**Java基础题**

**1. 什么是Java虚拟机？为什么Java被称作是“平台无关的编程语言”？**

* java虚拟机是执行字节码文件（.class）的虚拟机进程。。java源程序（.java）被编译器编译成字节码文件（.class）。然后字节码文件，将由java虚拟机，解释成机器码（不同平台的机器码不同）。利用机器码操作硬件和操作系统。
* 因为不同的平台装有不同的JVM，它们能够将相同的.class文件，解释成不同平台所需要的机器码。正是因为有JVM的存在，java被称为平台无关的编程语言。

**2.JDK和JRE的区别是什么？**

* Java开发工具包(JDK)是完整的Java软件开发包，包含了JRE，编译器和其他的工具(比如：JavaDoc，Java调试器)，可以让开发者开发、编译、执行Java应用程序。
* Java运行时环境(JRE)。它包括Java虚拟机、Java核心类库和支持文件。它不包含开发工具（JDK）、编译器、调试器和其他工具。

**3.”static”关键字是什么意思？Java中是否可以覆盖(override)一个private或者是static的方法？**

* “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的类的实例变量的情况下被访问。   
  Java中static方法不能被覆盖，因为方法覆盖是基于运行时动态绑定的，而static方法是编译时静态绑定的。static方法跟类的任何实例都不相关，所以概念上不适用。
* java中也不可以覆盖private的方法，因为private修饰的变量和方法只能在当前类中使用，如果是其他的类继承当前类是不能访问到private变量或方法的，当然也不能覆盖。

**4.是否可以在static环境中访问非static变量？**

static变量在Java中是属于类的，它在所有的实例中的值是一样的。当类被Java虚拟机载入的时候，会对static变量进行初始化。如果你的代码尝试不用实例来访问非static的变量，编译器会报错，因为这些变量还没有被创建出来，还没有跟任何实例关联上。

**5.Java支持的数据类型有哪些？什么是自动拆装箱？**

* Java支持的数据类型包括两种：一种是基本数据类型，包含byte，char, short, boolean ,int, long, float, double;另一种是引用类型：如String等，其实是对象的引用，JVM中虚拟栈中存的是对象的地址，创建的对象实质在堆中，通过地址来找到堆中的对象的过程，即为引用类型。
* 自动装箱就是Java编译器在基本数据类型和对应的对象包装类型间的转化，即int转化为Integer,自动拆箱是Integer调用其方法将其转化为int的过程。

**6.Java中的方法覆盖(Overriding)和方法重载(Overloading)是什么意思？**

* Java中的方法**重载**发生在同一个类里面两个或者是多个方法的**方法名相同但是参数不同**的情况。重载Override是一个类中多态性的一种表现。
* 方法**覆盖**是说子类重新定义了父类的方法。方法覆盖必须有**相同的方法名，参数列表和返回类型**。覆盖者可能不会限制它所覆盖的方法的访问。在java中，子类可继承父类的方法，则不需要重新编写相同的方法。但有时子类并不想原封不动继承父类的方法，而是想做一定的修改，这就采用方法重写。方法重写又称方法覆盖。

**7.Java中，什么是构造方法？什么是构造方法重载？什么是复制构造方法？**

* 当新对象被创建的时候，构造方法会被调用。每一个类都有构造方法。在程序员没有给类提供构造方法的情况下，Java编译器会为这个类创建一个默认的构造方法。
* Java中构造方法重载和方法重载很相似。可以为一个类创建多个构造方法。每一个构造方法必须有它自己唯一的参数列表。
* Java不支持复制构造方法，如果你不自己写构造方法的情况下，Java不会创建默认的复制构造方法。

**8. Java支持多继承么？**

Java中类不支持多继承，只支持单继承（即一个类只有一个父类）。但是java中的接口支持多继承，，即一个子接口可以有多个父接口。（接口的作用是用来扩展对象的功能，一个子接口继承多个父接口，说明子接口扩展了多个功能，当类实现接口时，类就扩展了相应的功能）。

**9. 接口和抽象类的区别是什么？**

**从设计层面来说，抽象是对类的抽象，是一种模板设计，接口是行为的抽象，是一种行为的规范。**

Java提供和支持创建抽象类和接口。它们的实现有共同点，不同点在于：

* 接口中所有的方法隐含的都是抽象的。而抽象类则可以同时包含抽象和非抽象的方法;
* 类可以实现很多个接口，但是只能继承一个抽象类;
* 类可以不实现抽象类和接口声明的所有方法，当然，在这种情况下，类也必须得声明成是抽象的;
* 抽象类可以在不提供接口方法实现的情况下实现接口;
* Java接口中声明的变量默认都是final的。抽象类可以包含非final的变量;
* Java接口中的成员函数默认是public的。抽象类的成员函数可以是private，protected或者是public;
* 接口是绝对抽象的，不可以被实例化。抽象类也不可以被实例化，但是，如果它包含main方法的话是可以被调用的。

