### ****Java 基础****

1. **JDK 和 JRE 有什么区别？**

JDK：Java Development Kit 的简称，java 开发工具包，提供了 java 的开发环境和运行环境。

JRE：Java Runtime Environment 的简称，java 运行环境，为 java 的运行提供了所需环境。

JDK 其实包含了 JRE

**2. == 和 equals 的区别是什么？**

**== 解读**

对于基本类型和引用类型 == 的作用效果是不同的，如下所示：

基本类型：比较的是值是否相同；

引用类型：比较的是引用是否相同；

**equals 解读**

equals 默认情况下是地址，重写equals 方法后比较值

1. **this和Super区别**



## **This：代表当前对象的引用**

代表使用该方法的对象的引用，他指向自身的这个对象  
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：子类调用父类构造方法**  
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

**4. final 在 java 中有什么作用？**

* final 修饰的类叫最终类，该类不能被继承。
* final 修饰的方法不能被重写。
* final 修饰的变量叫常量，常量必须初始化，初始化之后值就不能被修改。

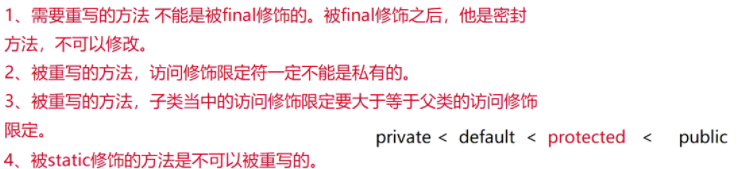
**5.重载与重写的区别**

**重写（override）：发生在有继承关系的类之间**

子类定义了与父类方法名称、参数类型及个数完全相同的方法。但不能够拥有比父类更为严格的访问控制权限（private<default<public）。

如果现在父类方法使⽤private定义，子类中使⽤public覆写，对吗？

答：如果现在父类方法使用了private定义，那么就表示该方法只能被父类使⽤，子类无法使用





**6. String 属于基础的数据类型吗？**

String 不属于基础类型，基础类型有 8 种：byte、boolean、char、short、int、float、long、double，而 String 属于对象。

**7. java 中操作字符串都有哪些类？它们之间有什么区别？**

操作字符串的类有：String、StringBuffer、StringBuilder。

1. String 和 StringBuffer、StringBuilder 的区别在String不可变，而 StringBuffer、StringBuilder 可以改变。
2. StringBuffer和StringBuilder的区别在于，StringBuffer 是线程安全的，而 StringBuilder 是非线程安全的，但 StringBuilder 的性能却高于 StringBuffer，所以在单线程环境下推荐使用 StringBuilder，多线程环境下推荐使用 StringBuffer。

**8. String str="i"与 String str=new String("i")一样吗？**

不一样，因为内存的分配方式不一样。String str="i"的方式，在常量池中；而 String str=new String("i") 则会被分到堆内存中。

**10. String 类的常用方法都有那些？**

indexOf()：返回指定字符的索引。

charAt()：返回指定索引处的字符。

getBytes()：返回字符串的 byte 类型数组。

toLowerCase()：将字符串转成小写字母。

toUpperCase()：将字符串转成大写字符。

substring()：截取字符串。

equals()：字符串比较。

replace()：字符串替换。

trim()：去除字符串两端空白。

split()：分割字符串，返回一个分割后的字符串数组。

length()：返回字符串长度。

**11.普通类和抽象类有哪些区别？**

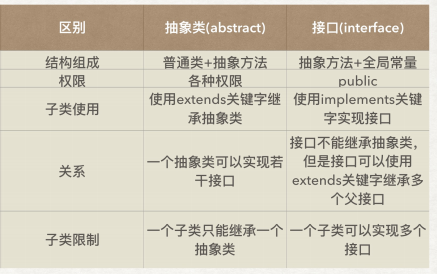
普通类不能包含抽象方法，抽象类可以包含抽象方法，也可以不包含。

抽象类不能直接实例化，普通类可以直接实例化。

**12. 抽象类能使用 final 修饰吗？**

不能，定义抽象类需要让其他类继承，如果定义为 final 该类就不能被继承

1. **接口和抽象类有什么区别？**



1. **深拷贝与浅拷贝区别**

浅拷贝：只是复制了对象的引用地址，两个对象指向同一个内存地址，所以修改其中任意的值，另一个值都会随之变化。

深拷贝：是将对象及值复制过来，两个对象修改其中任意的值另一个值不会改变，这就是深拷贝，但是此方法无法复制函数类型）

**15.关于异常？**

异常：指的就是程序在 **运行时** 出现错误通知调用者的一种机制.

