> 天之道，损有余而补不足，是故虚胜实，不足胜有余。

如背景中介绍，作者在一年之内参加过多场面试，应聘岗位均为Java开发方向。在不断的面试中，分类总结了Java开发岗位面试中的一些知识点。主要包括以下几个部分：

> 1. Java 基础知识点
> 2. Java 常见集合
> 3. 高并发编程（JUC 包）
> 4. JVM 内存管理
> 5. Java 8 知识点
> 6. 网络协议相关
> 7. 数据库相关
> 8. MVC 框架相关
> 9. 大数据相关
> 10. Linux 命令相关

面试，是大家从学校走向社会的第一步。互联网公司的校园招聘，从形式上说，面试一般分为2-3轮技术面试+1轮HR面试。但是一些公司确实是没有HR面试的，直接就是三轮技术面。技术面试中，面试官一般会先就你所应聘的岗位进行相关知识的考察，也叫基础知识和业务逻辑面试。只要你回答的不是特别差，面试官通常会说：“咱们写个代码吧”，这个时候就开始了算法面试。也就是说，一轮技术面试=基础知识和业务逻辑面试+算法面试。

关于算法面试的总结，各位可以查阅我先前交流的chat：“知名互联网公司校招中常见的算法题”。

本场chat，我们主要从技术面试聊起。技术面试包括：业务逻辑和基础知识面试。

首先是**业务逻辑面试**，也就是讲项目，面试官会对你简历上写的若干个项目其中之一拿出来和你聊聊。在期间，会针对你所做的东西进行深度挖掘，**包括：为什么要这么做？优缺点分析，假如重新让你做一次，你打算怎么做？**等等。这个环节主要考察我们对自己做过的项目（实习项目或者校内项目）是否有一个清晰的认识。关于业务逻辑面试的准备，建议在平时多多思考总结，对项目的数据来源、整体运行框架都应该熟悉掌握。比如说你在某公司实习过程中，就可以进行总结，而不必等到快离职的时候慌慌张张的去总结该项目。

接下来是**基础知识面试**。Java开发属于后台开发方向，有人说后台开发很坑，因为需要学习的东西太多了。没错，这个岗位就是需要学习好多东西。包括：本语言（Java/C++/PHP）基础、数据库、网络协议、Linux系统、计算机原理甚至前端相关知识都可以考察你，而且，并不**超纲**。有时候，你报的是后台开发岗，并且熟悉的是Java语言，但是面试官却是C++开发方向的，就是这么无奈~ 好了，闲话少说，让我们开始分类讲解常见面试知识点呗。

（一） Java基础知识点

1）面向对象的特性有哪些？

答：封装、继承和多态。

2）Java中覆盖和重载是什么意思？

解析：覆盖和重载是比较重要的基础知识点，并且容易混淆，所以面试中常见。 答：覆盖（Override）是指子类对父类方法的一种重写，只能比父类抛出更少的异常，访问权限不能比父类的小。被覆盖的方法不能是private的，否则只是在子类中重新定义了一个方法；

重载（Overload）表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同。

面试官：**那么构成重载的条件有哪些?**

答：参数类型不同、参数个数不同、参数顺序不同。

面试官：**函数的返回值不同可以构成重载吗？为什么?**

答：不可以，因为Java中调用函数并不需要强制赋值。举例如下：

如下两个方法：

```
        void f(){}
        int f(){ return 1;}
```

只要编译器可以根据语境明确判断出语义，比如在int x = f();中，那么的确可以据此区分重载方法。不过，**有时你并不关心方法的返回值，你想要的是方法调用的其他效果**（这常被称为“为了副作用而调用”），这时你可能会调用方法而忽略其返回值，所以如果像下面的调用：

```
        fun();
```

此时Java如何才能判断调用的是哪一个 f() 呢？别人如何理解这种代码呢？**所以，根据方法返回值来区分重载方法是行不通的。**

3）抽象类和接口的区别有哪些?

答：

1. 抽象类中可以没有抽象方法；接口中的方法必须是抽象方法；
2. 抽象类中可以有普通的成员变量；接口中的变量必须是static final类型的，必须被初始化,接口中只有常量，没有变量。
3. 抽象类只能单继承，接口可以继承多个父接口；
4. Java8中接口中会有default方法，即方法可以被实现。

面试官：抽象类和接口如何选择?

答：

1. 如果要创建不带任何方法定义和成员变量的基类，那么就应该选择接口而不是抽象类。
2. 如果知道某个类应该是基类，那么第一个选择的应该是让它成为一个接口，只有在必须要有方法定义和成员变量的时候，才应该选择抽象类。因为抽象类中允许存在一个或多个被具体实现的方法，只要方法没有被全部实现该类就仍是抽象类。

4）Java和C++的区别：

解析：虽然我们不太懂C++，但是就是会这么问，尤其是三面（总监级别）面试中。

答：

1. 都是面向对象的语言，都支持封装、继承和多态
2. 指针：Java不提供指针来直接访问内存，程序更加安全
3. 继承： Java的类是单继承的，C++支持多重继承； Java通过一个类实现多个接口来实现C++中的多重继承； Java中类不可以多继承，但是！！！ 接口可以多继承
4. 内存： Java有自动内存管理机制，不需要程序员手动释放无用内存

5）Java中的值传递和引用传递

解析：这类题目，面试官会手写一个例子，让你说出函数执行结果，详细举例请查阅我的博客：Java值传递和引用传递基础分析。

答：值传递是指对象被值传递，意味着传递了对象的一个副本，即使副本被改变，也不会影响源对象。

引用传递是指对象被引用传递，意味着传递的并不是实际的对象，而是对象的引用。因此，外部对引用对象的改变会反映到所有的对象上。

6）JDK中常用的包有哪些?

答：java.lang、java.util、java.io、java.net、java.sql

7）JDK，JRE和JVM的联系和区别：

答：JDK是 java开发工具包，是java开发环境的核心组件，并提供编译、调试和运行一个java程序所需要的所有工具，可执行文件和二进制文件，是一个平台特定的软件。

JRE是 java运行时环境，是JVM的实施实现，提供了运行java程序的平台。JRE包含了JVM，但是不包含java编译器/调试器之类的开发工具。

JVM是 java虚拟机，当我们运行一个程序时，JVM负责将字节码转换为特定机器代码，JVM提供了内存管理/垃圾回收和安全机制等。这种独立于硬件和操作系统，正是java程序可以一次编写多处执行的原因。

区别：

1. JDK用于开发，JRE用于运行java程序
2. JDK和JRE中都包含JVM
3. JVM是java编程语言的核心并且具有平台独立性

Others：限于篇幅的关系，面试中Java基础知识点还有：反射、泛型、注解等。详见我的博客：

- Java 注解简单学习
- Java泛型常见面试题
- Java反射机制

> 小结：本节主要阐述了Java基础知识点，这些问题主要是一面面试官在考察，难度不大，适当复习下，应该没什么问题。

（二）Java中常见集合

集合这方面的考察相当多，这部分是面试中必考的知识点。

1）说说常见的集合有哪些吧?

