

http://blog.cnblogs.com/shangrid256/article-list/dist http://blog.cnblogs.com/xiejiaoyi/article-list/dist
http://www.cnblogs.com/new0801/ http://www.cnblogs.com/hehe520/ http://www.cnblogs.com/1294/ 随笔 - 1294, 文章 - 0, 评论 - 0, 引用 - 0
http://www.cnblogs.com/muyuge/ http://www.cnblogs.com/nainange/ http://www.cnblogs.com/aiwz/
http://www.cnblogs.com/hehe001/ http://www.cnblogs.com/xiaomaohai/ http://www.cnblogs.com/daniaoge/
http://www.cnblogs.com/wuwuwei/ http://www.cnblogs.com/yueguo/ http://www.cnblogs.com/zhangyunlin/
http://www.cnblogs.com/wangfengju/ http://www.cnblogs.com/hulanshan/ http://www.cnblogs.com/fengju/
http://www.cnblogs.com/sesexoo/ http://www.cnblogs.com/wuwa/ http://www.cnblogs.com/cl1024cl/
http://www.cnblogs.com/wuyida/ http://www.cnblogs.com/49360/ http://www.cnblogs.com/10360/ 异常类的多项选择
http://www.cnblogs.com/wanghang/ http://www.cnblogs.com/feizhuliu/ http://www.cnblogs.com/xiaowangba/
http://www.cnblogs.com/nanrenzhuazhi/ http://www.cnblogs.com/caibi/ http://www.cnblogs.com/heiwa/
http://www.cnblogs.com/aiyuxi/ http://www.cnblogs.com/caibi/ http://www.cnblogs.com/heiwa/
http://www.cnblogs.com/aixiaomei/ http://www.cnblogs.com/haoxiaoz/ http://www.cnblogs.com/xiaomm/
http://www.cnblogs.com/liang123/ http://www.cnblogs.com/xianrou/ http://www.cnblogs.com/nonghu/
http://www.cnblogs.com/dajiji/