也可以参考JDK8中抽象类和接口的区别。

**10. 什么是值传递和引用传递？**

**一般认为,java内的传递都是值传递. java中实例对象的传递是引用传递。**

* 值传递是对基本型变量而言的,传递的是该**变量**的一个副本,改变副本不影响原变量；
* 引用传递一般是对于对象型变量而言的,传递的是该对象**地址**的一个副本, 并不是原对象本身。

**11. 进程与线程的区别？**

* 进程是执行着的应用程序，是程序的一种动态形式，是CPU、内存等资源占用的基本单位，而且进程之间相互独立，通信比较困难，进程在执行过程中，包含比较固定的入口，执行顺序，出口；
* 线程是进程内部的一个执行序列，隶属于某个进程，一个进程可以有多个线程，线程不能占有CPU、内存等资源，而且线程之间共享一块内存区域，通信比较方便，线程的入口执行顺序这些过程被应用程序所控制。

**12. 创建线程有几种不同的方式？你喜欢哪一种？为什么？**

有四种方式可以用来创建线程。

* 继承Thread类；
* 实现Runnable接口；
* 应用程序可以使用Executor框架来创建线程池；
* 实现Callable接口。

实现Runnable接口这种方式更受欢迎，因为这不需要继承Thread类。在应用设计中已经继承了别的对象的情况下，这需要多继承（而**Java不支持多继承**），只能实现接口。同时，线程池也是非常高效的，很容易实现和使用。

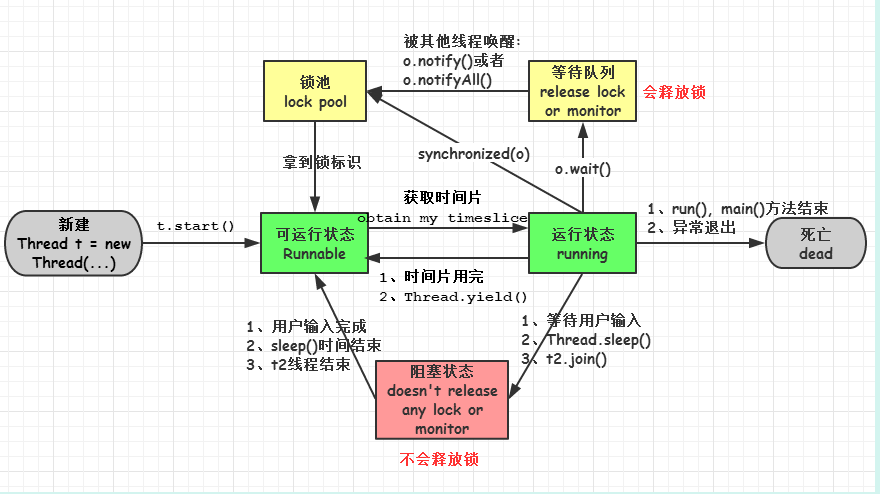
**13. 概括的解释下线程的几种可用状态。**

* **新建( new )**：新创建了一个线程对象。
* **可运行( runnable )**：线程对象创建后，其他线程(比如 main 线程）调用了该对象的start ()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获取cpu的使用权。
* **运行( running )**：可运行状态( runnable )的线程获得了cpu时间片（ timeslice ），执行程序代码。
* **阻塞( block )**：阻塞状态是指线程因为某种原因放弃了cpu使用权，也即让出了cpu timeslice ，暂时停止运行。直到线程进入可运行( runnable )状态，才有机会再次获得 cpu timeslice 转到运行( running )状态。阻塞的情况分三种：

**(一). 等待阻塞**：运行( running )的线程执行 o.wait ()方法，JVM会把该线程放入等待队列( waitting queue )中。

**(二). 同步阻塞**：运行( running )的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则 JVM 会把该线程放入锁池( lock pool )中。

**(三). 其他阻塞**: 运行( running )的线程执行Thread.sleep ( long ms )或 t.join ()方法，或者发出了 I/O 请求时，JVM会把该线程置为阻塞状态。当 sleep ()状态超时、join()等待线程终止或者超时、或者I/O 处理完毕时，线程重新转入可运行( runnable )状态。

* **死亡( dead )**：线程run()、 main() 方法执行结束，或者因异常退出了run()方法，则该线程结束生命周期。死亡的线程不可再次复生。   
  

**14. 同步方法和同步代码块的区别是什么？**

**为何使用同步？**   
java允许多线程并发控制，当多个线程同时操作一个可共享的资源变量时（增删改查），将会导致数据的不准确，相互之间产生冲突，因此加入同步锁以避免在该线程没有完成操作之前，被其他线程的调用，从而保证了该变量的唯一性和准确性。

