**1、 equals与==的区别：**

==是判断两个变量或实例是不是指向同一个内存空间

equals是判断两个变量或实例所指向的内存空间的值是不是相同

**2 、Object有哪些公用方法？**

方法equals测试的是两个对象是否相等

方法clone进行对象拷贝

方法getClass返回和当前对象相关的Class对象

方法notify,notifyall,wait都是用来对给定对象进行线程同步的

**3 、Hashcode的作用，与 equal 有什么区别**

同样用于鉴定2个对象是否相等的，java集合中有 list 和 set 两类，其中 set不允许元素重复实现，那个这个不允许重复实现的方法，如果用 equal 去比较的话，如果存在1000个元素，你 new 一个新的元素出来，需要去调用1000次 equal 去逐个和他们比较是否是同一个对象，这样会大大降低效率。hashcode实际上是返回对象的存储地址，如果这个位置上没有元素，就把元素直接存储在上面，如果这个位置上已经存在元素，这个时候才去调用equal方法与新元素进行比较，相同的话就不存了，散列到其他地址上

**集合问题**

1. **Linkedlist和arraylist的区别**

1.ArrayList是实现了基于动态数组的数据结构，LinkedList基于链表的数据结构。

2.对于随机访问get和set，ArrayList觉得优于LinkedList，因为LinkedList要移动指针。

3.对于新增和删除操作add和remove，LinedList比较占优势，因为ArrayList要移动数据。

1. **HashMap和TreeMap**

1.HashMap不是有序的，按照哈希值来的

2.LinkHashMap是按照插入或者访问的顺序 排序，LinkedHashMap 继承自 HashMap，在 HashMap 的基础上，又在内部增加了一个链表，用以存放元素的顺序。

3.TreeMap可以根据Key的大小进行排序，实现了 SortedMap 接口，这意味着它可以对元素进行排序；TreeMap的实现是红黑树算法的实现， 红黑树又称红-黑二叉树，它首先是一颗二叉树，它具体二叉树所有的特性。同时红黑树更是一颗自平衡的排序二叉树。（维持平衡的过程就是排序的过程）

1. **ArrayList、LinkedList、Vector的区别**

ArrayList 和Vector底层是采用数组方式存储数据，Vector由于使用了synchronized方法（线程安全）所以性能上比ArrayList要差

LinkedList使用双向链表实现存储，随机存取比较慢

HashMap的底层源码实现：当我们往HashMap中put元素的时候，先根据key的hashCode重新计算hash值，根据hash值得到这个元素在数组中的位置（即下标），如果数组该位置上已经存放有其他元素了，那么在这个位置上的元素将以链表的形式存放，新加入的放在链头，最先加入的放在链尾。如果数组该位置上没有元素，就直接将该元素放到此数组中的该位置上。

**4．HashMap和 HashTable 的区别**

HashTable比较老，是基于Dictionary 类实现的，HashTable 则是基于 Map接口实现的

HashTable 是线程安全的， HashMap 则是线程不安全的

HashMap可以让你将空值作为一个表的条目的key或value

**Java的虚拟机**

**1.GC:**

目前主流的编程语言(java,C#等)的主流实现中,都是称通过可达性分析(Reachability Analysis)来判定对象是否存活的。这个算法的基本思路就是通过一系列的称为“GC Roots”的对象作为起始点,从这些节点开始向下搜索,搜索所走过的路径称为引用链(Reference Chain),当一个对象到GC Roots没有任何引用链相连(用图论的话来说,就是从GC Roots到这个对象不可达)时,则证明此对象是不可用的。

时间：程序员不能具体控制时间，系统在不可预测的时间调用System.gc()函数的时候；当然可以通过调优，用NewRatio控制newObject和oldObject的比例，用MaxTenuringThreshold 控制进入oldObject的次数，使得oldObject 存储空间延迟达到full gc,从而使得计时器引发gc时间延迟OOM的时间延迟，以延长对象生存期。

对象：超出了作用域或引用计数为空的对象；从gc root开始搜索找不到的对象，而且经过一次标记、清理，仍然没有复活的对象。

做什么：删除不使用的对象，回收内存空间；运行默认的finalize,当然程序员想立刻调用就用dipose调用以释放资源如文件句柄，JVM用from survivor、to survivor对它进行标记清理，对象序列化后也可以使它复活。

