

Java 异常处理策略研究

陈红跃, 张宏军, 陈 刚

(解放军理工大学 工程兵工程学院, 江苏 南京 210007)

摘 要:异常处理机制是程序设计语言的重要标志之一,在程序设计过程中用来处理程序运行中的异常。传统的程序设计语言里异常处理较为繁杂。Java 是一种面向对象的程序设计语言,且引入了异常处理机制。合理完备的异常处理可以增强程序运行的可靠性、提高软件的健壮性,可以较为快速地确定错误的位置。文章分析了 Java 异常处理的逻辑,阐述了异常类、异常处理机制以及异常处理方法,提出了异常处理的一些策略。综合运用这些策略可以使编程人员编写出更加简洁、高效的程序代码。

关键词:Java 语言;异常处理机制;异常类;异常转译

中图分类号:TP312

文献标识码:A

文章编号:1673-629X(2012)07-0009-04

Study of Java Exception Handling Strategy

CHEN Hong-yue, ZHANG Hong-jun, CHEN Gang

(Engineering Institute of Engineering Corps, PLA University of Science & Technology, Nanjing 210007, China)

Abstract:Exception handling mechanism is an important indicator of programming languages, people use it in the program design process to deal with the exception. Exception handling in the traditional programming language is complex. Java is an object-oriented programming language, it introduces exception handling mechanism. A reasonable use of exception handling can enhance the reliability of the program and improve the software robustness and determine the location of the error more quickly. It describes the logic of Java exception handling, elaborates exception class, exception handling mechanism and the method of exception handling, proposes several strategies for exception handling. Integrated use of these strategies can make programmers write more concise and efficient code.

Key words:Java language; exception handling mechanism; exception class; exception translation

0 引言

Java 语言是 Sun Microsystems 公司开发的一种面向对象的、可移植的程序设计语言,因而其可以跨平台使用,较为广泛地应用在因特网上软件的开发^[1]。

由于 Java 程序是在网络环境中运行的,故安全性不容忽视。程序运行过程中难免会发生错误,为了能够及时有效地处理这些错误,需要在程序中加入一些对异常进行处理的代码,从而减少电脑发生死机、程序产生死循环甚至损坏操作系统的可能性,确保程序可以安全运行,故 Java 引入了异常处理机制来处理异常^[2]。

1 异常概述

“异常”是应用程序中违反 Java 语言规范的语义

限制时可能产生的可预测的、可恢复的问题^[3]。一般地,大多数异常表示轻度到中度的问题。

在实际应用中,Java 程序里是经常出现异常的,如除数为零、数组越界、文件找不到等。根据相关研究的结果,23.3% 的程序包含 try 结构,24.5% 的程序包含 throw 结构,31.7% 的程序包含 try 结构或者 throw 结构^[4]。这些数据证实了在 Java 程序中,异常出现的频率已经达到了不得不被考虑的程度。

在 Java 语言中,使用面向对象的方法来处理异常。Try 结构是标准的 Java 异常处理结构,由 try 语句、catch 语句(可选)、finally 语句(必选)组成,其可以使用一个或多个 catch 语句来处理程序运行时产生的异常,格式如下:

```
try
{<代码段> //引发异常的代码}
catch(exception1 e)
{<代码段> //捕获异常 exception1}
catch(exception2 e)
{<代码段> //捕获异常 exception 2}
.....
```

收稿日期:2011-11-17;修回日期:2012-02-22

基金项目:国家自然科学基金资助项目(70971137)

作者简介:陈红跃(1986-),男,硕士研究生,CCF 会员,研究方向为作战模拟、联合作战演练系统分析与集成;张宏军,教授,博士生导师,研究方向为军事运筹、系统分析。

```

catch(exception e)
{<代码段> //捕获其它异常}

finally
{<代码段> //必须执行的代码}

采用 try 结构处理异常的例子如下:

public class Example1 {
public static void main(String args[])
{ int x=5; int y=0;
try { System.out.println(x/y); }
catch( ArithmeticException e)
{ System.out.println("除数为 0 异常" + e.getMessage()); }
finally { System.out.println("必须执行的代码"); }
}
}

```

程序运行时产生除数为零的异常,故输出“除数为 0 异常”,执行 finally 语句,又输出“必须执行的代码”。

在编写 Java 程序时,使用 try 语句块来监视可能引发异常的代码。异常产生时,程序从产生异常的代码段处跳出,Java 虚拟机查找与当前 try 块相匹配的 catch 块,找到后程序将跳到 catch 块处继续运行(当有多个 catch 块时,程序跳到第一个匹配的 catch 块处继续运行)。若没有找到与当前 try 块相匹配的 catch 块,则执行所有的 finally 块、调用当前线程所属的 uncaughtException 方法,最后终止引发异常的当前线程。依据 Java 语言规范^[5]的要求,Java 异常的处理逻辑如图 1^[6]所示,其中:

