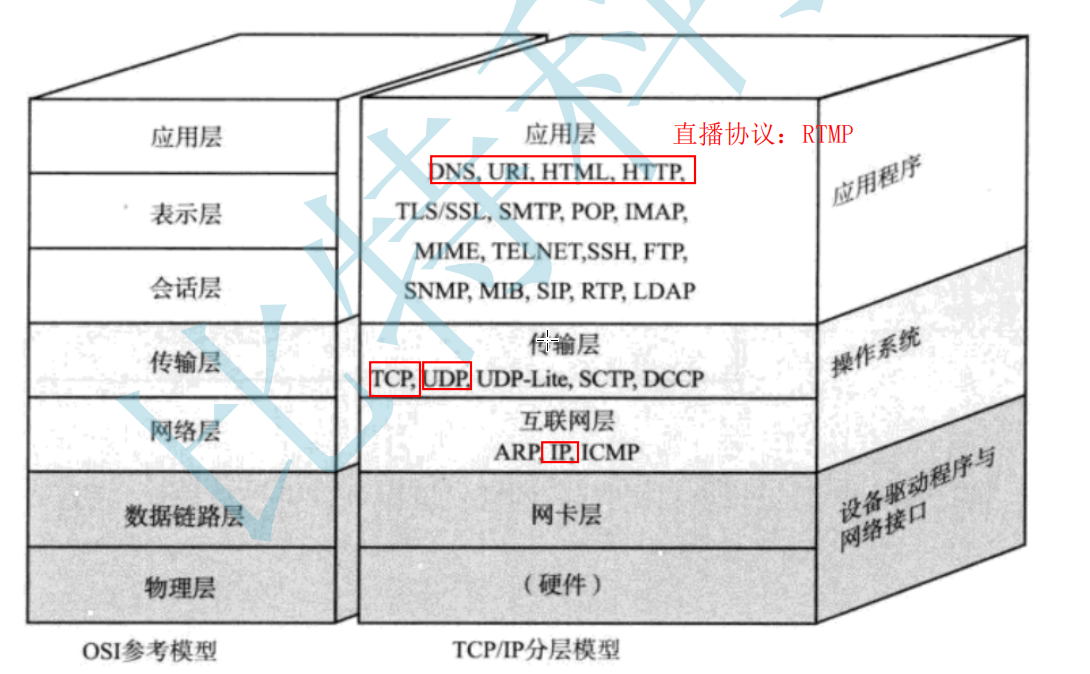
**OSI七层模型的作用：**



## **堆和栈的区别：**

1、数据结构的不同：堆可以看看作一棵树，这个数据结构常用于堆排序；栈是一种先进后出的数据结构；

2、可见度的不同：栈内存是线程的私有内存。而堆内存中的对象对所有线程可见。

3、存储变量的不同：基本数据类型、局部变量是存放在栈内存中的，用完自动消失；而new创建的实例对象以及数组是放在堆内存中的，用完靠垃圾回收机制去清理；

4、生命周期的不同：栈内存的更新速度要快于堆内存，因为局部变量的生命周期很短。栈内存存放的变量生命周期一旦结束就会被释放，而堆内存存放的实体会被垃圾回收机制不定时的回收。

5、所抛异常的不同：如果栈内存没有可用的空间存储方法调用和局部变量，JVM会抛出java.lang.StackOverFlowError；而如果是堆内存没有可用的空间存储生成的对象，JVM会抛出java.lang.OutOfMemoryError。

## 

## **栈和队列的区别：**

1、栈的插入和删除只允许在表尾一端进行，而队列允许在表尾一端进行插入，在表头一端进行删除；  
2、栈是先进后出的数据结构；队列是先进先出的数据结构；

## **List集合：**

一、**Arraylist**：实现了基于动态数组的数据结构,因为地址连续，一旦数据存储好了，查询操作效率会比较高，但因为地址连续， ArrayList要移动数据,所以插入和删除操作效率比较低。它是线程不安全的

二、**LinkedList**：基于链表的数据结构,地址是任意的，插入和删除速度快，访问速度慢。适用于要头尾操作或插入指定位置的场景。也是线程不安全的。

三、Vector：与ArrayList一样，也是通过数组实现的，不同的是它支持线程的同步，是线程安全的。(即某一时刻只有一个线程能够写Vector，避免多线程同时写而引起的不一致性，但实现同步需要很高的花费，因此，访问它比访问ArrayList慢。)

## **Set集合：**

HashSet与Treeset的适用场景

一、**TreeSet**：底层数据结构是红黑树。Treeset中的数据是自动排好序的，靠自然排序比较器保证有序。根据比较的返回值是否是0来保证元素的唯一性，不允许放入null值 。适用于排序场景。

二、**HashSet** 当向HashSet集合中存入一个元素时，HashSet会调用该对象的hashCode方法来得到该对象的hashCode值，然后根据该hashCode值决定该对象在HashSet中的存储位置。如果有两个元素通过equals方法比较true，但它们的hashCode方法返回的值不相等，HashSet将会把它们存储在不同位置，依然可以添加成功。线程不安全的  
**靠元素重写hashCode方法和equals方法来判断两个元素是否相等，如果相等则覆盖原来的元素，依此来确保元素的唯一性**

三、LinkedHashSet底层数据结构是链表和哈希表。(FIFO插入有序,唯一)1.由链表保证元素有序2.由哈希表保证元素唯一

## 

## ****Map集合：****

一、**HashMap**：基于哈希表实现。使用HashMap要求添加的键类明确定义了hashCode()和equals()[可以重写hashCode()和equals()]，为了优化HashMap空间的使用，您可以调优初始容量和负载因子。是非线程安全的 适用于Map中插入、删除和定位元素。

二**TreeMap**：非线程安全基于红黑树实现。TreeMap没有调优选项，因为该树总处于平衡状态。适用于按自然顺序或自定义顺序遍历键(key)。

**Java关键字：**

## **private(私有的)**

private：是访问权限最低的访问限制符，被它修饰的变量只能访问本类的对象，即除class自身外，其他都不得使用。可以修饰成员变量，成员方法，构造方法，被private修饰的成员只能在其修饰的本类中访问，在其他类中不能调用，但是被private修饰的成员可以通过set和get方法向外界提供访问方式