**区别**

* 同步方法默认用this或者当前类class对象作为锁；
* 同步代码块可以选择以什么来加锁，比同步方法要更细颗粒度，我们可以选择只同步会发生同步问题的部分代码而不是整个方法；
* 同步方法使用关键字 synchronized修饰方法，而同步代码块主要是修饰需要进行同步的代码，用synchronized（object）{代码内容}进行修饰；

**15. 在监视器(Monitor)内部，是如何做线程同步的？程序应该做哪种级别的同步？**

监视器和锁在Java虚拟机中是一块使用的。监视器监视一块同步代码块，确保一次==只有一个线程执行同步代码块==。每一个监视器都和一个对象引用相关联。线程在获取锁之前不允许执行同步代码。

**16.什么是死锁(deadlock)？**

所谓死锁是指多个进 程因==竞争资源==而造成的一种僵局（互相等待），若无外力作用，这些进程都将无法向前推进。死锁产生的4个必要条件：

* **互斥条件**：进程要求对所分配的资源（如打印机）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有。此时若有其他进程请求该资源，则请求进程只能等待。
* **不剥夺条件**：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（只能是主动释放)。
* **请求和保持条件**：进程已经保持了至少一个资源，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放。
* **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所请求。

**17. 如何确保N个线程可以访问N个资源同时又不导致死锁？**

使用多线程的时候，一种非常简单的避免死锁的方式就是：==指定获取锁的顺序==，并强制线程按照指定的顺序获取锁。因此，如果所有的线程都是以同样的顺序加锁和释放锁，就不会出现死锁了。

**18.Java集合类框架的基本接口有哪些？**

集合类接口指定了一组叫做元素的对象。集合类接口的每一种具体的实现类都可以选择以它自己的方式对元素进行保存和排序。有的集合类允许重复的键，有些不允许。   
Java集合类提供了一套设计良好的支持对一组对象进行操作的接口和类。Java集合类里面最基本的接口有：

* **Collection**：代表一组对象，每一个对象都是它的子元素。
* **Set**：不包含重复元素的Collection。
* **List**：有顺序的collection，并且可以包含重复元素。
* **Map**：可以把键(key)映射到值(value)的对象，键不能重复。

**19.为什么集合类没有实现Cloneable和Serializable接口？**

克隆(cloning)或者是序列化(serialization)的语义和含义是跟具体的实现相关的。因此，应该由集合类的具体实现来决定如何被克隆或者是序列化。

**20.什么是迭代器(Iterator)？**

Iterator接口提供了很多对集合元素进行迭代的方法。每一个集合类都包含了可以返回迭代器实例的   
迭代方法。迭代器可以在迭代的过程中删除底层集合的元素,但是不可以直接调用集合的   
remove(Object Obj)删除，可以通过迭代器的remove()方法删除。

**21.Iterator和ListIterator的区别是什么？**

下面列出了他们的区别：

* 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既可以前向也可以后向。
* ListIterator实现了Iterator接口，并包含其他的功能，比如：增加元素，替换元素，获取前一个和后一个元素的索引，等等。

**22.快速失败(fail-fast)和安全失败(fail-safe)的区别是什么？**

Iterator的安全失败是基于对底层集合做拷贝，因此，它**不受源集合上修改**的影响。**java.util包**下面的所有的集合类都是**快速失败**的，而**java.util.concurrent包**下面的所有的类都是**安全失败**的。快速失败的迭代器会抛出ConcurrentModificationException异常，而安全失败的迭代器永远不会抛出这样的异常。

**33.Java中的HashMap的工作原理是什么？**

Java中的HashMap是以键值对(key-value)的形式存储元素的。HashMap需要一个hash函数，它使用hashCode()和equals()方法来向集合/从集合添加和检索元素。当调用put()方法的时候，HashMap会计算key的hash值，然后把键值对存储在集合中合适的索引上。如果key已经存在了，value会被更新成新值。HashMap的一些重要的特性是它的**容量(capacity)，负载因子(load factor)和扩容极限(threshold resizing)**。

**34.hashCode()和equals()方法的重要性体现在什么地方？**

Java中的HashMap使用hashCode()和equals()方法来确定键值对的索引，当根据键获取值的时候也会用到这两个方法。如果没有正确的实现这两个方法，两个不同的键可能会有相同的hash值，因此，可能会被集合认为是相等的。而且，这两个方法也用来发现重复元素。所以这两个方法的实现对HashMap的精确性和正确性是至关重要的。

**35.HashMap和Hashtable有什么区别？**

HashMap和Hashtable都实现了Map接口，因此很多特性非常相似。但是，他们有以下不同点：

* HashMap允许**键和值是null**，而Hashtable不允许键或者值是null。
* Hashtable是同步的，而HashMap不是。因此，HashMap更适合于单线程环境，而Hashtable适合于多线程环境。
* HashMap提供了可供应用迭代的键的集合，因此，HashMap是快速失败的。另一方面，Hashtable提供了对键的列举(Enumeration)。
* 一般认为Hashtable是一个遗留的类。

