

找工作要面试，有面试就有对付面试的办法。以下一些题目来自我和我朋友痛苦的面试经历，提这些问题的公司包括 IBM, E\*Trade, Siebel, Motorola, SUN, 以及其它大小公司。

面试是没什么道理可讲的，它的题目有的不合情理、脱离实际。有在纸上写的，有当面考你的，也有在电话里问的，给你 IDE 的估计很少(否则你赶快去买彩票，说不定中)。所以如果你看完此文后，请不要抱怨说这些问题都能用 IDE 来解决。你必须在任何情况下准确回答这些问题，在面试中如果出现一两题回答不准确很有可能你就会被拒之门外了。

当然这些都是 Java 的基本题，那些面试的人大多数不会问你 Hibernate 有多先进，Eclipse 的三个组成部分，或 command design pattern，他们都是老一辈了，最喜欢问的就是基础知识。别小看了这些基础，我朋友水平一流，结果就栽在一到基础知识的问题下，和高薪无缘。

好了废话少说，开始正题。

第一，谈谈 final, finally, finalize 的区别。

**final** 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

**finally** 是异常处理语句结构的一部分，表示总是执行。

**finalize** 是 Object 类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

第二，Anonymous Inner Class (匿名内部类) 是否可以 extends(继承)其它类，是否可以 implements(实现)interface(接口)?

可以继承其他类或完成其他接口，在 swing 编程中常用此方式。

第三，Static Nested Class 和 Inner Class 的不同，说得越多越好(面试题有的很笼统)。

Static Nested Class 是被声明为静态 (static) 的内部类，它可以不依赖于外部类实例被实例化。而通常的内部类需要在外部类实例化后才能实例化。

第四，&和&&的区别。

&是位运算符，表示按位与运算，&&是逻辑运算符，表示逻辑与 (and)。

第五，HashMap 和 Hashtable 的区别。

HashMap 是 Hashtable 的轻量级实现 (非线程安全的实现)，他们都完成了 Map 接口，主要区别在于 HashMap 允许空 (null) 键值 (key)，由于非线程安全，效率上可能高于 Hashtable。

# 面试交流网

## http://www.msjl.net

第六，Collection 和 Collections 的区别。

Collection 是集合类的上级接口，继承与他的接口主要有 Set 和 List。  
Collections 是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

第七，什么时候用 assert。

1.4 新增关键字（语法），用于测试 boolean 表达式状态，可用于调试程序。

使用方法 `assert <boolean 表达式>`，表示如果表达式为真（true），则下面的语句执行，否则抛出 `AssertionError`。

另外的使用方式 `assert <boolean 表达式>:<other 表达式>`，表示如果表达式为真，后面的表达式忽略，否则后面表达式的值用于 `AssertionError` 的构建参数。

注意编译时要增加 `-source 1.4` 参数，否则报错。]运行时要增加 `-ea` 参数，否则 `assert` 行被忽略

第八，GC 是什么？为什么要有 GC？

GC 是垃圾收集的意思（Garbage Collection），内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java 提供的 GC 功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java 语言没有提供释放已分配内存的显示操作方法。

第九，`String s = new String("xyz");`创建了几个 String Object?  
两个

第十，`Math.round(11.5)`等於多少？`Math.round(-11.5)`等於多少？

`Math.round(11.5)==12`

`Math.round(-11.5)==-11`

`round` 方法返回与参数最接近的长整数，参数加 1/2 后求其 floor。

第十一，`short s1 = 1; s1 = s1 + 1;`有什么错？`short s1 = 1; s1 += 1;`有什么错？

`short s1 = 1; s1 = s1 + 1;`（`s1+1` 运算结果是 int 型，需要强制转换类型）

`short s1 = 1; s1 += 1;`（可以正确编译）

第十二，`sleep()` 和 `wait()` 有什么区别？

`sleep` 是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用 `sleep` 不会释放对象锁。  
`wait` 是 Object 类的方法，对此对象调用 `wait` 方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出 `notify` 方法（或 `notifyAll`）后本线程才进入对象锁定池准备获得对象锁进入运行状态。

第十三，Java 有没有 goto？

没有

很十三的问题，如果哪个面试的问到这个问题，我劝你还是别进这家公司。（完）  
(job.sohu.com)

第十四，数组有没有 `length()` 这个方法？`String` 有没有 `length()` 这个方法？

数组没有 `length()` 这个方法，有 `length` 的属性。

`String` 有 `length()` 这个方法。

第十五，`Overload` 和 `Override` 的区别。`Overloaded` 的方法是否可以改变返回值的类型？  
方法的重写 `Overriding` 和重载 `Overloading` 是 `Java` 多态性的不同表现。重写 `Overriding` 是父类与子类之间多态性的一种表现，重载 `Overloading` 是一个类中多态性的一种表现。  
如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (`Overriding`)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被“屏蔽”了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载 (`Overloading`)。`Overloaded` 的方法是可以改变返回值的类型。

第十六，`Set` 里的元素是不能重复的，那么用什么方法来区分重复与否呢？是用 `==` 还是 `equals()`？它们有何区别？

`Set` 里的元素是不能重复的，那么用 `iterator()` 方法来区分重复与否。`equals()` 是判断两个 `Set` 是否相等。

`equals()` 和 `==` 方法决定引用值是否指向同一对象。`equals()` 在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

