# 第五章 异常处理

## 5.1 什么是异常

程序运行时，发生的不被期望的事件，它阻止了程序按照程序员的预期正常执行，这就是异常。发生时，是任程序自生自灭，立刻退出终止。在Java中即，编译或运行或者运行过程中出现的错误。

Java提供了优秀的解决办法：异常处理机制。

异常处理机制能让程序发生异常时，按照预先设定的异常处理逻辑，针对性地处理异常，让程序尽最大可能恢复正常并继续执行。

## 5.2 常用关键字

* + try：将可能会发生异常的代码放入其中，则会监听这些代码，当try代码块内发生异常时，异常就被抛出或捕捉。
  + catch：用于捕获try代码块中发生的异常。
  + finally：finally代码块总是会被执行。它主要用于回收try块里使用的资源（IO读写，网络连接，数据库连接）。**Finally代码块执行完成之后，才会回来执行try或catch中return或throw语句，如果finally中使用了return或者throw等终止方法的语句，则不会跳回执行，而是直接停止。**
  + throw：用于抛出异常。
  + throws：用在方法头中，用于声明该方法可能抛出异常。主方法上也可以使用throws抛出异常，表示主方法里面可以不用强制性的进行异常的逻辑处理，如果出现了异常就交给JVM默认处理，则此时会导致程序中断执行。

## 5.3异常的类型

检查性异常：继承自Exception的是检查性异常，在编译期对异常的处理有强制性的要求，包括IOException，SQLException等等。不处理这些异常，会出现红色波浪线，编译不通过。当函数中存在抛出检查型异常的操作时该函数的函数声明中必须包含throws语句。调用改函数的函数也必须对该异常进行处理，如不进行处理则必须在调用函数上声明throws语句。

运行时异常：继承自RuntimeException或Error是运行时异常。对异常可以不需要强制处理，常见的有NullPointException,ClassCastException。可以不使用try…catch进行处理，但是如果有异常产生，则异常将由JVM默认处理。对于RuntimeException的子类最好也使用异常处理机制。虽然RuntimeException的异常可以不使用try...catch进行处理，但是如果一旦发生异常，则肯定会导致程序中断执行，所以，为了保证程序再出错后依然可以执行，在开发代码时最好使用try...catch的异常处理机制进行处理。

Error不是Exception，Exception是程序正常运行过程中可以预料到的意外情况，并且应该被开发者捕获处理；而Error是Java程序运行中不可预料的异常情况，这种情况不可能捕获到，比如OutOfMemoryError，NoClassDefFoundError等，由于这是属于JVM层的严重错误，导致JVM无法继续执行。

## 5.4如下是常见的Error和Exception

运行时异常（RuntimeException）也称为（非检查性异常UncheckedException）：

1. NullPointException：空指针异常
2. ClassCastException：类型强制转换异常
3. IllegalArgumentException：传递非法参数异常
4. IndexOutOfBoundsException：下标越界异常
5. NumberFormatException：数字格式异常
6. ArrayIndexOutOfBoundsException：数组越界异常
7. ArrayStoreException：数据存储异常，操作数组时类型不一致
8. ArithmeticException：算术异常
9. BufferOverflowException：缓冲区溢出异常

非运行时异常（CheckedException）也称检查性异常：

ClassNotFoundException：找不到指定class的异常

IOException：IO异常

FileNotFoundException：文件不存在异常

SQLException：SQL语句异常

InterruptedException：中断异常-调用线程睡眠的时候

错误（Error）：

NoClassDefFoundError：找不到class定义异常

StackOverflowError：深递归导致栈被耗尽而抛出的异常

OutofMemoryError：内存溢出异常

## 5.5 Error和Exception使用场景

NoClassDefFoundError和ClassNotFoundException的区别？

NoClassDefFoundError是Error，ClassNotFoundException和Exception。

NoClassDefFoundError它是JVM运行是通过加载类的时候，找不到对应的类而抛出的错误；ClassNotFoundException是在编译过程中如果可能出现此异常，在编译过程中必须将ClassNotFoundException异常抛出！

## 5.6 常见问题

**Error和Exception的区别：**

1. Error（错误）是系统中的错误，程序员是不能改变的和处理的，是在程序编译时出现的错误，只能通过修改程序才能修正。一般是指与虚拟机相关的问题，如系统崩溃，虚拟机错误，内存空间不足，方法调用栈溢等。对于这类错误的导致的应用程序中断，仅靠程序本身无法恢复和和预防，遇到这样的错误，建议让程序终止。
2. Exception（异常）表示程序可以处理的异常，可以捕获且可能恢复。遇到这类异常，应该尽可能处理异常，使程序恢复运行，而不应该随意终止异常。

**在catch捕获异常时，为什么不考虑使用Throwable类型，而只是使用Exception来进行接收？**

Throwable表示的范围要比Exception大。实际上程序使用Throwable来进行处理，没有任何语法问题，但是却会存在逻辑问题。因为此时出现的（或者说用户能够处理的）只有Exception类型，而如果使用Throwable接收，还会表示可以处理Error的错误，而用户是处理不了Error错误的，所以在开发中用户可以处理的异常都要求以Exception类为主。

**异常是一起处理好还是分开处理好？**

**根据实际的开发要求是否严格来决定。**在实际的项目开发项目工作中，所有的异常是统一使用Exception处理还是分开处理，完全根据开发者的项目开发标准来决定。如果项目开发环境严谨，基本上要求针对每一种异常分别进行处理，并且要详细记录下异常产生的时间以及产生的位置，这样可以方便程序维护人员进行代码的维护。**再次注意：处理多个异常时，捕获范围小的异常要放在捕获范围大的异常之前处理。**

**throw和throws的区别？**

* throw：指的是在方法中人为抛出一个异常对象（这个异常对象可能是自己实例化或者抛出已存在的）；
* throws：在方法的声明上使用，表示此方法在调用时必须处理异常。

**检查型异常（Checked Exception）与非检查型异常（Unchecked Exception）区别？**

* 所有的检查性异常都继承自java.lang.Exception；所有的非检查性异常都继承自java.lang.RuntimeException。
* 检查性异常和非检查性异常最主要的区别在于其处理异常的方式：检查性异常必须使用try catch或者throws等关键字进行处理，否则编译器会报错;非检查性异常一般是程序代码写的不够严谨而导致的问题，可以通过修改代码来规避。
* 常见的运行时异常：空指针异常(NullPointerException)、除零异常(ArithmeticException)、数组越界异常(ArrayIndexOutOfBoundsException)等;
* 常见的检查性异常：输入输出异常(IOException)、文件不存在异常(FileNotFoundException)、SQL语句异常(SQLException)等。

**assert关键字**

assert关键字语法很简单，有两种用法：

1. assert <boolean表达式>  
   如果<boolean表达式>为true，则程序继续执行。  
   如果为false，则程序抛出AssertionError，并终止执行。
2. assert <boolean表达式> : <错误信息表达式>  
   如果<boolean表达式>为true，则程序继续执行。  
   如果为false，则程序抛出java.lang.AssertionError，并输入<错误信息表达式>。

**finally块和return**

* 首先一个不容易理解的事实：在 try块中即便有return，break，continue等改变执行流的语句，finally也会执行。
* finally中的return 会覆盖 try 或者catch中的返回值。
* finally中的return或异常会抑制（消灭）前面try或者catch块中的异常。