多线程、并发及线程的基础问题：

1）Java 中能创建 volatile 数组吗？
能，Java 中可以创建 volatile 类型数组，不过只是一个指向数组的引用，而不是整个数组。我的意思是，如果改变引用指向的数组，将会受到 volatile 的保护，但是如果多个线程同时改变数组的元素，volatile 标示符就不能起到之前的保护作用了。

2）volatile 能使得一个非原子操作变成原子操作吗？
一个典型的例子是在类中有一个 long 类型的成员变量。如果你知道该成员变量会被多个线程访问，如计数器、价格等，你最好是将其设置为 volatile。为什么？因为 Java 中读取 long 类型变量不是原子的，需要分成两步，如果一个线程正在修改该 long 变量的值，另一个线程可能只能看到该值的一半（前 32 位）。但是对一个 volatile 型的 long 或 double 变量的读写是原子。

3）volatile 修饰符的有过什么实践？
一种实践是用 volatile 修饰 long 和 double 变量，使其能按原子类型来读写。double 和 long 都是 64 位宽，因此对这两种类型的读是分为两部分的，第一次读取第一个 32 位，然后再读剩下的 32 位，这个过程不是原子的，但 Java 中 volatile 型的 long 或 double 变量的读写是原子的。volatile 修复符的另一个作用是提供内存屏障（memory barrier），例如在分布式框架中的应用。简单的说，就是当你写一个 volatile 变量之前，Java 内存模型会插入一个写屏障（write barrier），读一个 volatile 变量之前，会插入一个读屏障（read barrier）。意思就是说，在你写一个 volatile 域时，能保证任何线程都能看到你写的值，同时，在写之前，也能保证任何数值的更新对所有线程是可见的，因为内存屏障会将其他所有写的值更新到缓存。

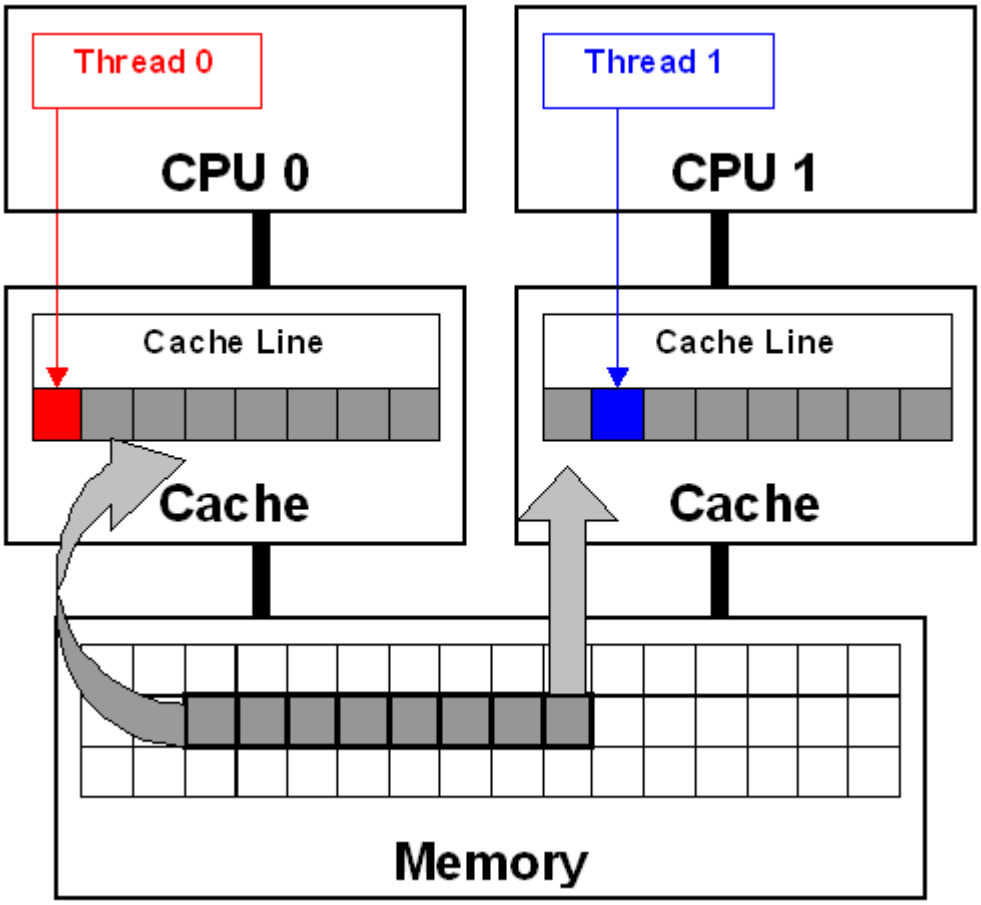
4）volatile 类型变量提供什么保证？
volatile 变量提供顺序和可见性保证，例如，JVM 或者 JIT 为了获得更好的性能会对语句重排序，但是 volatile 类型变量即使在没有同步块的情况下赋值也不会与其他语句重排序。volatile 提供 happens-before 的保证，确保一个线程的修改能对其他线程是可见的。某些情况下，volatile 还能提供原子性，如读 64 位数据类型，像 long 和 double 都不是原子的，但 volatile 类型的 double 和 long 就是原子的。