**36. 数组(Array)和列表(ArrayList)有什么区别？什么时候应该使用Array而不是ArrayList？**

* Array可以包含基本类型和对象类型，ArrayList只能包含对象类型。
* Array大小是固定的，ArrayList的大小是动态变化的。
* ArrayList提供了更多的方法和特性，比如：addAll()，removeAll()，iterator()等等。
* 对于基本类型数据，集合使用自动装箱来减少编码工作量。但是，当处理固定大小的基本数据类型的时候，这种方式相对比较慢。

**37. ArrayList和LinkedList有什么区别？**

ArrayList和LinkedList都实现了List接口，他们有以下的不同点：

* ArrayList是**基于索引**的数据接口，它的底层是数组。它可以以O(1)时间复杂度对元素进行随机访问。与此对应，LinkedList是以==元素列表==的形式存储它的数据，每一个元素都和它的前一个和后一个元素链接在一起，在这种情况下，查找某个元素的时间复杂度是O(n)。   
  相对于ArrayList，LinkedList的插入，添加，删除操作速度更快，因为当元素被添加到集合任意位置的时候，不需要像数组那样重新计算大小或者是更新索引。
* **LinkedList比ArrayList更占内存**，因为LinkedList为每一个节点存储了两个引用，一个指向前一个元素，一个指向下一个元素。

**38.Comparable和Comparator接口是干什么的？列出它们的区别。**

Java提供了只包含一个compareTo()方法的Comparable接口。这个方法可以个给两个对象排序。具体来说，它返回负数，0，正数来表明已经存在的对象小于，等于，大于输入对象。   
Java提供了包含compare()和equals()两个方法的Comparator接口。compare()方法用来给两个输入参数排序，返回负数，0，正数表明第一个参数是小于，等于，大于第二个参数。equals()方法需要一个对象作为参数，它用来决定输入参数是否和comparator相等。只有当输入参数也是一个comparator并且输入参数和当前comparator的排序结果是相同的时候，这个方法才返回true。

接：<https://www.nowcoder.com/questionTerminal/99f7d1f4f8374e419a6d6924d35d9530>   
来源：牛客网   
Comparable & Comparator 都是用来实现集合中元素的比较、排序的，只是 **Comparable是在集合内部定义的方法实现的排序**，Comparator 是在集合外部实现的排序，所以，如想实现排序，就需要在集合外定义 Comparator 接口的方法或在集合内实现 Comparable 接口的方法。 Comparator位于包java.util下，而Comparable位于包 java.lang下 Comparable 是一个对象本身就已经支持自比较所需要实现的接口（如 String、Integer 自己就可以完成比较大小操作，已经实现了Comparable接口） 自定义的类要在加入list容器中后能够排序，可以实现Comparable接口，在用Collections类的sort方法排序时，如果不指定Comparator，那么就以自然顺序排序， 这里的自然顺序就是实现Comparable接口设定的排序方式。 而 Comparator 是一个专用的比较器，当这个对象不支持自比较或者自比较函数不能满足你的要求时，你可以写一个比较器来完成两个对象之间大小的比较。 可以说一个是自已完成比较，一个是外部程序实现比较的差别而已。 用 Comparator 是策略模式（strategy design pattern），就是不改变对象自身，而用一个策略对象（strategy object）来改变它的行为。 比如：你想对整数采用绝对值大小来排序，Integer 是不符合要求的，你不需要去修改 Integer 类（实际上你也不能这么做）去改变它的排序行为，只要使用一个实现了 Comparator 接口的对象来实现控制它的排序就行了。

**39.什么是Java优先级队列(Priority Queue)？**

PriorityQueue是一个基于优先级堆的无界队列，它的元素是按照自然顺序(natural order)排序的。在创建的时候，我们可以给它提供一个负责给元素排序的比较器。PriorityQueue不允许null值，因为他们没有自然顺序，或者说他们没有任何的相关联的比较器。最后，PriorityQueue不是线程安全的，入队和出队的时间复杂度是O(log(n))。

**40.你了解大O符号(big-O notation)么？你能给出不同数据结构的例子么？**

大O符号描述了当数据结构里面的元素增加的时候，算法的规模或者是一个渐进上界 。   
大O符号也可用来描述其他的行为，比如：内存消耗。因为集合类实际上是数据结构，我们一般使用大O符号基于时间，内存和性能来选择最好的实现。大O符号可以对大量数据的性能给出一个很好的说明。

