避免在Java中使用CheckedException PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/490/2021_2022__E9_81_BF_E5_85_8D_E5_9C_A8J_c67_490701.htm 这篇文章指出了Java中checked Exception的一些缺点，提出应该在程序设计中避免使用checked Exception，对于需要处理checked Exception的代码，可以使用ExceptionAdapter这个类对checked Exception进行包装。这篇文章的概念和ExceptionAdapter这个类均源自Bruce Eckel的Does Java need Checked Exception. Java的Exception分为两类，一类是RuntimeException及其子类，另外一类就是checked Exception.Java要求函数对没有被catch处理掉的checked Exception，需要将其写在函数的声明部分。然而，这一要求常常给程序员带来一些不必要的负担。 为了避免在函数声明中写throws部分，在Java项目里面常常可以看到以下代码用来‘ 吞掉 ’Exception：try {// ...} catch (Exception ex) {ex.printStackTrace().} 这显然不是一个好的处理Exception办法，事实上，catch并处理一个Exception意味着让程序从发生的错误（Exception）中恢复过来。从这种意义上说，已上的代码只可能在一些很简单的情况下工作而不带来问题。 对于很多Exception，往往没有去处理它并让程序从错误中恢复出来的办法，这时唯一能做的事情可能就是在界面上显示一些提示信息给用户。这种情况下让程序抛出遇到的Exception是更为合理的做法。然而，这样做会使得一些函数的声明急剧膨胀。一个函数可能需要声明会抛出的7、8个checked Exception，而且每个调用它的函数也需要同样的声明。 比这更糟糕的是，这有可能破坏类设计的open-close原则。简单来说

，open-close原则是指当扩展一个模块的时候，可以不影响其现有的client.open-close原则是通过继承来实现的，当继承一个类的时候，我们既扩展了这个类，也不会影响原有的client（因为对这个类没有改动）。 现在考虑下面这种情况，有一个父类Base：public class Base {public void foo() throws ExceptionA {// ...}} 现在需要继承Base这个类并重载foo这个方法，在新的实现中，foo可能抛出ExceptionB：public class Extend extends Base {public void foo() throws ExceptionB {// ...}} 然而，这样写在Java里面是不合法的，因为Java把可能会抛出的Exception看作函数特征的一部分，子类声明抛出的Exception必须是父类的子集。 可以在Base类的foo方法中加入抛出ExceptionB的声明，然而，这样就破坏了open-close原则。而且，有时我们没有办法去修改父类，比如当重载一个Jdk里的类的时候。 另一个可能的做法是在Extend的foo方法中catch住ExceptionB，然后构造一个ExceptionA并抛出。这是个可行的办法但也只是一个权宜之计。 如果使用RuntimeException，这些问题都不会存在。这说明checked Exception并不是一个很实用的概念，也意味着在程序设计的时候，我们应该让自己的Exception类继承RuntimeException而不是Exception.（这和JDK的建议正好相反，但实践证明这样做代码的质量更好。） 对于那些需要处理checked Exception的代码，可以利用一个ExceptionAdapter的类把checked Exception包装成一个RuntimeException抛出。ExceptionAdapter来自Bruce Eckel的Does Java need Checked Exception这篇文章，在这里的ExceptionAdapter是我根据JDK 1.4修改过的：public class ExceptionAdapter extends RuntimeException {public

ExceptionAdapter(Exception ex) {super(ex).} public void printStackTrace(java.io.PrintStream s) {getCause().printStackTrace(s).}public void printStackTrace(java.io.PrintWriter s) {getCause().printStackTrace(s).}// rethrow()的作用是把被包装的Exception再次抛出。public void rethrow()throws Exception{throw (Exception) getCause().}} 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com