5）你是如何调用 wait（）方法的？使用 if 块还是循环？为什么？
wait() 方法应该在循环调用，因为当线程获取到 CPU 开始执行的时候，其他条件可能还没有满足，所以在处理前，循环检测条件是否满足会更好。下面是一段标准的使用 wait 和 notify 方法的代码：

```
// The standard idiom for using the wait method
synchronized (obj) {
while (condition does not hold)
obj.wait(); // (Releases lock, and reacquires on wakeup)
... // Perform action appropriate to condition
}
```

6）什么是多线程环境下的伪共享（false sharing）？

伪共享是多线程系统（每个处理器有自己的局部缓存）中一个众所周知的性能问题。伪共享发生在不同处理器的上的线程对变量的修改依赖于相同的缓存行，如下图所示：



7）什么是 Busy spin？我们为什么要使用它？

Busy spin 是一种在不释放 CPU 的基础上等待事件的技术。它经常用于避免丢失 CPU 缓存中的数据（如果线程先暂停，之后在其他 CPU 上运行就会丢失）。所以，如果你的工作要求低延迟，并且你的线程目前没有任何顺序，这样你就可以通过循环检测队列中的新消息来代替调用 sleep() 或 wait() 方法。它唯一的好处就是你只需等待很短的时间，如几微秒或几纳秒。LMAX 分布式框架是一个高性能线程间通信的库，该库有一个 BusySpinWaitStrategy 类就是基于这个概念实现的，使用 busy spin 循环 EventProcessors 等待屏障。

8）什么是线程局部变量？

线程局部变量是局限于线程内部的变量，属于线程自身所有，不在多个线程间共享。Java 提供 ThreadLocal 类来支持线程局部变量，是一种实现线程安全的方式。但是在管理环境下（如 web 服务器）使用线程局部变量的时候要特别小心，在这种情况下，工作线程的生命周期比任何应用变量的生命周期都要长。任何线程局部变量一旦在工作完成后没有释放，Java 应

用就存在内存泄露的风险。

9）Java 中 sleep 方法和 wait 方法的区别？
虽然两者都是用来暂停当前运行的线程，但是 sleep() 实际上只是短暂停顿，因为它不会释放锁，而 wait() 意味着条件等待，这就是为什么该方法要释放锁，因为只有这样，其他等待的线程才能在满足条件时获取到该锁。

10）什么是不可变对象（immutable object）？Java 中怎么创建一个不可变对象？
不可变对象指对象一旦被创建，状态就不能再改变。任何修改都会创建一个新的对象，如 String、Integer 及其它包装类。

11）我们能创建一个包含可变对象的不可变对象吗？
是的，我们是可以创建一个包含可变对象的不可变对象的，你只需要谨慎一点，不要共享可变对象的引用就可以了，如果需要变化时，就返回原对象的一个拷贝。最常见的例子就是对象中包含一个日期对象的引用。

数据类型和 Java 基础面试问题

12）Java 中应该使用什么数据类型来代表价格？
如果不是特别关心内存和性能的话，使用 BigDecimal，否则使用预定义精度的 double 类型。

13）怎么将 byte 转换为 String？
可以使用 String 接收 byte[] 参数的构造器来进行转换，需要注意的点是要使用的正确的编码，否则会使用平台默认编码，这个编码可能跟原来的编码相同，也可能不同。