答：Map接口和Collection接口是所有集合框架的父接口：

1. Collection接口的子接口包括：Set接口和List接口
2. Map接口的实现类主要有：HashMap、TreeMap、Hashtable、ConcurrentHashMap以及Properties等
3. Set接口的实现类主要有：HashSet、TreeSet、LinkedHashSet等
4. List接口的实现类主要有：ArrayList、LinkedList、Stack以及Vector等

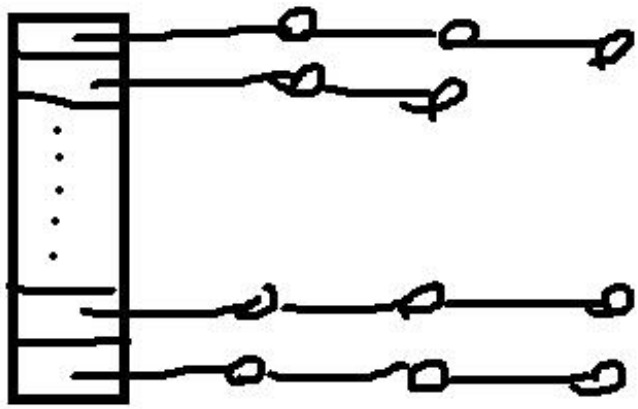
（2）HashMap和Hashtable的区别有哪些？（必问）

答：

1. HashMap没有考虑同步，是线程不安全的；Hashtable使用了synchronized关键字，是线程安全的；
2. 前者允许null作为Key；后者不允许null作为Key

3）HashMap的底层实现你知道吗?

答：在Java8之前，其底层实现是数组+链表实现，Java8使用了数组+链表+红黑树实现。此时你可以简单的在纸上画图分析：



4）ConcurrentHashMap和Hashtable的区别？必问

答：ConcurrentHashMap结合了HashMap和HashTable二者的优势。HashMap没有考虑同步，hashtable考虑了同步的问题。但是hashtable在每次同步执行时都要锁住整个结构。ConcurrentHashMap锁的方式是稍微细粒度的。 ConcurrentHashMap将hash表分为16个桶（默认值），诸如get,put,remove等常用操作只锁当前需要用到的桶。

面试官：ConcurrentHashMap的具体实现知道吗?

答：

1. 该类包含两个静态内部类HashEntry和Segment；前者用来封装映射表的键值对，后者用来充当锁的角色；
2. Segment是一种可重入的锁ReentrantLock，每个Segment守护一个HashEntry数组里得元素，当对HashEntry数组的数据进行修改时，必须首先获得对应的Segment锁。

5）HashMap的长度为什么是2的幂次方?

答：

1. 通过将Key的hash值与length-1进行&运算，实现了当前Key的定位，2的幂次方可以减少冲突（碰撞）的次数，提高HashMap查询效率
2. 如果length为2的次幂 则length-1 转化为二进制必定是11111……的形式，在于h的二进制与操作效率会非常的快，而且空间不浪费；如果length不是2的次幂，比如length为15，则length-1为14，对应的二进制为1110，在于h与操作，最后一位都为0，而0001，0011，0101，1001，1011，0111，1101这几个位置永远都不能存放元素了，空间浪费相当大，更糟的是这种情况中，数组可以使用的位置比数组长度小了很多，这意味着进一步增加了碰撞的几率，减慢了查询的效率！这样就会造成空间的浪费。

6）List和Set的区别是啥?

答：List元素是有序的，可以重复；Set元素是无序的，不可以重复。

7）List、Set和Map的初始容量和加载因子：

答：

1. List

- **ArrayList**的初始容量是10；加载因子为0.5； 扩容增量：原容量的 0.5倍+1；一次扩容后长度为16。
- **Vector**初始容量为10，加载因子是1。扩容增量：原容量的 1倍，如 Vector的容量为10，一次扩容后是容量为20。

2. Set

HashSet，初始容量为16，加载因子为0.75； 扩容增量：原容量的 1 倍； 如 HashSet的容量为16，一次扩容后容量为32

3. Map

HashMap，初始容量16，加载因子为0.75； 扩容增量：原容量的 1 倍； 如 HashMap的容量为16，一次扩容后容量为32

8）Comparable接口和Comparator接口有什么区别?

答：

1. 前者简单，但是如果需要重新定义比较类型时，需要修改源代码。
2. 后者不需要修改源代码，自定义一个比较器，实现自定义的比较方法。 具体解析参考我的博客：Java集合框架—Set

9）Java集合的快速失败机制“fail-fast”

答：

是java集合的一种错误检测机制，当多个线程对集合进行结构上的改变的操作时，有可能会产生fail-fast机制。

例如：假设存在两个线程（线程1、线程2），线程1通过Iterator在遍历集合A中的元素，在某个时候线程2修改了集合A的结构（是结构上面的修改，而不是简单的修改集合元素的内容），那么这个时候程序就会抛出 ConcurrentModificationException 异常，从而产生fail-fast机制。

原因：迭代器在遍历时直接访问集合中的内容，并且在遍历过程中使用一个 modCount 变量。集合在被遍历期间如果内容发生变化，就会改变modCount的值。每当迭代器使用hashNext()/next()遍历下一个元素之前，都会检测modCount变量是否为expectedmodCount值，是的话就返回遍历；否则抛出异常，终止遍历。

解决办法：

1. 在遍历过程中，所有涉及到改变modCount值得地方全部加上synchronized。
2. 使用CopyOnWriteArrayList来替换ArrayList

> 小结：本小节是Java中关于集合的考察，是Java岗位面试中必考的知识点，除了应该掌握以上的问题，包括各个集合的底层实现也建议各位同学阅读，加深理解。

（三）高并发编程-JUC包

> 在Java 5.0 提供了java.util.concurrent（简称JUC ）包，在此包中增加了在并发编程中很常用的实用工具类，用于定义类似于线程的自定义子系统，包括线程池、异步IO 和轻量级任务框架。

1）多线程和单线程的区别和联系：

答：

1. 在单核CPU中，将CPU分为很小的时间片，在每一时刻只能有一个线程在执行，是一种微观上轮流占用CPU的机制。
2. 多线程会存在线程上下文切换，会导致程序执行速度变慢，即采用一个拥有两个线程的进程执行所需要的时间比一个线程的进程执行两次所需要的时间要多一些

结论：即采用多线程不会提高程序的执行速度，反而会降低速度，但是对于用户来说，可以减少用户的响应时间。

2）如何指定多个线程的执行顺序?

解析：面试官会给你举个例子，如何让10个线程按照顺序打印0123456789？（写代码实现）

答：

1. 设定一个orderNum，每个线程执行结束之后，更新orderNum，指明下一个要执行的线程。并且唤醒所有的等待线程。
2. 在每一个线程的开始，要while判断orderNum是否等于自己的要求值！！不是，则wait，是则执行本线程。

3）线程和进程的区别：（必考）

答：

1. 进程是一个“执行中的程序”，是系统进行资源分配和调度的一个独立单位。
2. 线程是进程的一个实体，一个进程中拥有多个线程，线程之间共享地址空间和其它资源（所以通信和同步等操作线程比进程更加容易）
3. 线程上下文的切换比进程上下文切换要快很多。

- （1）进程切换时，涉及到当前进程的CPU环境的保存和新被调度运行进程的CPU环境的设置。
- （2）线程切换仅需要保存和设置少量的寄存器内容，不涉及存储管理方面的操作。

4）多线程产生死锁的4个必要条件?