**41.Enumeration接口和Iterator接口的区别有哪些？**

Enumeration速度是Iterator的2倍，同时占用更少的内存。但是，Iterator远远比Enumeration安全，因为其他线程不能够修改正在被iterator遍历的集合里面的对象。同时，Iterator允许调用者删除底层集合里面的元素，这对Enumeration来说是不可能的。

**42.HashSet和TreeSet有什么区别？**

* HashSet是由一个hash表来实现的，因此，它的元素是无序的。add()，remove()，contains()方法的时间复杂度是O(1)。
* 另一方面，TreeSet是由一个树形的结构来实现的，它里面的元素是有序的。因此，add()，remove()，contains()方法的时间复杂度是O(logn)。

**43.Java中垃圾回收有什么目的？什么时候进行垃圾回收？**

垃圾回收是在内存中存在没有引用的对象或超过作用域的对象时进行。   
垃圾回收的目的是识别并且丢弃应用不再使用的对象来释放和重用资源。

**44.System.gc()和Runtime.gc()会做什么事情？**

这两个方法用来提示JVM要进行垃圾回收。但是，立即开始还是延迟进行垃圾回收是取决于JVM的。

**45.finalize()方法什么时候被调用？析构函数(finalization)的目的是什么？**

垃圾回收器(garbage colector)决定回收某对象时，就会运行该对象的finalize()方法 但是在Java中很不幸，如果内存总是充足的，那么垃圾回收可能永远不会进行，也就是说filalize()可能永远不被执行，显然指望它做收尾工作是靠不住的。 那么finalize()究竟是做什么的呢？它最主要的用途是回收特殊渠道申请的内存。Java程序有垃圾回收器，所以一般情况下内存问题不用程序员操心。但有一种JNI(Java Native Interface)调用non-Java程序（C或C++），finalize()的工作就是回收这部分的内存。

**调用时机：**当垃圾回收器要宣告一个对象死亡时，至少要经过两次标记过程：如果对象在进行可达性分析后发现没有和GC Roots相连接的引用链，就会被第一次标记，并且判断是否执行finalizer( )方法，如果对象覆盖finalizer( )方法且未被虚拟机调用过，那么这个对象会被放置在F-Queue队列中，并在稍后由一个虚拟机自动建立的低优先级的Finalizer线程区执行触发finalizer( )方法，但不承诺等待其运行结束。   
finalization的目的：对象逃脱死亡的最后一次机会。（只要重新与引用链上的任何一个对象建立关联即可。）但是不建议使用，运行代价高昂，不确定性大，且无法保证各个对象的调用顺序。可用try-finally或其他替代。

**46.如果对象的引用被置为null，垃圾收集器是否会立即释放对象占用的内存？**

不会，在下一个垃圾回收周期中，这个对象将是可被回收的。

不会立即释放对象占用的内存。 如果对象的引用被置为null，只是断开了当前线程栈帧中对该对象的引用关系，而 垃圾收集器是运行在后台的线程，只有当用户线程运行到安全点(safe point)或者安全区域才会扫描对象引用关系，扫描到对象没有被引用则会标记对象，这时候仍然不会立即释放该对象内存，因为有些对象是可恢复的（在 finalize方法中恢复引用 ）。只有确定了对象无法恢复引用的时候才会清除对象内存。

**47.Java堆的结构是什么样子的？什么是堆中的永久代(Perm Gen space)?**

JVM的堆是运行时数据区，所有类的实例和数组都是在堆上分配内存。它在JVM启动的时候被创建。对象所占的堆内存是由自动内存管理系统也就是垃圾收集器回收。   
堆内存是由存活和死亡的对象组成的。存活的对象是应用可以访问的，不会被垃圾回收。死亡的对象是应用不可访问尚且还没有被垃圾收集器回收掉的对象。一直到垃圾收集器把这些对象回收掉之前，他们会一直占据堆内存空间。

永久代是用于存放静态文件，如Java类、方法等。持久代对垃圾回收没有显著影响，但是有些应用可能动态生成或者调用一些class，例如Hibernate 等，在这种时候需要设置一个比较大的持久代空间来存放这些运行过程中新增的类，永久代中一般包含：   
类的方法(字节码…)   
类名(Sring对象)   
.class文件读到的常量信息   
class对象相关的对象列表和类型列表 (e.g., 方法对象的array).   
JVM创建的内部对象   
JIT编译器优化用的信息

**48.串行(serial)收集器和吞吐量(throughput)收集器的区别是什么？**

吞吐量收集器使用并行版本的新生代垃圾收集器，它用于中等规模和大规模数据的应用程序。而串行收集器对大多数的小应用(在现代处理器上需要大概100M左右的内存)就足够了。

**49.在Java中，对象什么时候可以被垃圾回收？**

当一个对象到GC Roots不可达时，在下一个垃圾回收周期中尝试回收该对象，如果该对象重写了finalize()方法，并在这个方法中成功自救(将自身赋予某个引用)，那么这个对象不会被回收。但如果这个对象没有重写finalize()方法或者已经执行过这个方法，也自救失败，该对象将会被回收。

