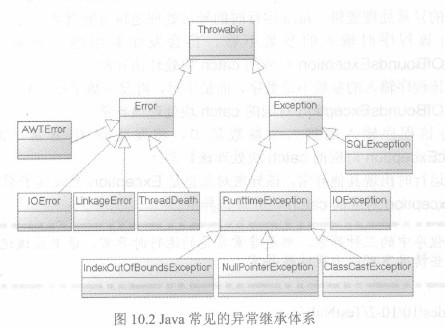
[**异常（try...catch...finally、throws、throw）**](http://www.cnblogs.com/wyl-study/archive/2012/02/14/2351308.html)



当出现程序无法控制的外部环境（文件不存在，文件内容损坏，网络不可用等）问题时，java就会用异常对象来描述。

Java中用2种方法处理异常：

1、 在发生异常的地方直接处理。

2、 将异常抛给调用者，让调用者处理。

异常的分类：

1、 检查性异常：java.lang.Exception 程序正确，但因为外在的环境条件不足引发

2、 运行期异常：java.lang.RuntimeException 这意味着程序出现bug，如数组越界，除以0等，这类异常需要更改程序来避免

3、 错误：java.lang.Error 可能源于程序的bug，但更可能源于环境问题，如内存耗尽、系统崩溃、动态链接失败等，这种错误无法恢复或不可能捕获，会导致程序中断，错误在程序中无序处理，而由运行环境处理。

 注意：进行异常捕获时所有父类异常的catch块都应该排在子类异常catch块的后面（先处理小异常，再处理大异常）。

访问异常信息：如果程序需要在catch块中访问对象的相关信息，可以通过调用catch后异常形参的方法来获取

1、 getMessage():返回异常的详细描述字符串。

2、 printStackTrace():将异常的跟踪栈信息输出到标准错误输出。

3、 printStackTrace(PrintStream s):将该异常的跟踪栈信息输出到指定输出流。

4、 getStackTrace():返回该异常的跟踪栈信息。

利用finally回收资源：

    程序在try里打开了一些物理资源（如数据库连接、网络连接等），这些物理资源必须显示回收。因为java的垃圾回收机制不会回收任何物理资源，只能回收堆内存中对象所占用的内存。为了保证一定能回收try中打开的物理资源，异常处理机制提供了finally块，不管是否出现异常，finally块总会被执行。异常处理语法结构：只有try块是必须的，catch和finally是可选的，但至少出现其中之一，catch可有多个。

以下情形，finally将不会被执行

1、 finally块中发生异常

2、 程序所在线程死亡

3、 在前面的代码中用了System.exit()

4、 关闭cpu

注意：一旦finally中使用了return或throw语句，将会导致try块、catch块中的return、throw语句失效。

（当java程序执行try块、catch块时遇到了return或throw语句，这两个语句会导致方法立即结束，所以系统并不会立即执行者两个语句，而是去寻找是否有finally块，如果没有，程序立即执行return或throw语句，方法终止；如果有finally，系统立即执行finall块，只有当finally块执行完成后系统才跳回来执行try、Catch块里的return、throw，如果finally里也使用了return或throw等导致方法终止的语句，则finally块已经终止了方法，系统将不会跳回执行try、catch块里的任何代码。）

使用throws声明抛出异常：throws只能在方法签名中使用，可以声明抛出多个异常类，以逗号隔开。

使用throws的情况：

1、当前方法不知道应该如何处理这种类型的异常。

2、该异常应该由上一级调用者处理。

抛出的异常将讲给JVM处理。处理方法是：打印异常跟踪栈信息并终止程序运行，这就是程序一道异常后自动结束的原因。

注意：使用throws抛出异常时有一个限制，就是方法重写时：子类方法中声明抛出的异常类型应该是父类抛出异常类型的子类或相等。

使用Checked异常至少存在两大不便之处：

1、对于程序中Checked异常，java要求必须显式的捕获并处理该异常或显式声明抛出，这就爱增加了编程的复杂度。

2、如果在方法中显式声明抛出Checked异常麻将会导致方法签名与异常耦合，如果方法是重写父类的方法，则该方法能抛出的异常还受被重写方法所抛出的异常限制。

使用throw抛出异常：

java允许程序自行抛出异常，由throw来完成。其抛出的不是异常类，而是一个异常实例，每次只能抛出一个异常实例。

如果throw抛出的异常时checked异常，则该throw语句要么处于try块里，显式捕获该异常，要么放在一个带有throws声明的方法中；如果throw语句抛出的异常时runtime异常，则该语句无须放在try块里或带throws抛出的方法中，程序可以显式使用try...catch来捕获，并处理异常，也可以完全不理会该异常，把该异常交给该方法调用者处理。

自定义异常类：需要继承Exception基类，如希望自定义Runtime异常则需继承RuntimeException基类，定义异常时要提供两种构造器，无参的和带一个字符串的，这字符串作为该异常对象的详细说明，即getMessaga方法的返回值。

总结：

1、处理异常，对异常采用合适的修补，然后绕过异常发生的地方继续执行；或者用别的数据进行计算以代替期望的方法返回值；或者提示用户重新操作等，总之，对于checked异常，程序应该尽量采用修复。

2、重新抛出异常，把当前运行环境下能做的事情尽量做完，然后进行异常转译，把异常包装成当前层的异常，重新抛出给上层调用者。

3、在合适的层处理异常。如果当前层不清楚如何处理异常，就不要在当前层使用catch来捕获异常，直接使用throws抛出异常，让上层调用者来负责处理异常。