**运行时异常（非受查异常）**：派生于 Error 类或 RuntimeException 类的所有异常

**编译时异常（受查异常）**：除非受查异常之外的其他异常，需要进行显示处理，否则编译不通过

**两者区别：**

非受查异常：java编译器不要求你一定要把它捕获或者一定要继续抛出

受查异常：要求你必须要在方法里面或者捕获或者继续抛出。

非受查：NullPointerException,ClassCastException,ArrayIndexsOutOfBoundsException,ArithmeticException(算术异常，除0溢出)

受查：Exception,FileNotFoundException,IOException,SQLException.

**16.Error与Expection区别？**

Error 指的是 Java 运行时内部错误和资源耗尽错误，应用程序不抛出此类异常. 告知用户并使程序终止。

Exception 是我们程序猿所使用的异常类的父类.

**17.java 中 IO 流分为几种？**

按功能来分：输入流（input）、输出流（output）。

按类型来分：字节流和字符流。

字节流和字符流的区别是：字节流按 8 位传输以字节为单位输入输出数据，字符流按 16 位传输以字符为单位输入输出数据。

1. **什么是面向对象？与面向过程有什么区别？**

1.面向过程：是一种以过程为中心的编程思想，自顶向下，逐步求精。

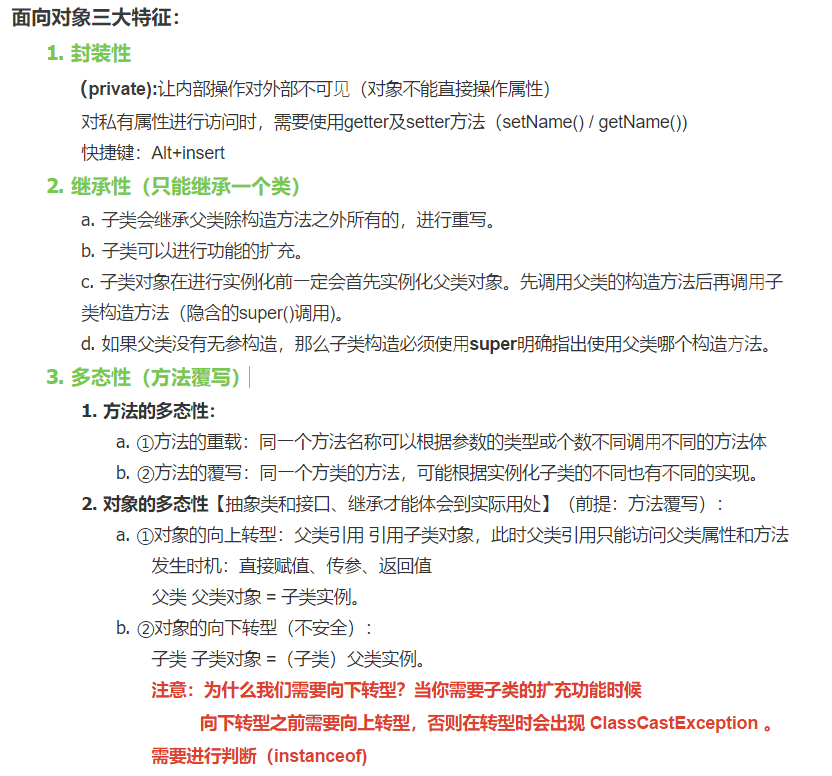
2.面向对象：是一类以对象作为基本程序结构单位的程序设计语言，指用于描述的设计是以对象为核心，而对象是程序运行时刻的基本成分。

