墨尔本,晴

lz12366

浏览: 8337 次

性别:

来自: 济南

内部类和static 内部类 | JAVA内存溢出解析

2010-04-30

Checked Exception与Runtime Exception 的区别（转）

文章分类:Java编程

原文：http://www.jspcn.net/htmlnews/11049321929371278.html

Java里有个很重要的特色是Exception ，也就是说允许程序产生例外状况。而在学Java 的时候，我们也只知道Exception 的写法，却未必真能了解不同种类的Exception 的区别。

首先，您应该知道的是Java 提供了两种Exception 的模式，一种是执行的时候所产生的Exception (Runtime Exception)，另外一种则是受控制的Exception (Checked Exception)。

所有的Checked Exception 均从java.lang.Exception 继承而来，而Runtime Exception 则继承java.lang.RuntimeException 或java.lang.Error (实际上java.lang.RuntimeException 的上一层也是java.lang.Exception)。

当我们撰写程序的时候，我们很可能会对选择某种形式的Exception 感到困扰，到底我应该选择Runtime Exception 还是Checked Exception ？

其实，在运作上，我们可以通过Class 的Method 如何产生某个Exception以及某个程序如何处理这个被产生来的Exception 来了解它们之间的差异。
首先我们先建立一个Exception

Java代码

```
1.  public class CException extends Exception
2.  {
3.  public CException() {}
4.  public CException(String message)
5.  {
6.  super(message);
7.  }
8.  }
```

然后我们撰写一个可能产生 CException 的 Class

Java代码

```
1.  public class testException
2.  {
3.  public void method1() throws CException
4.  {
5.  throw new CException("Test Exception");
6.  }
7.
8.  public void method2(String msg)
9.  {
10. if(msg == null)
11. {
12. throw new NullPointerException("Message is null");
13. }
14. }
15.
16. public void method3() throws CException
17. {
18. method1();
19. }
```

- JVM (5)
- SSH (5)
- 深入web' (1)
- 收藏的文章 (1)
- 摘抄的精彩英文技术文章 (2)

我的相册

5
共 46 张

其他分类

- 我的收藏 (20)
- 我的论坛主题贴 (0)
- 我的所有论坛贴 (49)
- 我的精华良好贴 (0)
- 我解决的问题 (1)

最近加入圈子

存档

- 2010-09 (5)
- 2010-08 (33)
- 2010-07 (9)
- 更多存档...

评论排行榜

- Java里的堆(heap)栈(stack)和方法区(metho ...
- jsp播放flv视频
- unicode
- mysql 中判断null
- 关于数组的创建方式

RSS
Add to Google
订阅到 鲜果
[什么是RSS?]


```
20.
21.  // 以下省略
22.  // ...
23.  }
```

在这三个method 中，我们看到了method1 和method2 的程序码内都会产生Exception，但method3 的程序码中(大括号内)，并没产生Exception，但在method3 的定义中，暗示了这个method 可能产生CException。

呼叫method1() 的程序，必须将method1() 包含在try 与catch 中，如：

Java代码

```
1.  public class runtest
2.  {
3.  // ....
4.  public static void main(String argv[])
5.  {
6.  testException te = new testException();
7.  try
8.  {
9.  te.method1();
10. }
11. catch(CException ce)
12. {
13. // ....
14. }
15. }
16. // ...
17. }
```

虽然包含在try 与catch 中，并不表示这段程序码一定会收到CException，但它的用意在于提醒呼叫者，执行这个method 可能产生的意外，而使用者也必须要能针对这个意外做出相对应的处理方式。

当使用者呼叫method2() 时，并不需要使用try 和catch 将程序码包起来，因为method2 的定义中，并没有throws 任何的Exception，如：

Java代码

```
1.  public class runtest
2.  {
3.  // ....
4.  public static void main(String argv[])
5.  {
6.
7.  testException te = new testException();
8.
9.  // 不会产生 Exception
10. te.method2("Hello");
11.
12. // 会产生 Exception
13. te.method2(null);
14. }
15. // ...
16. }
```

程序在执行的时候，也不见得会真的产生NullPointerException，这种Exception 叫做runtime exception 也有人称为unchecked exception，产生Runtime Exception 的method (在这个范例中是method2) 并不需要在宣告method 的时候定义它将会产生哪一种Exception。

在testException 的method3() 中，我们看到了另外一种状况，也就是method3里呼叫了method1()，但却没有将method1 包在try 和catch 之间。相反，在method3() 的定义中，它定义了CException，实际上就是如果method3 收到了CException，它将不处理这个CException，而将它往外丢。当然，由于method3 的定义中有throws CException，因此呼叫method3 的程序码也需要有try catch 才

行。

因此从程序的运作机制上看，Runtime Exception与Checked Exception 不一样，然而从逻辑上看，Runtime Exception 与Checked Exception 在使用的目的上也不一样。

一般而言，Checked Exception 表示这个Exception 必须要被处理，也就是说程序设计者应该已经知道可能会收到某个Exception(因为要try catch住) ，所以程序设计者应该能针对这些不同的Checked Exception 做出不同的处理。

而Runtime Exception 通常会暗示着程序上的错误，这种错误会导致程序设计者无法处理，而造成程序无法继续执行下去。

看看下面的例子：

Java代码

```java
String message[] = {"message1", "message2","message3"};
System.out.println(message[3]);
```

这段程序码在Compile 时并没问题，但在执行时则会出现ArrayIndexOutOfBoundException 的例外，在这种状况下，我们亦无法针对这个Runtime Exception 做出有意义的动作，这就像是我们呼叫了testException 中的method2 ，却引发了它的NullPointerException 一样，在这种状况下，我们必须对程序码进行修改，从而避免这个问题。

因此，实际上我们应该也必须要去抓取所有的Checked Exception，同时最好能在这些Checked Exception 发生的时候做出相对应的处理，好让程序能面对不同的状况。

然而对于Runtime Exception ，有些人建议将它catch 住，然后导向其它地方，让程序继续执行下去，这种作法并非不好，但它会让我们在某些测试工具下认为我们的程序码没有问题，因为我们将Runtime Exception "处理"掉了，事实却不然！譬如很多人的习惯是在程序的进入点后用个大大的try catch 包起来，如：

Java代码

```java
public class runtest1
{
public static void main(String argv[])
{
try
{
//...
}
catch(Exception e)
{
}
}
}
```

在这种情况下，我们很可能会不知道发生了什么Exception 或是从哪一行发出的，因此在面对不同的Checked Exception时，我们可已分别去try catch它。而在测试阶段时，如果碰到Runtime Exception ，我们可以让它就这样发生，接着再去修改我们的程序码，让它避免Runtime Exception，否则，我们就应该仔细追究每一个Exception ，直到我们可以确定它不会有Runtime Exception 为止！

对于Checked Exception 与Runtime Exception ，我想应该有不少人会有不同的观点，无论如何，程序先要能执行，这些Exception 才有机会产生。因此，我们可以把这些Exception 当成是Bug ，也可以当成是不同的状况(Checked Exception)，或当成是帮助我们除错的工具(Runtime Exception)，但前提是我们需要处理这些Exception ，如果不处理，那么问题或状况就会永远留在那里。

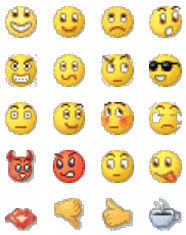
内部类和static 内部类 | JAVA内存溢出解析

11:03 | 浏览 (69) | 评论 (0) | 分类: 基础递进 | 相关推荐

评论

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交