- (1) try 块中无异常产生,未定义 finally 块。
- (2) try 块中无异常产生,已定义 finally 块。
- (3) try 块中产生异常,无匹配 catch 块,未定义 finally 块。
- (4) try 块中产生异常,无匹配 catch 块,已定义 finally 块。
- (5) try 块中产生异常,有匹配 catch 块并捕捉异常。
- (6) catch 块捕捉异常,已定义 finally 块。
- (7) catch 块捕捉异常,未定义 finally 块。
- (8) 内层嵌套块有未处理的异常,有匹配 catch 块捕捉。
- (9) catch 块中产生异常,已定义 finally 块。
- (10) 内层嵌套块有未处理的异常,无匹配 catch 块,已定义 finally 块。
- (11) catch 块中产生异常,未定义 finally 块。
- (12) 内层嵌套块有未处理的异常,无匹配 catch 块,未定义 finally 块。
- (13) finally 块中无异常产生。
- (14) finally 块中产生异常。

(15) finally 块中产生异常或传递前一个异常。

函数或封闭 try 结构

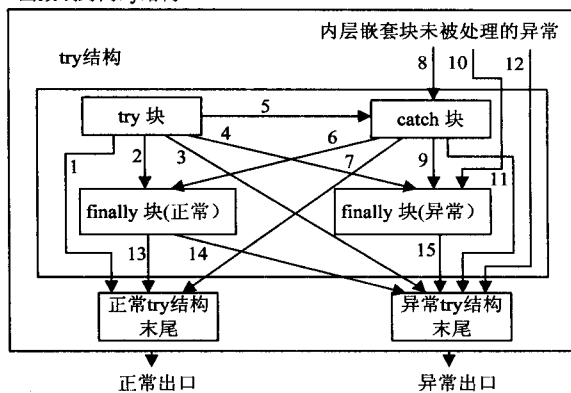


图 1 Java 异常处理的流程

2 异常类层次

Java 将异常封装到类中,这些类包含了运行程序时引发异常的相关信息及对异常进行处理的方法等。

Java 中的每个异常都是一个对象,Throwable 类是所有异常的基类。Java 的异常类都是 Throwable 类的子类,在 Throwable 类中定义了所有异常类共同需要的内容,如接收传入的字符串信息(该信息通常是对异常的描述)方法 getMessage()等。它派生出两个子类:Error(错误)和 Exception(异常)。

Error 表示应用程序中产生的由系统保留的较严重、不能恢复的问题^[7]。Java API 文档记录给出的定义是:“Throwable 的一个子类,代表严重问题,合理应用程序不应试图捕获它。大多数此类错误属反常情况。”例如,当 JVM 不再有继续执行操作所需的内存资源时,将出现 OutOfMemoryError。

Exception 表示应用程序中产生的可预测、可恢复的轻度到中度的问题。Java API 文档记录给出的定义是:“合理应用程序可能需要捕获的情况。”当 Java 内置的异常不能反应某些特定问题时,需要创建自定义的异常来说明这些问题。自定义异常必须是直接或间接派生自 Throwable 类,通常的作法是从 Exception 类直接派生,方法如下:

```
public class exceptionName extends Exception{//类成员}
```

用户需要在程序中使用 throw 关键字来抛出自定义异常对象,处理自定义异常 myException 的主要代码如下:

```

class myException extends Exception
{
myException(String msg)
{ super(msg); //调用父类构造方法 }
}

public class creatMyException
{ static void test() throws myException

```

```
throw new myException(" new exception");//抛出自定义异常  
  
public static void main(String args[])  
{  
    try  
    {test();}  
    catch(myException e)  
    {System.out.println("产生异常:" + e.getMessage());}  
}
```

程序运行结果为产生异常:new exception

Java 语言采用继承方式组织所有异常类,异常类的继承图如图 2 所示:

