**Final在java中的作用？**

Final可以修饰类、类属性、类方法和变量

Final修饰的类叫最终类，不能被继承；final修饰的方法不能被重写；final修饰的变量，必须初始化且初始化后值不能改变。

**抽象类能使用final继承吗？**

不能，定义抽象类就是为了让其他类继承的，如果定义为final该类就不能被继承。

**接口和抽象类有什么区别？**

实现：抽象类的子类使用extend是来继承；接口必须使用implements来实现接口；

构造函数：抽象类可以有构造函数，接口不能有；

实现数量：类可以实现很多个接口，但是只能继承一个抽象类；

访问修饰符：接口中的方法默认使用public修饰；抽象类中的方法可以是任意访问修饰符；

**BIO、NIO、AIO有什么区别？**

BIO：block IO,同步阻塞式IO，就是我们平常使用的传统IO，特点是模式使用方便，并发处理能力非常低；

NIO：New IO同步非阻塞IO，是传统IO 的升级，客户端和服务器端通过channel通讯，实现了多路复用；

AIO：Asynchronous IO是NIO的升级，也叫NIO2，实现了异步非阻塞IO，异步IO的操作基于事件和回调机制；

**Array和ArrayList有何区别？**

Array可以容纳基本类型和对象，而ArrayList只能容纳对象

Array是指定大小的，而ArrayList大小是不固定的；

Array没有提供ArrayList那么多功能

**Iterator和ListIterator有什么区别？**

Iterator可用来遍历Set和List集合，但是ListIterator只能用来遍历List；

Iterator对集合只能是前向遍历，ListIterator既可以前向又可以后向；

ListIterator实现了Iterator接口，并包含其他功能。

**并发和并行有什么区别？**

并行是指两个或多个事件在同一时刻发送，而并发是指两个或多个事件在同一时间间隔发生；

并行是在不同实体上的多个事件，并发是在同一实体上的多个事件；

**线程和进程的区别？**

进程是程序运行和资源分配的基本单位，一个程序至少有一个进程，一个进程至少有一个线程。进程在执行过程中拥有独立的内存单元，而多个线程共享内存资源，减少切换次数，从而效率更高。线程是进程的一个实体，是cpu调度和分派的基本单元，是比程序更小的能独立运行的基本单元。同一个进程中的多个线程可以并发执行。

**Java如何保证多线程安全？**

线程安全体现在三个方面：

原子性：提供互斥访问，同一时刻只能有一个线程对数据进行操作(atomic,synchronized)；

可见性：一个线程对主内存的修改可以及时被其他线程看到(synchronized,volatile)；

有序性：一个线程观察其他线程中的指令执行顺序，由于指令重排序，该观察结果一般杂乱无序。

**事务的四种隔离级别？**

事务隔离级别 脏读 不可重复度 幻读

读未提交(Read uncommited) 是 是 是

读提交(Read committed) 否 是 是

重复读(Reapeatable read) 否 否 是

串行化(serializable) 否 否 否

**脏读：**

读到别的事务回滚前的脏数据，例如事务B在执行过程中修改了数据x，在未提交前，事务A读取了数据x，而事务B却回滚了。这样事务A就形成了回滚。

**不可重复读：**

事务A首先读取了一条数据，然而执行逻辑的时候，事务B将这条数据改变了，然后事务A再次读取的时候，发现不匹配了，这就是所谓的不可重复读。

**幻读：**

事务A首先根据条件索引得到了N条数据，然后事务B改变了N条之外的数据M条或者添加了M条符合事务A搜索条件的数据，导致事务A再次搜索发现有N+M条数据，也就是读取了其他事务新增的数据。

**ACID的区别？**

**Atomicity:原子性**，一个事务(transaction)中的所有操作，或者全部完成，或者全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被恢复(Rollback)到事务开始前的状态，就像这个事务从来没有执行过一样，即事务不可分割、不可简约。