## **default(默认的)**

defalut：即不写任何关键字，它可以修饰类，成员变量，成员方法，构造方法。被默认权限修饰后，其只能被本类以及同包下的其他类访问。

## **protected(受保护的)**

protected：可以修饰成员变量，成员方法，构造方法，但不能修饰类(此处指的是外部类，内部类不加以考虑)。被protected修饰后，只能被同包下的其他类或者自身的子类访问

## **public(公共的)**

public是权限最大的修饰符，他可以修饰类，成员变量，成员方法，构造方法。被public修饰后，可以再任何一个类中，不管同不同包，任意使用。

## **abstract**

****修饰类****：用abstract修饰的类表示抽象类，抽象类不能被实例化，但可以有构造方法，构造方法可以被子类构造函数调用，需要子类继承并覆盖其中的抽象方法，一个子类只能继承一个抽象类。  
****修饰方法****：用abstract修饰的方法表示抽象方法,抽象方法没有方法体，只声明，不实现。子类继承抽象类的话，必须覆写抽象类的所有抽象方法，否则他还是抽象类。抽象关键字不能private，static，final并列修饰同一方法。

## **Final：**

****修饰变量****：包括成员变量和局部变量,对于基本数据类型变量时，变量中的数值不可以改变，对于引用类型来说，变量中的地址不可以改变。对于成员变量来讲因为成员表变量有默认值，使用final修饰后不再有默认值，所以我们必须在声明时或者构造方法中对它赋值;  
****修饰方法****：表示该方法无法被重写;  
****修饰类****：表示该类无法被继承。

## **static**

****修饰变量****：称为静态变量，静态变量是随着类加载时被完成初始化的，且初始化时间要早于非静态变量，它在内存中仅有一个，且JVM也只会为它分配一次内存，同时类所有的实例都共享静态变量，可以直接通过类名来访问它。但是实例变量则不同，它是伴随着实例的，每创建一个实例就会产生一个实例变量，它与该实例同生共死。  
****修饰方法****：称之为静态方法，可以通过类名对其进行直接调用（class.method),不依附于任何对像,所以在静态方法内不能访问非静态东西，但在非静态成员方法内可以访问静态成员方法/变量。  
****修饰代码块****：被static修饰的代码块，我们称之为静态代码块，可以在程序中随意放，在类初次被加载时依次执行。静态块可以优化程序性能。

## Synchronized 关键字，代表这个方法加锁,相当于不管哪一个线程运行到这个方法时,都要检查有没有其它线程正在用这个方法，有的话要等正在使用synchronized方法运行完这个方法后再运行此线程没有的话,锁定调用者,然后直接运行

## **extends**

1、extends是继承父类（final修饰的和抽象类除外），  
2、Java中不支持多重继承，但是可以用接口来实现，所以引出了implements。  
3、java不允许多继承，继承只能继承一个类，但implements可以实现多个接口，用逗号分开就行了比如

class A extends B implements C,D,E

* 1

## **implements**

implements是一个类实现一个接口用的关键字，他是用来实现接口中定义的抽象方法。还有几点需要注意：  
1、接口可以被多重implements,抽象类只能被单一extends  
2、接口只有定义,抽象类可以有定义和实现  
3、接口的字段定义默认为:public static final, 抽象类字段默认是"friendly"(本包可见)  
4、在java中implements表示子类继承父类，如类A继承类B写成 class A implements B{}

## **new**

类名称 对象名称 =new 类名称（）；  
a、通过new计算对象需要的内存大小，开辟空间初始化为0  
b、Person(“bite”,2,4)是调用构造方法，对属性初始化  
c、Person p:构造方法调用结束，返回一个指向新对象的引用

## **this**

代表使用该方法的对象的引用，他指向自身的这个对象  
而this 关键字用来访问本类内容时，用法也有三种：  
1.在本类的成员方法中，访问本类的成员变量。  
2.在本类的成员方法中，访问本类的另一个成员方法。  
3.在本类的构造方法中，访问本类的另一个构造方法

## **super**

super关键字的用法有三种：  
1.在子类的成员方法中，访问父类的成员变量。  
2.在子类的成员方法中，访问父类的成员方法。  
3.在子类的构造方法中，访问父类的构造方法

## **instanceof**

instanceof运算符的前一个操作符是一个引用变量，后一个操作数通常是一个类（可以是接口），用于判断前面的对象是否是后面的类，或者其子类、实现类的实例。如果是返回true，否则返回false。  
也就是说：使用instanceof关键字做判断时， instanceof 操作符的左右操作数必须有继承或实现关系

## **什么是索引？**

简单来说，比如你想在图书馆借一本莫言的小说，一楼存的数学类，二楼医学类，三楼文学类。在三楼一排存的徐志摩的，二排存的莫言的，三排存的林徽因的。索引就相当于引导，有人告诉你三楼二排存的莫言的，你就不用从一楼一本一本的去往上找，这就大大增加了查找速度。  
****来官方一点****：索引能快速找出在某一列中有一特定值的行。不使用索引，mysql必须从第一条记录开始查找，直到找到相关的行。如果表中查询的列有一个索引，mysql能快速到达某个位置去搜索数据文件，而不必查看所有数据。索引是对数据库中一列或多列的值进行排序的一种结构。

****优点:****  
1.大大加快数据的检索速度;  
2.创建唯一性索引，保证数据库表中每一行数据的唯一性;  
3.加速表和表之间的连接;  
4.在使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间  
****缺点：****  
1.索引需要占物理空间。  
2.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低了数据的维护速度。

## ****索引有哪些类型：****

****唯一索引****：不允许其中任何两行具有相同索引值的索引  
当现有数据中存在重复的键值时，大多数数据库不允许将新创建的唯一索引与表一起保存。数据库还可能防止添加将在表中创建重复键值的新数据。例如，如果在employee表中职员的姓(lname)上创建了唯一索引，则任何两个员工都不能同姓。