**垃圾回收机制：**

标记回收法：遍历对象图并且记录可到达的对象，以便删除不可到达的对象，一般使用单线程工作并且可能产生内存碎片

标记-压缩回收法：前期与第一种方法相同，只是多了一步，将所有的存活对象压缩到内存的一端，这样内存碎片就可以合成一大块可再利用的内存区域，提高了内存利用率

复制回收法：把现有内存空间分成两部分，gc运行时，它把可到达对象复制到另一半空间，再清空正在使用的空间的全部对象。这种方法适用于短生存期的对象，持续复制长生存期的对象则导致效率降低。

分代回收发：把内存空间分为两个或者多个域，如年轻代和老年代，年轻代的特点是对象会很快被回收，因此在年轻代使用效率比较高的算法。当一个对象经过几次回收后依然存活，对象就会被放入称为老年的内存空间，老年代则采取标记-压缩算法

**2、类加载器**

**有哪些类加载器加载什么类：**

启动类加载器（Bootstrap class loader）：这个类装载器是在JVM启动的时候创建的。它负责装载Java API（java 核心类库），包含Object对象。和其他的类装载器不同的地方在于这个装载器是通过native code来实现的，而不是用Java代码。

扩展类加载器（Extension class loader）：它装载除了基本的Java API以外的扩展类。它也负责装载其他的安全扩展功能。

系统类加载器（System class loader）：如果说bootstrap class loader和extension class loader负责加载的是JVM的组件，那么system class loader负责加载的是应用程序类。它负责加载用户在$CLASSPATH里指定的类。

**类加载器之间的父子关系：**

双亲委派模型，委托关系

**什么是双亲委派模型：**

双亲委派模型的工作过程为：如果一个类加载器收到了类加载的请求，它首先不会自己去尝试加载这个类，而是把这个请求委派给父类加载器去完成，每一个层次的加载器都是如此，因此所有的类加载请求都会传给顶层的启动类加载器，只有当父加载器反馈自己无法完成该加载请求（该加载器的搜索范围中没有找到对应的类）时，子加载器才会尝试自己去加载。

双亲委派模型是通过loadClass()方法来实现的，根据代码以及代码中的注释可以很清楚地了解整个过程其实非常简单：先检查是否已经被加载过，如果没有则调用父加载器的loadClass()方法，如果父加载器为空则默认使用启动类加载器作为父加载器。如果父类加载器加载失败，则先抛出ClassNotFoundException，然后再调用自己的findClass()方法进行加载。

**为什么使用这个模型：**

保证统一了，Java类随着它的类加载器一起具备了一种带有优先级的层次关系。例如java.lang.Object类，无论哪个类加载器去加载该类，最终都是由启动类加载器进行加载，因此Object类在程序的各种类加载器环境中都是同一个类。

**如何自定义自己的类加载器：**

**3、内存**

**程序计数器：**

程序计数器是一块较小的内存空间，可以看作是当前线程所执行的字节码的行号指示器。如果线程正在执行的是一个Java 方法，这个计数器记录的是正在执行的虚拟机字节码指令的地址；如果正在执行的是Natvie 方法，这个计数器值则为空（Undefined）。此内存区域是唯一一个在Java 虚拟机规范中没有规定任何OutOfMemoryError情况的区域。

**虚拟机栈：**

线程私有，它的生命周期与线程相同。虚拟机栈描述的是Java 方法执行的内存模型：每个方法被执行的时候都会同时创建一个栈帧（Stack Frame）用于存储局部变量表、操作栈、动态链接、方法出口等信息。

**本地方法栈：**

本地方法栈（Native MethodStacks）与虚拟机栈所发挥的作用是非常相似的，其区别不过是虚拟机栈为虚拟机执行Java 方法（也就是字节码）服务，而本地方法栈则是为虚拟机使用到的Native 方法服务。