**Consistency(一致性)**：在事务开始之前和事务结束之后，数据库的完整性没有被破坏。

**Isolation(隔离性)**：数据库允许多个并发事务同时对其诗句进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。

**Durability(持久性)**：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

**数据库的三范式是什么？**

第一范式：强调的是列的原子性，即数据库表的第一列都是不可分割的原子数据项。

第二范式：要求实体的属性完全依赖于主关键字。所谓完全依赖导师制不能存在仅依赖关键字一部分的属性。

第三属性：任何非属性不依赖于其他非主属性。

**Mysql的内连接、左连接、右连接的区别？**

内连接关键字：inner join;左连接:left join;右连接:right join

内连接是把匹配的关联数据显示出来；左连接是左边的表全部显示出来，右边的表显示符合条件的数据；右连接是把右边的表全部显示出来，左边的表显示符合条件的数据。

**线程死锁的四个必要条件？**

**互斥条件：**进程要求对所分配的资源进行排他性控制，即在一段时间内某资源仅为一个进程所占用。此时若有其他进程请求该资源，则请求进程智能等待。

**不可剥夺条件：**进程所获得的资源在未使用完毕之前，不能被其他进程强行夺走，即只能由获得该资源的进程自己来释放。

**请求与保持条件：**进程已经保持了至少一个条件，但又提出了新的资源请求，而该资源已被其他进程占有，此时请求进程被阻塞，但对自己已获得的资源保持不变。

**循环等待条件：**存在一种进程资源的循环等待链，链中每一个进程已获得的资源同时被链中下一个进程请求。

**Synchronized底层原理？**

synchronized修饰的方法在字节码中添加了一个ACC\_SYNCHRONIZED的flags，同步代码块则是在同步代码块前插入monitorenter，在同步代码块结束后插入monitorexit。这两者的处理是分别是这样的：当线程执行到某个方法时，JVM会去检查该方法的ACC\_SYNCHRONIZED访问标志是否被设置，如果设置了那线程会去获取这个对象所对应的monitor对象（每一个对象都有且仅有一个与之对应的monitor对象）,获取成功后才执行方法体，方法执行完再释放monitor对象，在这一期间，任何其他线程都无法获得这个monitor对象。而线程执行同步代码块时遇到的monitorenter和monitorexit指令依赖monitor对象完成。这两者实现的方式本质上无区别，只是方法的同步是一种隐式的方式，不通过字节码实现

**Volatile底层原理？**

Volatile使用了lock指令前缀，能确保在多处理器系统或多线程竞争的环境下互斥地使用这个内存地址。当指令执行完毕，这个锁定动作也就会消失。lock前缀指令其实就相当于一个内存屏障。

内存屏障是一组CPU处理指令，用来实现对内存操作的顺序限制。volatile的底层就是通过内存屏障来实现的。编译器和执行器 可以在保证输出结果一样的情况下对指令重排序，使性能得到优化。插入一个内存屏障，相当于告诉CPU和编译器先于这个命令的必须先执行，后于这个命令的必须后执行。

内存屏障细说来有写屏障、读屏障、读写屏障，而且内存屏障的实现依赖于编译器和机器两部分。

**synchronized 和 volatile 的区别是什么？**

Volatile的本质是告诉jvm当前变量在寄存器/工作内存中的值是不确定的，需要从主存中读取；synchronized则是锁定当前变量，只有当前线程可以访问该变量，其他线程被阻塞。

Volatile仅能使用在变量级别；synchronized则可以使用在变量、方法、和类级别的。

Volatile仅能实现变量的修改可见性，不能保证原子性；而synchronized则可以保证变量的修改可见性和原子性。

Volatile不会造成线程的阻塞；synchronized可能会造成线程的阻塞。

Volatile标记的变量不会被编译器优化；synchronized标记的变量可以被编译器优化。

**Synchronized和Lock有什么区别？**

Synchronized是java内置关键字，在jvm层面，Lock是个java类；

Synchronized无法判断是否获取锁的状态，Lock可以判断是否获取到锁；

Synchronized会自动释放锁，lock需要在finally中手工释放锁(unlock)，否则容易造成死锁；

用Synchronized关键字的两个线程1和线程2，如果当前线程1获得锁，线程2线程等待。如果线程1阻塞，线程2则会一直等待下去，而lock锁不一定会等待下去，如果尝试获取不到锁，线程可以不用一直等待就结束了了。