****主键索引：****  
在数据库关系图中为表定义主键将自动创建主键索引，主键索引是唯一索引的特定类型。该索引要求主键中的每个值都唯一。当在查询中使用主键索引时，它还允许对数据的快速访问。

****聚簇索引：****  
在聚集索引中，表中行的物理顺序与键值的逻辑（索引）顺序相同。一个表只能包含一个聚集索引， 即如果存在聚集索引，就不能再指定CLUSTERED 关键字。一个表中只能有一个聚集索引，但表中的每一列都可以有自己的非聚集索引。  
非聚集索引：则表中行的物理顺序与键值的逻辑（索引）顺序不匹配。与非聚集索引相比，聚集索引通常提供更快的数据访问速度。聚集索引更适用于对很少对基表进行增删改操作的情况。

## 

## ****创建和删除索引****

索引的建立是在数据库的某些列上面，  
1、在经常需要搜索的列上，可以加快搜索速度，  
2、在作为主键的列上，强制该列的唯一性，  
3、在经常用作连接的列上，可以加快连接的速度  
4、在经常需要排序的列上，利用索引的排序，加快排序时间，  
5、在经常使用where子句中的列上，可以加快条件的判断速度

****怎样建立和删除？****

1使用create index语句在已存在的表上创建和删除索引*// 创建索引*

create index index\_name on table\_name(column\_name);*// 删除索引*

drop index index\_name on table\_name;(内部被映射成alter table语句)

2使用alter table 语句创建和删除索引*//创建索引*

alter table table\_name add index(column\_name);*// 删除索引*

alter table table\_name drop index index\_name;

****什么是事务？****

事务是作为一个逻辑单元执行的一系列操作，一个逻辑工作单元必须有四个属性，称为 ACID（原子性、一致性、隔离性和持久性）属性，只有这样才能成为一个事务。

****原子性：****事务中的全部操作在数据库中是不可分割的；对于其数据修改，要么全都执行，要么全都不执行。

****一致性：****事务在完成时，必须使所有的数据都保持一致状态。比如两个杯子倒水，A往B里倒一点，B往A里倒一点，到最后，A杯跟B杯加起来的水是没有少的。

****隔离性：****事务的隔离性是多个用户并发访问数据库时，数据库为每一个用户开启的事务，不能被其他事务的操作数据所干扰，多个并发事务之间要相互隔离。

****持久性：****事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

### **脏读、幻读不可重复读**

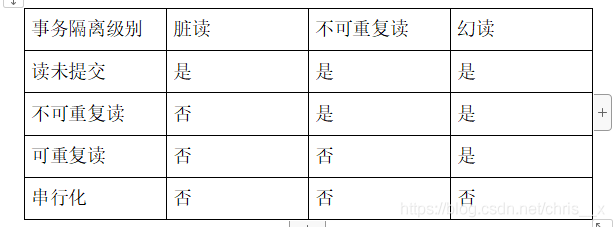
在并发状态下事务会出现一些问题，像上面这三种情况；

****脏读****：某个事务已更新一份数据，另一个事务在此时读取了同一份数据，由于某些原因，前一个回滚了操作，则后一个事务所读取的数据就是脏数据  
举例说明：A 给 B 转了100块钱，但是 A 还没有提交，此时 B 查询自己账户，多了100万。然后 A 发现转错人了，回滚了事物。然后 B 100块钱就没了。在这个过程中 B 查到了没有提交的那100块钱，这就是脏读。

****不可重复读****：在一个事务的两次查询之中数据不一致，这可能是两次查询过程中间插入了一个事务更新的原有的数据。  
举例：A账户有100块钱，A第一次查询时是100，B取走了50，A在查询就是50；对A来说两次结果不一致就是不可重复读；

****幻读****：在一个事务的两次查询中数据不一致，发现了原来没有的数据，  
举例：例如有一个事务A查询了几列数据，而另一个事务B却在此时插入了新的几列数据，先前的事务在接下来的查询中，就会发现有几列数据是它先前所没有的,对A来说就产生了幻读；  
ps:不可重复读和幻读很像，不可重复读侧重于修改，幻读侧重于新增和删除；

### **4、事物的隔离级别**

SQL 标准定义了四个隔离级别去解决上述问题，隔离级别越高，产生的问题就越少；  


****读取未提交****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。

****读取已提交****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。（Oracle默认的隔离级别）

****可重复读****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。（****mysql默认的隔离级别****）

****可串行化****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。

因为隔离级别越低，事务请求的锁越少，所以大部分数据库系统的隔离级别都是READ-COMMITTED(读取提交内容):，但是你要知道的是InnoDB 存储引擎默认使用 \*\*REPEATABLE-READ（可重读）\*\*并不会有任何性能损失。

## **死锁**

### **1、死锁的概念**

死锁是指两个或两个以上的进程（线程）在运行过程中因争夺资源而造成的一种僵局，若无外力作用，这些进程（线程）都将无法向前推进 ，这时就形成了死锁。处于死锁状态的进程称为死锁进程；比如一扇门，你要出我要进，你在等我让，我在等你让，这时就陷入了死循环，就形成了死锁。

### **2、死锁产生的条件**

****先来看两个概念： 可抢占资源和不可抢占资源****  
可抢占资源：指某进程在获得这类资源后，该资源可以再被其他进程或系统抢占。对于这类资源是不会引起死锁的。CPU 和主存均属于可抢占性资源。

不可抢占资源：一旦系统把某资源分配给该进程后，就不能将它强行收回，只能在进程用完后自行释放。磁带机、打印机等属于不可抢占性资源

竞争不可抢占资源引起死锁如：共享文件时引起死锁系统中拥有两个进程P1和P2，它们都准备写两个文件F1和F2。而这两者都属于可重用和不可抢占性资源。如果进程P1在打开F1的同时，P2进程打开F2文件，当P1想打开F2时由于F2已结被占用而阻塞，当P2想打开1时由于F1已结被占用而阻塞，此时就会无线等待下去，形成死锁。进程推进顺序不当时也会引起死锁

产生死锁的四个必要条件  
（1）互斥条件：进程对所分配到的资源不允许其他进程进行访问，若其他进程访问该资源，只能等待，直至占有该资源的进程使用完成后释放该资源；比如有一间单人自习室，我在里面学习，你就不能进来学习，你要进来得等我学完出来了