二、特点不同

1.面向过程：就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。

2.面向对象语言：识认性，系统中的基本构件可识认为一组可识别的离散对象，对象具有唯一的静态类型和多个可能的动态类型，在基本层次关系的不同类中共享数据和操作。

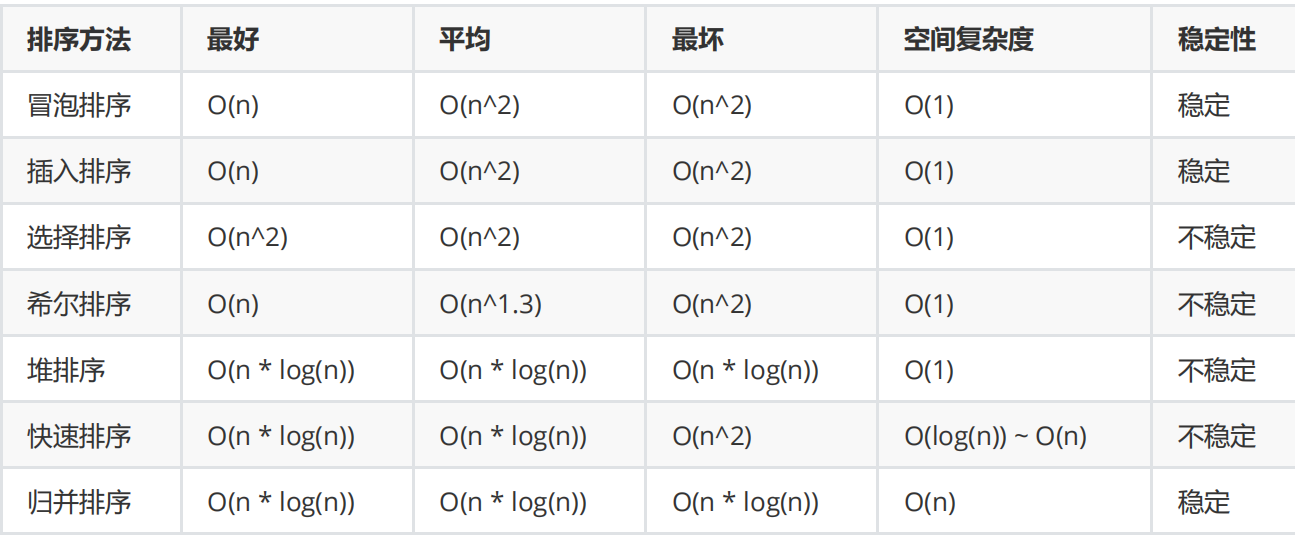
面向过程：不支持丰富的“面向对象”特性（比如继承、多态），并且不允许混合持久化状态和域逻辑。



**19. throw 和 throws 的区别？**

throws是用来声明一个方法可能抛出的所有异常信息，throws是将异常声明但是不处理，而是将异常往上传，谁调用我就交给谁处理。而throw则是指抛出的一个具体的异常类型。

**数据结构**



**1. Collection 和 Collections 有什么区别？**

java.util.Collection 是一个集合接口（集合类的一个顶级接口），其直接继承接口有List与Set。

Collections则是集合类的一个工具类/帮助类，其中提供了一系列静态方法，用于对集合中元素进行排序、搜索以及线程安全等各种操作。

**2.List、Set、Map 之间的区别是什么？**



List以索引来存取元素，有序的，元素是允许重复的，可以插入多个null。  
Set不能存放重复元素，无序的，只允许一个null  
Map保存键值对映射，映射关系可以一对一、多对一  
List有基于数组、链表实现两种方式  
Set、Map容器有基于哈希存储和红黑树两种方式实现  
Set基于Map实现，Set里的元素值就是Map的键值

**3.HashMap 和 Hashtable 有什么区别？**

hashTable同步的，而HashMap是非同步的，效率上逼hashTable要高。

hashMap允许空键值，而hashTable不允许。

HashMap更适合于单线程环境，而Hashtable适合于多线程环境

**HashMap**

* 1. 底层由链表+数组+红黑树实现
  2. 可以存储null键和null值
  3. 线性不安全
  4. 初始容量为16，扩容每次都是2的n次幂

**HashTable**

1. HashTable的底层也是由链表+数组+红黑树实现。
2. 无论key还是value都不能为null
3. 它是线性安全的，使用了synchronized关键字。

**ConcurrentHashMap**

* + - * 1. ConcurrentHashMap的底层是数组+链表+红黑树
        2. 不能存储null键和值
        3. ConcurrentHashMap是线程安全的

**4.如何决定使用 HashMap 还是 TreeMap？**

对于在Map中插入、删除和定位元素这类操作，HashMap是最好的选择。

需要对一个有序的key集合进行遍历，TreeMap是更好的选择。

## **5.堆和栈的区别？**

1.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.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.栈内存的更新速度要快于堆内存，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。

## **6.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；

3.栈底层用顺序表或者链表实现，队列采用链表或者数组实现

**7. ArrayList 和 LinkedList 的区别是什么？**

ArrrayList底层的数据结构是数组，支持随机访问，而 LinkedList 的底层数据结构是链表，不支持随机访问。

使用下标访问一个元素，ArrayList 的时间复杂度是 O(1)，而 LinkedList 是 O(n)。

**8. 如何实现数组和 List 之间的转换？**

List转换成为数组：调用ArrayList的toArray方法。

数组转换成为List：调用Arrays的asList方法。

**9.final、finally.finaalized区别？**

**Final：**可以用来修饰类、方法、成员变量。修饰类表示该类不能被集成，修饰方法表示该方法不能被重写，修饰成员变量表示该成员变量的值不能被修改。  
**finally：**一般配合try……catch使用，用来保证程序不论是正常执行或是抛出异常或是中途return，都会执行finally中的代码。  
**finalized：**这是配合GC的一个方法，在GC回收前都会先执行该方法。但由于致命缺陷已被废弃。