答：

1. 互斥条件：一个资源每次只能被一个线程使用
2. 请求与保持条件：一个线程因请求资源而阻塞时，对已获得的资源保持不放
3. 不剥夺条件：进程已经获得的资源，在未使用完之前，不能强行剥夺
4. 循环等待条件：若干线程之间形成一种头尾相接的循环等待资源关系

面试官：如何避免死锁？（经常接着问这个问题哦~）

答：指定获取锁的顺序，举例如下：

1. 比如某个线程只有获得A锁和B锁才能对某资源进行操作，在多线程条件下，如何避免死锁?
2. 获得锁的顺序是一定的，比如规定，只有获得A锁的线程才有资格获取B锁，按顺序获取锁就可以避免死锁！！！

5）sleep( )和wait(n)、wait( )的区别：

答：

1. sleep方法：是Thread类的静态方法，当前线程将睡眠n毫秒，线程进入阻塞状态。当睡眠时间到了，会解除阻塞，进行可运行状态，等待CPU的到来。睡眠不释放锁（如果有的话）
2. wait方法：是Object的方法，必须与synchronized关键字一起使用，线程进入阻塞状态，当notify或者notifyall被调用后，会解除阻塞。但是，只有重新占用互斥锁之后才会进入可运行状态。睡眠时，释放互斥锁。

6）synchronized关键字：

答：底层实现：

1. 进入时，执行monitorenter，将计数器+1，释放锁monitorexit时，计数器-1；
2. 当一个线程判断到计数器为0时，则当前锁空闲，可以占用；反之，当前线程进入等待状态

含义：（monitor机制）

Synchronized是在加锁，加对象锁。对象锁是一种重量锁（monitor），synchronized的锁机制会根据线程竞争情况在运行时会有偏向锁（单一线程）、轻量锁（多个线程访问synchronized区域）、对象锁（重量锁，多个线程存在竞争的情况）、自旋锁等。该关键字是一个几种锁的封装。

7）volatile关键字

答：该关键字可以保证可见性不保证原子性。

功能：

1. 主内存和工作内存，直接与主内存产生交互，进行读写操作，保证可见性
2. 禁止JVM进行的指令重排序

解析：关于指令重排序的问题，可以查阅DCL双检锁失效相关资料。

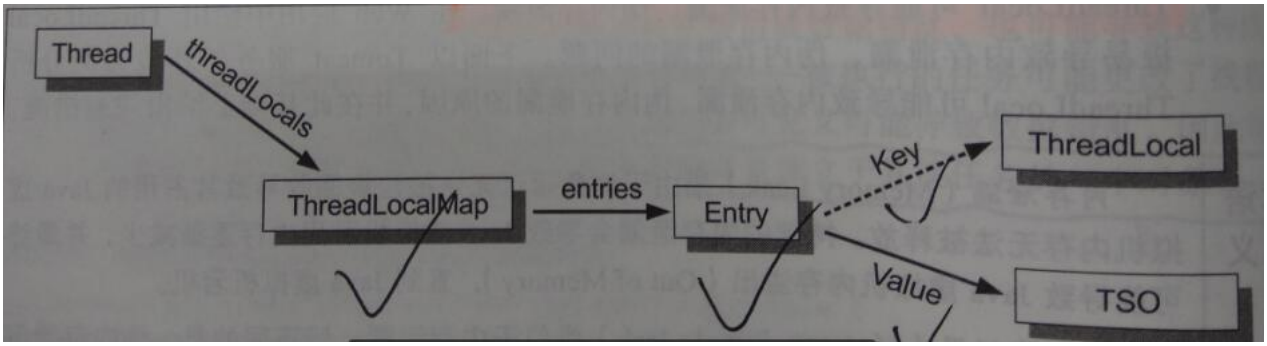
8）ThreadLocal（线程局部变量）关键字：

答：当使用ThreadLocal维护变量时，其为每个使用该变量的线程提供独立的变量副本，所以每一个线程都可以独立的改变自己的副本，而不会影响其他线程对应的副本。

ThreadLocal内部实现机制：

1. 每个线程内部都会维护一个类似HashMap的对象，称为ThreadLocalMap，里边会包含若干了

Entry（K-V键值对），相应的线程被称为这些Entry的属主线程
2. Entry的Key是一个ThreadLocal实例，Value是一个线程特有对象。Entry的作用即是：为其属主线程建立起一个ThreadLocal实例与一个线程特有对象之间的对应关系
3. Entry对Key的引用是弱引用；Entry对Value的引用是强引用。



9）Atomic关键字：

答：可以使基本数据类型以原子的方式实现自增自减等操作。参考我的博客：concurrent.atomic包下的类AtomicInteger的使用

10）线程池有了解吗？**（必考）**

答：java.util.concurrent.ThreadPoolExecutor类就是一个线程池。客户端调用ThreadPoolExecutor.submit(Runnable task)提交任务，线程池内部维护的工作者线程的数量就是该线程池的线程池大小，有3种形态：

> - **当前线程池大小**：表示线程池中实际工作者线程的数量
> - **最大线程池大小**（maxinumPoolSize）：表示线程池中允许存在的工作者线程的数量上限
> - **核心线程大小**（corePoolSize ）：表示一个不大于最大线程池大小的工作者线程数量上限

1. 如果运行的线程少于corePoolSize，则Executor始终首选添加新的线程，而不进行排队
2. 如果运行的线程等于或者多于corePoolSize，则Executor始终首选将请求加入队列，而不是添加新线程
3. 如果无法将请求加入队列，即队列已经满了，则创建新的线程，除非创建此线程超出maxinumPoolSize，**在这种情况下，任务将被拒绝。**

限于篇幅有限，更多高并发编程中的问题，请各位参阅我的博客：

- Java多线程编程实战指南（核心篇）读书笔记（一）
- Java多线程编程实战指南（核心篇）读书笔记（二）
- Java多线程编程实战指南（核心篇）读书笔记（三）
- Java多线程编程实战指南（核心篇）读书笔记（四）
- Java多线程编程实战指南（核心篇）读书笔记（五）

> 小结：本小节内容涉及到Java中多线程编程，线程安全等知识，是面试中的重点和难点。

（四）JVM内存管理

既然是Java开发面试，那么对JVM的考察当然也是必须的，面试官一般会问你对JVM有了解吗？我通常都会把我所了解的都说一遍，包括：JVM内存划分、JVM垃圾回收的含义，有哪些GC算法，年轻代和老年代各自的特点统统阐述一遍。

1）JVM内存划分：

1. 方法区（线程共享）：常量、静态变量、JIT(即时编译器)编译后的代码也都在方法区
2. 堆内存（线程共享）：垃圾回收的主要场所。
3. 程序计数器： 当前线程执行的字节码的位置指示器。
4. 虚拟机栈（栈内存）：保存局部变量、基本数据类型变量以及堆内存中某个对象的引用变量
5. 本地方法栈： 为JVM提供使用native 方法的服务

2）类似-Xms、-Xmn这些参数的含义：

答：

堆内存分配：

1. JVM初始分配的内存由-Xms指定，默认是物理内存的1/64
2. JVM最大分配的内存由-Xmx指定，默认是物理内存的1/4
3. 默认空余堆内存小于40%时，JVM就会增大堆直到-Xmx的最大限制；空余堆内存大于70%时，JVM会减少堆直到 -Xms的最小限制。
4. 因此服务器一般设置-Xms、-Xmx相等以避免在每次GC 后调整堆的大小。对象的堆内存由称为垃圾回收器的自动内存管理系统回收。

非堆内存分配：

1. JVM使用-XX:PermSize设置非堆内存初始值，默认是物理内存的1/64；
2. 由XX:MaxPermSize设置最大非堆内存的大小，默认是物理内存的1/4。
3. -Xmn2G：设置年轻代大小为2G。
4. -XX:SurvivorRatio，设置年轻代中Eden区与Survivor区的比值。

3）垃圾回收算法有哪些?