（2）请求和保持条件：进程获得一定的资源之后，又对其他资源发出请求，但是该资源可能被其他进程占有，此事请求阻塞，但又对自己获得的资源保持不放；我在现在这个自习室里，我还觉得不够，又对你所在的自习室发出请求，但你正在用，这时我又不愿意把自己的自习室让出来，这就是请求和保持；

（3）不可剥夺条件：是指进程已获得的资源，在未完成使用之前，不可被剥夺，只能在使用完后自己释放，就是我在自习室里你不能把我撵出来；

（4）循环等待条件：是指进程发生死锁后，必然存在一个进程–资源之间的环形链；

### **3、处理死锁的基本方法**

****1.预防死锁****：通过设置一些限制条件，去破坏产生死锁必要条件的一个或多个来预防死锁，让系统不会产生死锁，两种办法  
①资源一次性分配：可以破坏请求和保持条件；  
可剥夺资源：即当某进程新的资源未满足时，释放已占有的资源（破坏不可剥夺条件）  
②资源有序分配法：系统给每类资源赋予一个编号，每一个进程按编号递增的顺序请求资源，释放则相反（破坏环路等待条件）

****2.避免死锁****：在资源分配过程中，使用某种方法避免系统进入不安全的状态，从而避免发生死锁，在使用前判断，只允许不会产生死锁的进程申请资源  
①银行家算法  
原理简介：把操作系统看作是银行家，操作系统管理的资源相当于银行家管理的资金，进程向操作系统请求分配资源相当于用户向银行家贷款。  
为保证资金的安全，银行家规定：  
(1) 当一个顾客对资金的最大需求量不超过银行家现有的资金时就可接纳该顾客；  
(2) 顾客可以分期贷款，但贷款的总数不能超过最大需求量；  
(3) 当银行家现有的资金不能满足顾客尚需的贷款数额时，对顾客的贷款可推迟支付，但总能使顾客在有限的时间里得到贷款；  
(4) 当顾客得到所需的全部资金后，一定能在有限的时间里归还所有的资金.

操作系统按照银行家制定的规则为进程分配资源，当进程首次申请资源时，要测试该进程对资源的最大需求量，如果系统现存的资源可以满足它的最大需求量则按当前的申请量分配资源，否则就推迟分配。当进程在执行中继续申请资源时，先测试该进程本次申请的资源数是否超过了该资源所剩余的总量。若超过则拒绝分配资源，若能满足则按当前的申请量分配资源，否则也要推迟分配。

②避免多次锁定：尽量避免同一个线程对多个 锁进行锁定。

③具有相同的加锁顺序：（给锁添加顺序）如果多个线程需要对多个 Lock 进行锁定，则应该保证它们以相同的顺序请求加锁。

④使用定时锁：程序在调用 acquire() 方法加锁时可指定 timeout 参数，该参数指定超过 timeout 秒后会自动释放对 Lock 的锁定，这样就可以解开死锁了。

⑤死锁检测：死锁检测是一种依靠算法机制来实现的死锁预防机制，它主要是针对那些不可能实现按序加锁，也不能使用定时锁的场景的。

****3.检测死锁****：允许死锁的发生，但是通过系统的检测之后，采取一些措施，将死锁清除掉

****4.解除死锁****：当检测出死锁后，便采取适当措施将进程从死锁状态中解脱出来。主要有两个方法  
①从别的进程中抢占足够数量的资源分配给死锁进程来解除死锁状态  
②终止进程，终止或撤销系统中一个或多个死锁进程；

## **抽象类和接口的区别**

抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是public static final类型的；

1、抽象类要被子类继承，接口要被类实现，一个类只能继承一个抽象类，但可以实现多个接口  
2、接口只能做方法声明，抽象类可以做方法声明，也可以做方法实现  
3、接口中不能含有静态代码块以及静态方法，而抽象类可以有静态代码块和静态方法；  
4、接口中可以含有 变量和方法。但接口中的变量会被隐式地指定为public static final变量，不能是其他；而方法会被隐式地指定为public abstract方法且只能是public 。抽象类中的变量是普通变量。  
5、修饰符不同，抽象方法可以有public,protected,default,j接口默认public  
6、抽象类可以提供成员方法的实现细节，而接口中只能存在public abstract 方法；  
7、实现抽象类的方法时, 如果方法是抽象的,子类必须重写抽象的方法. 如果方法不是抽象的, 子类可以选择继承 VS 实现了接口 就必须实现接口中的所有方法, 因为接口中的方法默认的全部都是抽象的方法 ；所以这里可以说, 接口是抽象类的一种, 也可以说接口是抽象类的延伸

1、==操作符既可以用于比较基本的数据类型，也可以用于比较对象, 如果作用于基本数据类型的变量，则直接比较其存储的 “值”是否相等；如果作用于引用类型的变量，则比较的是所指向的对象的地址。而equals只可以用于对象之间的比较

2、对于equals方法，注意：equals方法不能作用于基本数据类型的变量，equals继承Object类，比较的是是否是同一个对象。如果没有对equals方法进行重写，则比较的是引用类型的变量所指向的对象的地址；诸如String、Date等类对equals方法进行了重写的话，比较的是所指向的对象的内容。

3、其他类型的对象进行比较时, equals方法默认的原理是判断两者的内存地址是否相同，所以默认情况下与=操作符返回的结果相同，但是这里应该发现差别，也就是equals方法可以被重写一用户可以定制自己的equals方法，而==操作符不能被重写

**输入URL会发生什么：**

****DNS解析****  
浏览器能访问的资源都是通过IP访问的，但是为了方便我们记忆和使用网站都是使用的域名，所以当我们输入一个域名的时候就需要浏览器进行域名解析。浏览器解析域名的流程是  
a、先从浏览器缓存中查找是否有该域名（浏览器缓存），  
b、浏览器缓存中没有则从系统缓存中查找（系统缓存），  
c、系统缓存中没有则从路由缓存中查找，（路由器缓存）  
d、路由缓存没有则从本地DNS服务器中查找，（ISP DNS缓存）  
e、本地DNS服务器中没有则从其他DNS服务器中查找，其他服务器中查找的规则是递归查找，查找的顺序是（根域名服务器，一级域名服务器，二级域名服务器，三级域名服务器，是递归查询）