Synchronized的锁可重入、不可中断、非公平，而lock锁可重入、可判断、可公平。

Lock锁适合大量同步的代码同步问题，Synchronized适合代码少量的同步问题。

Java锁有哪些？

Synchronized、volatile、自旋锁、偏向锁、轻量级锁、重量级锁

自旋锁：如果持有锁的线程能在很短时间内释放锁资源，那么那些等待竞争锁的线程就不需要做内核态和用户态之间的切换进入阻塞挂起状态，它们只需要等一等（自旋），等持有锁的线程释放锁后即可立即获取锁，这样就避免用户线程和内核的切换的消耗。

偏向锁：（顾名思义，它会偏向于第一个访问锁的线程，如果在运行过程中，同步锁只有一个线程访问，不存在多线程争用的情况，则线程是不需要触发同步的，这种情况下，就会给线程加一个偏向锁）

轻量级锁：轻量级锁是由偏向所升级来的，偏向锁运行在一个线程进入同步块的情况下，当第二个线程加入锁争用的时候，偏向锁就会升级为轻量级锁。

**Concurrenthashtable的get put?**

1.7 由segment和hashentry组成，数组+链表

Put方法：

首先通过key定位到segment,之后在对应的segment中进行具体的put，hashentry中的value都是volatile 关键词修饰，但是不能保证并发的原子性，所以put时会尝试获取锁；(获取失败，存在其他锁竞争，自旋获取锁)

将当前 Segment 中的 table 通过 key 的 hashcode 定位到 HashEntry。

遍历该 HashEntry，如果不为空则判断传入的 key 和当前遍历的 key 是否相等，相等则覆盖旧的 value。

不为空则需要新建一个HashEntry并加入到Segment中，同时会先判断是否需要扩容。

最后会解除在最开始所获取当前 Segment 的锁。

Get方法：

只需要将 Key 通过 Hash 之后定位到具体的 Segment ，再通过一次 Hash 定位到具体的元素上。

由于 HashEntry 中的 value 属性是用 volatile 关键词修饰的，保证了内存可见性，所以每次获取时都是最新值。

ConcurrentHashMap的 et 方法是非常高效的，**因为整个过程都不需要加锁**。

1.8 cas+synchronized hashentry改为Node

Put :cas写入数据，失败则自旋保证成功

Get:

根据计算出来的hashcode寻址，如果就在桶上那么直接返回值。

如果是红黑树那就按照树的方式获取值。

就不满足那就按照链表的方式遍历获取值。

**什么情况下出现栈内存溢出？**

如果线程请求的栈深度大于虚拟机所允许的深度，将抛出异常StackOverFlowError

**什么时候出发GC？**

新生代有一个Eden区和两个survivor区，如果空间不足就向其中一个survivor区上放，如果仍然放不下就会引发一次新生代minor gc；

每次在执行minor gc的时候应该对晋升到老年代的对象进行分析，如果晋升到老年代的平均大小大于老年代的剩余空间，则发生一次full gc。

Good morning/afternoon.

It is my great honor to have this opportunity to introduce myself .and I hope I could make a good performance today,eventually become a member of ZTE. Now let me introduce myself please. I am a third-year graduate student of nanjing university of posts and telecommunications,my name is dingliang ,24years old ,born in Jiangsu Changzhou .my major is electronics and communication engineering and I will graduate next June.

I got my bachelor degree after my graduation in the year of 2017.I spent most of my time on study, I have passed CET6.and I have acquired basic knowledge of my major during my school time.

In July 2018, I became a Java intern of Beijing telesound communication electronics. During this period, my professional level was further improved.

Zte is a professional global company, so I feel I can gain the most from working in this kind of environment. That's why I came here to compete for this position. I think I'm a good team player and I'm a person of great honesty to others.

That’s all.