答：

1. **引用计数**：原理是此对象有一个引用，即增加一个计数，删除一个引用则减少一个计数。垃圾回收时，只用收集计数为0的对象。此算法最致命的是无法处理循环引用的问题。
2. **标记-清除**：此算法执行分两阶段。第一阶段从引用根节点开始标记所有被引用的对象，第二阶段遍历整个堆，把未标记的对象清除。此算法需要暂停整个应用，同时，会产生内存碎片。
3. **复制算法**：此算法把内存空间划为两个相等的区域，每次只使用其中一个区域。垃圾回收时，遍历当前使用区域，把正在使用中的对象复制到另外一个区域中。此算法每次只处理正在使用中的对象，因此复制成本比较小，同时复制过去以后还能进行相应的内存整理，不会出现“碎片”问题。当然，此算法的缺点也是很明显的，就是需要两倍内存空间。
4. **标记-整理**：此算法结合了“标记-清除”和“复制”两个算法的优点。也是分两阶段，第一阶段从根节点开始标记所有被引用对象，第二阶段遍历整个堆，把清除未标记对象并且把存活对象“压缩”到堆的其中一块，按顺序排放。此算法避免了“标记-清除”的碎片问题，同时也避免了“复制”算法的空间问题。

4）root搜索算法中，哪些可以作为root?

答：

- 被启动类（bootstrap加载器）加载的类和创建的对象
- JavaStack中的引用的对象(栈内存中引用的对象)。
- 方法区中静态引用指向的对象。
- 方法区中常量引用指向的对象。
- Native方法中JNI引用的对象。

5）GC什么时候开始?

答：GC经常发生的区域是堆区，堆区还可以细分为新生代、老年代，新生代还分为一个Eden区和两个Survivor区。

1. 对象优先在Eden中分配，当Eden中没有足够空间时，虚拟机将发生一次Minor GC，因为Java大多数对象都是朝生夕灭，所以Minor GC非常频繁，而且速度也很快；
2. Full GC，发生在老年代的GC，当老年代没有足够的空间时即发生Full GC，发生Full GC一般都会有一次Minor GC。大对象直接进入老年代，如很长的字符串数组，虚拟机提供一个-XX:PretenureSizeThreadhold参数，令大于这个参数值的对象直接在老年代中分配，避免在Eden区和两个Survivor区发生大量的内存拷贝；
3. 发生Minor GC时，虚拟机会检测之前每次晋升到老年代的平均大小是否大于老年代的剩余空间大小，如果大于，则进行一次Full GC，如果小于，则查看HandlePromotionFailure设置是否允许担保失败，如果允许，那只会进行一次Minor GC，如果不允许，则改为进行一次Full GC。

6）内存泄漏和内存溢出

答：

概念：

1. 内存溢出指的是内存不够用了。
2. 内存泄漏是指对象可达，但是没用了。即本该被GC回收的对象并没有被回收
3. 内存泄露是导致内存溢出的原因之一；内存泄露积累起来将导致内存溢出。

内存泄漏的原因分析：

1. 长生命周期的对象引用短生命周期的对象
2. 没有将无用对象置为null

> 小结：本小节涉及到JVM虚拟机，包括对内存的管理等知识，相对较深。除了以上问题，面试官会继续问你一些比较深的问题，可能也是为了看看你的极限在哪里吧。比如：内存调优、内存管理，是否遇到过内存泄漏的实际案例、是否真正关心过内存等。由于本人实际项目经验不足，这些深层次问题并没有接触过，各位有需要可以上网查阅。

（五）Java 8相关知识点

关于Java8中新知识点，面试官会让你说说Java8你了解多少，下边主要阐述我所了解，并且在面试中回答的Java8新增知识点。

1）HashMap的底层实现有变化：HashMap是数组+链表+红黑树（JDK1.8增加了红黑树部分）实现。

2）JVM内存管理方面，由元空间代替了永久代。

区别：

1. 元空间并不在虚拟机中，而是使用本地内存
2. 默认情况下，元空间的大小仅受本地内存限制
3. 也可以通过-XX：MetaspaceSize指定元空间大小

3）Lambda表达式（也称为闭包），允许我们将函数当成参数传递给某个方法，或者把代码本身当做数据处理。

4）函数式接口：指的是只有一个函数的接口，java.lang.Runnable和java.util.concurrent.Callable就是函数式接口的例子；java8提供了一个特殊的注解@FunctionalInterface来标明该接口是一个函数式接口。

5）引入重复注解：Java 8中使用@Repeatable注解定义重复注解。

6）接口中可以实现方法default方法。

7） 注解的使用场景拓宽： 注解几乎可以使用在任何元素上：局部变量、接口类型、超类和接口实现类，甚至可以用在函数的异常定义上。

8） 新的包java.time包

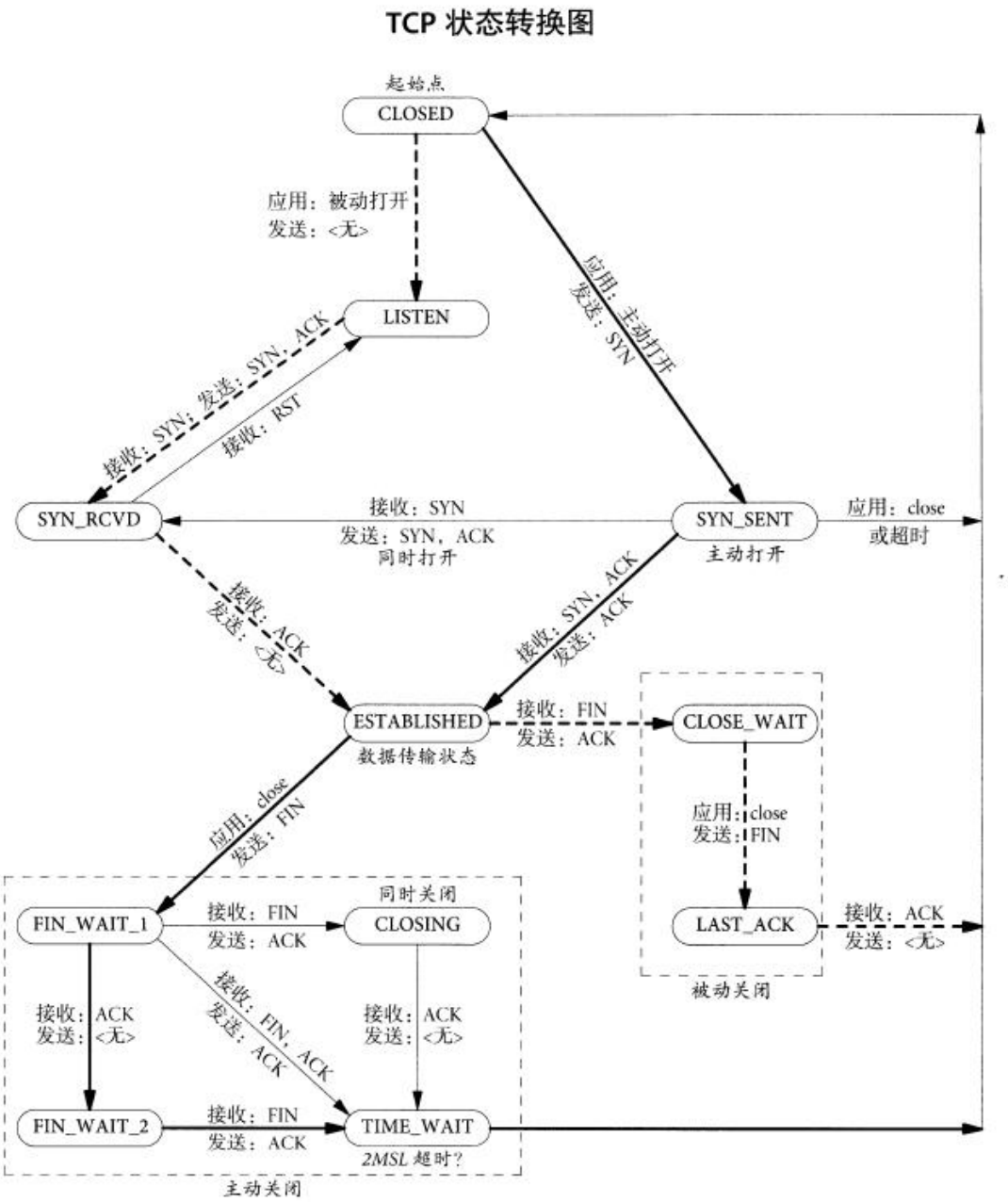
1. 包含了所有关于日期、时间、时区、持续时间和时钟操作的类
2. 这些类都是不可变的、线程安全的。