****TCP三次握手建立连接****

①、Client首先发送一个连接试探，ACK=0 表示确认号无效，SYN = 1 表示这是一个连接请求或连接接受报文，同时表示这个数据报不能携带数据，seq = x 表示Client自己的初始序号（seq = 0 就代表这是第0号包），这时候Client进入syn\_sent状态，表示客户端等待服务器的回复

②、Server监听到连接请求报文后，如同意建立连接，则向Client发送确认。TCP报文首部中的SYN 和 ACK都置1 ，ack = x + 1表示期望收到对方下一个报文段的第一个数据字节序号是x+1，同时表明x为止的所有数据都已正确收到（ack=1其实是ack=0+1,也就是期望客户端的第1个包），seq = y 表示Server 自己的初始序号（seq=0就代表这是服务器这边发出的第0号包）。这时服务器进入syn\_rcvd，表示服务器已经收到Client的连接请求，等待client的确认。

⑤、Client收到确认后还需再次发送确认，同时携带要发送给Server的数据。ACK 置1 表示确认号ack= y + 1 有效（代表期望收到服务器的第1个包），Client自己的序号seq= x + 1（表示这就是我的第1个包，相对于第0个包来说的），一旦收到Client的确认之后，这个TCP连接就进入Established状态，就可以发起http请求了。  
为什么不是两次握手？****防止了服务器的一直等待而浪费资源****

****三、发送HTTP请求并响应****  
连接成功之后就可以开始传输数据了，传输数据需要将用户输入的URL封装成HTTP Request请求报文，发送到服务器，服务器收到请求后会发出应答，即响应数据。  
HTTP请求报文格式：请求报头和请求主体。请求报头包括请求方式（GET/POST/DELETE/PUT）、请求资源路径、HTTP版本号，返回的信息是否需要缓存，以及客户端是否发送cookie等

****四、断开连接，TCP四次挥手****  
第一次挥手  
因此当主动方发送断开连接的请求（即FIN报文）给被动方时，仅仅代表主动方不会再发送数据报文了，但主动方仍可以接收数据报文。  
第二次挥手  
被动方此时有可能还有相应的数据报文需要发送，因此需要先发送ACK报文，告知主动方“我知道你想断开连接的请求了”。这样主动方就不会因为没有收到应答而重复发送断开连接的请求。  
第三次挥手  
被动方在处理完数据报文后，便发送给主动方FIN报文；这样可以保证数据通信正常可靠地完成。发送完FIN报文后，被动方进入LAST\_ACK阶段（超时等待）。  
第四挥手  
主动方发送ACK报文进行连接中断的确认，被动方接收后，连接就正式中断了。

****五、浏览器解析渲染页面****\*\*

****Linux查看端口：****

## **PS命令**

1. ps -a:查看所有  
   ②、ps -u:以用户的格式显示  
   ③、ps -x:显示后台进程运行参数  
   ④、ps -d:创建人、创建时间、进程号

## **netstat命令**

①、-t 仅显示和tcp相关的  
②、-u 仅显示和udp相关的  
③、-n 不限时别名，能显示数字的全部转换为数字  
④、-l 仅显示出于Listen(监听)状态的  
⑤、-p 显示建立这些连接的程序名

****线程：****  
进程是资源分配的最小单位，线程是资源调度的最小单位  
比如说银行来了很多顾客，忙不过来了，这时行长张三叫来了李四王五，这时李四王五张三就是线程，人是张三叫的，他就是主线程，来的那一个个顾客就是一个个进程。进程和线程的最大区别就是线程间是数据共享的。

****什么时候用线程？****  
1、有可能提升速度的时候（原理：多一个调度单位，就多一个机会抢到cpu）  
2、线程有极限，如果操作系统中已经有太多你的线程，那在添加线程意义也就不大了  
*注意*：本身创建线程销毁线程也需要时间，所以不能使劲叫人，很简单，叫工人是需要开工资的。

****Thread t= new MyThread();****  
什么时候线程t才会真的运行起来？因为t这个线程被分配了CPU资源，结合进程的调度知识，  
1、cpu上现在线程被调度下来了  
2、t被从就绪队列中选中了  
3、t被调度到CPU上了，  
那什么时候线程可能会被从cpu上调度下来的五种情况“  
1、被高优先级抢了  
2、执行结束  
3、在等IO  
4、主动放弃  
5、时间片耗尽

****线程的创建****

一、继承Thread类

class MyThread extends Thread{

@Override

public void run(){

//这里放希望线程执行的代码

}

}

二、实现Runnable接口

class MyThread implements Runnable{

@Override

public void run(){

//这里放希望线程执行的代码

}

}

****怎么样才是线程安全的？****  
多线程程序的运行结果可以保证100%正确性，才算安全  
1、多线程之间有共享资源  
2、多线程之间有针对共享资源的修改  
这两种情况下才会出现线程不安全

****回想一下变量的类型****  
变量有两种类型，一种是基本数据类型，一种是引用数据类型（变量的类型无法决定该变量是否是线程之间共享的）  
*变量的形态*：形态才是决定变量存放位置以及是否是线程共享的主要因素#  
a、形参和局部变量存放在栈帧里，它们是线程之间私有的  
b、属性存放在对象里，对象存放在堆中，所以是线程之间共享的  
c、静态属性：在类中，类在方法区中，是线程之间共享的

Synchronized是java的内置锁，每个JAVA对象都可以用作一个实现同步的锁，线程进入同步代码块或方法的时候会自动获得该锁，在退出同步代码块或方法时会释放该锁

加锁：实现两个线程之间互斥  
1、作为方法的修饰符，可以写在定义的方法之前：这个方法在同一时刻只能被一个线程访问，从而保证了多线程访问的安全性。然而，当一个方法体规模非常大时，把该方法声明为synchronized会大大影响程序的执行效率

