

## 题目：Java 反射机制的用途和缺点

### 1. 反射的用途 Uses of Reflection

Reflection is commonly used by programs which require the ability to examine or modify the runtime behavior of applications running in the Java virtual machine. This is a relatively advanced feature and should be used only by developers who have a strong grasp of the fundamentals of the language. With that caveat in mind, reflection is a powerful technique and can enable applications to perform operations which would otherwise be impossible.

反射被广泛地用于那些需要在运行时检测或修改程序行为的程序中。这是一个相对高级的特性，只有那些语言基础非常扎实的开发人员才应该使用它。如果能把这句警示时刻放在心里，那么反射机制就会成为一项强大的技术，可以让应用程序做一些几乎不可能做到的事情。

### 2. 反射的缺点 Drawbacks of Reflection

Reflection is powerful, but should not be used indiscriminately. If it is possible to perform an operation without using reflection, then it is preferable to avoid using it. The following concerns should be kept in mind when accessing code via reflection.

尽管反射非常强大，但也不能滥用。如果一个功能可以不用反射完成，那么最好就不用。在我们使用反射技术时，下面几条内容应该牢记于心：

#### 1.1. 性能第一 Performance Overhead

Because reflection involves types that are dynamically resolved, certain Java virtual machine optimizations can not be performed. Consequently, reflective operations have slower performance than their non-reflective counterparts, and should be avoided in sections of code which are called frequently in performance-sensitive applications.

反射包括了一些动态类型，所以 JVM 无法对这些代码进行优化。因此，反射操作的效率要比那些非反射操作低得多。我们应该避免在经常被执行的代码或对性能要求很高的程序中使用反射。

#### 1.2. 安全限制 Security Restrictions

Reflection requires a runtime permission which may not be present when running under a

security manager. This is an important consideration for code which has to run in a restricted security context, such as in an Applet.

使用反射技术要求程序必须在一个没有安全限制的环境中运行。如果一个程序必须在有安全限制的环境中运行，如 Applet，那么这就是个问题。

### 1.3. 内部暴露 Exposure of Internals

Since reflection allows code to perform operations that would be illegal in non-reflective code, such as accessing private fields and methods, the use of reflection can result in unexpected side-effects, which may render code dysfunctional and may destroy portability. Reflective code breaks abstractions and therefore may change behavior with upgrades of the platform.

由于反射允许代码执行一些在正常情况下不被允许的操作（比如访问私有的属性和方法），所以使用反射可能会导致意料之外的副作用——代码有功能上的错误，降低可移植性。反射代码破坏了抽象性，因此当平台发生改变的时候，代码的行为就有可能也随着变化。