应用技巧:Java中的异常处理Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/560/2021_2022__E5_BA_94_E7_94_A8_E6_8A_80_E5_c104_560156.htm 异常处理是初学者经常忽视执行的编程技巧。然而，当他们完成了一项大型项目后，就会发现仅仅停留在表面的工作是不够的。在本文中，我们将对异常处理进行讨论，并为大家解释其重要性，尤其是要告诉大家怎样处理这些情况。 首先，让我们解释一下什么是异常情况，通常也称例外。正如在其他的编程语言中那样，它也适用于Java：异常情况是那些在运行时出现错误的情况。这些错误并非真正的错误，因为他们是一些例外。我们可以将这些情况理解为必须解决的异常事件，否则程序将无法继续执行。所以我们就有必要了解怎样处理异常事件。在异常事件的例子中最显著的应该是在程序执行时，运行时的分配变成了零。这样程序就无法执行，于是Java就会抛出一个异常事件，确切点说是ArithmeticException。从Java程序员的角度来看，异常事件是对象。抛出异常事件类似于抛出对象。但是，并非所有的对象都可以被抛出。 为了充分理解可抛出的异常事件，整个类层次结构的一些部分要被提交。主要的类称为Throwable。这个类拥有两个子类：Exception和Error。一个异常事件对象应从Throwable的类中传出。意味着它应该是Exception子类或Error子类的一个对象实例。这些都可以在 java.lang数据包中找到。 异常处理就是捕捉可能在运行时被抛出的异常事件的一项技术。Java通过try-catch-finally的异常处理语句为我们提供了强大的异常处理解决方案。而在另一方面，你也可以使用已经声明的异常

事件，如ArithmeticException, NullPointerException等。其他类扩展了Exception 类，如IOException子类。 此外，我们应该注意到异常事件包含了两种情况：检查过的和没检查的。技术上，我们认为没检查过的异常事件RuntimeExceptions。这些不需要在抛出的语句中作出声明，而且对它们的捕捉也是选择性的。不过，它们一般不会有什么影响，如果程序员根本不能发现它们的存在。在大多数情况下，这些都是逻辑性的编程错误，如NullPointerException或者ArrayIndexOutOfBounds。 同时，对异常事件进行技术性检查也迫使程序员对其进行处理和管理，意味着要对其进行单独捕捉并覆盖。这些都来自Exceptions类和它的子类，包括我们之前讨论过的RuntimeExceptions。检查过的异常事件要求异常事件处理因为它们有可能导致程序终止。 现在，我们对异常事件有了个基本的了解，下面就让我们启动集成开发环境开始编码吧!异常处理 前面我们提到了异常处理就是指处理代码中的异常事件，或者在运行时向运行引擎抛出异常事件，在引擎末端它会搜索异常事件处理例程。它使用包含了一系列方法调用的调用堆栈进行搜索。 一般而言，异常事件可能因为包含一个异常活动或其他异步异常导致的。我们讨论的异常事件包括了一些基本的处理议题：怎样捕捉和处理这些异常事件。Java允许我们创建自己的Exception对象和类，但是会有一个关键的请求。这些对象和类必须是扩展的Exception类。编码标准要求异常事件应该充分命名，意味着它们的名字就代表了其本身。 throw new Exception(“ This is an exception! ”) 下面，我们看看要怎样捕捉和处理这些异常事件。检查以下代码：try{ // this is the block of code where the exception happens //

sometimes called as source/root of exception // or even called as tricky block or tricky method } catch{Exception_Typel e) { // dealing with this kind of exception } Catch (Exception_Type2 e) { // dealing witn this kind of exception } //... unlimited number of catches are possible finally { // this block of code is always executed } try-catch-finally语句的第一个部分是尝试阻止。这是异常事件有可能发生的部分。通常，我们建议代码行用最小的数量来编写，因为它们只会在异常事件发生的时候执行。这种情况发生时，执行会跳转去捕捉那些异常事件被比较的块中。如果它们匹配，那么就可以处理异常事件。 不论尝试阻止的时候，异常事件会不会发生，或不管能不能得到处理，阻止总会执行。由于它总是被执行，所以我们推荐你在这里做一些清理。因此，正如所预料的那样，执行起来就是具有选择性的。 Try-catch模块的结构类似于switch-case的结构。在检查过的需要处理的异常事件中，是有可能在相同方法中将其处理或者抛出的。后者可以通过关键词抛出。在这种情况下，异常事件的种类必须在方法签名中被指定。看这个例子：Void myMethod () throws SomeKindOfException{ // method goes here } 接下来，我们将为大家展示更多的异常处理实例。 初学者常常与非匹配数据类型纠缠不清。通常，它们会引发一些问题，例如，在做加法时出现非数字型代码。下面给大家展示的代码中，出现了异常处理的工作环境。检查该网页以完成嵌入式Exception种类的清单。现在，我们要处理NumberFormatException 的发生。 public static void main (String args[]) { double sum= 0. for (int i=0. iargs. length. 1) try { sum = Double.parseDboule (args[i]). } Catch

(NumberFormatException e) { Ststem.out.printIn(args[i] “ non-numeric data on ”). } System.out.printIn(“ Total sum: “ sum). } 正如你所见到的，它和命令行参数一起运行，而且一旦轮到非数字型参数，它就会写入system.out，意指出现的问题。但是项目会继续进行，因为try模块是循环的。否则，没有合适的异常处理，项目就会终止。用这种方式总和还是可以计算处理并在最后显示处理。 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