**常见的Java问题**

**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虚拟机是一个可以执行Java字节码的虚拟机进程。Java源文件被编译成能被Java虚拟机执行的字节码文件。

　　Java被设计成允许应用程序可以运行在任意的平台，而不需要程序员为每一个平台单独重写或者是重新编译。Java虚拟机让这个变为可能，因为它知道底层硬件平台的指令长度和其他特性。

**2．**

**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。

**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语言支持的8中基本数据类型是：

　　byte

　　short

　　int

　　long

　　float

　　double

　　boolean

　　char

　　自动装箱是Java编译器在基本数据类型和对应的对象包装类型之间做的一个转化。比如：把int转化成Integer，double转化成double，等等。反之就是自动拆箱。

**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。与此相对，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。(不确定)

方法重载跟返回值类型和修饰符无关

**7.Java中，什么是构造函数？什么是构造函数重载？什么是复制构造函数？**

　　当新对象被创建的时候，构造函数会被调用。每一个类都有构造函数。在程序员没有给类提供构造函数的情况下，Java编译器会为这个类创建一个默认的构造函数。

　　Java中构造函数重载和方法重载很相似。可以为一个类创建多个构造函数。每一个构造函数必须有它自己唯一的参数列表。

　　Java不支持像C++中那样的复制构造函数，这个不同点是因为如果你不自己写构造函数的情况下，Java不会创建默认的复制构造函数。

**8.Java支持多继承么？**

　　不支持，Java不支持多继承。每个类都只能继承一个类，但是可以实现多个接口。

**9.接口和抽象类的区别是什么？**

　　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

　　类如果要实现一个接口，它必须要实现接口声明的所有方法。但是，类可以不实现抽象类声明的所有方法，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？**

　　对象被值传递，意味着传递了对象的一个副本。因此，就算是改变了对象副本，也不会影响源对象的值。

　　对象被引用传递，意味着传递的并不是实际的对象，而是对象的引用。因此，外部对引用对象所做的改变会反映到所有的对象上。

　　Java线程

**11.进程和线程的区别是什么？**

　　进程是执行着的应用程序，而线程是进程内部的一个执行序列。一个进程可以有多个线程。线程又叫做轻量级进程。

**12.创建线程有几种不同的方式？你喜欢哪一种？为什么？**

　　有三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。**

　　线程在执行过程中，可以处于下面几种状态：

　　就绪(Runnable):线程准备运行，不一定立马就能开始执行。

　　运行中(Running)：进程正在执行线程的代码。

　　等待中(Waiting):线程处于阻塞的状态，等待外部的处理结束。

　　睡眠中(Sleeping)：线程被强制睡眠。

　　I/O阻塞(Blocked on I/O)：等待I/O操作完成。

　　同步阻塞(Blocked on Synchronization)：等待获取锁。

　　死亡(Dead)：线程完成了执行。

**14.同步方法和同步代码块的区别是什么？**

　　在Java语言中，每一个对象有一把锁。线程可以使用synchronized关键字来获取对象上的锁。synchronized关键字可应用在方法级别(粗粒度锁)或者是代码块级别(细粒度锁)。

**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)？**

　　两个进程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是两个进程都陷入了无限的等待中。

**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。

　　Java集合类

**18.Java集合类框架的基本接口有哪些？**

　　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？**

　　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。

**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。

　　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。

**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。

**23.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)。

**24.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。

**25.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。

**26.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？**

　　下面列出了Array和ArrayList的不同点：

　　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。

**27.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。

　　也可以参考ArrayList vs. LinkedList。

**28.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，正数来表明输入对象小于，等于，大于已经存在的对象。

　　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。

**29.什么是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))。

**30.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，算法的规模或者是性能在最坏的场景下有多么好。

　　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。

**31.如何权衡是使用无序的数组还是有序的数组？**

　　有序数组最大的好处在于查找的时间复杂度是O(log n)，而无序数组是O(n)。有序数组的缺点是插入操作的时间复杂度是O(n)，因为值大的元素需要往后移动来给新元素腾位置。相反，无序数组的插入时间复杂度是常量O(1)。

**32.Java集合类框架的最佳实践有哪些？**

　　根据应用的需要正确选择要使用的集合的类型对性能非常重要，比如：假如元素的大小是固定的，而且能事先知道，我们就应该用Array而不是ArrayList。

　　有些集合类允许指定初始容量。因此，如果我们能估计出存储的元素的数目，我们可以设置初始容量来避免重新计算hash值或者是扩容。

　　为了类型安全，可读性和健壮性的原因总是要使用泛型。同时，使用泛型还可以避免运行时的ClassCastException。

　　使用JDK提供的不变类(immutable class)作为Map的键可以避免为我们自己的类实现hashCode()和equals()方法。

　　编程的时候接口优于实现。

　　底层的集合实际上是空的情况下，返回长度是0的集合或者是数组，不要返回null。

**33.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。

**34.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)。

　　垃圾收集器(Garbage Collectors)