14）我们能将 int 强制转换为 byte 类型的变量吗？如果该值大于 byte 类型的范围，将会出现什么现象？
是的，我们可以做强制转换，但是 Java 中 int 是 32 位的，而 byte 是 8 位的，所以，如果强制转化是，int 类型的高 24 位将会被丢弃，byte 类型的范围是从 -128 到 128。

15）Java 中 ++ 操作符是线程安全的吗？
不是线程安全的操作。它涉及到多个指令，如读取变量值，增加，然后存储回内存，这个过程可能会出现多个线程交差。

16）a = a + b 与 a += b 的区别？
+= 隐式的将加操作的结果类型强制转换为持有结果的类型。如果两这个整型相加，如 byte、short 或者 int，首先会将它们提升到 int 类型，然后在执行加法操作。如果加法操作的结果比 a 的最大值要大，则 a+b 会出现编译错误，但是 a += b 没问题，如下：

```
byte a = 127;
byte b = 127;
b = a + b; // error : cannot convert from int to byte
b += a; // ok
```

注：其实无论 a+b 的值为多少，编译器都会报错，因为 a+b 操作会将 a、b 提升为 int 类

型，所以将 int 类型赋值给 byte 就会编译出错

17）我能在不进行强制转换的情况下将一个 double 值赋值给 long 类型的变量吗？
不行，你不能在没有强制类型转换的前提下将一个 double 值赋值给 long 类型的变量，因为 double 类型的范围比 long 类型更广，所以必须要进行强制转换。

18）3*0.1 == 0.3 将会返回什么？true 还是 false？
false，因为有些浮点数不能完全精确的表示出来。

19）int 和 Integer 哪个会占用更多的内存？
Integer 对象会占用更多的内存。Integer 是一个对象，需要存储对象的元数据。但是 int 是一个原始类型的数据，所以占用的空间更少。

20）为什么 Java 中的 String 是不可变的（Immutable）？
Java 中的 String 不可变是因为 Java 的设计者认为字符串使用非常频繁，将字符串设置为不可变可以允许多个客户端之间共享相同的字符串。

21）Java 中的构造器链是什么？
当你从一个构造器中调用另一个构造器，就是 Java 中的构造器链。这种情况只在重载了类的构造器的时候才会出现。

JVM 底层 与 GC（Garbage Collection） 的面试问题

22）64 位 JVM 中，int 的长度是多数？
Java 中，int 类型变量的长度是一个固定值，与平台无关，都是 32 位。意思就是说，在 32 位 和 64 位 的 Java 虚拟机中，int 类型的长度是相同的。

23）Serial 与 Parallel GC 之间的不同之处？
Serial 与 Parallel 在 GC 执行的时候都会引起 stop-the-world。它们之间主要不同 serial 收集器是默认的复制收集器，执行 GC 的时候只有一个线程，而 parallel 收集器使用多个 GC 线程来执行。

24）32 位和 64 位的 JVM，int 类型变量的长度是多数？
32 位和 64 位的 JVM 中，int 类型变量的长度是相同的，都是 32 位或者 4 个字节。

25）Java 中 WeakReference 与 SoftReference 的区别？
虽然 WeakReference 与 SoftReference 都有利于提高 GC 和 内存的效率，但是 WeakReference ，一旦失去最后一个强引用，就会被 GC 回收，而软引用虽然不能阻止被回收，但是可以延迟到 JVM 内存不足的时候。

26）WeakHashMap 是怎么工作的？
WeakHashMap 的工作与正常的 HashMap 类似，但是使用弱引用作为 key，意思就是当 key 对象没有任何引用时，key/value 将会被回收。

27）JVM 选项 -XX:+UseCompressedOops 有什么作用？为什么要使用？
当你将你的应用从 32 位的 JVM 迁移到 64 位的 JVM 时，由于对象的指针从 32 位增加到了 64 位，因此堆内存会突然增加，差不多要翻倍。这也会对 CPU 缓存（容量比内存小很多）的数据产生不利的影响。因为，迁移到 64 位的 JVM 主要动机在于可以指定最大堆大小，通过压缩 OOP 可以节省一定的内存。通过 -XX:+UseCompressedOops 选项，JVM 会使用 32 位的 OOP，而不是 64 位的 OOP。

28）怎样通过 Java 程序来判断 JVM 是 32 位 还是 64 位？
你可以检查某些系统属性如 sun.arch.data.model 或 os.arch 来获取该信息。