2017年8月

日	一	二	三	四	五	六
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

公告

昵称: 海南一哥
园龄: 8个月
粉丝: 1
关注: 0
+加关注

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

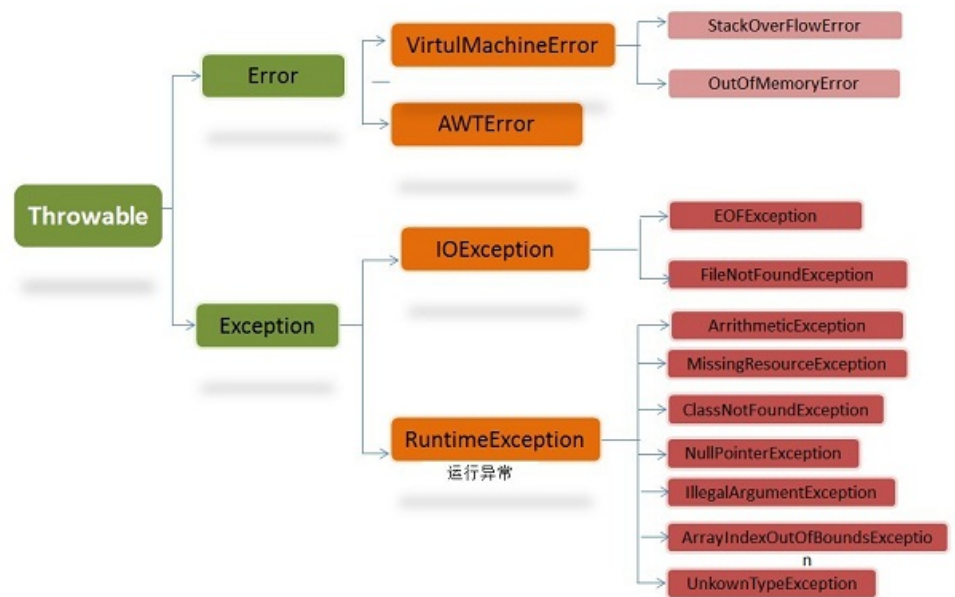
随笔档案

- 2017年1月 (2)
- 2016年12月 (2)
- 2016年11月 (54)
- 2016年10月 (15)
- 2016年9月 (17)
- 2016年8月 (19)
- 2016年7月 (70)
- 2016年6月 (38)
- 2016年5月 (10)
- 2016年4月 (19)
- 2016年3月 (59)
- 2016年2月 (3)
- 2015年12月 (2)
- 2015年11月 (2)
- 2015年10月 (1)
- 2015年9月 (2)
- 2015年8月 (8)
- 2015年7月 (15)
- 2015年6月 (11)
- 2015年5月 (8)
- 2015年4月 (3)
- 2015年2月 (19)
- 2015年1月 (12)

java异常体系结构详解

在参加网易49360的在线考试的时候,看到一道关于java中异常类的多项选择题,这几天翻看了相关书籍和网上的资料,结合自己的理解与思考,将自己的一些收获记录下来。

先来看看java中异常的体系结构图解:



首先说明一点, java中的Exception类的子类不仅仅是像上图所示只包含IOException和RuntimeException这两大类,事实上Exception的子类很多很多,主要可概括为: 运行时异常与非运行时异常。

一java异常体系结构

从上述图示可以看到,

Throwable类(表示可抛出)是所有异常和错误的超类,两个直接子类为Error和Exception,分别表示错误和异常。其中异常类Exception又分为运行时异常(RuntimeException)和非运行时异常,这两种异常有很大的区别,也称之为不检查异常(Unchecked Exception)和检查异常(Checked Exception)。下面将详细讲述这些异常之间的区别与联系:

1、Error与Exception

Error是程序无法处理的错误,它是由JVM产生和抛出的,比如OutOfMemoryError、ThreadDeath等。这些异常发生时,Java虚拟机(JVM)一般会选择线程终止。

Exception是程序本身可以处理的异常,这种异常分两大类运行时异常和非运行时异常。程序中应当尽可能去处理这些异常。

2、运行时异常和非运行时异常

运行时异常都是RuntimeException类及其子类异常,如NullPointerException、IndexOutOfBoundsException等,这些异常是不检查异常,程序中可以选择不捕获处理,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。

2014年12月 (2)
2014年11月 (4)
2014年10月 (21)
2014年9月 (2)
2014年8月 (4)
2014年7月 (5)
2014年6月 (5)
2014年5月 (3)
2014年4月 (15)
2014年3月 (12)
2014年2月 (9)
2014年1月 (13)
2013年12月 (6)
2013年11月 (3)
2013年10月 (3)
2013年9月 (2)
2013年8月 (1)
2013年7月 (3)
2013年6月 (9)
2013年5月 (29)
2013年4月 (9)
2013年3月 (8)
2013年2月 (2)
2013年1月 (6)
2012年12月 (6)
2012年11月 (9)
2012年10月 (6)
2012年9月 (14)
2012年8月 (2)
2012年7月 (4)
2012年6月 (13)
2012年5月 (3)
2012年4月 (6)
2012年3月 (6)
2012年2月 (2)
2012年1月 (2)
2011年12月 (4)
2011年11月 (4)
2011年10月 (6)
2011年9月 (12)
2011年8月 (9)
2011年7月 (15)
2011年6月 (11)
2011年5月 (14)
2011年4月 (22)
2011年3月 (14)
2011年2月 (10)
2011年1月 (10)
2010年12月 (6)
2010年11月 (4)
2010年10月 (5)
2010年9月 (6)
2010年8月 (6)
2010年7月 (4)
2010年6月 (2)
2010年5月 (3)
2010年4月 (9)
2010年3月 (10)
2010年2月 (10)
2010年1月 (11)
2009年12月 (5)
2009年11月 (10)
2009年10月 (14)
2009年9月 (16)
2009年8月 (24)
2009年7月 (28)
2009年6月 (28)
2009年5月 (28)
2009年4月 (23)
2009年3月 (27)
2009年2月 (3)
2009年1月 (5)
2008年10月 (4)

