# java异常

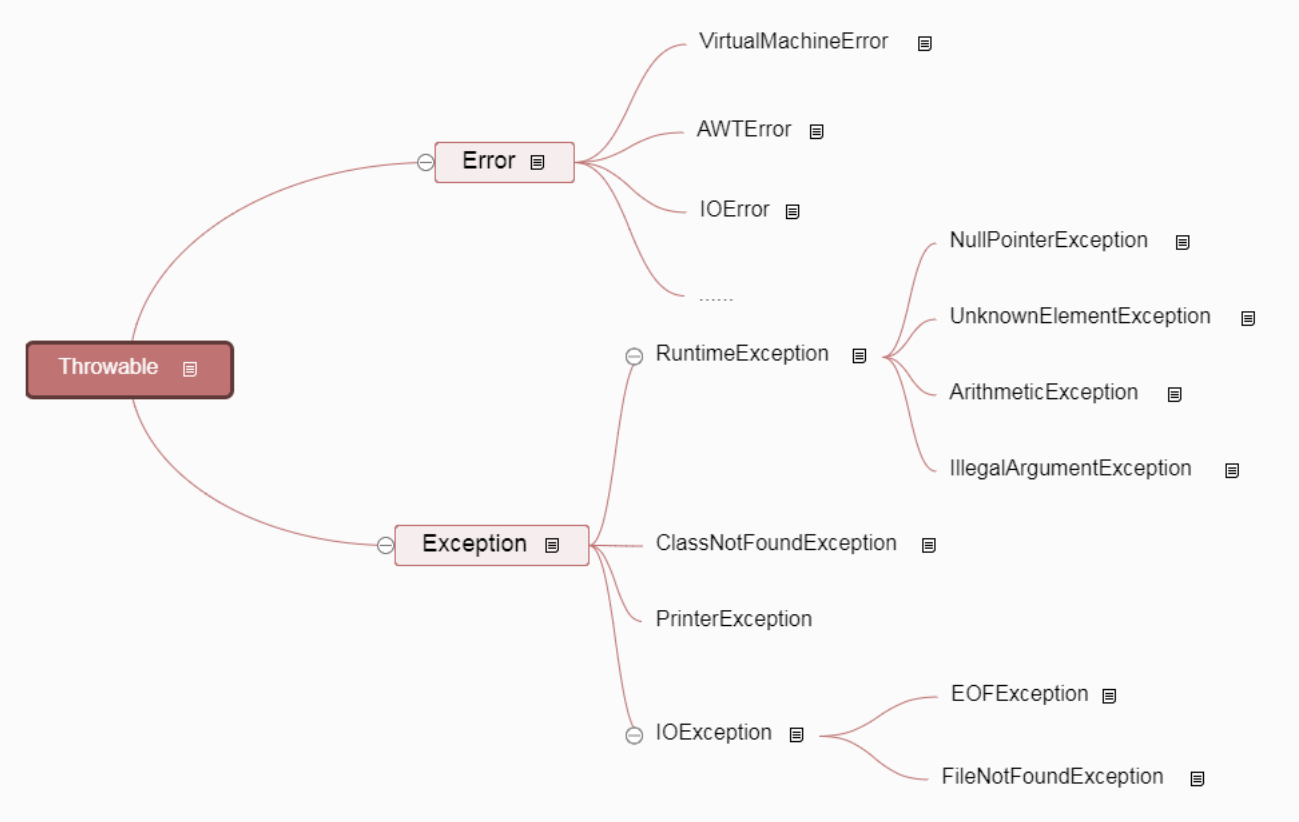
java异常

## 一、定义

**异常**指不期而至的各种状况，如：文件找不到、网络连接失败、非法参数等。异常是一个事件，它发生在程序运行期间，干扰了正常的指令流程。Java通 过API中Throwable类的众多子类描述各种不同的异常。因而，*Java异常都是对象*，是Throwable子类的实例，描述了出现在一段编码中的 错误条件。当条件生成时，错误将引发异常。

## 二、分类

Java通 过API中Throwable类的众多子类描述各种不同的异常。




不可查异常： 不可查异常(编译器不要求强制处置的异常):包括运行时异常（RuntimeException与其子类）和错误（Error）。在程序中可以不抛出和捕获。