**10.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数组长度时指定的，而ArrayList大小是固定的。

Array没有提供ArrayList那么多功能，比如addAll、removeAll和iterator等。

**11.在 Queue中poll()和remove()有什么区别？**

poll() 和 remove() 都是从队列中取出一个元素，但是 poll() 在获取元素失败的时候会返回空，但是 remove() 失败的时候会抛出异常。

**12.哪些集合类是线程安全的？**

vector：就比arraylist多了个同步化机制（线程安全），因为效率较低，现在已经不太建议使用。在web应用中，特别是前台页面，往往效率（页面响应速度）是优先考虑的。

statck：堆栈类，先进后出。

hashtable：就比hashmap多了个线程安全。

enumeration：枚举，相当于迭代器。

**13.HashSet与Treeset？**

**TreeSet**：底层数据结构是红黑树，数据是自动排好序的，不允许放入null值 。

## **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，线程不安全

1. **hash表**

**哈希表（哈希表的插入/删除/查找时间复杂度是O(1) ）**

1.内部数据结构是数组

2.关键字经过变换（hash函数)得到int类型的变量

3.int类型值变为一个合法数组下标

4.把关键字放入数组的该下标

**1.哈希冲突：**不同关键字通过哈希函数得到相同的哈希地址

**2.冲突避免**：冲突无法避免（因为哈希表底层数组的容量往往是小于实际要存储的关键字的数量），只能尽量降低冲突率。

负载因子=填入表中的数据/表的长度

降低冲突率方法：

1. 哈希函数的设计使得下标均匀分布（除留余数法、平方取中法、直接定制法）
2. 调节负载因子（放入的元素个数 / 散列表长度）间接降低冲突
3. 扩容并进行重新哈希（遍历数组再次计算下标）

**3.解决冲突：开散列与闭散列**

**1.闭散列/开放定址法：把key存放到冲突位置中的“下一个” 空位置中去**

查找成功的平均查找长度：将每个元素需要经过查找的次数相加 / 元素个数

查找失败的平均查找长度：数组每个下标经过查找时找不到的次数相加（即从该位置开始遇到存在空白的位置需要次数) / 数组长度

**2.线性探测：**从发生冲突的位置开始，依次向后探测，直到寻找到下一个空位置为止。

缺陷：产生冲突的数据堆积在一块

**3.二次探测：**从冲突的为值开始，向后移动i^2的位置，i为第几次冲突

**4.开散列/哈希桶/链地址法：具有相同地址的关键码放入一个集合，每一个子集合称为一个桶，桶中的元素通过单链表链接起来，各链表的头结点存储在哈希表中。（头插）**

冲突严重时可以改进：

每个桶的背后是另一个哈希表

每个桶的背后是一棵搜索树

**4.与java类集关系**

HashMap 和 HashSet 即 java 中利用哈希表实现的 Map 和 Set

java 中使用的是哈希桶方式解决冲突的

**java 会在冲突链表长度大于一定阈值（8）后，将链表转变为搜索树（红黑树）**

java 中**计算哈希值实际上是调用的类的 hashCode 方法**，**进行 key 的相等性比较是调用 key 的 equals 方法**。所以如果要用自定义类作为 HashMap 的 key 或者 HashSet 的值，**必须覆写 hashCode 和 equals 方法**，而且要做到 equals 相等的对象，hashCode 一定是一致的。

不进行覆写得到的hashcode值和equals对象是不一样的

equals相等 hashcode相等反之，不一定可能存在冲突

### ****多线程****

**1. 并行和并发有什么区别？**

并行是指两个或者多个事件在同一时刻发生；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。

**2. 线程和进程的区别？**

进程是程序运行和资源分配的最小单位，一个程序至少有一个进程，一个进程至少有一个线程。

线程是是cpu调度和分配的最小单位。同一进程中的多个线程之间可以并发执行。

**3. 创建线程有哪几种方式？**

①. 继承Thread类

定义Thread类的子类，并重写该类的run方法，该run方法的方法体就代表了线程要完成的任务。调用线程对象的start()方法来启动该线程。

②. 通过Runnable接口

定义runnable接口的实现类，并重写该接口的run()方法，创建 Runnable实现类的实例，并依此实例作为Thread的target来创建Thread对象，调用线程对象的start()方法来启动该线程。

③. 通过Callable和FutureTask接口

创建Callable接口的实现类，并实现call()方法。创建Callable实现类的实例，使用FutureTask类来包装Callable对象，该FutureTask对象封装了该Callable对象的call()方法的返回值。