**50.JVM的永久代中会发生垃圾回收么？**

垃圾回收不会发生在永久代，如果永久代满了或者是超过了临界值，会触发完全垃圾回收(Full GC)。如果你仔细查看垃圾收集器的输出信息，就会发现永久代也是被回收的。这就是为什么正确的永久代大小对避免Full GC是非常重要的原因。请参考下Java8：从永久代到元数据区   
(注：Java8中已经移除了永久代，新加了一个叫做元数据区的native内存区)

**51.Java中的两种异常类型是什么？他们有什么区别？**

Java中有两种异常：受检查的(checked)异常和不受检查的(unchecked)异常。不受检查的异常不需要在方法或者是构造函数上声明，就算方法或者是构造函数的执行可能会抛出这样的异常，并且不受检查的异常可以传播到方法或者是构造函数的外面。相反，受检查的异常必须要用throws语句在方法或者是构造函数上声明。这里有Java异常处理的一些小建议。

接：<https://www.nowcoder.com/questionTerminal/3ded1983c85c4ae197e005bd31777bc7>   
来源：牛客网

Throwable包含了错误(Error)和异常(Excetion两类)   
Exception又包含了运行时异常(RuntimeException, 又叫非检查异常)和非运行时异常(又叫检查异常)   
(1) Error是程序无法处理了, 如果OutOfMemoryError等, 这些异常发生时, java虚拟机一般会终止线程 .   
(2) 运行时异常都是RuntimeException类及其子类,如 NullPointerException、IndexOutOfBoundsException等, 这些异常是不检查的异常, 是在程序运行的时候可能会发生的, 所以程序可以捕捉, 也可以不捕捉. 这些错误一般是由程序的逻辑错误引起的, 程序应该从逻辑角度去尽量避免.   
(3) 检查异常是运行时异常以外的异常, 也是Exception及其子类, 这些异常从程序的角度来说是必须经过捕捉检查处理的, 否则不能通过编译. 如IOException、SQLException等。

**52.Java中Exception和Error有什么区别？**

Exception和Error都是Throwable的子类。Exception用于用户程序可以捕获的异常情况。Error定义了不期望被用户程序捕获的异常。

**53.throw和throws有什么区别？**

throw关键字用来在程序中明确的抛出异常，相反，throws语句用来表明方法不能处理的异常。每一个方法都必须要指定哪些异常不能处理，所以方法的调用者才能够确保处理可能发生的异常，多个异常是用逗号分隔的。

1、Throw用于方法内部，Throws用于方法声明上；   
2、Throw后跟异常对象，Throws后跟异常类型；   
3、Throw后只能跟一个异常对象，Throws后可以一次声明多种异常类型。

**54. 异常处理完成以后，Exception对象会发生什么变化？、**

Exception对象会在下一个垃圾回收过程中被回收掉。

**55. finally代码块和finalize()方法有什么区别？**

无论是否抛出异常，finally代码块都会执行，它主要是用来释放应用占用的资源。finalize()方法是Object类的一个protected方法，它是在对象被垃圾回收之前由Java虚拟机来调用的。

**56. 什么是Applet？**

java applet是能够被包含在HTML页面中并且能被启用了java的客户端浏览器执行的程序。Applet主要用来创建动态交互的web应用程序。

**57. 解释一下Applet的生命周期。**

applet可以经历下面的状态：   
Init：每次被载入的时候都会被初始化。   
Start：开始执行applet。   
Stop：结束执行applet。   
Destroy：卸载applet之前，做最后的清理工作。

**58. 当applet被载入的时候会发生什么？**

首先，创建applet控制类的实例，然后初始化applet，最后开始运行。

**59. Applet和普通的Java应用程序有什么区别？**

applet是运行在启用了java的浏览器中，Java应用程序是可以在浏览器之外运行的独立的Java程序。但是，它们都需要有Java虚拟机。   
进一步来说，Java应用程序需要一个有特定方法签名的main函数来开始执行。Java applet不需要这样的函数来开始执行。   
最后，Java applet一般会使用很严格的安全策略，Java应用一般使用比较宽松的安全策略。

**60. Java applet有哪些限制条件？**

主要是由于安全的原因，给applet施加了以下的限制：   
applet不能够载入类库或者定义本地方法；不能在宿主机上读写文件；不能读取特定的系统属性；不能发起网络连接，除非是跟宿主机；不能够开启宿主机上其他任何的程序。

