题目：115 个 Java 面试题和答案——终极（上）

本文我们将要讨论 Java 面试中的各种不同类型的面试题，它们可以让雇主测试应聘者的 Java 和通用的面向对象编程的能力。下面的章节分为上下两篇，第一篇将要讨论面向对象编程和它的特点，关于 Java 和它的功能的常见问题，Java 的集合类，垃圾收集器，第二篇主要讨论异常处理，Java 小应用程序，Swing，JDBC，远程方法调用(RMI)，Servlet 和 JSP。

目录

面向对象编程（OOP）

Java 是一个支持并发、基于类和面向对象的计算机编程语言。下面列出了面向对象软件开发的优点：

代码开发模块化，更易维护和修改。
代码复用。
增强代码的可靠性和灵活性。
增加代码的可理解性。
面向对象编程有很多重要的特性，比如：封装，继承，多态和抽象。下面的章节我们会逐个分析这些特性。

封装

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。

多态

多态是编程语言给不同的底层数据类型做相同的接口展示的一种能力。一个多态类型上的操作可以应用到其他类型的值上面。

继承

继承给对象提供了从基类获取字段和方法的能力。继承提供了代码的重用行，也可以在不修改类的情况下给现存的类添加新特性。

抽象

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 封装的不同点

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。

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.JDK 和 **JRE** 的区别是什么 ？

Java 运行时环境(JRE)是将要执行 Java 程序的 Java 虚拟机。它同时也包含了执行 applet 需要的浏览器插件。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。

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。

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
(译者注：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)