非运行时异常是`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。

二异常的捕获和处理

异常处理的步骤：

`throw try catch finally throws`下面是在网络通信中运用`socket`的一段代码：

```
1. <span style="white-space:pre">                </span>try {  
2.     mSocket=new Socket(ip,port);  
3.     if(mSocket!=null)  
4.     {  
5.         Log.i("Client","socket is create");  
6.         clientInput=new ClientInputThread(mSocket);  
7.         clientOutput=new ClientOutputThread(mSocket)  
8.     ;  
9.         clientInput.setStart(true);  
10.        clientOutput.setStart(true);  
11.        clientInput.start();  
12.        clientOutput.start();  
13.    }  
14.    }  
15.    else {  
16.        Log.i("Client","socket is not create");  
17.        // Toast.makeText( "亲，服务器端连接出  
18.        错",0).show();  
19.    }  
20.    } catch (UnknownHostException e) {  
21.        // TODO Auto-generated catch block  
22.        e.printStackTrace();  
23.    } catch (IOException e) {  
24.        // TODO Auto-generated catch block  
25.        e.printStackTrace();  
26.    }finally{}
```

从上述代码可以看到异常处理的步骤为

```
try  
{  
    可能出现异常的代码块  
}  
catch (ExceptionName1 e)  
{  
    当产生ExceptionName1 异常时的处理措施  
}  
catch (ExceptionName2 e)  
{  
    当产生ExceptionName2 异常时的处理措施  
}  
.....  
finally  
{  
    无论是否捕捉到异常都必须处理的代码  
}
```

2、 `try`、`catch`、`finally`三个语句块应注意的问题

第一： `try`、`catch`、`finally`三个语句块均不能单独使用，三者可以组成

2008年9月 (1)
2008年7月 (9)
2008年6月 (5)
2008年5月 (5)
2008年4月 (2)
2008年3月 (11)
2008年2月 (5)
2008年1月 (17)
2007年12月 (1)
2007年11月 (5)
2007年10月 (14)
2007年9月 (33)
2007年8月 (27)
2007年7月 (10)
2007年6月 (12)
2007年5月 (12)
2007年4月 (7)
2007年2月 (2)
2006年9月 (2)
2006年8月 (1)
2006年7月 (5)
2006年6月 (5)
2006年5月 (7)
2006年4月 (6)
2006年3月 (10)
2006年2月 (3)
2006年1月 (3)
2005年12月 (1)
2005年11月 (2)
2005年10月 (1)
2005年6月 (1)
2005年5月 (1)
2005年3月 (2)
2005年1月 (1)
2004年12月 (3)

阅读排行榜

1. java异常体系结构详解 (1368)
2. 基于寄存器与基于栈的虚拟机(376)
3. CSDN发表文章后老是待审核的原因(124)
4. LOG4J日志级别详解(77)
5. 【图的DFS】图的DFS非递归算法(74)

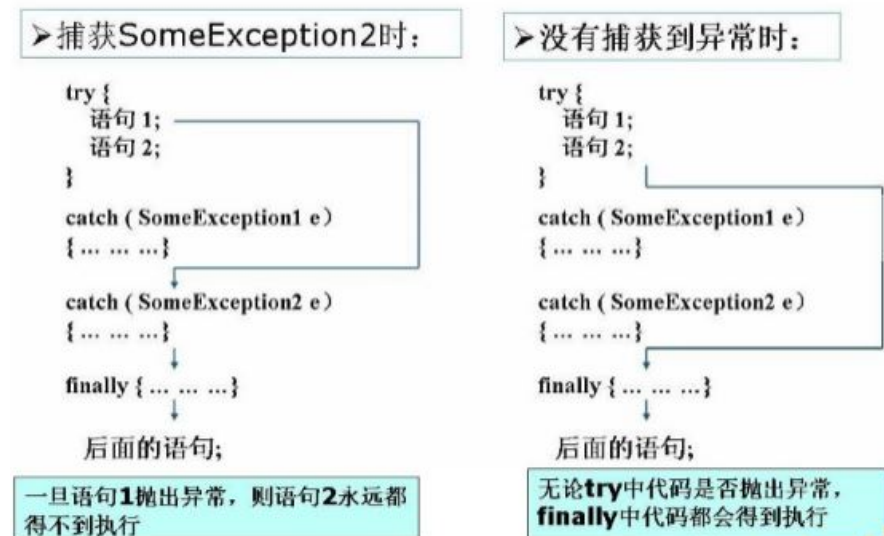
try...catch...finally、try...catch、try...finally三种结构，catch语句可以有一个或多个，finally语句最多一个。

