关注性能：异常的异常 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022__E5_85_B3_E6_B3_A8_E6_80_A7_E8_c104_252375.htm 编码的艰难决择：应该这样做还是那样做？性能讨论组中充斥着类似于这样的问题“ 我应该像大多数人那样编写代码，还是为了得到更好的性能那样编写代码？ ”一般专家会建议应该避免早期的优化，并且直到性能测试显示需要优化的时候才使用最优方法，但实际情况是我们每写一行代码都在做出会影响到性能的决定。JavaRanch 上的一项讨论调查了确保类型安全的两种选择，一种是抛出异常，另一种是用 instanceof，并提出了“ 哪种方法更好 ”的问题。清单 1 和 2 显示了这两种方法。清单 1. 使用 instanceof 来分支 Listing 1: using instanceof to branch for (int i = 0. i { Object obj = myVector.elementAt(i). if (obj instanceof MySpecialClass) { // do this } }清单 2. 抛出异常来分支 for (int i = 0. i try { MySpecialClass myClass = (MySpecialClass)myVector.elementAt(i). // do this } catch (ClassCastException cce) { continue. // for loop } }提这种问题的危险之一是，当您想把问题浓缩成一个简单例子时可能会失去很多上下文。没有充分的上下文通常会把讨论弄得长而混乱，因为每一个阅读了问题的回答者都会用他们自己的上下文来联系问题。所有这种额外的上下文都会增添含义，这会把我们从问题的出发点转移出来。头脑里有了这些东西之后，就让我们来看能否从这一思路中找到的消息线索中筛选出某些真理来。异常的特征提起异常大多数开发者首先要说的就是它们很昂贵。如果您继续追问为什么它们很昂贵，最普

遍的答案是我们需要捕获异常堆栈的当前状态。尽管这是开销的很大一部分，但通过列出异常的一些特征，我们可以知道这只是故事的开始。下面是异常的一些特征：☆ 可以被抛出。 ☆ 可以被捕获。 ☆ 可以被程序化地创建。 ☆ 可以被 JVM 创建。 ☆ 被表示为第一级对象。 ☆ 继承的深度从 3 开始。 ☆ 由 String（和来自 1.4 的 StackTraceElements）组成。 ☆ 依靠本机方法 fillInStackTrace()。 异常与其他对象的主要区别是异常可以被抛出和捕获。让我们从调查当异常被抛出时所触发的事件过程来开始我们的研究。 