29）32 位 JVM 和 64 位 JVM 的最大堆内存分别是多数？
理论上说上 32 位的 JVM 堆内存可以到达 2^32，即 4GB，但实际上会比这个小很多。不同操作系统之间不同，如 Windows 系统大约 1.5 GB，Solaris 大约 3GB。64 位 JVM 允许指定最大的堆内存，理论上可以达到 2^64，这是一个非常大的数字，实际上你可以指定堆内存大小到 100GB。甚至有的 JVM，如 Azul，堆内存到 1000G 都是可能的。

30）JRE、JDK、JVM 及 JIT 之间有什么不同？
JRE 代表 Java 运行时（Java run-time），是运行 Java 引用所必须的。JDK 代表 Java 开发工具（Java development kit），是 Java 程序的开发工具，如 Java 编译器，它也包含 JRE。JVM 代表 Java 虚拟机（Java virtual machine），它的责任是运行 Java 应用。JIT 代表即时编译（Just In Time compilation），当代码执行的次数超过一定的阈值时，会将 Java 字节码转换为本地代码，如，主要的热点代码会被准换为本地代码，这样有利大幅度提高 Java 应用的性能。

31）解释 Java 堆空间及 GC？
当通过 Java 命令启动 Java 进程的时候，会为它分配内存。内存的一部分用于创建堆空间，当程序中创建对象的时候，就从对空间中分配内存。GC 是 JVM 内部的一个进程，回收无效对象的内存用于将来的分配。

32）你能保证 GC 执行吗？
不能，虽然你可以调用 System.gc() 或者 Runtime.gc()，但是没有办法保证 GC 的执行。

33）怎么获取 Java 程序使用的内存？堆使用的百分比？
可以通过 java.lang.Runtime 类中与内存相关方法来获取剩余的内存，总内存及最大堆内存。通过这些方法你也可以获取到堆使用的百分比及堆内存的剩余空间。Runtime.freeMemory() 方法返回剩余空间的字节数，Runtime.totalMemory() 方法总内存的字节数，Runtime.maxMemory() 返回最大内存的字节数。

34）Java 中堆和栈有什么区别？
JVM 中堆和栈属于不同的内存区域，使用目的也不同。栈常用于保存方法帧和局部变量，而对象总是在堆上分配。栈通常都比堆小，也不会在多个线程之间共享，而堆被整个 JVM 的所有线程共享。

Java 基本概念面试题

35）“a==b”和”a.equals(b)”有什么区别？
如果 a 和 b 都是对象，则 a==b 是比较两个对象的引用，只有当 a 和 b 指向的是堆中的同一个对象才会返回 true，而 a.equals(b) 是进行逻辑比较，所以通常需要重写该方法来提供逻辑一致性的比较。例如，String 类重写 equals() 方法，所以可以用于两个不同对象，但是包含的字母相同的比较。

36）a.hashCode() 有什么用？与 a.equals(b) 有什么关系？
hashCode() 方法是相应对象整型的 hash 值。它常用于基于 hash 的集合类，如 Hashtable、HashMap、LinkedHashMap 等等。它与 equals() 方法关系特别紧密。根据 Java 规范，两个使用 equal() 方法来判断相等的对象，必须具有相同的 hash code。

37）final、finalize 和 finally 的不同之处？
final 是一个修饰符，可以修饰变量、方法和类。如果 final 修饰变量，意味着该变量的值在初始化后不能被改变。finalize 方法是在对象被回收之前调用的方法，给对象自己最后一个复活的机会，但是什么时候调用 finalize 没有保证。finally 是一个关键字，与 try 和 catch 一起用于异常的处理。finally 块一定会被执行，无论在 try 块中是否有发生异常。

38）Java 中的编译期常量是什么？使用它又什么风险？
公共静态不可变（public static final ）变量也就是我们所说的编译期常量，这里的 public 可选的。实际上这些变量在编译时会被替换掉，因为编译器知道这些变量的值，并且知道这些变量在运行时不能改变。这种方式存在的一个问题是你使用了一个内部的或第三方库中的公有编译时常量，但是这个值后面被其他人改变了，但是你的客户端仍然在使用老的值，甚至你已经部署了一个新的 jar。为了避免这种情况，当你在更新依赖 JAR 文件时，确保重新编译你的程序。