**35.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。

**36.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。

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

　　在释放对象占用的内存之前，垃圾收集器会调用对象的finalize()方法。一般建议在该方法中释放对象持有的资源。

**38.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。

**39.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。

**40.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。

**41.在Java中，对象什么时候可以被垃圾回收？**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**42.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

**43.Java类的加载时机，类什么时候初始化？**

　　1）创建类的实例，也就是new一个对象   
　　2）访问某个类或接口的静态变量，或者对该静态变量赋值   
　　3）调用类的静态方法   
　　4）反射（Class.forName(“com.lyj.load”)）   
　　5）初始化一个类的子类（会首先初始化子类的父类）   
　　6）JVM启动时标明的启动类，即文件名和类名相同的那个类 。

**44.在Java中，类的初始化顺序是怎样的？**

　　1）如果这个类还没有被加载和链接，那先进行加载和链接   
　　2）假如这个类存在直接父类，并且这个类还没有被初始化（注意：在一个类加载器中，类只能初始化一次），那就初始化直接的父类（不适用于接口）   
　　3）加入类中存在初始化语句（如static变量和static块），那就依次执行这些初始化语句。

　　4）总的来说，初始化顺序依次是：（静态变量、静态初始化块）–>（变量、初始化块）–> 构造器；

如果有父类，则顺序是：父类static方法 –> 子类static方法 –> 父类构造方法- -> 子类构造方法

**45.在Java中，反射机制？**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**46.在Java中，IO，NIO？**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**47.单例模式？**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**48.代理模式？，两种实现方法**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**48.线程池原理，参数方法**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**48.线程池中怎么保证核心线程不被销毁**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**49.阻塞队列**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**50.分布式系统列**

　　当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**51.springboot的ioc aop，bean**

当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

**52.mysql中创建索引时的索引类型**

normal：表示普通索引

unique：表示唯一的，不允许重复的索引，如果该字段信息保证不会重复例如身份证号用作索引时，可设置为unique

full text: 表示 全文搜索的索引。 FULLTEXT 用于搜索很长一篇文章的时候，效果最好。用在比较短的文本，如果就一两行字的，普通的 INDEX 也可以。

spatial： 空间索引

空间索引是对空间数据类型的字段建立的索引，MYSQL中的空间数据类型有4种，分别是GEOMETRY、POINT、LINESTRING、POLYGON。MYSQL使用SPATIAL关键字进行扩展，空间索引只能在存储引擎为MYISAM的表中创建

**53.btree索引和hash索引的区别**

1、BTREE（B树（可以是多叉树）） {主流使用}

2、HASH（key,value） 这种方式对范围查询支持得不是很好

hash 索引结构的特殊性，其检索效率非常高，索引的检索可以一次定位，不像B-Tree 索引需要从根节点到枝节点，最后才能访问到页节点这样多次的IO访问，所以 Hash 索引的查询效率要远高于 B-Tree 索引。

可 能很多人又有疑问了，既然 Hash 索引的效率要比 B-Tree 高很多，为什么大家不都用 Hash 索引而还要使用 B-Tree 索引呢？任何事物都是有两面性的，Hash 索引也一样，虽然 Hash 索引效率高，但是 Hash 索引本身由于其特殊性也带来了很多限制和弊端，主要有以下这些。

（1）Hash 索引仅仅能满足”=”,”IN”和”<=>”查询，不能使用范围查询。

由于 Hash 索引比较的是进行 Hash 运算之后的 Hash 值，所以它只能用于等值的过滤，不能用于基于范围的过滤，因为经过相应的 Hash 算法处理之后的 Hash 值的大小关系，并不能保证和Hash运算前完全一样。

（2）Hash 索引无法被用来避免数据的排序操作。

由于 Hash 索引中存放的是经过 Hash 计算之后的 Hash 值，而且Hash值的大小关系并不一定和 Hash 运算前的键值完全一样，所以数据库无法利用索引的数据来避免任何排序运算；

（3）Hash 索引不能利用部分索引键查询。

对于组合索引，Hash 索引在计算 Hash 值的时候是组合索引键合并后再一起计算 Hash 值，而不是单独计算 Hash 值，所以通过组合索引的前面一个或几个索引键进行查询的时候，Hash 索引也无法被利用。

（4）Hash 索引在任何时候都不能避免表扫描。

前面已经知道，Hash 索引是将索引键通过 Hash 运算之后，将 Hash运算结果的 Hash 值和所对应的行指针信息存放于一个 Hash 表中，由于不同索引键存在相同 Hash 值，所以即使取满足某个 Hash 键值的数据的记录条数，也无法从 Hash 索引中直接完成查询，还是要通过访问表中的实际数据进行相应的比较，并得到相应的结果。