Thank you for giving me the chance

Nanjing university of posts and telecommunications, located at no. 9, wenyuan road, qixia district, nanjing, is a comprehensive key university with electronic information as its feature, engineering as its main subject, science, engineering, economics, management, culture, education, art, law and other disciplines integrated, postdoctoral, doctoral, master's degree and other multi-level education coordinated development

**设计模式**

**单例模式**

简单点说就是一个应用程序中，某个类的实例对象只有一个，你没有办法去new，因为构造器是被private修饰的，一般通过getInstance()的方法来获取他们的实例。getInstance()的返回值是一个对象的引用，并不是一个新的实例。

**观察者模式**

对象间一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新

**装饰者模式**

对已有的业务逻辑进一步的封装，使其增加额外的功能

**适配器模式**

将两种完全不同的事物联系到一起

**工厂模式**

简单工厂模式：一个抽象的接口（抽象类），多个抽象接口的实现类，一个工厂类，用来实例化抽象的接口

工厂方法模式：有四个角色，抽象工厂模式，具体工厂模式，抽象产品模式，具体产品模式。不再是由一个工厂类去实例化具体的产品，而是由抽象工厂的子类去实例化产品

抽象工厂模式：

与工厂方法模式不同的是，工厂方法模式中的工厂只生产单一的产品，而抽象工厂模式中的工厂生产多个产品

**代理模式(proxy)**

**简单工厂和抽象工厂有什么区别？**

简单工厂模式：

这个模式本身很简单而且使用在业务较简单的情况下。一般用于小项目或者具体产品很少扩展的情况（这样工厂类才不用经常更改）。

它由三种角色组成：

工厂类角色：这是本模式的核心，含有一定的商业逻辑和判断逻辑，根据逻辑不同，产生具体的工厂产品。如例子中的Driver类。

抽象产品角色：它一般是具体产品继承的父类或者实现的接口。由接口或者抽象类来实现。如例中的Car接口。

具体产品角色：工厂类所创建的对象就是此角色的实例。在java中由一个具体类实现，如例子中的Benz、Bmw类。

抽象工厂模式：

可以这么说，它和工厂方法模式的区别就在于需要创建对象的复杂程度上。而且抽象工厂模式是三个里面最为抽象、最具一般性的。抽象工厂模式的用意为：给客户端提供一个接口，可以创建多个产品族中的产品对象。

来看看抽象工厂模式的各个角色（和工厂方法的如出一辙）：

抽象工厂角色： 这是工厂方法模式的核心，它与应用程序无关。是具体工厂角色必须实现的接口或者必须继承的父类。在java中它由抽象类或者接口来实现。

具体工厂角色：它含有和具体业务逻辑有关的代码。由应用程序调用以创建对应的具体产品的对象。在java中它由具体的类来实现。

抽象产品角色：它是具体产品继承的父类或者是实现的接口。在java中一般有抽象类或者接口来实现。

具体产品角色：具体工厂角色所创建的对象就是此角色的实例。在java中由具体的类来实现

JVM内存结构主要分为三大块：堆、栈和方法区。每个线程都有自己的一个虚拟机栈，栈保存着局部变量以及所有调用的方法的参数和返回值。其他线程无法访问该线程的栈中数据。栈仅能保存基本类型和对象引用，对象是存放在堆中的。堆内存和方法区中的静态变量等数据可以被线程共享，而同步处理针对的正是共享数据。

**Mysql调优方式？**

选择合适的存储引擎：InnoDB

充分使用索引，为搜索字段创建索引

避免使用select \*,列出要查询的字段

垂直分割表：把一个表中特别长的字段拆分成多张表，然后在查询时进行关联查询，这样可以有效的提高查询效率

使用union来代替临时表

**Java设计模式？分为三大类**

创建型模式（15种）：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式

结构型模式（11种）：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式

行为模式（11种）：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式

**Ioc和aop？**

Ioc：类与类之间的相互协作来完成特定的业务逻辑，每个类都需要负责管理与自己有交互的类的引用和依赖，代码将会变的异常难以维护和极度的高耦合，通过IOC将这些相互依赖对象的创建、协调工作交给Spring容器去处理，每个对象只需要关注其自身的业务逻辑关系就可以了。