Java 集合框架的面试题

39) List、Set、Map 和 Queue 之间的区别？
List 是一个有序集合，允许元素重复。它的某些实现可以提供基于下标值的常量访问时间，但是这不是 List 接口保证的。Set 是一个无序集合。

40）poll() 方法和 remove() 方法的区别？
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。

41）Java 中 LinkedHashMap 和 PriorityQueue 的区别是什么？
PriorityQueue 保证最高或者最低优先级的的元素总是在队列头部，但是 LinkedHashMap 维持的顺序是元素插入的顺序。当遍历一个 PriorityQueue 时，没有任何顺序保证，但是 LinkedHashMap 课保证遍历顺序是元素插入的顺序。

42）ArrayList 与 LinkedList 的不区别？
最明显的区别是 ArrrayList 底层的数据结构是数组，支持随机访问，而 LinkedList 的底层数据结构书链表，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)。

43）用哪两种方式来实现集合的排序？
你可以使用有序集合，如 TreeSet 或 TreeMap，你也可以使用有顺序的的集合，如 list，然后通过 Collections.sort() 来排序。

44）Java 中怎么打印数组？
你可以使用 Arrays.toString() 和 Arrays.deepToString() 方法来打印数组。由于数组没有实现 toString() 方法，所以如果将数组传递给 System.out.println() 方法，将无法打印出数组的内容，但是 Arrays.toString() 可以打印每个元素。

45) Hashtable 与 HashMap 有什么不同之处？
这两个类有许多不同的地方，下面列出了一部分：
a) Hashtable 是 JDK 1 遗留下来的类，而 HashMap 是后来增加的。
b）Hashtable 是同步的，比较慢，但 HashMap 没有同步策略，所以会更快。
c）Hashtable 不允许有个空的 key，但是 HashMap 允许出现一个 null key。

46）Java 中的 HashSet，内部是如何工作的？
HashSet 的内部采用 HashMap 来实现。由于 Map 需要 key 和 value，所以所有 key 的都有一个默认 value。类似于 HashMap，HashSet 不允许重复的 key，只允许有一个 null key，意思就是 HashSet 中只允许存储一个 null 对象。

47）写一段代码在遍历 ArrayList 时移除一个元素？
该问题的关键在于面试者使用的是 ArrayList 的 remove() 还是 Iterator 的 remove()方法。这有一段示例代码，是使用正确的方式来实现在遍历的过程中移除元素，而不会出现 ConcurrentModificationException 异常的示例代码。

48）我们能自己写一个容器类，然后使用 for-each 循环码？
可以，你可以写一个自己的容器类。如果你想使用 Java 中增强的循环来遍历，你只需要实现 Iterable 接口。如果你实现 Collection 接口，默认就具有该属性。

49）ArrayList 和 HashMap 的默认大小是多数？
在 Java 7 中，ArrayList 的默认大小是 10 个元素，HashMap 的默认大小是 16 个元素（必须是 2 的幂）。这就是 Java 7 中 ArrayList 和 HashMap 类的代码片段：

```
// from ArrayList.java JDK 1.7
private static final int DEFAULT_CAPACITY = 10;

//from HashMap.java JDK 7
static final int DEFAULT_INITIAL_CAPACITY = 1 << 4; // aka 16
```

50）有没有可能两个不相等的对象有有相同的 hashcode？
有可能，两个不相等的对象可能会有相同的 hashcode 值，这就是为什么在 hashmap 中会有冲突。相等 hashcode 值的规定只是说如果两个对象相等，必须有相同的 hashcode 值，

但是没有关于不相等对象的任何规定。

51）两个相同的对象会有不同的的 hash code 吗？
不能，根据 hash code 的规定，这是不可能的。

52）我们可以在 hashcode() 中使用随机数字吗？
不行，因为对象的 hashcode 值必须是相同的。参见答案获取更多关于 Java 中重写 hashCode() 方法的知识。

53）Java 中，Comparator 与 Comparable 有什么不同？
Comparable 接口用于定义对象的自然顺序，而 comparator 通常用于定义用户定制的顺序。Comparable 总是只有一个，但是可以有多个 comparator 来定义对象的顺序。

54）为什么在重写 equals 方法的时候需要重写 hashCode 方法？
因为有强制的规范指定需要同时重写 hashcode 与 equal 是方法，许多容器类，如 HashMap、HashSet 都依赖于 hashcode 与 equals 的规定。