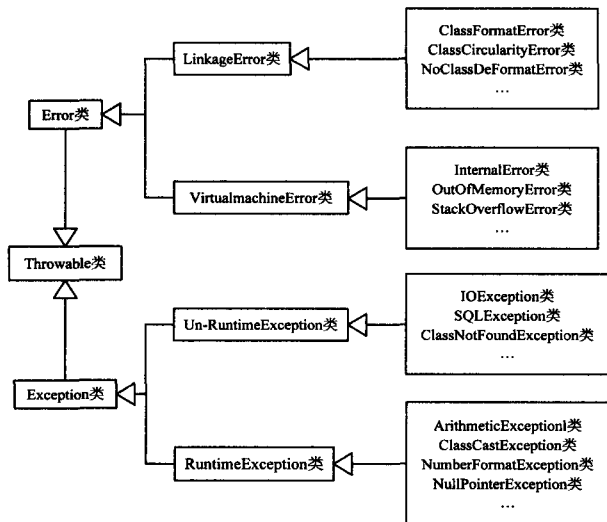


图 2 Java 异常类继承图

图 2 中异常类的含义如表 1、表 2 所示:

表 1 Error 类异常

Error 类	异常类名称	异常类含义
LinkageError (结合错误)	ClassFormatError	类格式错误
	ClassCircularityError	无限循环错误
	NoClassDeFormatError	无类定义错误
VirtualmachineError (虚拟机错误)	InternalError	内部错误
	OutOfMemoryError	内存不足错误
	StackOverflowError	堆栈溢出错误

表 2 Exception 类异常

Exception 类	异常类名称	异常类含义
Un-RuntimeException (非运行时异常)	IOException	输入输出类异常
	SQLException	数据库操作类异常
	ClassNotFoundException	未找到相应类异常
RuntimeException (运行时异常)	ArithmeticException	算术运算类异常
	ClassCastException	类型强制转换类异常
	NumberFormatException	字符串转换为数字类异常
	NullPointerException	引用不存在的对象类异常

3 异常机制

Java 通过异常类及其子类对编程中出现的异常提供了一个标准的、简便的抛出和处理异常的机制,该机制通过面向对象的方法进行异常处理,通过该机制开发人员可以发现程序中出现的异常,以便及时修正。Java 异常处理机制分为三个步骤^[8]:

(1)抛出异常:运行应用程序时,若发生错误,则生成一个相应的异常类并抛出到运行时系统。

(2)捕获异常:抛出异常后,系统会查找与其相匹配的语句块去捕获该异常。

(3)处理异常:找到匹配的异常处理语句块处理异常,如若未找到,则终止程序的运行,最后对异常进行缺省处理。

Java 的异常机制通过 try、catch、finally、throw 和 throws 5 个关键字来处理异常,关键字的作用如表 3:

表 3 关键字的作用

关键字	作用
try	存放可能产生异常的代码
catch	捕获可能出现的某种异常
finally	程序运行结束前,始终执行的代码
throw	将异常抛出
throws	列出所有可能的异常

4 异常处理策略

应用程序中产生异常时,其可以自行跳转至相应的异常处理模块中进行一些尝试性修复处理,使得程序能够正常运行或降级运行或安全地终止,以提高应用系统的可靠性^[9]。分析数据表明,对异常的不当处理会引起系统崩溃^[10]。以下是异常处理的一些策略。

4.1 尽可能处理异常

在条件允许的情况下尽可能去处理产生的异常,如果条件达不到,导致异常无法处理,则声明异常。对异常声明时,必须将其添加到相应方法块的结束位置,即输入部分之后。以下是声明异常的示例:

```
public void errorProneMethod ( int input ) throws java. io.
SQLException {
    // Code for the method, including one or more method
    // calls that may produce an SQLException
}
```

4.2 异常不能影响对象的状态

在异常处理中,异常不能影响对象的状态^[11]。不可变对象发生异常后其状态不会变,可变对象必须在编程中保证发生异常后不会影响到其状态。在编程时,可以通过三种方法保证异常不会影响到对象状态:

(1)将可能引发异常的程序代码和“正常”代码分离。

(2) 如果异常代码不易分离, 调用定义好的 `recover` 方法来修复被改变的类变量。

(3) 拷贝对象时, 如果产生异常, 不会影响对象本身。

4.3 对异常分层次处理

对异常进行处理时, 要对其进行层次的区分^[12], 不能够把特殊情况下的异常放到更高层次的异常中去处理^[13]。顶层系统对一些底层具体的异常不做处理, 需要进行异常转译。异常转译就是将继承 `Throwable` 类的一种子异常类转换为一种新的异常类^[14], 例如:

```
public class ResourceLoader {
    public getResource(String name) throws ResourceLoadException {
        try { // try to load the resource from the database {
            catch (SQLException e) {
                throw new ResourceLoadException(e.toString());
            }
        }
    }
}
```