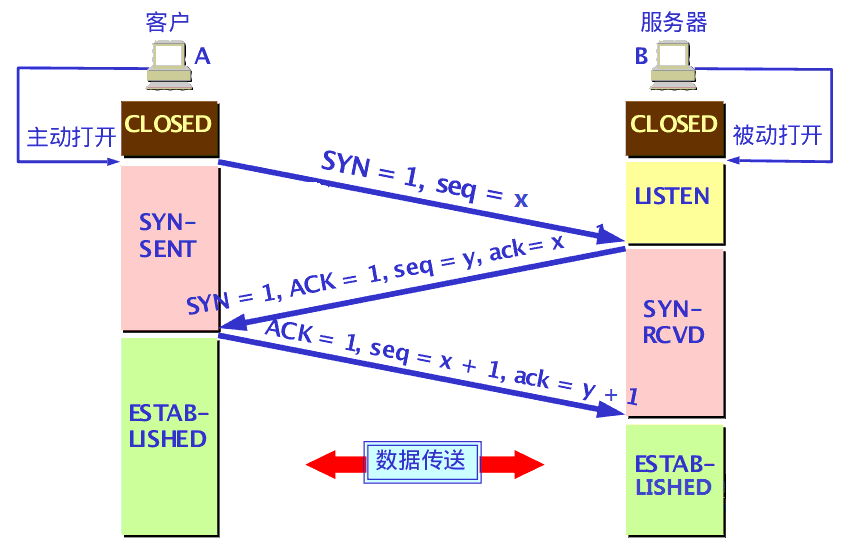
**堆：**

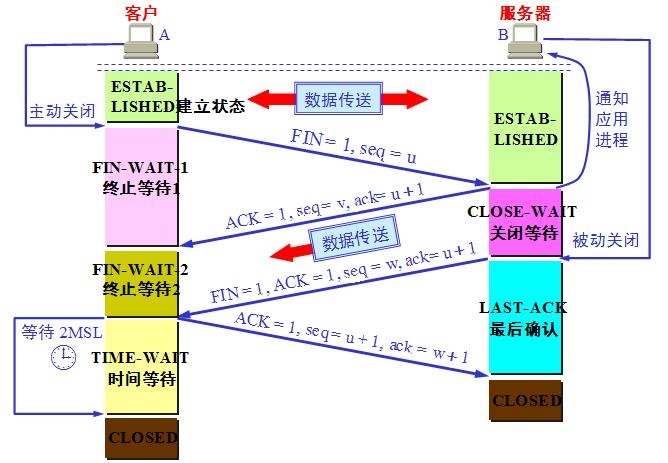
堆是Java 虚拟机所管理的内存中最大的一块。Java 堆是被所有线程共享的一块内存区域，在虚拟机启动时创建。此内存区域的唯一目的就是存放对象实例，几乎所有的对象实例都在这里分配内存。**堆是垃圾收集器管理的主要区域，因此很多时候也被称做“GC 堆”**。

**方法区：**

方法区在一个jvm实例的内部，类型信息被存储在一个称为方法区的内存逻辑区中。类型信息是由类加载器在类加载时从类文件中提取出来的。类(静态)变量也存储在方法区中。一个.class文件是类被java虚拟机使用之前的表现形式，一旦这个类要被使用，java虚拟机就会对其进行装载、连接（验证、准备、解析）和初始化。而装载（后的结果就是由.class文件转变为方法区中的一段特定的数据结构。

**TCP/IP协议**





**1、SYN Flood 攻击原理及防御**

SYN Flood是当前最流行的DoS（拒绝服务攻击）与DDoS（分布式拒绝服务攻击）的方式之一，这是一种利用TCP协议缺陷，发送大量伪造的TCP连接请求，从而使得被攻击方资源耗尽（CPU满负荷或内存不足）的攻击方式。

**原理：**

问题出在TCP连接的三次握手中，恶意的攻击者大量发送SYN报文，服务器端将为了维护一个非常大的半连接列表而消耗非常多的资源----数以万计的半连接，即使是简单的保存并遍历也会消耗非常多的CPU时间和内存，何况还要不断对这个列表中的IP进行SYN+ACK的重试。实际上如果服务器的TCP/IP栈不够强大，最后的结果往往是堆栈溢出崩溃---即使服务器端的系统足够强大，服务器端也将忙于处理攻击者伪造的TCP连接请求而无暇理睬客户的正常请求（毕竟客户端的正常请求比率非常之小），此时从正常客户的角度看来，服务器失去响应，这种情况我们称作：服务器端受到了SYN Flood攻击（SYN洪水攻击）。

**攻击方式：**

Direct Attack 攻击方使用固定的源地址发起攻击，这种方法对攻击方的消耗最小

Spoofing Attack 攻击方使用变化的源地址发起攻击，这种方法需要攻击方不停地修改源地址，实际上消耗也不大

Distributed Direct Attack 这种攻击主要是使用僵尸网络进行固定源地址的攻击

**防御方法：**

1、无效连接监视释放

不停监视系统的半开连接和不活动连接，当达到一定阈值时拆除这些连接，从而释放系统资源。

2、延缓TCB分配方法

从前面SYN Flood原理可以看到，消耗服务器资源主要是因为当SYN数据报文一到达，系统立即分配TCB，从而占用了资源。而SYN Flood由于很难建立起正常连接，因此，当正常连接建立起来后再分配TCB则可以有效地减轻服务器资源的消耗。常见的方法是使用Syn Cache和Syn Cookie技术。