**61. 什么是不受信任的applet？**

不受信任的applet是不能访问或是执行本地系统文件的Java applet，默认情况下，所有下载的applet都是不受信任的。

**62. 从网络上加载的applet和从本地文件系统加载的applet有什么区别？**

当applet是从网络上加载的时候，applet是由applet类加载器载入的，它受applet安全管理器的限制。   
当applet是从客户端的本地磁盘载入的时候，applet是由文件系统加载器载入的。   
从文件系统载入的applet允许在客户端读文件，写文件，加载类库，并且也允许执行其他程序，但是，却通不过字节码校验。

**63. applet类加载器是什么？它会做哪些工作？**

当applet是从网络上加载的时候，它是由applet类加载器载入的。类加载器有自己的java名称空间等级结构。类加载器会保证来自文件系统的类有唯一的名称空间，来自网络资源的类有唯一的名称空间。   
当浏览器通过网络载入applet的时候，applet的类被放置于和applet的源相关联的私有的名称空间中。然后，那些被类加载器载入进来的类都是通过了验证器验证的。验证器会检查类文件格式是否遵守Java语言规范，确保不会出现堆栈溢出(stack overflow)或者下溢(underflow)，传递给字节码指令的参数是正确的。

**64. applet安全管理器是什么？它会做哪些工作？**

applet安全管理器是给applet施加限制条件的一种机制。浏览器可以只有一个安全管理器。安全管理器在启动的时候被创建，之后不能被替换覆盖或者是扩展。

**65. 弹出式选择菜单(Choice)和列表(List)有什么区别？**

Choice是以一种紧凑的形式展示的，需要下拉才能看到所有的选项。Choice中一次只能选中一个选项。List同时可以有多个元素可见，支持选中一个或者多个元素。

**66. 什么是布局管理器？**

布局管理器用来在容器中组织组件。

**67. 滚动条(Scrollbar)和滚动面板(JScrollPane)有什么区别？**

Scrollbar是一个组件，不是容器。而ScrollPane是容器。ScrollPane自己处理滚动事件。

**68. 哪些Swing的方法是线程安全的？**

只有3个线程安全的方法： repaint(), revalidate(), and invalidate()。

**69. 说出三种支持重绘(painting)的组件。**

Canvas, Frame, Panel,和Applet支持重绘。

**70. 什么是JDBC？**

JDBC是允许用户在不同数据库之间做选择的一个抽象层。JDBC允许开发者用JAVA写数据库应用程序，而不需要关心底层特定数据库的细节。

JDBC（Java DataBase Connectivity）,是一套面向对象的应用程序接口（API），制定了统一的访问各类关系数据库的标准接口，为各个数据库厂商提供了标准的实现。通过JDBC技术，开发人员可以用纯Java语言和标准的SQL语句编写完整的数据库应用程序，并且真正地实现了软件的跨平台性。   
通常情况下使用JDBC完成以下操作：   
1.同数据库建立连接；   
2.向数据库发送SQL语句；   
3.处理从数据库返回的结果；   
JDBC具有下列优点：   
1.JDBC与ODBC(Open Database Connectivity，即开放数据库互连）十分相似，便于软件开发人员理解；   
2.JDBC使软件开发人员从复杂的驱动程序编写工作中解脱出来，可以完全专注于业务逻辑开发；   
3.JDBC支持多种关系型数据库，大大增加了软件的可移植性；   
4.JDBC API是面向对象的，软件开发人员可以将常用的方法进行二次封装，从而提高代码的重用性；

**71. 数据库连接池是什么意思？**

像打开关闭数据库连接这种和数据库的交互可能是很费时的，尤其是当客户端数量增加的时候，会消耗大量的资源，成本是非常高的。可以在应用服务器启动的时候建立很多个数据库连接并维护在一个池中。连接请求由池中的连接提供。在连接使用完毕以后，把连接归还到池中，以用于满足将来更多的请求。

**72. 让RMI程序能正确运行有哪些步骤？**

为了让RMI程序能正确运行必须要包含以下几个步骤：   
编译所有的源文件。   
使用rmic生成stub。   
启动rmiregistry。   
启动RMI服务器。   
运行客户端程序。

**73. 解释下Marshalling和demarshalling。**

当应用程序希望把内存对象跨网络传递到另一台主机或者是持久化到存储的时候，就必须要把对象在内存里面的表示转化成合适的格式。这个过程就叫做Marshalling，反之就是demarshalling。

**74. 解释下Serialization和Deserialization。**

Java提供了一种叫做对象序列化的机制，他把对象表示成一连串的字节，里面包含了对象的数据，对象的类型信息，对象内部的数据的类型信息等等。因此，序列化可以看成是为了把对象存储在磁盘上或者是从磁盘上读出来并重建对象而把对象扁平化的一种方式。反序列化是把对象从扁平状态转化成活动对象的相反的步骤。

**75. 什么是Servlet？**

Servlet是用来处理客户端请求并产生动态网页内容的Java类。Servlet主要是用来处理或者是存储HTML表单提交的数据，产生动态内容，在无状态的HTTP协议下管理状态信息。