第二：try、catch、finally三个代码块中变量的作用域为代码块内部，分别独立而不能相互访问。如果要在三个块中都可以访问，则需要将变量定义到这些块的外面。

第三：多个catch块时候，最多只会匹配其中一个异常类且只会执行该catch块代码，而不会再执行其它的catch块，且匹配catch语句的顺序为从上到下，也可能所有的catch都没执行。

第四：先Catch子类异常再Catch父类异常。

用示意图表示如下：



Finally的作用

- 无论try所指定的程序块中是否抛出异常，也无论catch语句的异常类型是否与所抛弃的异常的类型一致，finally中的代码一定会得到执行
- finally语句为异常处理提供一个统一的出口，使得在控制流程转到程序的其他部分以前，能够对程序的状态作统一的管理
- 通常在finally语句中可以进行资源的清除工作，如关闭打开的文件、删除临时文件等

3、throw、throws关键字

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。throw关键字用法如下：

```
1. public static void test() throws Exception
2. {
3.     throw new Exception("方法test中的Exception");
4. }
```

throws关键字用于方法体外部的的方法声明部分，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。

■ 假设f方法抛出了A异常，则f方法有两种方式来处理A异常

1. throws A

- 谁调用f方法，谁处理A异常，f方法本身不处理A异常

2. try{} catch () {}

- f方法本身自己来处理A异常

■ 要抛出的异常 必须得是Throwable的子类

■ throws A表示调用f方法时f方法可能会抛出A类异常，建议您调用f方法时最好对f方法可能抛出的A类异常进行捕捉

■ throws A 不表示f方法一定会抛出A类异常 throws A，f方法也可以不抛出A类异常

■ throws A 不表示调用f方法时，必须的对A异常进行捕捉

- 假设A是RuntimeException子类异常
- 由于RuntimeException的子类异常可以处理也可以不处理，所以编译器允许你调用f方法时，对f方法抛出的RuntimeException子类异常不进行处理

■ 强烈建议你

- 对throws出的所有异常进行处理
- 如果一个方法内部已经对A异常进行了处理，则就不要再throws A

注意一个方法throws出某个异常但是该方法内部可以不throw出该异常，代码如下：

```
1. class ER extends RuntimeException
2. {
3.
4. }
5. class SomeClass
6. {
7.     public void fun()throws ER
8.     {
9.         System.out.println("AAAA");
10.
11.     }
12. }
13.
14. public class ExceptionTest {
15.
16.     public static void main(String[] args) {
17.         // TODO Auto-generated method stub
18.         SomeClass A=new SomeClass();
19.         A.fun();
20.     }
21. }
```

程序运行结果如下：AAAA。

好文要顶

关注我

收藏该文



海南一哥

关注 - 0

粉丝 - 1

+加关注

0

推荐

0

反对

« 上一篇: [TCP协议三次握手与四次挥手详解](#)

» 下一篇: [算法之大整数乘法](#)

posted on 2016-03-31 11:41 海南一哥 阅读(1367) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

发表评论

昵称:  克己、

评论内容:



提交评论

退出 订阅评论

[Ctrl+Enter快捷键提交]

最新**IT**新闻:

- 摩拜“扫码开锁”涉嫌侵权案开庭 当庭上演“解码风云”
 - 英特尔透露使用**10nm+**工艺制造的**Ice Lake**架构
 - 吴恩达又一新项目曝光! 募资**1.5**亿美元 要做**AI**风投
 - **Facebook**引入新的**UI**设计 为的就是让你上瘾
 - 靠无人机发家的大疆, 也许正悄悄入局自动驾驶
- » 更多新闻...

最新知识库文章:

- 做到这一点, 你也可以成为优秀的程序员
 - 写给立志做码农的大学生
 - 架构腐化之谜
 - 学会思考, 而不只是编程
 - 编写**Shell**脚本的最佳实践
- » 更多知识库文章...

Powered by:

博客园

Copyright © 海南一哥