Syn Cache技术：

这种技术是在收到SYN数据报文时不急于去分配TCB，而是先回应一个SYN ACK报文，并在一个专用HASH表（Cache）中保存这种半开连接信息，直到收到正确的回应ACK报文再分配TCB。

Syn Cookie技术：

Syn Cookie技术则完全不使用任何存储资源，它使用一种特殊的算法生成Sequence Number，这种算法考虑到了对方的IP、端口、己方IP、端口的固定信息，以及对方无法知道而己方比较固定的一些信息，如MSS、时间等，在收到对方的ACK报文后，重新计算一遍，看其是否与对方回应报文中的（Sequence Number-1）相同，从而决定是否分配TCB资源。

3、使用SYN Proxy防火墙

防火墙中提供一种SYN代理的功能，主要原理是对试图穿越的SYN请求进行验证后才放行

**2、TCP/IP协议的工作流程如下：**

* 源主机上，应用层将一串应用数据流传送给传输层。
* 输层将应用层的数据流截成分组，并加上TCP报头形成TCP段，送交网络层。
* 网络层给TCP段加上包括源、目的主机IP地址的IP报头，生成一个IP数据包，并将IP数据包送交链路层。
* 路层在其MAC帧的数据部分装上IP数据包，再加上源、目的主机的MAC地址和帧头，并根据其目的MAC地址，将MAC帧发往目的主机或IP路由器。
* 目的主机，链路层将MAC帧的帧头去掉，并将IP数据包送交网络层。
* 网络层检查IP报头，如果报头中校验和与计算结果不一致，则丢弃该IP数据包；若校验和与计算结果一致，则去掉IP报头，将TCP段送交传输层。
* 传输层检查顺序号，判断是否是正确的TCP分组，然后检查TCP报头数据。若正确，则向源主机发确认信息；若不正确或丢包，则向源主机要求重发信息。
* 目的主机，传输层去掉TCP报头，将排好顺序的分组组成应用数据流送给应用程序。这样目的主机接收到的来自源主机的字节流，就像是直接接收来自源主机的字节流一样。

**3、ARP的中文意思是（地址解析协议）工作原理。**

　 1. 首先，每台主机都会在自己的ARP缓冲区 (ARP Cache)中建立一个 ARP列表，以表示IP地址和MAC地址的对应关系。

　　2. 当源主机需要将一个数据包要发送到目的主机时，会首先检查自己 ARP列表中是否存在该 IP地址对应的MAC地址，如果有﹐就直接将数据包发送到这个MAC地址；如果没有，就向本地网段发起一个ARP请求的广播包，查询此目的主机对应的MAC地址。此ARP请求数据包里包括源主机的IP地址、硬件地址、以及目的主机的IP地址。

　　3. 网络中所有的主机收到这个ARP请求后，会检查数据包中的目的IP是否和自己的IP地址一致。如果不相同就忽略此数据包；如果相同，该主机首先将发送端的MAC地址和IP地址添加到自己的ARP列表中，如果ARP表如果ARP表中已经存在该IP的信息，则将其覆盖，然后给源主机发送一个 ARP响应数据包，告诉对方自己是它需要查找的MAC地址；

4. 源主机收到这个ARP响应数据包后，将得到的目的主机的IP地址和MAC地址添加到自己的ARP列表中，并利用此信息开始数据的传输。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

**4、P和UDP的区别：**

TCP协议是有连接的，有连接的意思是开始传输实际数据之前TCP的客户端和服务器端必须通过三次握手建立连接，会话结束之后也要结束连接。而UDP是无连接的

TCP协议保证数据按序发送，按序到达，提供超时重传来保证可靠性，但是UDP不保证按序到达，甚至不保证到达，只是努力交付，即便是按序发送的序列，也不保证按序送到。

TCP协议所需资源多，TCP首部需20个字节（不算可选项），UDP首部字段只需8个字节。

TCP有流量控制和拥塞控制，UDP没有，网络拥堵不会影响发送端的发送速率

TCP是一对一的连接，而UDP则可以支持一对一，多对多，一对多的通信。

TCP面向的是字节流的服务，UDP面向的是报文的服务。