Java：详解Java中的异常(Error与Exception)

置顶 王小二(IT) 2018-06-22 16:54:26 76642 收藏 574

分类专栏： 编程语言(Java基础) 文章标签： try catch finally Error

版权

《Java：详解Java中的异常(Error与Exception)》

《Java：简述Java中的自定义异常》

《Java：简述throw-throws异常抛出》

《Java：简述try-catch-finally异常捕获》

《Java：简述try-catch-finally中return返回》

一、 异常机制的概述

      异常机制是指当程序出现错误后，程序如何处理。具体来说，异常机制提供了程序退出的安全通道。当出现错误后，程序执行的流程发生改变，程序的控制权转移到异常处理器。

      程序错误分为三种：1.编译错误；2.运行时错误；3.逻辑错误。

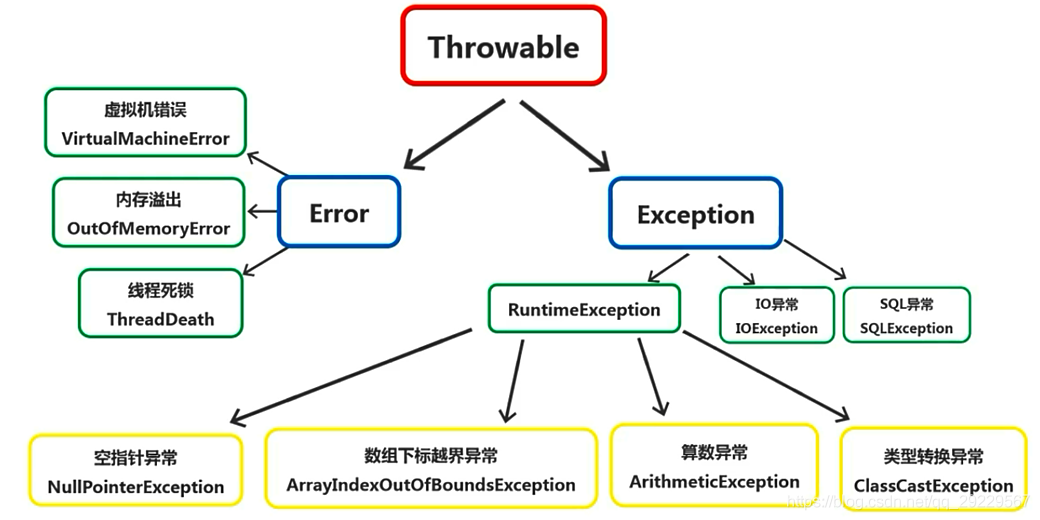
      （1）编译错误是因为程序没有遵循语法规则，编译程序能够自己发现并且提示我们错误的原因和位置，这个也是大家在刚接触编程语言最常遇到的问题。

      （2）运行时错误是因为程序在执行时，运行环境发现了不能执行的操作。

      （3）逻辑错误是因为程序没有按照预期的逻辑顺序执行。异常也就是指程序运行时发生错误，而异常处理就是对这些错误进行处理和控制。

二、 异常的结构

       在 Java 中，所有的异常都有一个共同的祖先 Throwable（可抛出）。Throwable 指定代码中可用异常传播机制通过 Java 应用程序传输的任何问题的共性。



       Throwable： 有两个重要的子类：Exception（异常）和 Error（错误），二者都是 Java 异常处理的重要子类，各自都包含大量子类。异常和错误的区别是：异常能被程序本身可以处理，错误是无法处理。

      Throwable类中常用方法如下：

1. 返回异常发生时的详细信息

public string getMessage();

2. 返回异常发生时的简要描述

public string toString();

3. 返回异常对象的本地化信息。使用Throwable的子类覆盖这个方法，可以声称本地化信息。如果子类没有覆盖该方法，则该方法返回的信息与getMessage（）返回的结果相同

public string getLocalizedMessage();

4. 在控制台上打印Throwable对象封装的异常信息

public void printStackTrace();