Java 最佳实践的面试问题

55）Java 中，编写多线程程序的时候你会遵循哪些最佳实践？
a）给线程命名，这样可以帮助调试。
b）最小化同步的范围，而不是将整个方法同步，只对关键部分做同步。
c）如果可以，更偏向于使用 volatile 而不是 synchronized。
d）使用更高层次的并发工具，而不是使用 wait() 和 notify() 来实现线程间通信，如 BlockingQueue，CountDownLatch 及 Semeaphore。
e）优先使用并发集合，而不是对集合进行同步。并发集合提供更好的可扩展性。

56）说出几点 Java 中使用 Collections 的最佳实践？
a）使用正确的集合类，例如，如果不需要同步列表，使用 ArrayList 而不是 Vector。
b）优先使用并发集合，而不是对集合进行同步。并发集合提供更好的可扩展性。
c）使用接口代表和访问集合，如使用 List 存储 ArrayList，使用 Map 存储 HashMap 等等。
d）使用迭代器来循环集合。
e）使用集合的时候使用泛型。

57）说出在 Java 中使用线程的最佳实践？
a）对线程命名
b）将线程和任务分离，使用线程池执行器来执行 Runnable 或 Callable。
c）使用线程池

58）说出 IO 的最佳实践？
a）使用有缓冲区的 IO 类，而不要单独读取字节或字符。
b）使用 NIO 和 NIO2
c）在 finally 块中关闭流，或者使用 try-with-resource 语句。
d）使用内存映射文件获取更快的 IO。

59）列出应该遵循的 JDBC 最佳实践？
a）使用批量的操作来插入和更新数据
b）使用 PreparedStatement 来避免 SQL 异常，并提高性能。
c）使用数据库连接池
d）通过列名来获取结果集，不要使用列的下标来获取。

60）说出几条 Java 中方法重载的最佳实践？
a）不要重载这样的方法：一个方法接收 int 参数，而另个方法接收 Integer 参数。
b）不要重载参数数量一致，而只是参数顺序不同的方法。
c）如果重载的方法参数个数多于 5 个，采用可变参数。

Date、Time 及 Calendar 的面试题

61）在多线程环境下，SimpleDateFormat 是线程安全的吗？
不是，非常不幸，DateFormat 的所有实现，包括 SimpleDateFormat 都不是线程安全的，因此你不应该在多线程序中使用，除非是在对外线程安全的环境中使用，如 将 SimpleDateFormat 限制在 ThreadLocal 中。如果你不这么做，在解析或者格式化日期的时候，可能会获取到一个不正确的结果。因此，从日期、时间处理的所有实践来说，我强力推荐 joda-time 库。

62）Java 中如何格式化一个日期？如格式化为 ddMMyyyy 的形式？
Java 中，可以使用 SimpleDateFormat 类或者 joda-time 库来格式日期。DateFormat 类允许你使用多种流行的格式来格式化日期。参见答案中的示例代码，代码中演示了将日期格式化成不同的格式，如 dd-MM-yyyy 或 ddMMyyyy。

关于 OOP 和设计模式的面试题

63）接口是什么？为什么要使用接口而不是直接使用具体类？
接口用于定义 API。它定义了类必须得遵循的规则。同时，它提供了一种抽象，因为客户端只使用接口，这样可以有多重实现，如 List 接口，你可以使用可随机访问的 ArrayList，也可以使用方便插入和删除的 LinkedList。接口中不允许写代码，以此来保证抽象，但是 Java 8 中你可以在接口声明静态的默认方法，这种方法是具体的。

64）Java 中，抽象类与接口之间有什么不同？
Java 中，抽象类和接口有很多不同之处，但是最重要的一个是 Java 中限制一个类只能继承一个类，但是可以实现多个接口。抽象类可以很好的定义一个家族类的默认行为，而接口能更好的定义类型，有助于后面实现多态机制。关于这个问题的讨论请查看答案。

65）除了单例模式，你在生产环境中还用过什么设计模式？
这需要根据你的经验来回答。一般情况下，你可以说依赖注入，工厂模式，装饰模式或者观察者模式，随意选择你使用过的一种即可。不过你要准备回答接下的基于你选择的模式的问题。