使用FutureTask对象作为Thread对象的target创建并启动新线程。

调用FutureTask对象的get()方法来获得子线程执行结束后的返回值。

**4. 说一下 runnable 和 callable 有什么区别？**

Runnable接口中的run()方法的返回值是void

Callable接口中的call()方法是有返回值的，是一个泛型，和Future、FutureTask配合可以用来获取异步执行的结果。

**5. 线程有哪些状态？**

线程通常都有五种状态，创建、就绪、运行、阻塞和死亡。

**6. sleep() 和 wait() 有什么区别？**

sleep()：方法是线程类（Thread）的静态方法，对象的锁没有被释放,没有锁也不请求

wait()：wait()是Object类的方法，释放对象的锁，使得其他线程能够访问，可以通过notify，notifyAll方法来唤醒等待的线程

sleep（）方法需要捕获异常，wait()方法不需要捕获异常

**7. notify()和 notifyAll()有什么区别？**

如果线程调用了对象的 wait()方法，那么线程便会处于该对象的等待池中，等待池中的线程不会去竞争该对象的锁。

调用了notify后只要一个线程会由等待池进入锁池，而notifyAll会将该对象等待池内的所有线程移动到锁池中，等待锁竞争。

**8.线程的 run()和 start()有什么区别？**

start()方法来启动一个线程，此时线程是处于就绪状态，并没有运行。

run()方法相当于一个任务清单，结束后进入Terminated状态

**9. 创建线程池有哪几种方式？**

①newFixedThreadPool(int nThreads)

创建一个**固定长度的线程池**，每当提交一个任务就创建一个线程，直到达到线程池的最大数量，这时线程规模将不再变化，当线程发生未预期的错误而结束时，线程池会补充一个新的线程。

②newCachedThreadPool()

创建一个**可缓存的线程池**，如果线程池的规模超过了处理需求，将自动回收空闲线程，而当需求增加时，则可以自动添加新线程，线程池的规模不存在任何限制。

③newSingleThreadExecutor()

这是一个**单线程的Executor**，它创建单个工作线程来执行任务，如果这个线程异常结束，会创建一个新的来替代它；它的特点是能确保依照任务在队列中的顺序来串行执行。

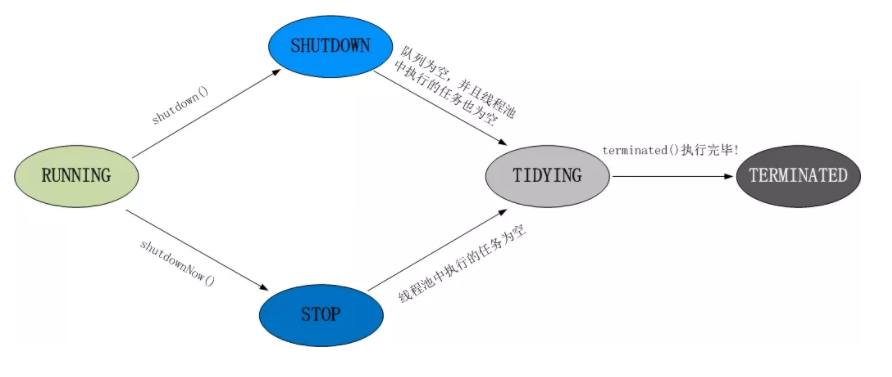
④newScheduledThreadPool(int corePoolSize)

创建了一个**固定长度的线程池**，而且**以延迟或定时的方式**来执行任务，类似于Timer。

**10. 线程池都有哪些状态？**

线程池有5种状态：Running、ShutDown、Stop、Tidying、Terminated。

线程池各个状态切换框架图：



**11. 线程池中 submit()和 execute()方法有什么区别？**

接收的参数不一样。submit有返回值，而execute没有，submit方便Exception处理

**12. 在 java 程序中怎么保证多线程的运行安全？**

线程安全在三个方面体现：

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地被其他线程看到，（synchronized,volatile）；

代码重排序：书写的代码顺序怒一定是执行顺序。

**13. 什么是死锁？**

死锁是指两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象。这些永远在互相等待的进程称为死锁进程。

**14. 怎么防止死锁？**

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
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
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
4. **循环等待条件**：是指进程发生死锁后，若干进程之间形成一种头尾相接的循环等待资源关系

**15.说一下 synchronized 底层实现原理？**

synchronized可以保证方法或者代码块在运行时，同一时刻只有一个方法可以进入到临界区，同时它还可以保证共享变量的内存可见性。

Java中每一个对象都可以作为锁，这是synchronized实现同步的基础：

普通同步方法，锁是当前实例对象

静态同步方法，锁是当前类的class对象

同步方法块，锁是括号里面的对象

**16. synchronized 和 volatile 的区别是什么？**

* 1. volatile告诉jvm当前变量在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，其他线程被阻塞住。

