泛型只在编译阶段有效

泛型有三种使用方式，分别为：泛型类、泛型接口、泛型方法

上界<? extends T>不能往里存，只能往外取,就是?必须是T的子类

下界<? super T>往外取只能赋值给Object变量，不影响往里存,?必须是T本身或它的父类

容器

> 一般问你对集合体系的理解,然后说一下你熟悉的集合底层实现

HashMap的底层实现

1.主要问HashMap的底层实现?数据是怎么插入的,过程描述一下

2.扩展到1.8的红黑树

3.深入点问你红黑树的特性?然后介绍一下红黑树

4.刁钻点的会问你影响因子之类的问题

由HashMap会引出ConcurrentHashMap的问题

concurrentHashMap的底层实现,锁分段!然后你可以顺便说说HashTable的实现为什么会比较慢,因为他是全部加锁

HashSet的底层实现(这个问的不多)

知道底层是用HashMap实现的就好了,稍微看一下

多线程

volatile关键字

1.java内存模型(就是从主线程读取数据那个)
2.重排序和happens-before
3.作用,线程可见性以及防止重排序

ThreadLocal

1.会让你介绍这个关键字,怎么用的?

2.然后说一下原理

Synchronized的关键字

这个需要看一下,内容挺多的,也常问

Lock锁,就是从Synchronized关键字引发的,问你对其他锁的了解

这个可以看一下那个RetreenLock和Synchronized关键字的对比,然后说一下

多线程的创建方式

1.继承Thread类
2.实现Runable接口
3.实现Callable接口
4.线程池的实现

如果你上面提了线程池,会问你线程池的几个问题

1.你对线程池的池化技术的了解

2.创建线程池的几个参数,ThreadPoolExecutor的几个参数
corePoolSize,maximumPoolSize,workQueue

3.常见的几种线程池

4.如果你说的参数里包含了饱和策略,可能问你几个饱和策略,

乐观锁和悲观锁

1.会问什么是乐观锁和悲观锁?

2.会问你怎么用的

3.准备的好的话,你可以引申一些其他的锁什么公平锁非公平锁等

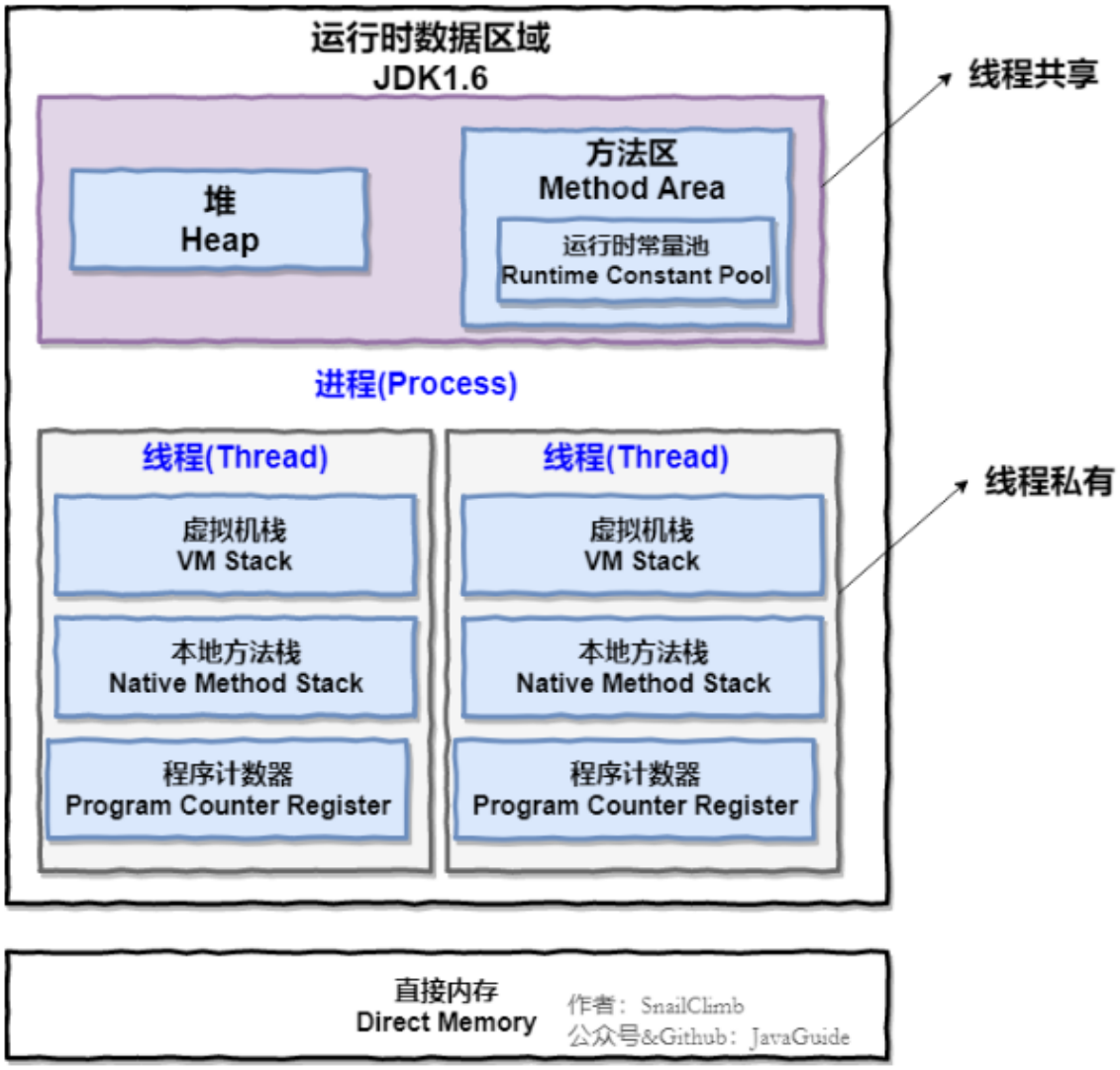
AtomicInteger这种的原子类

```
1.一般会问你对这个类的了解?

2.问这个实现原理,cas(compare and swap)这个东西
```