> 小结：Java8的一些新特性，面试官一般情况下不要求你有多么精通，主要是看看你有没有一些了解。

（六）网络协议相关

> 网络协议方面，考察最多的包括服务器和客户端在三次握手、四次挥手过程中的状态变化；还有网络拥塞控制，及其解决办法等。

1）三次握手、四次挥手示意图：



总共有四种状态：主动建立连接、主动断开连接、被动建立连和被动断开连接 两两组合还是4种组合：

1. 主动建立连接、主动断开连接会经历的状态： SYNC*SENT*----*ESTABLISHED*---*FIN*WAIT*1*---*FIN*WAIT*2*---*TIME*WAIT
2. 主动建立连接、被动断开连接会经历的状态： SYNC*SENT*----*ESTABLISHED*---*CLOSE*WAIT---LAST_ACK
3. 被动建立连接、主动断开连接会经历的状态： LISTEN---SYN*RCVD*---*ESTABLISHED*---*FIN*WAIT*1*---*FIN*WAIT*2*---*TIME*WAIT
4. 被动建立连接、被动断开连接会经历的状态： LISTEN---SYN*RCVD*---*ESTABLISHED*---*CLOSE*WAIT---LAST_ACK

2）滑动窗口机制

由发送方和接收方在三次握手阶段，互相将自己的最大可接收的数据量告诉对方。也就是自己的数据接收缓冲池的大小。这样对方可以根据已发送的数据量来计算是否可以接着发送。在处理过程中，当接收缓冲池的大小发生变化时，要给对方发送更新窗口大小的通知。

3）拥塞避免机制

拥塞：

对资源的需求超过了可用的资源。若网络中许多资源同时供应不足，网络的性能就要明显变坏，整个网络的吞吐量随之负荷的增大而下降。

拥塞控制： 防止过多的数据注入到网络中，使得网络中的路由器或链路不致过载。

拥塞控制方法：

- 慢开始+拥塞避免
- 快重传+快恢复

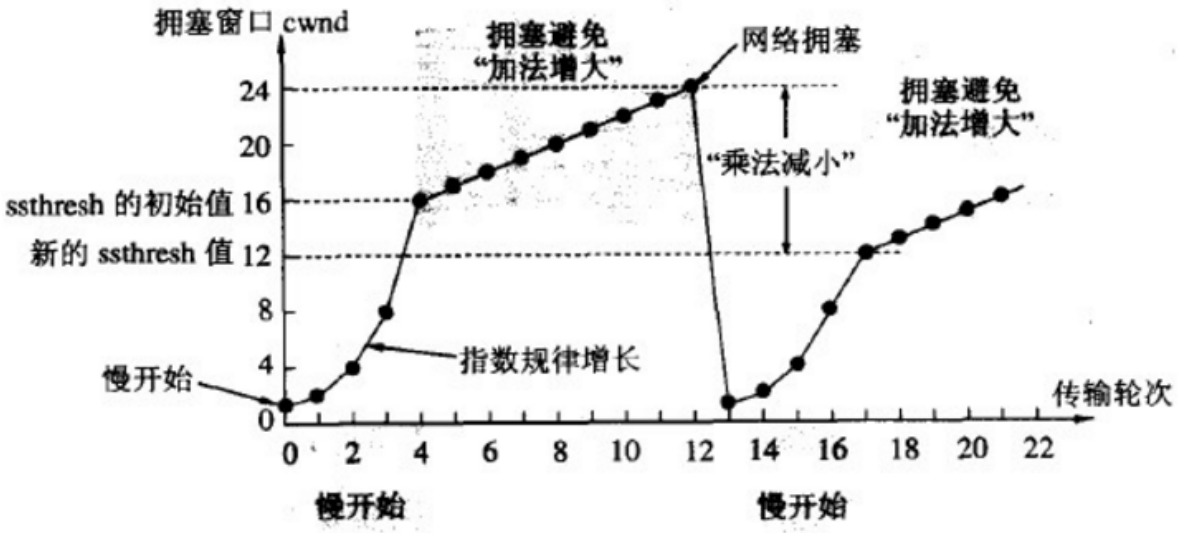


图 5-25　慢开始和拥塞避免算法的实现举例

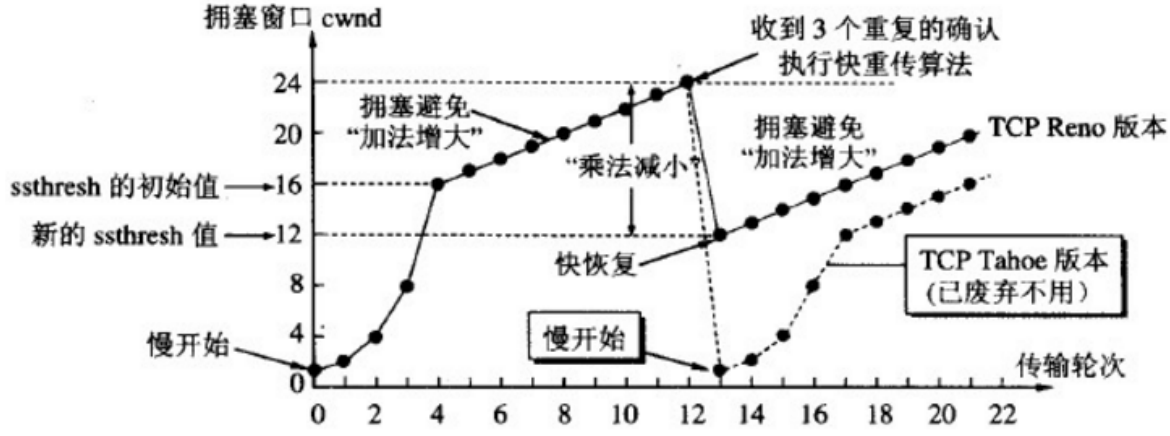


图 5-27　从连续收到三个重复的确认转入拥塞避免

4）浏览器中输入："www.xxx.com"之后都发生了什么？请详细阐述。

解析：经典的网络协议问题。

答：

1. 由域名→IP地址 寻找IP地址的过程依次经过了浏览器缓存、系统缓存、hosts文件、路由器缓存、递归搜索根域名服务器。
2. 建立TCP/IP连接（三次握手具体过程）

3. 由浏览器发送一个HTTP请求
4. 经过路由器的转发，通过服务器的防火墙，该HTTP请求到达了服务器
5. 服务器处理该HTTP请求，返回一个HTML文件
6. 浏览器解析该HTML文件，并且显示在浏览器端
7. 这里需要注意：

- HTTP协议是一种基于TCP/IP的应用层协议，进行HTTP数据请求必须先建立TCP/IP连接
- 可以这样理解：HTTP是轿车，提供了封装或者显示数据的具体形式；Socket是发动机，提供了网络通信的能力。
- 两个计算机之间的交流无非是两个端口之间的数据通信,具体的数据会以什么样的形式展现是以不同的应用层协议来定义的。