* 1. 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。
  2. volatile仅能实现变量的可见性，不能保证原子性；

synchronized则可以保证变量的可见性和原子性。

* 1. volatile不会造成线程的阻塞；synchronized可能会造成线程的阻塞。

**17.懒汉模式**



**18.**什**么是 java 序列化？什么情况下需要序列化？**

**序列化:对象转为字节序列的过程**

**反序列化：字节序列恢复为对象的过程**

序列化的目的；为了持久的保存在内存中的各种对象的状态（也就是实例变量，不是方法）

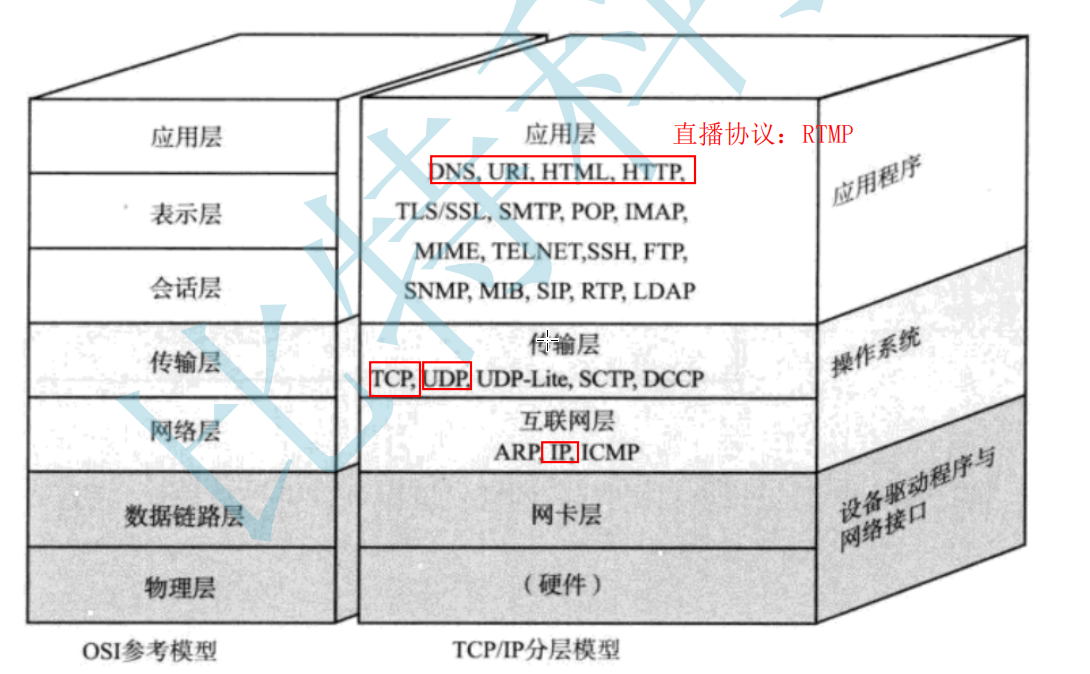
实现方式： java对象序列化、Json序列化。XML、FST、Hession

Java对象序列化：

序列化:ObjectOutPutStream.writeObject(objece obj)

反序列化：ObjectOutPutStream.readObject()

**JavaWeb**



**1.http 响应码**

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

4XX:客户端错误  
400：请求错误  
401：需要认证  
403：没权限

404：找不到 — 服务器找不到给定的资源；    
5xx  服务器中出现的错误    
500  内部错误 — 因为意外情况，服务器不能完成请求。

**2**.** forward 和 redirect 的区别？**

Forward和Redirect代表了两种请求转发方式：直接转发和间接转发。

**直接转发方式（Forward）**，客户端和浏览器只发出一次请求，Servlet、HTML、JSP或其它信息资源，由第二个信息资源响应该请求，在请求对象request中，保存的对象对于每个信息资源是共享的。

**间接转发方式（Redirect）**实际是两次HTTP请求，服务器端在响应第一次请求的时候，让浏览器再向另外一个URL发出请求，从而达到转发的目的。

**3. 简述TCP和UDP的区别？**

1. TCP可靠 有链接 面向数据流

UDP不可靠 无连接的，面向数据报

1. UDP传输效率高于UDP
2. TCP可靠，保证数据无差错，不丢失，不重复，且按序到达;

UDP尽最大努力交付，不保证可靠交付。

1. TCP连接是点到点的;

UDP支持一对一，一对多，多对一和多对多的交互通信。

**4.TCP为什么要三次握手，两次不行吗？为什么？**

为了实现可靠数据传输， TCP 协议的通信双方， 都必须维护一个序列号， 以标识发送出去的数据包中， 哪些是已经被对方收到的。 三次握手的过程即是通信双方相互告知序列号起始值， 并确认对方已经收到了序列号起始值的必经步骤。