**76. 说一下Servlet的体系结构。**

所有的Servlet都必须要实现的核心的接口是javax.servlet.Servlet。每一个Servlet都必须要直接或者是间接实现这个接口，或者是继承javax.servlet.GenericServlet或者javax.servlet.http.HTTPServlet。最后，Servlet使用多线程可以并行的为多个请求服务。

**77. 什么是Web应用程序？**

Web应用程序是对Web或者是应用服务器的动态扩展。有两种类型的Web应用：面向表现的和面向服务的。面向表现的Web应用程序会产生包含了很多种标记语言和动态内容的交互的web页面作为对请求的响应。而面向服务的Web应用实现了Web服务的端点(endpoint)。一般来说，一个Web应用可以看成是一组安装在服务器URL名称空间的特定子集下面的Servlet的集合。

**78. 如何知道是哪一个客户端的机器正在请求你的Servlet？**

ServletRequest类可以找出客户端机器的IP地址或者是主机名。getRemoteAddr()方法获取客户端主机的IP地址，getRemoteHost()可以获取主机名。

**79. HTTP响应的结构是怎么样的？**

HTTP响应由三个部分组成：   
状态码(Status Code)：描述了响应的状态。可以用来检查是否成功的完成了请求。请求失败的情况下，状态码可用来找出失败的原因。如果Servlet没有返回状态码，默认会返回成功的状态码HttpServletResponse.SC\_OK。   
HTTP头部(HTTP Header)：它们包含了更多关于响应的信息。比如：头部可以指定认为响应过期的过期日期，或者是指定用来给用户安全的传输实体内容的编码格式。如何在Serlet中检索HTTP的头部看这里。   
主体(Body)：它包含了响应的内容。它可以包含HTML代码，图片，等等。主体是由传输在HTTP消息中紧跟在头部后面的数据字节组成的。

**80. 什么是cookie？session和cookie有什么区别？**

cookie是Web服务器发送给浏览器的一块信息。浏览器会在本地文件中给每一个Web服务器存储cookie。以后浏览器在给特定的Web服务器发请求的时候，同时会发送所有为该服务器存储的cookie。下面列出了session和cookie的区别：   
无论客户端浏览器做怎么样的设置，session都应该能正常工作。客户端可以选择禁用cookie，但是，session仍然是能够工作的，因为客户端无法禁用服务端的session。   
在存储的数据量方面session和cookies也是不一样的。session能够存储任意的Java对象，cookie只能存储String类型的对象。

**81. JSP请求是如何被处理的？**

浏览器首先要请求一个以.jsp扩展名结尾的页面，发起JSP请求，然后，Web服务器读取这个请求，使用JSP编译器把JSP页面转化成一个Servlet类。需要注意的是，只有当第一次请求页面或者是JSP文件发生改变的时候JSP文件才会被编译，然后服务器调用servlet类，处理浏览器的请求。一旦请求执行结束，servlet会把响应发送给客户端。

**82. 封装的定义和好处有哪些？**

封装给对象提供了隐藏内部特性和行为的能力。对象提供一些能被其他对象访问的方法来改变它内部的数据。在Java当中，有3种修饰符：public，private和protected。每一种修饰符给其他的位于同一个包或者不同包下面对象赋予了不同的访问权限。   
下面列出了使用封装的一些好处：   
通过隐藏对象的属性来保护对象内部的状态。   
提高了代码的可用性和可维护性，因为对象的行为可以被单独的改变或者是扩展。   
禁止对象之间的不良交互提高模块化。   
参考这个文档获取更多关于封装的细节和示例。

或：

一是用private把类的细节与外界隔离起来，从而实现数据项和方法的隐藏，而要访问这些数据项和方法唯一的途径就是通过类本身，类才有资格调用它所拥有的资源（方法，数据项属性等等）。所以第一个好处就是数据的安全性提高了。   
二是通过隐藏隔离，只允许外部对类做有限的访问，开发者可以自由的改变类的内部实现，而无需修改使用该类的那些程序。只要那些在类外部就能被调用的方法保持其外部特征不变，内部代码就可以自由改变，各取所需，利于分工。   
三就是提高了代码的重用性，封装成工具类以后能够减少很多繁琐的步骤。

**83.抽象的定义？抽象和封装的不同点？**

* 抽象是把想法从具体的实例中分离出来的步骤，因此，要根据他们的功能而不是实现细节来创建类。Java支持创建只暴漏接口而不包含方法实现的抽象的类。这种抽象技术的主要目的是把类的行为和实现细节分离开。
* 抽象和封装是互补的概念。一方面，抽象关注对象的行为。另一方面，封装关注对象行为的细节。一般是通过隐藏对象内部状态信息做到封装，因此，封装可以看成是用来提供抽象的一种策略。