Java内存模型

说一下java内存模型




```
主要就是结合这张图开始说,说清楚线程私有的和线程共享的,然后介绍每一块大致干啥的
```

JVM里创建对象的过程

```
1.类加载检查
2.分配内存
3.初始化零值
4.设置对象头
5.执行init方法
```

内存模型会引申出几个问题

- 怎么判断对象死亡
  - 引用计数算法
  - 可达性分析
- 垃圾回收算法
  - 标记-清除算法(内存碎片化)
  - 复制算法(只能用一半内存)
  - 标记-整理算法
  - 分代收集算法
- 垃圾收集器
  - serial
  - parNew
  - parallel scavenge
  - serial old
  - parallel old
  - cms
  - g1

    > 刁钻点的会问你cms和paralel scavenge的比较,一个并行收集,一个是并发收集,可以看看

如果你简历写了jvm调优可能会问你那些工具(我没写)

参考这里

设计模式

> 我简历里写了设计模式,就问了我这个

1.你了解的设计模式

2.根据设计模式问你场景

3.问spring里用了哪些设计模式

4.你怎么用设计模式的

网络

TCP的三次握手,确立连接,四次挥手断开连接

问Tcp问的不多

Linux

linux的常见命令(因为我简历里写了这些,所以问了)

数据结构与算法

几种常见的排序的时间复杂度和空间复杂度

冒泡排序,快速排序,插入排序等,稍微要有点印象

数据库

> 因为在中控用过oracle和mysql

问你oracle和mysql的差异

隔离级别之类的

事务的隔离级别

事务隔离级别	脏读	不可重复读	幻读
读未提交（read-uncommitted）	是	是	是
不可重复读（read-committed）	否	是	是
可重复读（repeatable-read）	否	否	是
串行化（serializable）	否	否	否

数据库的调优

参考链接

索引相关

- 在哪些地方加索引
- 索引怎么会失效?
- 索引是不是越多越好
- mysql索引的底层实现B+树

Redis

参考我之前分享给你们的那篇公众号文章

系统设计

主要问你公司的系统架构,到时候可以一起说说

常用框架

spring的AOP实现

```
就是动态代理
jdk的动态代理看一下(面向接口)
基于cglib的动态代理
```

Spring的bean生命周期

```

```

spring 的bean的作用域

```
单例
多例
request
session
```

sprig事务的隔离级别

TransactionDefinition.ISOLATION_DEFAULT: 使用后端数据库默认的隔离级别，Mysql 默认采用的 REPEATABLE_READ隔离级别 Oracle 默认采用的 READ_COMMITTED隔离级别.

TransactionDefinition.ISOLATION_READ_UNCOMMITTED: 最低的隔离级别，允许读取尚未提交的数据变更，可能会导致脏读、幻读或不可重复读

TransactionDefinition.ISOLATION_READ_COMMITTED: 允许读取并发事务已经提交的数据，可以阻止脏读，但是幻读或不可重复读仍有可能发生

TransactionDefinition.ISOLATION_REPEATABLE_READ: 对同一字段的多次读取结果都是一致的，除非数据是被本身事务自己所修改，可以阻止脏读和不可重复读，但幻读仍有可能发生。

TransactionDefinition.ISOLATION_SERIALIZABLE: 最高的隔离级别，完全服从ACID的隔离级别。所有的事务依次逐个执行，这样事务之间就完全不可能产生干扰，也就是说，该级别可以防止脏读、不可重复读以及幻读。但是这将严重影响程序的性能。通常情况下也不会用到该级别。

spring运用了哪些设计模式

springboot相关

- springboot的热部署
- springboot是怎么启动的
- springboot的一些常用注解之类的

springcloud相关

- springcloud的基本组件
- springcloud的未来发展发祥和前景
- springcloud的rpc
- springcloud与dubbo的比较(我简历里写了dubbo,所以问了,你们自己考量,会说一点也好,可能会问对市面常见微服务框架的了解)

ElasticSearch

> 如果你用到了,你得会将es的一些基本知识

倒排索引是怎么实现的?

消息队列

问你用过几种消息队列,为什么选择用这个

除了kafka,你再说一个其他的,然后对比说一下

怎么保证消息不丢失

根据你用的队列来,kafka可以看一下的

怎么保证消息不重复消费

Zookeeper

zookeeper的服务注册与发现的了解

zookeeper一些知识总结

Docker和K8s

> 我写了那个基线的,就问我k8s相关的概念了

你对Docker技术的理解

k8s那些pod等概念的理解

简历

参考那个javaguide里面的

HR问题

问你为什么离职?

> 你们的好回答,不想干外包了...

是想从事技术,还是想从事业务?

> 这个自己权衡,想清楚就好