如果只是两次握手， 至多只有连接发起方的起始序列号能被确认， 另一方选择的序列号则得不到确认。

**5.说一下 tcp 粘包是怎么产生的？**

**①. 发送方产生粘包**

发送的数据包太小，那么TCP协议默认的会将这些较小的数据包进行合并发送

**②. 接收方产生粘包**

接收方不能及时的把缓冲区中的数据拿出来，而下一个数据又到来并有一部分放入的缓冲区末尾，等我们读取数据时就是一个粘包。

**6.get 和 post 请求有哪些区别？**

**1.Get不带请求体，post带请求体**

**Get带请求体在QueryString/segment中**

2.GET在浏览器回退时是无害的，而POST会再次提交请求。

3.GET请求会被浏览器主动cache，而POST不会，除非手动设置。

**7.jsp 和 servlet 有什么区别？**

JSP的本质就是Servlet，JVM只能识别java的类，不能识别JSP的代码，Web容器将JSP的代码编译成JVM能够识别的java类

jsp更擅长表现于页面显示，servlet更擅长于逻辑控制。

Servlet中没有内置对象，Jsp中的内置对象都是必须通过HttpServletRequest对象，HttpServletResponse对象以及HttpServlet对象得到。

**8.session 和 cookie 有什么区别？**

由于HTTP协议是无状态的协议，所以服务端需要记录用户的状态时，就需要Session.

Session:保存在服务端的，有一个唯一标识。

Cookie:服务端识别特定的客户这个时候,保存在客户端。

Session是在服务端保存的一个数据结构，用来跟踪用户的状态，这个数据可以保存在集群、数据库、文件中；Cookie是客户端保存用户信息的一种机制，用来记录用户的一些信息，也是实现Session的一种方式。

**9.说一下 session 的工作原理？**

session是一个存在服务器上的类似于一个散列表格的文件。里面存有我们需要的信息，在我们需要用的时候可以从里面取出来。类似于一个大号的map吧，里面的键存储的是用户的sessionid，用户向服务器发送请求的时候会带上这个sessionid。这时就可以从中取出对应的值了。

**数据库**

1. **索引**

索引是满足某种特定查找算法的数据结构，而这些数据结构会以某种方式指向数据，从而实现高效查找数据。

具体来说 MySQL 中的索引，不同的数据引擎实现有所不同，但目前主流的数据库引擎的索引都是 B+ 树实现的，B+ 树的搜索效率，可以到达二分法的性能，找到数据区域之后就找到了完整的数据结构了，所有索引的性能也是更好的。

**2.事务的四大特性ACID 是什么？**

Atomicity（原子性）：事务不可分割、不可约简。

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

Isolation（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。

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。

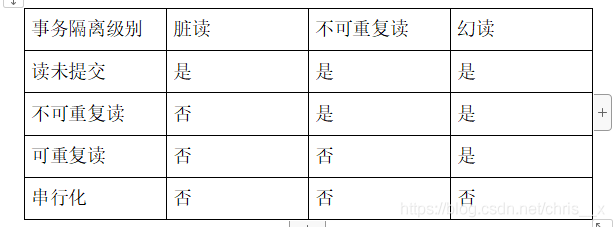
在并发状态下事务会出现一些问题，像上面这三种情况；

****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；

1. ****说一下数据库的事务隔离？****

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

**4.mysql 的内连接、左连接、右连接有什么区别？**

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，右边的表显示出符合条件的数据；右连接正好相反。

**5.怎么验证 mysql 的索引是否满足需求？**

使用 explain 查看 SQL 是如何执行查询语句的，从而分析你的索引是否满足需求。

explain 语法：explain select \* from table where type=1。

**6.说一下乐观锁和悲观锁？**

**乐观锁：**每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在提交更新的时候会判断一下在此期间别人有没有去更新这个数据。

**悲观锁：**每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会阻止，直到这个锁被释放。

**7.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阻塞处于死锁状态的进程称为死锁进程；

### **2.死锁产生的条件**

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
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
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
4. **循环等待条件**：是指进程发生死锁后，若干进程之间形成一种头尾相接的循环等待资源关系

### **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。

****2.避免死锁****：银行家算法

****3.检测死锁****：确定是否存在“循环等待”条件

****4.解除死锁****：

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；

**JVM及垃圾回收**

**1. 说一下 jvm 的主要组成部分？及其作用？**

类加载器、运行时数据区、执行引擎、本地库接口

组件的作用： 首先通过类加载器把 Java 代码转换成字节码，运行时数据区再把字节码加载到内存中，而字节码文件只是 JVM 的一套指令集规范，并不能直接交个底层操作系统去执行，因此需要特定的命令解析器执行引擎，将字节码翻译成底层系统指令，再交由 CPU 去执行，而这个过程中需要调用其他语言的本地库接口来实现整个程序的功能。

**2.. 说一下 jvm 运行时数据区？**

程序计数器

虚拟机栈

本地方法栈

堆

方法区

**3..说一下堆栈的区别？**

1. 栈内存存储的是局部变量而堆内存存储的是实体；

2. 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；

3. 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。

**4.. 说一下类加载的执行过程？**

类加载分为以下 5 个步骤：

1. 加载：根据查找路径找到相应的 class 文件然后导入；
2. 检查：检查加载的 class 文件的正确性；
3. 准备：给类中的静态变量分配内存空间；
4. 解析：虚拟机将常量池中的符号引用替换成直接引用的过程。符号引用就理解为一个标示，而在直接引用直接指向内存中的地址；
5. 初始化：对静态变量和静态代码块执行初始化工作。