被volatile修饰的变量具备可见性：意思就是说，在多线程环境下，某个共享变量如果被其中一个线程给修改了，其他线程能够立即知道这个共享变量已经被修改了，当其他线程要读取这个变量的时候，最终会去内存中读取，而不是从自己的工作空间中读取。  
volatile long a;  
a=10L//内部要求必须把修改刷新到内存中  
****保证可见性的原理****：观察加入volatile关键字和没有加入volatile关键字时所生成的汇编代码发现，加入volatile关键字时，会多出一个lock前缀指令  
lock前缀指令实际上相当于一个内存屏障（也成内存栅栏），内存屏障会提供3个功能：  
　1）它确保指令重排序时不会把其后面的指令排到内存屏障之前的位置，也不会把前面的指令排到内存屏障的后面；即在执行到内存屏障这句指令时，在它前面的操作已经全部完成；

2）它会强制将对缓存的修改操作立即写入主存；

3）如果是写操作，它会导致其他CPU中对应的缓存行无效。

其实理论上 wait 和 sleep 完全是没有可比性的，因为一个是用于线程之间的通信的，一个是让线程阻塞一段时间，唯一的相同点就是都可以让线程放弃执行一段时间。用生活中的例子说的话就是婚礼时会吃糖，和家里自己吃糖之间有差别。说白了放弃线程执行只是 wait 的一小段现象。

****当然为了面试的目的，我们还是总结下：****  
1、wait 之前需要请求锁，而wait执行时会先释放锁，等被唤醒时再重新请求锁。这个锁是 wait 对像上的 monitor lock  
2.、sleep 是无视锁的存在的，即之前请求的锁不会释放，没有锁也不会请求。  
3.、wait 是 Object 的方法。sleep 是 Thread 的静态方法  
4、sleep（）方法需要捕获异常，wait()方法不需要捕获异常  
5、Sleep可以再任何地方使用，而wait只能在synchronized方法或synchronized块中使用。

## **单元测试(Unit Testing)**

单元测试又称模块测试，是对软件的基本组成单元进行测试，其目的是检验软件基本组成单位的正确性。  
测试阶段：编码后或者编码前（测试驱动开发、测试先行）  
测试对象：最小模块  
测试人员：白盒测试工程师或开发工程师  
测试依据：代码和注释+详细设计文档  
测试方法：白盒测试  
测试内容：模块接口测试、局部数据结构测试、路径测试、错误处理测试、边界测试

## **集成测试(Integration Testing)**

集成测试也称联合测试（联调）、组装测试，主要目的是检查软件单位之间的接口是否正确。将程序模块采用适当的集成策略组装起来，对系统的接口及集成后的功能进行正确性检测的测试工作。  
测试阶段：一般单元测试之后进行  
测试对象：模块间的接口  
测试人员：白盒测试工程师或开发工程师  
测试依据：单元测试的模块+概要设计文档  
测试方法：黑盒测试与白盒测试相结合  
测试内容：模块之间数据传输、模块之间功能冲突、模块组装功能正确性、全局数据结构、单模块缺陷对系统的影响

## **系统测试(System Testing)**

将软件系统看成是一个系统的测试。包括对功能、性能以及软件所运行的软硬件环境进行测试。时间大部分在系统测试执行阶段，包括回归和冒烟。  
测试阶段：集成测试通过之后  
测试对象：整个系统（软、硬件）  
测试人员：黑盒测试工程师  
测试依据：需求规格说明文档  
测试方法：黑盒测试  
测试内容：功能、界面、可靠性、易用性、性能、兼容性、安全性等

****回归测试(Regression Testing)****  
回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。比如修手机之后第一件事就是看手机修好没，这就是回归测试自动回归测试将大幅降低系统测试、维护升级等阶段的成本。  
****冒烟测试（smoke testing）****  
这一术语源自硬件行业。对一个硬件或硬件组件进行更改或修复后，直接给设备加电。如果没有冒烟，则该组件通过了测试。  
冒烟测试的对象是每一个新编译的需要正式测试的软件版本，目的是确认软件基本功能正常，可以进行后续的正式测试工作。冒烟测试的执行者是*版本编译人员。*

## **验收测试(Acceptance Testing)**

验收测试是部署软件之前的最后一个测试操作。它是技术测试的最后一个阶段，也称为交付测试。验收测试的目的是确保软件准备就绪，按照项目合同、任务书、双方约定的验收依据文档，向软件购买都展示该软件系统满足原始需求。  
测试阶段：系统测试通过之后  
测试对象：整个系统（包括软硬件）。  
测试人员：主要是最终用户或者需求方。  
测试依据：用户需求、验收标准  
测试方法：黑盒测试  
测试内容：同系统测试(功能…各类文档等)

## **为什么要有GC?**

电脑的内存大小的不变的，当我们使用对象的时候，如使用New关键字的时候，就会在内存中生产一个对象，但是我们在使用JAVA开发的时候，当一个对象使用完毕之后我们并没有手动的释放那个对象所占用的内存，就这样在使用程序的过程中，对象越来越多，当内存存放不了这么多对象的时候，电脑就会崩溃了，JAVA为了解决这个问题就推出了这个自动清除无用对象的功能。

## **3、怎么判定垃圾**

引用计数法  
顾名思义，整一个count来给对象计数，有引用指向时就+1，引用离开时就-1，count=0时，说明对象死了。这么简单，为什么JVM却不用它来管理内存呢？很好理解，要是三个人之间形成了三角恋，你觉着谁能脱单？引用计数法就没法解决这种具有相互循环引用的对象。  
可达性分析法  
那java采用什么方法呢，就是可达性分析法。简单来说就是找一个根节点，往下遍历，它能到的地方就是活对象，到不了的jvm就会对其进行回收。这个根节点称为GC roots;

在Java语言中，可作为GC Roots的对象包含下面几种:  
1、虚拟机栈(栈帧中的本地变量表)中引用的对象  
2、方法区中类静态属性引用的对象  
3、方法区中常量引用的对象  
4、本地方法栈中JNI(Native方法)引用的对象

## **四种引用介绍**

强引用: 指的是在程序代码之中普遍存在的，类似于"Object obj = new Object()"这类的引用，只要强引用还存在，垃圾回收器永远不会回收掉被引用的对象实例。