5）常见HTTP状态码

1. 1xx（临时响应）
2. 2xx（成功）
3. 3xx（重定向）：表示要完成请求需要进一步操作
4. 4xx（错误）：表示请求可能出错，妨碍了服务器的处理
5. 5xx（服务器错误）：表示服务器在尝试处理请求时发生内部错误
6. 常见状态码：

- 200（成功）
- 304（未修改）：自从上次请求后，请求的网页未修改过。服务器返回此响应时，不会返回网页内容
- 401（未授权）：请求要求身份验证
- 403（禁止）：服务器拒绝请求
- 404（未找到）：服务器找不到请求的网页

6）TCP和UDP的区别：

答：

1. 回答发送数据前是否存在建立连接的过程。
2. ＴＣＰ过确认机制，丢包可以重发，保证数据的正确性；ＵＤＰ不保证正确性，只是单纯的负责发送数据包。
3. UDP是面向报文的。发送方的UDP对应用程序交下来的报文，在添加首部后就向下交付给IP层。既不拆分，也不合并，而是保留这些报文的边界，因 此，应用程序需要选择合适的报文大小。
4. UDP的头部，只有8个字节，相对于TCP头部的20个字节信息包的额外开销很小。

限于篇幅，更多网络协议相关知识，请参阅我的博客：[TCP/IP协议面试常问知识点，倾心总结]

> 小结：必须熟练掌握TCP和UDP的区别、三次握手和四次挥手的状态切换，必考。

（七）数据库知识点

> 既然是后端开发，那么与数据库相关的知识点也是必不可少的。

1）MySQL和MongoDB的区别有哪些？如何选择?

2）MongoDB的优缺点有哪些?

（ps本人对这一块不是很熟悉，就不附上参考答案了，请各位小伙伴自行学习哈~）

3）听说过事务吗？（必考）**

答：作为单个逻辑工作单元执行的一系列操作，满足四大特性：

1. 原子性（Atomicity）：事务作为一个整体被执行，要么全部执行，要么全部不执行
2. 一致性（Consistency）：保证数据库状态从一个一致状态转变为另一个一致状态
3. 隔离性（Isolation）：多个事务并发执行时，一个事务的执行不应影响其他事务的执行
4. 持久性（Durability）：一个事务一旦提交，对数据库的修改应该永久保存

4）事务的并发问题有哪几种?

答：丢失更新、脏读、不可重复读以及幻读。

5）数据库中的锁有哪几种? 答：独占锁、排他锁以及更新锁。

6）事务的隔离级别有哪几种?

答：读未提交、读已提交、可重复读和序列化。

扩展问题：MySQL事务默认隔离级别是哪个?

答：可重复读

解析：关于问题（4）（5）（6）的详细解答，请参阅我的博客：数据库并发机制和事务的隔离级别详解

（ps，关于数据库事务方面的深层次考察还有分布式事务即两段提交和三段提交等，限于本人水平，请各位自行学习）

7）数据库的索引有什么作用？（必考）**底层数据结构是什么，为什么使用这种数据结构?**

答：

1. **索引**是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。
2. 底层数据结构是B+树。
3. 使用B+树的原因：查找速度快、效率高，在查找的过程中，每次都能抛弃掉一部分节点，减少遍历个数。（**此时，你应该在白纸上画出什么是B+树**）

扩展问题：聚簇索引和非聚簇索引的区别?

8）MyISAM和InnoDB的区别有哪些?

答：

- MyISAM不支持事务，InnoDB是事务类型的存储引擎
- MyISAM只支持表级锁，BDB支持页级锁和表级锁，默认为页级锁；而InnoDB支持行级锁和表级锁，默认为行级锁
- MyISAM引擎不支持外键，InnoDB支持外键
- MyISAM引擎的表在大量高并发的读写下会经常出现表损坏的情况
- 对于count( )查询来说MyISAM更有优势
- InnoDB是为处理巨大数据量时的最大性能设计，它的CPU效率可能是任何其它基于磁盘的关系数据库引擎所不能匹敌的
- MyISAM支持全文索引（FULLTEXT），InnoDB不支持
- MyISAM引擎的表的查询、更新、插入的效率要比InnoDB高

> 最主要的区别是：MyISAM表不支持事务、不支持行级锁、不支持外键。 InnoDB表支持事务、支持行级锁、支持外键。**（可直接回答这个）**

9）数据库中Where、group by、having关键字：

答： **关键字作用：**

1. where子句用来筛选from子句中指定的操作所产生的的行
2. group by 子句用来分组where子句的输出
3. having子句用来从分组的结果中筛选行

having和where的区别：

1. 语法类似，where搜索条件在进行分组操作之前应用；having搜索条件在进行分组操作之后应用。
2. having可以包含聚合函数sum、avg、max等；
3. having子句限制的是组，而不是行

当同时含有where子句、group by 子句 、having子句及聚集函数时，执行顺序如下：

1. 执行where子句查找符合条件的数据
2. 使用group by 子句对数据进行分组；对group by 子句形成的组运行聚集函数计算每一组的值；最后用having 子句去掉不符合条件的组

10）还有一些问题，如MySQL和SQL Server用法上的区别、limit关键字的使用等问题。

> 小结：数据库方面还是事务机制、隔离级别比较重要，当然了数据库索引是必考的问题。偶尔也会给你几个表，让你现场写SQL语句，主要考察group by 和having等关键字。

（八）MVC框架相关知识点

> 我在项目中使用的框架有Spring MVC和MyBatis，所以在简历上只写了这两种框架，面试官主要针对这两种框架进行提问。以下问题供小伙伴们参考。

JavaWeb开发经典的3层框架：Web层、Service层（业务逻辑层）和Dao层（数据访问层）

- Web层：包含JSP和Servlet等与Web相关的内容
- 业务层：只关心业务逻辑
- 数据层：封装了对数据库的访问细节

Spring知识点

1）Spring的IOC和AOP有了解吗?

答：

- IOC：控制反转，（解耦合）将对象间的依赖关系交给Spring容器，使用配置文件来创建所依赖的对象，由主动创建对象改为了被动方式。
- AOP：面向切面编程，将功能代码从业务逻辑代码中分离出来。