       Error（错误）:是程序无法处理的错误，表示运行应用程序中较严重问题。大多数错误与代码编写者执行的操作无关，而表示代码运行时 JVM（Java 虚拟机）出现的问题。例如，Java虚拟机运行错误（Virtual MachineError），当 JVM 不再有继续执行操作所需的内存资源时，将出现 OutOfMemoryError。这些异常发生时，Java虚拟机（JVM）一般会选择线程终止。这些错误表示故障发生于虚拟机自身、或者发生在虚拟机试图执行应用时，如Java虚拟机运行错误（Virtual MachineError）、类定义错误（NoClassDefFoundError）等。这些错误是不可查的，因为它们在应用程序的控制和处理能力之 外，而且绝大多数是程序运行时不允许出现的状况。对于设计合理的应用程序来说，即使确实发生了错误，本质上也不应该试图去处理它所引起的异常状况。在 Java中，错误通过Error的子类描述。

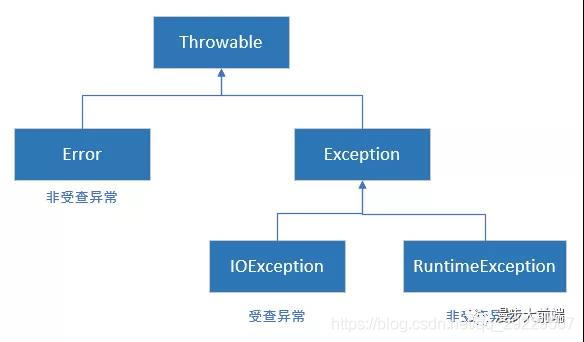
      Exception（异常）:是程序本身可以处理的异常。Exception 类有一个重要的子类 RuntimeException。RuntimeException 类及其子类表示“JVM 常用操作”引发的错误。例如，若试图使用空值对象引用、除数为零或数组越界，则分别引发运行时异常（NullPointerException、ArithmeticException）和 ArrayIndexOutOfBoundException。

      Exception（异常）分两大类：运行时异常和非运行时异常(编译异常)。程序中应当尽可能去处理这些异常。

      1.运行时异常：都是RuntimeException类及其子类异常，如NullPointerException(空指针异常)、IndexOutOfBoundsException(下标越界异常)等，这些异常是不检查异常，程序中可以选择捕获处理，也可以不处理。这些异常一般是由程序逻辑错误引起的，程序应该从逻辑角度尽可能避免这类异常的发生。运行时异常的特点是Java编译器不会检查它，也就是说，当程序中可能出现这类异常，即使没有用try-catch语句捕获它，也没有用throws子句声明抛出它，也会编译通过。

      2.非运行时异常 （编译异常）：是RuntimeException以外的异常，类型上都属于Exception类及其子类。从程序语法角度讲是必须进行处理的异常，如果不处理，程序就不能编译通过。如IOException、SQLException等以及用户自定义的Exception异常，一般情况下不自定义检查异常。

       通常，Java的异常（Throwable）分为可查的异常（checked exceptions）和不可查的异常（unchecked exceptions）。



       1. 可查异常（编译器要求必须处置的异常）：正确的程序在运行中，很容易出现的、情理可容的异常状况。除了Exception中的RuntimeException及RuntimeException的子类以外，其他的Exception类及其子类(例如：IOException和ClassNotFoundException)都属于可查异常。这种异常的特点是Java编译器会检查它，也就是说，当程序中可能出现这类异常，要么用try-catch语句捕获它，要么用throws子句声明抛出它，否则编译不会通过。

      2. 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）。RuntimeException表示编译器不会检查程序是否对RuntimeException作了处理，在程序中不必捕获RuntimException类型的异常，也不必在方法体声明抛出RuntimeException类。RuntimeException发生的时候，表示程序中出现了编程错误，所以应该找出错误修改程序，而不是去捕获RuntimeException。

三、 异常处理的机制

      在 Java 应用程序中，异常处理机制为：抛出异常，捕捉异常。

      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。详细信息请查看《简述throw-throws异常抛出》。