软引用: 是用来描述一些还有用但是不是必须的对象。对于软引用关联着的对象，在系统将要发生内存溢出之前，会把这些对象列入回收范围之中进行第二次回收。如果这次回收还是没有足够的内存，才会抛出内存溢出异常。在JDK1.2之后，提供了SoftReference类来实现软引用。

弱引用: 弱引用也是用来描述非必需对象的。但是它的强度要弱于软引用。被弱引用关联的对象只能生存到下一次垃圾回收发生之前。当垃圾回收器开始进行工作时，无论当前内容是否够用，都会回收掉只被弱引用关联的对象。在JDK1.2之后提供了WeakReference类来实现弱引用。

虚引用: 虚引用也被称为幽灵引用或者幻影引用，它是最弱的一种引用关系。一个对象是否有虚引用的存在，完全不会对其生存时间构成影响，也无法通过虚引用来取得一个对象实例。为一个对象设置虚引用的唯一目的就是能在这个对象被收集器回收时收到一个系统通知。在JDK1.2之后，提供了PhantomReference类来实现虚引用。

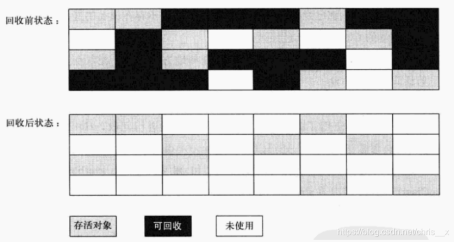
## **5、回收方法区**

方法区(永久代)的垃圾回收主要收集两部分内容 :  
废弃常量和无用的类  
回收废弃常量和回收Java堆中的对象十分类似。以常量池中字面量(直接量)的回收为例，假如一个字符串"abc"已经  
进入了常量池中，但是当前系统没有任何一个String对象引用常量池的"abc"常量，也没有在其他地方引用这个字面  
量，如果此时发生GC并且有必要的话，这个"abc"常量会被系统清理出常量池。常量池中的其他类(接口)、方法、  
字段的符号引用也与此类似。  
****判断一个类是无用类****  
1、该类所有实例都已经被回收(即在Java堆中不存在任何该类的实例)  
2.、加载该类的ClassLoader已经被回收  
3.、该类对应的Class对象没有在任何其他地方被引用，无法在任何地方通过反射访问该类的方法

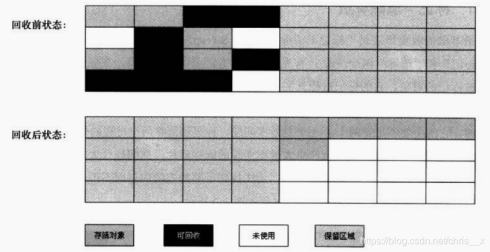
# **三、 垃圾回收算法**

## **1、 标记-清除算法**

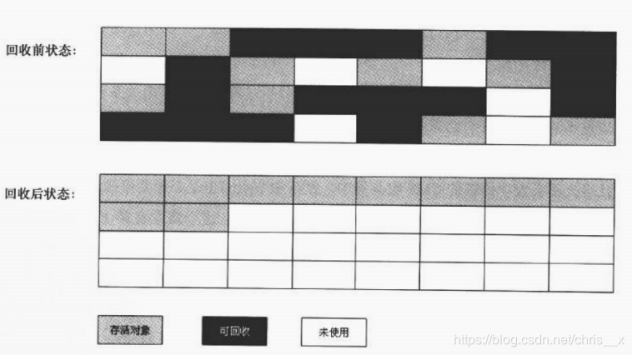
分为"标记"和"清除"两个阶段 : 首先标记出所有需要回收的对象，然后统一回收。有两点不足  
1、 效率问题 : 标记和清除这两个过程的效率都不高  
2.、空间问题 : 标记清除后会产生大量不连续的内存碎片，空间碎片太多可能会导致以后在程序运行中需要分配较大对象时，无法找到足够连续内存而不得不提前触发另一次垃圾收集。



## **2 、复制算法(新生代回收算法)**

将可用内存按容量划分为大小相等的两块，每次只使用其中的一块。当这一块的内存用完了，就将还存活着的对象复制到另外一块上面，然后再把已使用过的内存空间一次清理掉。这样做的好处是每次都是对整个半区进行内存回收，内存分配时也就不需要考虑内存碎片等复杂情况，只需要移动堆顶指针，按顺序分配即可  


## **3 、标记-整理算法(老年代回收算法)**

复制收集算法在对象存活率较高时会进行比较多的复制操作，效率会变低。因此在老年代一般不能使用复制算法。  
针对老年代的特点，提出了一种称之为"标记-整理算法"。标记过程仍与"标记-清除"过程一致，但后续步骤不是直接对可回收对象进行清理，而是让所有存活对象都向一端移动，然后直接清理掉端边界以外的内存。流程图如下:  


## **4、 分代收集算法**

当前JVM垃圾收集都采用的是"分代收集(Generational Collection)"算法，这个算法并没有新思想，只是根据对象存  
活周期的不同将内存划分为几块。一般是把Java堆分为新生代和老年代。在新生代中，每次垃圾回收都有大批对象死去，只有少量存活，因此我们采用复制算法；而老年代中对象存活率高、没有额外空间对它进行分配担保，就必须采用"标记-整理"算法。

# **四、 七种垃圾收集器**

1、Serial收集器（新生代，串行）：最古老，最稳定，它工作时其他线程必须暂停。  
2、ParNew收集器（新生代，并行）：是Serial收集器的多线程版本。  
3、Parallel Scavenge（新生代，并行）Parallel收集器更关注系统的吞吐量，使用复制算法。  
4、Serial Old收集器(老年代，串行)：使用标记整理，是Serial的老年版  
5、Parallel Old 收集器（老年代，并行），Parallel Old是Parallel Scavenge收集器的老年代版本，使用多线程和“标记－整理”算法  
6、CMS收集器（老年代，并行），是一种以获取最短回收停顿时间为目标的收集器。  
7、G1收集器，G1 (Garbage-First)是一款面向服务器的垃圾收集器,主要针对配备多颗处理器及大容量内存的机器. 以极高概率满足GC停顿时间要求的同时,还具备高吞吐量性能特征