2）AOP的实现方式有哪几种？如何选择？（必考）**

答：JDK动态代理实现和cglib实现。

选择：

1. 如果目标对象实现了接口，默认情况下会采用JDK的动态代理实现AOP，也可以强制使用cglib实现

AOP

2. 如果目标对象没有实现接口，必须采用cglib库，Spring会自动在JDK动态代理和cglib之间转换

扩展：JDK动态代理如何实现？（加分点）**

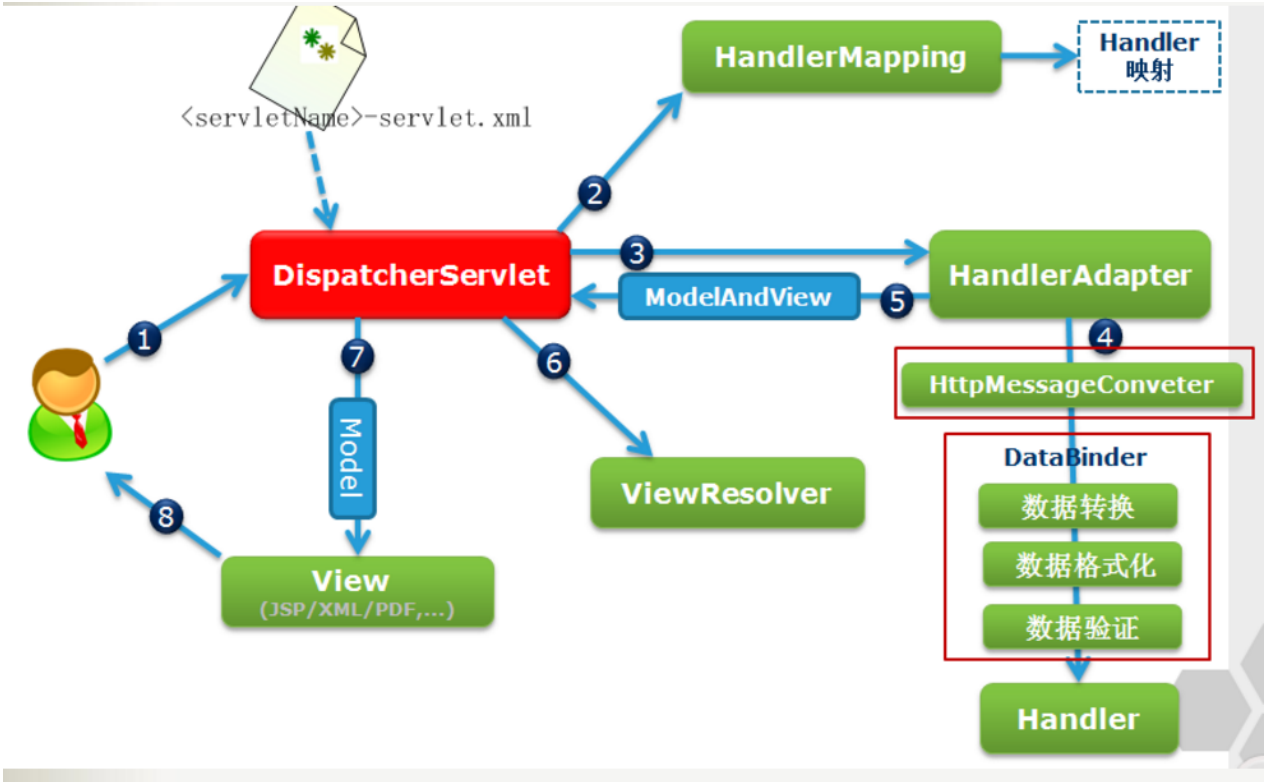
答：JDK动态代理，只能对实现了接口的类生成代理，而不是针对类，该目标类型实现的接口都将被代理。原理是通过在运行期间创建一个接口的实现类来完成对目标对象的代理。

1. 定义一个实现接口InvocationHandler的类
2. 通过构造函数，注入被代理类
3. 实现invoke（ Object proxy, Method method, Object[] args）方法
4. 在主函数中获得被代理类的类加载器
5. 使用Proxy.newProxyInstance( )产生一个代理对象
6. 通过代理对象调用各种方法

解析：关于IOC和AOP的详细阐述，请各位参阅我的博客：Spring核心AOP（面向切面编程）总结，Spring 框架学习—控制反转（IOC）

3）Spring MVC的核心控制器是什么？消息处理流程有哪些?

答：核心控制器为DispatcherServlet。消息流程如下：



4）其他问题包括：重定向和转发的区别、动态代理和静态代理的区别等。

Mybatis知识点

关于MyBatis主要考察占位符#和$的区别，区别如下：

1. #符号将传入的数据都当做一个字符串，会对自动传入的数据加一个双引号
2. $符号将传入的数据直接显示生成SQL中。
3. #符号存在预编译的过程，，对问号赋值，防止SQL注入。
4. 符号是直译的方式，一般用在orderby符号是直译的方式，一般用在orderby{列名}语句中。
5. 能用#号就不要用$符号

> 小结：限于作者水平，MVC框架方面了解不是太多，实战能力欠缺。面试官偶尔问框架底层实现原理等都知之甚少，有能力的小伙伴可以多加学习。

（九）大数据相关知识点

> 大数据相关是因为我的简历上写了KafKa相关项目，所以面试官会进行提问KafKa相关知识点，我也进行了一些简单概念总结，深层次的实现原理因为并没有特别多的实战经验，所以并不了解。以下概念总结供小伙伴参考。

1）KafKa基本特性：

答：快速持久化、支持批量读写消息、支持消息分区，提高了并发能力、支持在线增加分区、支持为每个分区创建多个副本。

扩展：为什么可以实现快速持久化?

答：KafKa将消息保存在磁盘中，并且读写磁盘的方式是顺序读写，避免了随机读写磁盘（寻道时间过长）导致的性能瓶颈；磁盘的顺序读写速度超过内存随机读写。

2）核心概念：

答：

- **生产者（Producer）：** 生产消息，并且按照一定的规则推送到Topic的分区中。
- **消费者（Consumer）：** 从Topic中拉去消息，并且进行消费。
- **主题（Topic）：** 用于存储消息的逻辑概念，是一个消息集合。
- **分区（partition）：**

1. 每个Topic可以划分为多个分区，每个消息在分区中都会有一个唯一编号offset
2. kafka通过offset保证消息在分区中的顺序
3. 同一Topic的不同分区可以分配在不同的Broker上
4. partition以文件的形式存储在文件系统中。

副本（replica）：

1. KafKa对消息进行了冗余备份，每个分区有多个副本，每个副本中包含的消息是“一样”的。
2. 每个副本中都会选举出一个Leader副本，其余为Follower副本，Follower副本仅仅将数据从Leader副本拉去到本地，然后同步到自己的Log中。

消费者组（Consumer Group）： 每个 consumer 都属于一个 consumer group，每条消息只能被 consumer group 中的一个 Consumer 消费，但可以被多个 consumer group 消费。

Broker：

1. 一个单独的server就是一个Broker
2. 主要工作：接收生产者发过来的消息，分配offset，并且保存到磁盘中；

Cluster&Controller：

1. 多个Broker可以组成一个Cluster，每个集群选举一个Broker来作为Controller，充当指挥中心
2. Controller负责管理分区的状态，管理每个分区的副本状态，监听ZooKeeper中数据的变化等工作

保留策略和日志压缩：

1. 不管消费者是否已经消费了消息，KafKa都会一直保存这些消息（持久化到磁盘）

2. 通过保留策略，定时删除陈旧的消息
3. 日志压缩，只保留最新的Key-Value对

关于副本机制：（加分点）**

ISR集合：表示当前"可用"且消息量与Leader相差不多的副本集合。满足条件如下：

1. 副本所在节点必须维持着与ZooKeeper的连接
2. 副本最后一条信息的offset与Leader副本的最后一条消息的offset之间的差值不能超过指定的阈值

HW&LEO：

1. HW标记了一个特殊的offset，当消费者处理消息的时候，只能拉取到HW之前的消息
2. HW也是由Leader副本管理的
3. LEO（Log End Offset）是所有副本都会有的一个offset标记

ISR、HW和LEO的工作配合：

1. producer向此分区中推送消息
2. Leader副本将消息追加到Log中，并且递增其LEO
3. Follower副本从Leader副本中拉取消息进行同步
4. Follower副本将消息更新到本地Log中，并且递增其LEO
5. 当ISR集合中的所有副本都完成了对offset的消息同步，Leader副本会递增其HW

KafKa的容灾机制：通过分区的副本Leader副本和Follower副本来提高容灾能力

> 小结：请小伙伴根据自己的简历自行准备学习大数据相关知识点。

（十）Linux常见命令

> 作者对这一方面不是很精通，知识点来源于网络总结以及面试官的提问，仅供小伙伴参考。

1）grep、sed以及awk命令

解析：awk命令如果可以掌握，是面试中的一个**加分点。**

2）文件和目录：

> pwd 显示当前目录

ls 显示当前目录下的文件和目录：

1. ls -F 可以区分文件和目录
2. ls -a 可以把隐藏文件和普通文件一起显示出来
3. ls -R 可以递归显示子目录中的文件和目录
4. ls -l 显示长列表
5. ls -l test 过滤器，查看某个特定文件信息。可以只查看test文件的信息