第十七，给我一个你最常见到的 `runtime exception`。

`ArithmeticException`, `ArrayStoreException`, `BufferOverflowException`,  
`BufferUnderflowException`, `CannotRedoException`, `CannotUndoException`,  
`ClassCastException`, `CMMException`, `ConcurrentModificationException`, `DOMException`,  
`EmptyStackException`, `IllegalArgumentException`, `IllegalMonitorStateException`,  
`IllegalPathStateException`, `IllegalStateException`,  
`ImageOpException`, `IndexOutOfBoundsException`, `MissingResourceException`,  
`NegativeArraySizeException`, `NoSuchElementException`, `NullPointerException`,  
`ProfileDataException`, `ProviderException`, `RasterFormatException`, `SecurityException`,  
`SystemException`, `UndeclaredThrowableException`, `UnmodifiableSetException`,  
`UnsupportedOperationException`

第十八，`error` 和 `exception` 有什么区别？

`error` 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。

`exception` 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。

第十九，`List`, `Set`, `Map` 是否继承自 `Collection` 接口？

`List`, `Set` 是

`Map` 不是

第二十，`abstract class` 和 `interface` 有什么区别？

声明方法的存在而不去实现它的类被叫做抽象类 (**abstract class**)，它用于要创建一个体现某些基本行为的类，并为该类声明方法，但不能在该类中实现该方法的情况。不能创建 **abstract** 类的实例。然而可以创建一个变量，其类型是一个抽象类，并让它指向具体子类的一个实例。不能有抽象构造函数或抽象静态方法。**Abstract** 类的子类为它们父类中的所有抽象方法提供实现，否则它们也是抽象类为。取而代之，在子类中实现该方法。知道其行为的其它类可以在类中实现这些方法。

接口 (**interface**) 是抽象类的变体。在接口中，所有方法都是抽象的。多继承性可通过实现这样的接口而获得。接口中的所有方法都是抽象的，没有一个有程序体。接口只可以定义 **static final** 成员变量。接口的实现与子类相似，除了该实现类不能从接口定义中继承行为。当类实现特殊接口时，它定义 (即将程序体给予) 所有这种接口的方法。然后，它可以在实现了该接口的类的任何对象上调用接口的方法。由于有抽象类，它允许使用接口名作为引用变量的类型。通常的动态联编将生效。引用可以转换到接口类型或从接口类型转换，**instanceof** 运算符可以用来决定某对象的类是否实现了接口。

第二十一，**abstract** 的 **method** 是否可同时是 **static**，是否可同时是 **native**，是否可同时是 **synchronized**？  
都不能

第二十二，接口是否可继承接口？抽象类是否可实现(**implements**)接口？抽象类是否可继承实体类(**concrete class**)？

接口可以继承接口。抽象类可以实现(**implements**)接口，抽象类是否可继承实体类，但前提是实体类必须有明确的构造函数。

第二十三，启动一个线程是用 **run()** 还是 **start()**？

启动一个线程是调用 **start()** 方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由 **JVM** 调度并执行。这并不意味着线程就会立即运行。**run()** 方法可以产生必须退出的标志来停止一个线程。

第二十四，构造器 **Constructor** 是否可被 **override**？

构造器 **Constructor** 不能被继承，因此不能重写 **Overriding**，但可以被重载 **Overloading**。

第二十五，是否可以继承 **String** 类？

**String** 类是 **final** 类故不可以继承。

第二十六，当一个线程进入一个对象的一个 **synchronized** 方法后，其它线程是否可进入此对象的其它方法？

不能，一个对象的一个 **synchronized** 方法只能由一个线程访问。

第二十七，**try {}** 里有一个 **return** 语句，那么紧跟在这个 **try** 后的 **finally {}** 里的 **code** 会不会被执行，什么时候被执行，在 **return** 前还是后？

会执行，在 **return** 前执行。

# 面试交流网

## http://www.msjl.net

第二十八，编程题：用最有效率的方法算出 2 乘以 8 等於几?  
有 C 背景的程序员特别喜欢问这种问题。

$2 \ll 3$

第二十九，两个对象值相同(`x.equals(y) == true`)，但却可有不同的 hash code，这句话对不对?

不对，有相同的 hash code。

第三十，当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?

是值传递。Java 编程语言只由值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对对象的引用是永远不会改变的。

第三十一，switch 是否能作用在 byte 上，是否能作用在 long 上，是否能作用在 String 上?

switch (expr1) 中，expr1 是一个整数表达式。因此传递给 switch 和 case 语句的参数应该是 int、short、char 或者 byte。long,string 都不能作用于 switch。

第三十二，编程题：写一个 Singleton 出来。

Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。一般 Singleton 模式通常有几种形式:

第一种形式: 定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例化，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
public class Singleton {
    private Singleton(){}
    //在自己内部定义自己一个实例，是不是很奇怪?
    //注意这是 private 只供内部调用
    private static Singleton instance = new Singleton();
    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
    public static Singleton getInstance() {
        return instance;
    }
}
```

第二种形式:

```
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance() {
        //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
        //使用时生成实例，提高了效率!
        if (instance==null)
```

```
        instance=new Singleton();  
return instance;    }  
}
```

其他形式:

定义一个类，它的构造函数为 **private** 的，所有方法为 **static** 的。

一般认为第一种形式要更加安全些