Java类谜题53：按你的意愿行事 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022_Java_E7_B1_BB_E8_B0_9C_c97_138481.htm 现在该轮到你写一些代码了。假设你有一个称为Thing的库类，它唯一的构造器将接受一个int参数： public class Thing { public Thing(int i) { ... } ... } Thing实例没有提供任何可以获取其构造器参数的值的途径。因为Thing是一个库类，所以你不具有访问其内部的权限，因此你不能修改它。 假设你想编写一个称为MyThing的子类，其构造器将通过调用SomeOtherClass.func()方法来计算超类构造器的参数。这个方法返回的值被一个个的调用以不可预知的方式所修改。最后，假设你想将这个曾经传递给超类构造器的值存储到子类的一个final实例域中，以供将来使用。那么下面就是你自然会写出的代码： public class MyThing extends Thing { private final int arg. public MyThing() { super(arg = SomeOtherClass.func()). ... } } 遗憾的是，这个程序是非法的。如果你尝试着去编译它，那么你将得到一条像下面这样的错误消息： MyThing.java: can’t reference arg before supertype constructor has been called super(arg = SomeOtherClass.func()). ^ 你怎样才能重写MyThing以实现想要的效果呢？MyThing()构造器必须是线程安全的：多个线程可能会并发地调用它。 这个解决方案内在地就是线程安全的和优雅的，它涉及对MyThing中第二个私有的构造器的运用： public class MyThing extends Thing { private final int arg. public MyThing() { this(SomeOtherClass.func()). } private MyThing(int i) { super(i). arg = i. } } 这个解决方案使用了交替构造器调用机制（alternate

constructor invocation）[JLS 8.8.7.1]。这个特征允许一个类中的某个构造器链接调用同一个类中的另一个构造器。在本例中，MyThing()链接调用了私有构造器MyThing(int)，它执行了所需的实例初始化。在这个私有构造器中，表达式SomeOtherClass.func()的值已经被捕获到了变量i中，并且它可以在超类构造器返回之后存储到final类型的域param中。 通过本谜题所展示的私有构造器捕获（Private Constructor Capture）惯用法是一种非常有用的模式，你应该把它添加到你的技巧库中。我们已经看到了某些真的是很丑陋的代码，它们本来是可以通过使用本模式而避免如此丑陋的。