（5）Hash 索引遇到大量Hash值相等的情况后性能并不一定就会比B-Tree索引高。

对于选择性比较低的索引键，如果创建 Hash 索引，那么将会存在大量记录指针信息存于同一个 Hash 值相关联。这样要定位某一条记录时就会非常麻烦，会浪费多次表数据的访问，而造成整体性能低下。

原文：https://blog.csdn.net/guo\_qiangqiang/article/details/88794971

**54.应该选取表中哪些字段作为索引？**

为了使索引的使用效率更高，在创建索引时，必须考虑在哪些字段上创建索引和创建什么类型的索引,有7大原则：

1．选择唯一性索引

2．为经常需要排序、分组和联合操作的字段建立索引

3．为常作为查询条件的字段建立索引

4．限制索引的数目

5．尽量使用数据量少的索引

6．尽量使用前缀来索引

7．删除不再使用或者很少使用的索引

8. 经常更新修改的字段不要建立索引（针对mysql说，因为字段更改同时索引就要重新建立，排序，而Orcale好像是有这样的机制字段值更改了，它不立刻建立索引，排序索引，而是根据更改个数，时间段去做平衡索引这件事的）

9、不推荐在同一列建多个索引

**55.innodb和myaim？**

**56.聚簇索引与非聚簇索引？**

**57.索引最左匹配**

**58.阻塞队列**

自从Java 1.5之后，在java.util.concurrent包下提供了若干个阻塞队列，主要有以下几个：

　　ArrayBlockingQueue：基于数组实现的一个阻塞队列，在创建ArrayBlockingQueue对象时必须制定容量大小。并且可以指定公平性与非公平性，默认情况下为非公平的，即不保证等待时间最长的队列最优先能够访问队列。

　　LinkedBlockingQueue：基于链表实现的一个阻塞队列，在创建LinkedBlockingQueue对象时如果不指定容量大小，则默认大小为Integer.MAX\_VALUE。

　　PriorityBlockingQueue：以上2种队列都是先进先出队列，而PriorityBlockingQueue却不是，它会按照元素的优先级对元素进行排序，按照优先级顺序出队，每次出队的元素都是优先级最高的元素。注意，此阻塞队列为无界阻塞队列，即容量没有上限（通过源码就可以知道，它没有容器满的信号标志），前面2种都是有界队列。

　　DelayQueue：基于PriorityQueue，一种延时阻塞队列，DelayQueue中的元素只有当其指定的延迟时间到了，才能够从队列中获取到该元素。DelayQueue也是一个无界队列，因此往队列中插入数据的操作（生产者）永远不会被阻塞，而只有获取数据的操作（消费者）才会被阻塞。

**59.阻塞队列中的方法 VS 非阻塞队列中的方法**

1.非阻塞队列中的几个主要方法：

　　add(E e):将元素e插入到队列末尾，如果插入成功，则返回true；如果插入失败（即队列已满），则会抛出异常；

　　remove()：移除队首元素，若移除成功，则返回true；如果移除失败（队列为空），则会抛出异常；

　　offer(E e)：将元素e插入到队列末尾，如果插入成功，则返回true；如果插入失败（即队列已满），则返回false；

　　poll()：移除并获取队首元素，若成功，则返回队首元素；否则返回null；

　　peek()：获取队首元素，若成功，则返回队首元素；否则返回null

　　对于非阻塞队列，一般情况下建议使用offer、poll和peek三个方法，不建议使用add和remove方法。因为使用offer、poll和peek三个方法可以通过返回值判断操作成功与否，而使用add和remove方法却不能达到这样的效果。注意，非阻塞队列中的方法都没有进行同步措施。

2.阻塞队列中的几个主要方法：

阻塞队列包括了非阻塞队列中的大部分方法，上面列举的5个方法在阻塞队列中都存在，但是要注意这5个方法在阻塞队列中都进行了同步措施。除此之外，阻塞队列提供了另外4个非常有用的方法：

put方法用来向队尾存入元素，如果队列满，则等待；

　　take方法用来从队首取元素，如果队列为空，则等待；

　　 offer(E e,long timeout, TimeUnit unit)

方法用来向队尾存入元素，如果队列满，则等待一定的时间，当时间期限达到时，如果还没有插入成功，则返回false；否则返回true；

　　 poll(long timeout, TimeUnit unit)

方法用来从队首取元素，如果队列空，则等待一定的时间，当时间期限达到时，如果取到，则返回null；否则返回取得的元素；

**60.wait和sleep**

1.非阻塞队列中的几个主要方法：

**61.异常，线程sleep和return之间的关系**

1.非阻塞队列中的几个主要方法：

**61.IO和NIO**

1.非阻塞队列中的几个主要方法：

**62.hashcode与equals方法；**