可查异常：所有继承自Exception并且不是RuntimeException的异常都是checked Exception，如ClassNotFoundException。JAVA 语言规定必须对checked Exception作处理，编译器会对此作

检查，要么在方法体中声明抛出checked Exception,要么使用catch语句捕获checked Exception进行处理，不然不能通过编译。

## 三、异常处理机制

***3.1抛出异常***：当一个方法出现错误引发异常时，方法创建异常对象并交付运行时系统，异常对象中包含了异常类型和异常出现时的程序状态等异常信息。运行时系统负责寻找处置异常的代码并执行。

3.1.1throws和throw抛出：

throw关键字是用于方法体内部，用来抛出一个Throwable类型的异常。如果抛出了检查异常，则还应该在方法头部声明方法可能抛出的异常类型。该 方法的调用者也必须检查处理抛出的异常。如果所有方法都层层上抛获取的异常，最终JVM会进行处理，处理也很简单，就是打印异常消息和堆栈信息。如果抛出 的是Error或RuntimeException，则该方法的调用者可选择处理该异常。

throws关键字用于方法体外部的方法声明部分，用来声明方法可能会抛出某些异常。仅当抛出了检查异常，该方法的调用者才必须处理或者重新抛出该异常。 当方法的调用者无力处理该异常的时候，应该继续抛出。

(com.mx.test2.throws)

3.1.2异常关系链

在实际开发过程中经常在捕获一个异常之后抛出另外一个异常，并且我们希望在新的异常对象中保存原始异常对象的信息，实际上就是异常传递，即把底层的异常对象传给上层，一级一级，逐层抛出。当程序捕获了一个底层的异常，而在catch处理异常的时候选择将该异常抛给上层…这样异常的原因就会逐层传递，形成一个由低到高的异常链。但是

异常链在实际应用中一般不建议使用，同时异常链每次都需要就将原始的异常对象封装为新的异常对象，消耗大量资源。现在（jdk 1.4之后）所有的Throwable的子类构造中都可以接受一个cause对象，这个cause也就是原始的异常对象。(com.mx.test6)

firstThrow()产生MyException对象->异常冒泡至调用其的secondThrow()->异常冒泡至调用secondThrow()的TestExceptionChain的构造方法->冒泡至printtry的main()方法。注意到：异常对象一直被抛出，直至在printtry的mian()方法中被try-catch捕获！

3.1.3异常转译

异常转译就是将一种类型的异常转成另一种类型的异常，然后再抛出异常。

通常而言，更为合理的转换方式是：com.mx.test5

1、 Error——>Exception

2、 Error——>RuntimeException

3、 Exception——>RuntimeException,

***3.2捕获异常***：在方法抛出异常之后，运行时系统将转为寻找合适的异常处理器（exception handler）。潜在的异常处理器是异常发生时依次存留在调用栈中的方法的集合。当异常处理器所能处理的异常类型与方法抛出的异常类型相符时，即为合适 的异常处理器。运行时系统从发生异常的方法开始，依次回查调用栈中的方法，直至找到含有合适异常处理器的方法并执行。当运行时系统遍历调用栈而未找到合适 的异常处理器，则运行时系统终止。同时，意味着Java程序的终止。

***3.2.1捕获异常***：try、catch和finally

1.***try、catch语句***:关键词try后的一对大括号将一块可能发生异常的代码包起来，称为监控区域。Java方法在运行过程中出现异常，则创建异常对象。将异常抛出监控区域之 外，由Java运行时系统试图寻找匹配的catch子句以捕获异常。若有匹配的catch子句，则运行其异常处理代码，try-catch语句结束。

匹配的原则是：如果抛出的异常对象属于catch子句的异常类，或者属于该异常类的子类，则认为生成的异常对象与catch块捕获的异常类型相匹配。

1.1.捕获throw语句抛出的异常。com.mx.test Test1

1.2.捕获系统运行时自动抛出的异常。com.mx.test Test2

1.3.不捕获也不声明抛出异常。com.mx.test Test3

注意：一旦某个catch捕获到匹配的异常类型，将进入异常处理代码。一经处理结束，就意味着整个try-catch语句结束。其他的catch子句不再有匹配和捕获异常类型的机会。com.mx.test Test5