66）适配器模式是什么？什么时候使用？
适配器模式提供对接口的转换。如果你的客户端使用某些接口，但是你有另外一些接口，你就可以写一个适配去来连接这些接口。

67）构造器注入和 setter 依赖注入，那种方式更好？
每种方式都有它的缺点和优点。构造器注入保证所有的注入都被初始化，但是 setter 注入提供更好的灵活性来设置可选依赖。如果使用 XML 来描述依赖，Setter 注入的可读写会更强。经验法则是强制依赖使用构造器注入，可选依赖使用 setter 注入。

68）依赖注入和工程模式之间有什么不同？
虽然两种模式都是将对象的创建从应用的逻辑中分离，但是依赖注入比工程模式更清晰。通过依赖注入，你的类就是 POJO，它只知道依赖而不关心它们怎么获取。使用工厂模式，你的类需要通过工厂来获取依赖。因此，使用 DI 会比使用工厂模式更容易测试。

69）适配器模式和装饰器模式有什么区别？
虽然适配器模式和装饰器模式的结构类似，但是每种模式的出现意图不同。适配器模式被用于桥接两个接口，而装饰模式的目的是在不修改类的情况下给类增加新的功能。

70）适配器模式和代理模式之前有什么不同？
这个问题与前面的类似，适配器模式和代理模式的区别在于他们的意图不同。由于适配器模式和代理模式都是封装真正执行动作的类，因此结构是一致的，但是适配器模式用于接口之间　的转换，而代理模式则是增加一个额外的中间层，以便支持分配、控制或智能访问。

71）什么是模板方法模式？
模板方法提供算法的框架，你可以自己去配置或定义步骤。例如，你可以将排序算法看做是一个模板。它定义了排序的步骤，但是具体的比较，可以使用 Comparable 或者其语言中类似东西，具体策略由你去配置。列出算法概要的方法就是众所周知的模板方法。

72）什么时候使用访问者模式？
访问者模式用于解决在类的继承层次上增加操作，但是不直接与之关联。这种模式采用双派发的形式来增加中间层。

73）什么时候使用组合模式？
组合模式使用树结构来展示部分与整体继承关系。它允许客户端采用统一的形式来对待单个对象和对象容器。当你想要展示对象这种部分与整体的继承关系时采用组合模式。

74）继承和组合之间有什么不同？
虽然两种都可以实现代码复用，但是组合比继承共灵活，因为组合允许你在运行时选择不同的实现。用组合实现的代码也比继承测试起来更加简单。

75）描述 Java 中的重载和重写？
重载和重写都允许你用相同的名称来实现不同的功能，但是重载是编译时活动，而重写是运行时活动。你可以在同一个类中重载方法，但是只能在子类中重写方法。重写必须要有继承。

76）Java 中，嵌套公共静态类与顶级类有什么不同？
类的内部可以有多个嵌套公共静态类，但是一个 Java 源文件只能有一个顶级公共类，并且顶级公共类的名称与源文件名称必须一致。

77) OOP 中的 组合、聚合和关联有什么区别？
如果两个对象彼此有关系，就说他们是彼此相关联的。组合和聚合是面向对象中的两种形式的关联。组合是一种比聚合更强力的关联。组合中，一个对象是另一个的拥有者，而聚合则是指一个对象使用另一个对象。如果对象 A 是由对象 B 组合的，则 A 不存在的话，B 一定不存在，但是如果 A 对象聚合了一个对象 B，则即使 A 不存在了，B 也可以单独存在。

78）给我一个符合开闭原则的设计模式的例子？
开闭原则要求你的代码对扩展开放，对修改关闭。这个意思就是说，如果你想增加一个新的功能，你可以很容易的在不改变已测试过的代码的前提下增加新的代码。有好几个设计模式是基于开闭原则的，如策略模式，如果你需要一个新的策略，只需要实现接口，增加配置，不需要改变核心逻辑。一个正在工作的例子是 Collections.sort() 方法，这就是基于策略模式，遵循开闭原则的，你不需为新的对象修改 sort() 方法，你需要做的仅仅是实现你自己的 Comparator 接口。

79）什么时候使用享元模式？
享元模式通过共享对象来避免创建太多的对象。为了使用享元模式，你需要确保你的对象是不可变的，这样你才能安全的共享。JDK 中 String 池、Integer 池以及 Long 池都是很好的使用了享元模式的例子。

其他基本问题：
1、如何原地交换两个变量的值
int a = 5, b = 10; a = a+b; b = a-b; a = a – b;