**5. 怎么判断对象是否可以被回收？**

一般有两种方法来判断：

1. 引用计数器：为每个对象创建一个引用计数，有对象引用时计数器 +1，引用被释放时计数 -1，当计数器为 0 时就可以被回收。它有一个缺点不能解决循环引用的问题；
2. 可达性分析：从 GC Roots 开始向下搜索，搜索所走过的路径称为引用链。当一个对象到 GC Roots 没有任何引用链相连时，则证明此对象是可以被回收的。

**6. 说一下 jvm 有哪些垃圾回收算法？**

标记-清除算法

标记-整理算法

复制算法

分代算法

****7. 说一下 jvm 有哪些垃圾回收器？****

Serial：最早的单线程串行垃圾回收器。

Serial Old：Serial 垃圾回收器的老年版本，同样也是单线程的，可以作为 CMS 垃圾回收器的备选预案。

ParNew：是 Serial 的多线程版本。

Parallel 和 ParNew 收集器类似是多线程的，但 Parallel 是吞吐量优先的收集器，可以牺牲等待时间换取系统的吞吐量。

Parallel Old 是 Parallel 老生代版本，Parallel 使用的是复制的内存回收算法，Parallel Old 使用的是标记-整理的内存回收算法。

CMS：一种以获得最短停顿时间为目标的收集器，非常适用 B/S 系统。

G1：一种兼顾吞吐量和停顿时间的 GC 实现，是 JDK 9 以后的默认 GC 选项。

**8. 详细介绍一下 CMS 垃圾回收器？**

CMS 是以牺牲吞吐量为代价来获得最短回收停顿时间的垃圾回收器。对于要求服务器响应速度的应用上，这种垃圾回收器非常适合。在启动 JVM 的参数加上“-XX:+UseConcMarkSweepGC”来指定使用 CMS 垃圾回收器。

CMS 使用的是标记-清除的算法实现的，所以在 gc 的时候回产生大量的内存碎片，当剩余内存不能满足程序运行要求时，系统将会出现 Concurrent Mode Failure，临时 CMS 会采用 Serial Old 回收器进行垃圾清除，此时的性能将会被降低。

****9.新生代垃圾回收器和老生代垃圾回收器都有哪些？有什么区别？****

新生代回收器：Serial、ParNew、Parallel Scavenge

老年代回收器：Serial Old、Parallel Old、CMS

整堆回收器：G1

新生代垃圾回收器一般采用的是复制算法，复制算法的优点是效率高，缺点是内存利用率低；老年代回收器一般采用的是标记-整理的算法进行垃圾回收。

**10. 说一下 jvm 调优的工具？**

JDK 自带了很多监控工具，都位于 JDK 的 bin 目录下，其中最常用的是 jconsole 和 jvisualvm 这两款视图监控工具。

jconsole：用于对 JVM 中的内存、线程和类等进行监控；

jvisualvm：JDK 自带的全能分析工具，可以分析：内存快照、线程快照、程序死锁、监控内存的变化、gc 变化等。

**11. 常用的 jvm 调优的参数都有哪些？**

-Xms2g：初始化推大小为 2g；

-Xmx2g：堆最大内存为 2g；

-XX:NewRatio=4：设置年轻的和老年代的内存比例为 1:4；

-XX:SurvivorRatio=8：设置新生代 Eden 和 Survivor 比例为 8:2；

–XX:+UseParNewGC：指定使用 ParNew + Serial Old 垃圾回收器组合；

-XX:+UseParallelOldGC：指定使用 ParNew + ParNew Old 垃圾回收器组合；

-XX:+UseConcMarkSweepGC：指定使用 CMS + Serial Old 垃圾回收器组合；

-XX:+PrintGC：开启打印 gc 信息；

-XX:+PrintGCDetails：打印 gc 详细信息。

**12..垃圾回收算法优缺点**：

（1）标记清除算法：

优点：此算法实现起来较容易。

缺点：效率问题。标记和清除两个过程的效率都不高。

空间问题。会产生大量不连续的内存碎片。空间碎片过多会导致内存不够，从而触发垃圾收集。

（2）复制算法：

优点：实现简单，运行高效，且不容易产生内存碎片。

缺点：是将内存缩小为了原来的一半。

（3）标记整理算法：和标记清除算法 标记过程一样，但是在标记完成之后，它是将存活的的对象都向另一端移动，然后清理掉端边界以外的内存。

（4）分代收集算法：当前JVM垃圾收集都采用的是分代收集算法，此算法并没有新的思想，只是根据对象存货周期的不同将内存划分为几块。