`getResource` 方法捕获了 `SQLException`, 并将其转换为 `ResourceLoadException`。

合理的转译关系图应该如图 3 所示:

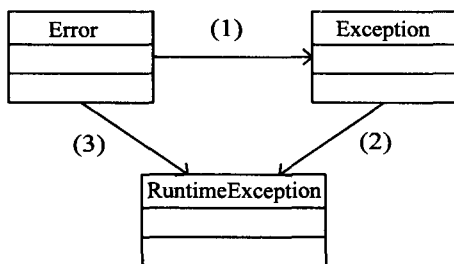


图 3 转译关系图

图 3 中描述的异常转译有以下三种:

(1) `Error` 转译为 `Exception`: 错误转译为异常并抛出, 可以减小因错误造成的不良影响。

(2) `Exception` 转译为 `RuntimeException`: 异常转译为运行时异常, 可以简化程序代码。

(3) `Error` 转译为 `RuntimeException`: 错误转译为运行时异常, 便于程序统一处理错误和异常信息。

4.4 遵循 try-catch-finally 规则

· 只有在可能产生异常或代码无法预料时才使用 `try` 语句。

- `try` 块后至少有一个 `catch` 或 `finally` 块。
- 遵守 `try-catch-finally` 块的顺序。
- 有多个 `catch` 块时, 执行第一个匹配的 `catch` 块。
- 充分利用 `finally` 块来关闭并释放不能被垃圾回收器自动回收的资源。
- 尽量精简 `try` 块。

4.5 指定具体的异常类型

`catch` 语句表示可能出现某种异常, 并去捕获该异常。异常类的作用是告诉 Java 编译器处理哪一种异

常, 由于绝大多数异常都直接或间接派生自 `Throwable` 类, 人们试图用一个 `catch` 语句去捕获所有的异常, 这是不太可能实现的。故在 `catch` 语句中尽量指定具体的异常类型。

5 结束语

Java 语言其自身的异常处理机制很强大, 通过 `Throwable` 类接口进行异常处理, 用到的异常类均为 `Throwable` 类的子类。Java 采用继承方式对异常进行了分类, 使得我们可以方便、灵活地使用异常。

在 Java 程序设计过程中, 合理利用异常处理机制、适时进行异常转译、及时处理产生的异常, 可以增强程序的可读性、优雅性, 提高程序的质量。这里对 Java 异常处理的方法和流程做了一些研究, 并对异常处理时的一些注意事项做了较详细论述, 希望对程序设计人员有所帮助。

参考文献:

- [1] 魏晓玲, 杨占海. Java 语言的异常处理机制[J]. 科技信息, 2009(2): 504-505.
- [2] 张洪波, 高伟. Java 的异常处理机制[J]. 唐山师范学院学报, 2008, 30(2): 77-78.
- [3] 卢丹, 小林良岳, 中山健, 等. Java 字节码异常处理中信息流的分析[J]. 应用科技, 2007, 34(2): 28-29.
- [4] Sinhas. Analysis of programs with exception-handling constructs [C]//Proc of International Conference of Software Maintenance. Maryland, USA: [s. n.], 1998.
- [5] James G, Bill J, Guy S, et al. The Java Language Specification [M]. 3rd ed. [s. l.]: Addison Wesley, 2006.
- [6] Saurabh S, Jean M H. Analysis and Testing of Programs with Exception-handling Constructs [J]. IEEE Transactions on Software Engineering, 2000, 26(9): 849-871.
- [7] 王伟胜, 陈志刚. Java 异常处理机制研究[J]. 电脑与信息技术, 2008, 16(1): 31-32.
- [8] 阳小兰, 钱程. 基于 Java 的异常处理技术与应用[J]. 软件导刊, 2009, 8(11): 69-70.
- [9] 王新雨, 须文波, 柴志雷. Java 虚拟机中异常机制实时性的研究及实现[J]. 计算机工程与应用, 2008, 44(34): 84-85.
- [10] Perry D E. Current trends in exception handling [J]. IEEE Trans on Software Engineering, 2000, 26(9): 817-819.
- [11] 赵化冰, 唐英, 唐文彬, 等. Java 异常处理[J]. 计算机应用, 2003, 23(12): 349-350.
- [12] 黄艳峰. 基于 Java 的异常处理机制研究[J]. 河南科学, 2009, 27(10): 1268-1269.
- [13] 杨厚群, 陈静. Java 异常处理机制的研究[J]. 计算机科学, 2007, 34(3): 286-289.
- [14] 孙卫琴. Java 面向对象编程[M]. 北京: 电子工业出版社, 2007.