Aop：利用横切技术，剖解开封装的对象内部，并将那些影响到多个类的公共行为封装到一个可重用模块，即方面，简单的来说，将与业务无关的，却为业务模块所共用的逻辑和责任封装起来，便于减少系统的重复代码，降低模块间的耦合度。

**内存泄漏和内存溢出？**

内存泄漏是被分配的对象无用但可达，GC无法回收

内存溢出是无法申请到足够的内存而产生的错误

内存泄漏场景：

创建和应用生命周期一样的单例对象

创建匿名内部类的静态对象

未关闭资源

长时间存在的集合容器中创建生命周期短的对象

内存溢出场景：

堆内存溢出

线程栈溢出（递归）

方法区内存溢出（反射，静态变量）

**使用线程池的好处？**

减少在创建和销毁线程上所花时间及系统资源的开销

如果不使用线程池，有可能造成系统创建大量线程而导致消耗完内存以及过度切换

降低资源消耗，提高响应速度，提高线程的可管理性

MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架，其主要就完成2件事情：

封装JDBC操作

利用反射打通Java类与SQL语句之间的相互转换

**Static关键字**

Static关键字表明一个成员变量或者是成员方法可以在没有所属类的实例变量的情况下访问。

**Java反射**

在java运行时环境中，对于任意一个类，都能知道这个类的所有属性和方法，对于任意一个对象，都能调用它的任意方法和属性。

**索引失效的情况？**

条件中有or

多列索引，不是使用的第一部分

Like查询是以%开头

需要类型转换

Where中索引列有运算

Where中索引列使用了函数

如果mysql觉得全表扫描更快时

**Cookie和session的区别？**

都是为了存储用户相关信息，cookie存储在本地，session存储在服务器，存储在服务器更加安全

**String，StringBuffer，StringBuilder区别？**

String的值是不可变的，每次对String的操作都会生成新的String对象，这样效率低下，浪费有限的内存空间

StringBuffer和StringBuilder类的对象能被多次修改，并且不产生新的未使用的对象，StringBuffer的线程是安全的，对字符串的操作都加了synchronized，Stringbuilder不保证线程安全。

**Java Start和run的区别？**

1） start：  
　　用start方法来启动线程，真正实现了多线程运行，这时无需等待run方法体代码执行完毕而直接继续执行下面的代码。通过调用Thread类的start()方法来启动一个线程，这时此线程处于就绪（可运行）状态，并没有运行，一旦得到cpu时间片，就开始执行run()方法，这里方法 run()称为线程体，它包含了要执行的这个线程的内容，Run方法运行结束，此线程随即终止。  
2） run：  
　　run()方法只是类的一个普通方法而已，如果直接调用Run方法，程序中依然只有主线程这一个线程，其程序执行路径还是只有一条，还是要顺序执行，还是要等待run方法体执行完毕后才可继续执行下面的代码，这样就没有达到写线程的目的。总结：调用start方法方可启动线程，而run方法只是thread的一个普通方法调用，还是在主线程里执行。这两个方法应该都比较熟悉，把需要并行处理的代码放在run()方法中，start()方法启动线程将自动调用 run()方法，这是由jvm的内存机制规定的。并且run()方法必须是public访问权限，返回值类型为void

**Get和Post的区别？**

Get在浏览器回退是无害的，post会再次提交请求，

Get请求会被浏览器主动缓存，post不可以，

Get只能进行url编码，post支持多种编码方式，

Get请求的参数会被保留在浏览器的历史记录里，post的参数不会，

Get请求在URL中传递的参数是有长度限制的，而POST没有，

Get比post更安全，参数直接暴露在url中，get通过url传参，而post通过request body

Get产生一个TCP数据包，post产生两个tcp数据包，对于get请求，浏览器会把http header和data一并发送出去，而对于post，浏览器先发送header，再发送data。