2.***try-catch-finally***： try-catch语句还可以包括第三部分，就是finally子句。它表示无论是否出现异常，都应当执行的内容。com.mx.test Test4

在以下4种特殊情况下，finally块不会被执行：

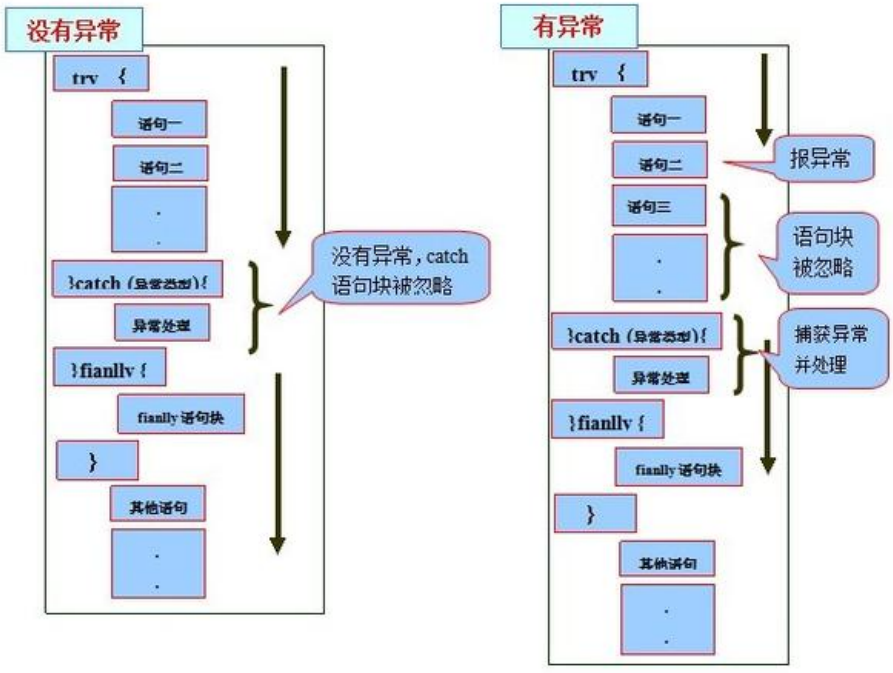
1）在finally语句块中发生了异常。

2）在前面的代码中用了System.exit()退出程序。

3）程序所在的线程死亡。

4）关闭CPU。

3执行顺序：



***3.3自定义异常***：步骤 com.mx.test2 MyExcception Throw

（1）创建自定义异常类。

（2）在方法中通过throw关键字抛出异常对象。

（3）如果在当前抛出异常的方法中处理异常，可以使用try-catch语句捕获并处理；否则在方法的声明处通过throws关键字指明要抛出给方法调用者的异常，继续进行下一步操作。

（4）在出现异常方法的调用者中捕获并处理异常。

***3.4Throwable类中的常用方法***：

注意：catch关键字后面括号中的Exception类型的参数e。Exception就是try代码块传递给catch代码块的变量类型，e就是变量名。catch代码块中语句"e.getMessage();"用于输出错误性质。通常异常处理常用3个函数来获取异常的有关信息:

getCause()：返回抛出异常的原因。如果 cause 不存在或未知，则返回 null。

getMessage()：返回异常的消息信息。

printStackTrace()：对象的堆栈跟踪输出至错误输出流，作为字段 System.err 的值。com.mx.test4 Test2

关于异常，在继承中还要注意两点：

一个方法被覆盖时，覆盖它的方法必须扔出相同的异常或异常的子类。

如果父类抛出多个异常，那么重写（覆盖）方法必须扔出那些异常的一个子集,也就是说，不能扔出新的异常。

com.mx.test3 Test1

# 四、处理异常原则

4.1 ***不要忽略checked Exception***(捕获一个checked Exception的时候，必须对异常进行处理；如果认为不必要在这里作处理，就不要捕获该异常，在方法体中声明方法抛出异常，由上层调用者来处理该异常。)

4.2 ***不要捕获unchecked Exception***(没有恰当的处理方式。)

4.3 ***不要一次捕获所有的异常***(需要不同的处理和恢复措施。com.mx.test4 Test3)

4.4 ***提早抛出***(明确。com.mx.test4.Test1)

4.5 ***不要让try块过于庞大***(影响效率。)