3）处理文件方面的命令有：touch、cp、 ln、mv、rm、

4）处理目录方面的命令：mkdir

5）查看文件内容：file、cat、more、less、tail、head

6）监测程序命令：ps、top

eg.找出进程名中包括java的所有进程：ps -ef | grep java

top命令 实时监测进程

top命令输出的第一部分：显示系统的概括。

1. 第一行显示了当前时间、系统的运行时间、登录的用户数和系统的平均负载（平均负载有3个值：最近1min 5min 15min）
2. 第二行显示了进程的概要信息，有多少进程处于运行、休眠、停止或者僵化状态。
3. 第三行是CPU的概要信息
4. 第四行是系统内存的状态

7）ps和top命令的区别：

1. ps看到的是命令执行瞬间的进程信息,而top可以持续的监视
2. ps只是查看进程,而top还可以监视系统性能,如平均负载,cpu和内存的消耗
3. 另外top还可以操作进程,如改变优先级(命令r)和关闭进程(命令k)
4. ps主要是查看进程的，关注点在于查看需要查看的进程
5. top主要看cpu,内存使用情况，及占用资源最多的进程由高到低排序，关注点在于资源占用情况

8） 压缩数据

1. tar -xvf 文件名
2. tar -zxvf 文件名
3. tar -cvzf 文件名

9）结束进程：kill PID或者kill all

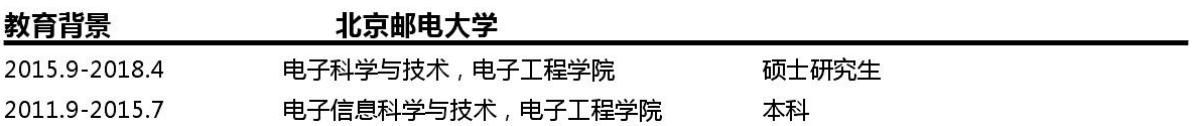
至此，从十个不同的方面阐述了Java开发面试岗位中所涉及到的重要知识点。加上我上次发布的关于算法面试的chat，我大概将最近一年的时间内的面试笔试经验给大家做了总结分享。接下来，为了给大家提供更多的帮助，我想针对简历方面和大家聊聊，主要包括：**制作简历**和**投递简历**两方面。

制作简历：

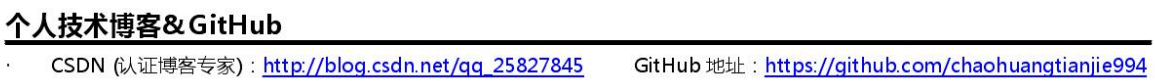
首先，我想先介绍下我的简历都包括哪些部分：

1. 个人信息：

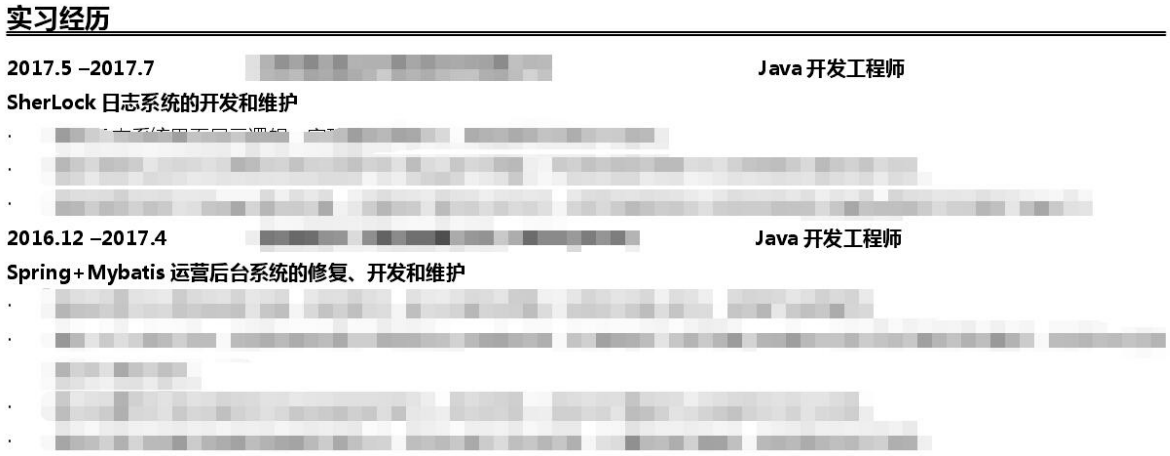
2. 教育背景：



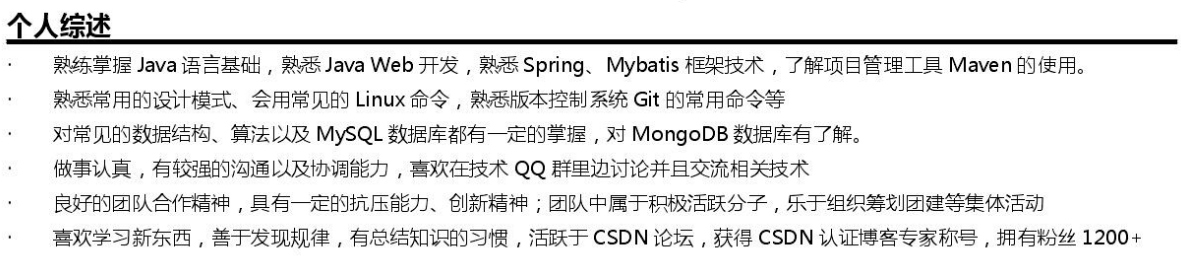
3. 个人亮点：



4. 实习项目：



5. 校内项目
6. 奖励、语言及IT能力
7. 个人综述：



简历应该突出表现自己优秀的地方，所以我将个人技术博客和github地址放到了相对靠前的位置，使面试官或者HR可以一眼看到。各位小伙伴如果在校期间参加过什么大型竞赛并且获奖，也可以写在明显的位置。

给大家一个小提示，那就是必须对简历上所写的知识点有一定了解，不懂的就不要写上去了，因为你会被问的很惨（即使一面面试官不问，三面面试官也会问的）。 比如说你在简历上写了一个技术A，说自己在项目中使用过技术A，那么面试官就会问该技术A的底层实现、原理等等。如果你回答出来，确实，这是加分项，但是很多时候我们是回答不出来的。

投递简历

投递简历同样很重要，对于并不是出身名校的小伙伴来说，在投递一些互联网公司的时候，可能会面临着简历被刷的可能。这时候内推的重要性就体现出来了。

内推是内部推荐的意思，**内推的好处一般有两种：**

1. 内推免简历筛选，直接进入笔试环节。
2. 内推可以直接获得面试资格，面试失败可以继续参加后续校招大流程。

内推时间点：

1. 暑期实习校园招聘的内推一般从每年3月份开始。
2. 秋招的内推从每年8月份开始，持续到九月份。

如何内推?

内推，意味着你要找到公司内部人员进行推荐，内推渠道主要是找自己的师兄师姐。另外可以时常关注号称“全国最大的高校论坛”-**北邮人论坛**。北邮人论坛资源相当丰富，每年招聘季的内推帖子数不胜数，各位如果有需要，可以关注一下。在牛客网以及赛码网的讨论区内也存在着大量的内推消息，大家可以关注。