      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。详细信息请查看《Java：简述try-catch-finally异常捕获》。

      对于错误、运行时异常、可查异常，Java技术所要求的异常处理方式有所不同。

      1. 错误：对于方法运行中可能出现的Error，当运行方法不欲捕捉时，Java允许该方法不做任何抛出声明。因为，大多数Error异常属于永远不能被允许发生的状况，也属于合理的应用程序不该捕捉的异常。

      2. 运行时异常：由于运行时异常的不可查性，为了更合理、更容易地实现应用程序，Java规定，运行时异常将由Java运行时系统自动抛出，允许应用程序忽略运行时异常。

      3. 可查异常：对于所有的可查异常，Java规定：一个方法必须捕捉，或者声明抛出方法之外。也就是说，当一个方法选择不捕捉可查异常时，它必须声明将抛出异常。

        能够捕捉异常的方法，需要提供相符类型的异常处理器。所捕捉的异常，可能是由于自身语句所引发并抛出的异常，也可能是由某个调用的方法或者Java运行时 系统等抛出的异常。也就是说，一个方法所能捕捉的异常，一定是Java代码在某处所抛出的异常。简单地说，异常总是先被抛出，后被捕捉的。

        异常抛出：任何Java代码都可以抛出异常，如：自己编写的代码、来自Java开发环境包中代码，或者Java运行时系统。无论是谁，都可以通过Java的throw语句抛出异常。从方法中抛出的任何异常都必须使用throws子句。

        异常捕获：捕捉异常通过try-catch语句或者try-catch-finally语句实现。

        总体来说，Java规定：对于可查异常必须捕捉、或者声明抛出。允许忽略不可查的RuntimeException和Error。

三、Java常见异常

      1. RuntimeException子类:

序号 异常名称 异常描述

1 java.lang.ArrayIndexOutOfBoundsException 数组索引越界异常。当对数组的索引值为负数或大于等于数组大小时抛出。

2 java.lang.ArithmeticException  算术条件异常。譬如：整数除零等。

3 java.lang.SecurityException  安全性异常

4 java.lang.IllegalArgumentException 非法参数异常

5 java.lang.ArrayStoreException  数组中包含不兼容的值抛出的异常

6 java.lang.NegativeArraySizeException 数组长度为负异常

7 java.lang.NullPointerException 空指针异常。当应用试图在要求使用对象的地方使用了null时，抛出该异常。譬如：调用null对象的实例方法、访问null对象的属性、计算null对象的长度、使用throw语句抛出null等等。

      2.IOException

序号 异常名称 异常描述

1 IOException 操作输入流和输出流时可能出现的异常

2 EOFException 文件已结束异常

3 FileNotFoundException 文件未找到异常

      3. 其他

序号 异常名称 异常描述

1 ClassCastException 类型转换异常类

2 ArrayStoreException 数组中包含不兼容的值抛出的异常

3 SQLException 操作数据库异常类

4 NoSuchFieldException 字段未找到异常

5 NoSuchMethodException 方法未找到抛出的异常

6 NumberFormatException 字符串转换为数字抛出的异常

7 StringIndexOutOfBoundsException 字符串索引超出范围抛出的异常

8 IllegalAccessException 不允许访问某类异常

9 InstantiationException

 当应用程序试图使用Class类中的newInstance()方法创建

一个类的实例，而指定的类对象无法被实例化时，抛出该异常

10 java.lang.ClassNotFoundException 找不到类异常。当应用试图根据字符串形式的类名构造类，而在遍历CLASSPAH之后找不到对应名称的class文件时，抛出该异常。

四、相关的问题

      1. 为什么要创建自己的异常？

      答：当Java内置的异常都不能明确的说明异常情况的时候，需要创建自己的异常。

      2. 应该在声明方法抛出异常还是在方法中捕获异常？

      答：捕捉并处理知道如何处理的异常，而抛出不知道如何处理的异常。

————————————————

版权声明：本文为CSDN博主「王小二(IT)」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_29229567/article/details/80773970