# **五、 java 内存分配与回收以及 Minor GC 和Major GC**

答：对象优先在堆的 Eden 区分配，大对象直接进入老年代。长期存活的对象将直接进入老年代.，当 Eden 区没有足够的空间进行分配时，虚拟机会执行一次 Minor GC。Minor Gc 通常发生在新生代的 Eden 区，在这个区的对象生存期短，往往发生 Gc 的频率较高，回收速度比较快;Full Gc/Major GC 发生在老年代，一般情况下，触发老年代 GC的时候不会触发 Minor GC,但是通过配置，可以在 Full GC 之前进行一次 MinorGC 这样可以加快老年代的回收速度。  
面试题: 请问了解Minor GC和Full GC么，这两种GC有什么不一样吗  
1、Minor GC又称为新生代GC : 指的是发生在新生代的垃圾收集。因为Java对象大多都具备朝生夕灭的特性，因此Minor GC(采用复制算法)非常频繁，一般回收速度也比较快。  
2、Full GC 又称为 老年代GC或者Major GC : 指发生在老年代的垃圾收集。出现了Major GC，经常会伴随至少一次的Minor GC(并非绝对，在Parallel Scavenge收集器中就有直接进行Full GC的策略选择过程)。  
3、Major GC的速度一般会比Minor GC慢10倍以上。

**HTTP状态码：**

200:ok,客户端请求成功，正常响应  
204：正常响应，但没有实体  
206：范围请求，返回部分数据  
301:永久重定向  
302：临时重定向  
303：和302类似，但必须使用GET方法  
304：状态未改变  
307：临时重定向，不该改变请求方法  
400：请求报文语法错误  
401：需要认证  
403：服务器拒绝访问对应的资源  
404：请求的资源不存在  
500：服务器故障  
503：服务器处于超负载或正在停机维护

## **HTTP和HTTPS：**

简单理解,http传送数据（包括账号和密码），都是明文传送，很容易被窃取或者侦听，在现有的互联网应用中，很明显有不安全因素，所以有了https，可以简单理解成https多了一层加密解密层，  
简单来说，HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全。  
http+加密+认证+完整性保护=https  
HTTPS的特点  
内容加密：采用混合加密技术，中间者无法直接查看明文内容  
验证身份：通过证书认证客户端访问的是自己的服务器  
保护数据完整性：防止传输的内容被中间人冒充或者篡改  
HTTPS和HTTP的区别主要如下:  
1、https协议需要到ca申请证书，一般免费证书较少，因而需要一-定费用。  
2、http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。  
3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80,后者是443。  
4、http的连接很简单，是无状态的; HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比htt

https加密方式  
先来了解一下对称加密和非对称加密  
****对称加密****：加密和解密都是同一个密匙。即发送方和接收方都必须使用相同的密钥对明文进行加密解密；加密速度比较快；

****非对称加密****：有两个密钥，公开秘钥（publickey）和私有密钥（privatekey）是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。加密速度比较慢，非对称加密将公钥暴露，供客户端加密，服务端使用私钥解密。

对称加密和非对称加密结合方式  
浏览器使用Https的URL访问服务器，建立SSL链接，服务器收到SSL链接，发送非对称加密的公钥A返回给浏览器，浏览器生成随机数，作为对称加密的密钥B；浏览器使用公钥A，对自己生成的密钥B进行加密，得到密钥C；浏览器将钥C，发送给服务器；服务器用私钥D对接受的密钥C进行解密，得到对称加密钥B。浏览器和服务器之间可以用密钥B作为对称加密密钥进行通信。

## **前置知识泛型**

比如你平时装米的袋子只能装米，装面的袋子只能装面，现在这个袋子啥都能装。  
语法格式；<>  
1、泛型是为了解决某些容器、算法等代码的通用性而引入，并且能在编译期间做类型检查。  
2.、泛型利用的是 Object 是所有类的祖先类，并且父类的引用可以指向子类对象的特定而工作。  
3.、泛型是一种编译期间的机制，即 MyArrayList 和 MyArrayList 在运行期间是一个类型。  
4、泛型是 java 中的一种合法语法，标志就是尖括号 <>  
5、<>尖括号里面可以有多个变量，用逗号隔开。

## **Arraylist**

底层数据结构：动态数组（顺序表）  
元素类型：Object,即可以存放所有类型  
特点：地址连续，一旦数据存储好了，查询操作效率会比较高，因为有下标，但同时也因为地址连续， ArrayList要移动数据,所以插入和删除操作效率比较低。它是线程不安全的

## **怎样定义Arraylist**

1、创建一个无参的ArrayList集合

List<String> list = new ArrayList<>();

2、创建一个初始化长度为100的ArrayList集合

List<String> initlist = new ArrayList<>(100);

3、将其他类型的集合转为ArrayList，有参构造方法

List<String> setList = new ArrayList<>(new HashSet());

## **ArrayList的常用方法**

1、arr.add(object);添加一个元素  
2、arr.get(index);取出集合中的元素，在get方法的参数中，写入索引。  
3、arr.size();返回集合的长度，即存储元素的个数。  
4、arr.set(object);设置一个元素  
5、arr.remove();移除一个元素  
6、arr.clear():清空掉这个集合

为什么线程不安全？

## **LinkedList：**

底层数据结构：链表  
特点：地址是任意的，插入和删除速度快，访问速度慢。适用于要头尾操作或插入指定位置的场景。也是线程不安全的。

ArrayList和LinkedList  
共性：都是List接口的实现类，因此都实现了List的所有未实现的方法，只是实现的方式有所不同。  
区别：List接口的实现方式不同  
ArrayList实现了List接口，以数组的方式来实现的，因此对于快速的随机取得对象的需求，使用ArrayList实现执行效率上会比较好。LinkedList是采用链表的方式来实现List接口的，因此在进行insert和remove动作时效率要比ArrayList高。适合用来实现Stack(堆栈)与Queue(队列)。

## **Vector**

与ArrayList一样，也是通过数组实现的，不同的是它支持线程的同步，是线程安全的。(即某一时刻只有一个线程能够写Vector，避免多线程同时写而引起的不一致性，但实现同步需要很高的花费，因此，访问它比访问ArrayList慢。)