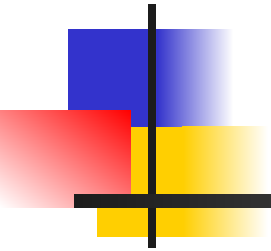


Java程序设计

第7章　异常处理

第 7 章　内部类与异常处理

7.1 异　常

异常是指程序在运行时产生的错误。比如在进行除法运算时，若除数为0，则运行时Java会自动抛出算术异常、若对一个值为null的引用变量进行操作，则会抛出空指针异常、若访问一个大小为2的一维数组中的第3个元素，则会抛出数组下标越界异常等。

Java语言中的异常也是通过一个对象来表示的，程序运行时抛出的异常，实际上就是一个异常对象。该对象中不仅封装了错误信息，还提供了一些处理方法，如getMessage()方法获取异常信息、printStackTrace()方法输出对异常的详细描述信息等。

异常

对于可能出现的异常，都需要预先进行处理，保证程序的有效运行，否则程序会出错。

在Java语言中已经提供了一些异常用来描述经常发生的错误，对于这些异常，有的需要程序员进行捕获处理或声明抛出，称为“受检查异常”；有的是由Java虚拟机自动进行捕获处理，称为“运行时异常”或“不受检异常”。

Java中常见的异常如下表所示：

异常类名称	异常类含义
ArithmeticException	算术异常类
ArrayIndexOutOfBoundsException	数组下标越界异常类
ArrayStoreException	将与数组类型不兼容的值赋值给数组元素时抛出的异常
ClassCastException	类型强制转换异常类
ClassNotFoundException	为找到相应类异常
EOFException	文件已结束异常类
FileNotFoundException	文件未找到异常类
IllegalAccessException	访问某类被拒绝时抛出的异常
InstantiationException	试图通过newInstance() 方法创建一个抽象类或抽象接口的实例时抛出的异常
IOException	输入输出异常类
NegativeArraySizeException	建立元素个数为负数的数组异常类
NullPointerException	空指针异常类
NumberFormatException	字符串转换为数字异常类
NoSuchFieldException	字段未找到异常类
NoSuchMethodException	方法未找到异常类
SecurityException	小应用程序(Applet) 执行浏览器的安全设置禁止的动作时抛出的异常
SQLException	操作数据库异常类

7.2 异常处理

异常产生后，若不做任何处理，则程序就会被终止，为了保证程序有效的执行，就需要对产生的异常进行相应处理。

在Java语言中，若某个方法抛出异常，既可以在当前方法中进行捕获，然后处理该异常，也可以将异常向上抛出，由方法的调用者来处理。

下面来介绍Java中的异常处理方法。

异常处理

使用try…catch语句

在Java语言中，对容易发生异常的代码，可通过try…catch语句捕获。在try语句块中编写可能发生异常的代码，然后在catch语句块中捕获执行这些代码时可能发生的异常。

一般格式格式为：

```
try{
    可能产生异常的代码
}catch(异常类 异常对象){
    异常处理代码
}
```

异常处理

使用try…catch语句

try语句块中的代码可能同时存在多种异常，那么到底捕获的是哪一种类型的异常，是由catch语句中的“异常类”参数来指定的。catch语句类似于方法的声明，包括一个异常类型和该类的一个对象，异常类必须是Throwable类的子类，用来指定了catch语句要捕获的异常，异常类对象可在catch语句块中被调用，例如调用对象的getMessage() 方法获取对异常的描述信息。

异常处理

使用try…catch语句

将一个字符串转换为整型，可通过Integer类的parseInt()方法来实现。当该方法的字符串参数包含非数字字符时，parseInt()方法会抛出异常。Integer类的parseInt()方法的声明如下：

```
public static int parseInt(String s)  throws NumberFormatException{…}
```

代码中通过throws语句抛出了NumberFormatException异常，所以在应用parseInt()方法时可通过try…catch语句来捕获该异常，从而进行相应的异常处理。

异常处理

使用try…catch语句

例如将字符串“24L”转换为Integer类型，并捕获转换中产生的数字格式异常，可以使用如下代码：

```
try{
    int age=Integer.parseInt("24L"); //抛出NumberFormatException异常
     System.out.println("打印1");
}catch(NumberFormatException e){ //捕获NumberFormatException异常
    System.out.println("年龄请输入整数！ ");
    System.out.println("错误: "+e.getMessage() );
}
System.out.println("打印2");
```

异常处理

使用try…catch语句

因为程序执行到“Integer.parseInt("24L")”时抛出异常，直接被catch语句捕获，程序流程跳转到catch语句块内继续执行，所以“System.out.println("打印1")”代码行不会被执行；而异常处理结束后，会继续执行try…catch语句后面的代码。

说明：若不知代码抛出的是哪种异常，可指定它们的父类Exception。

异常处理

使用try…catch语句

在try…catch语句中，可以同时存在多个catch语句块。

一般格式为：

```
try{
    可能产生异常的代码
}catch(异常类1 异常对象){
    异常1处理代码
}catch(异常类2 异常对象){
    异常2处理代码
}
```

代码中的每个catch语句块都用来捕获一种类型的异常。若try语句块中的代码发生异常，则会由上而下依次来查找能够捕获该异常的catch语句块，并执行该catch语句块中的代码

异常处理

使用try…catch语句

在使用多个catch语句捕获try语句块中的代码抛出的异常时，需要注意catch语句的顺序。若多个catch语句所要捕获的异常类之间具有继承关系，则用来捕获子类的catch语句要放在捕获父类的catch语句的前面。否则，异常抛出后，先由捕获父类异常的catch语句捕获，而捕获子类异常的catch语句将成为执行不到的代码，在编译时会出错。例如：

```
try{
    int age=Integer.parseInt("24L") ; //抛出NumberFormatException异常
}catch(Exception e) { //先捕获Exception异常
    System.out.println(e.getMessage() ) ;
}catch(NumberFormatException e) { //捕获异常类Exception的子类异常
    System.out.println(e.getMessage() ) ;
}
```

异常处理

使用try…catch语句

代码中第二个catch语句捕获的NumberFormatException异常是Exception异常类的子类，所以try语句块中的代码抛出异常后，先由第一个catch语句块捕获，其后的catch语句块成为执行不到的代码，编译时发生如下异常：

```
执行不到的 NumberFormatException 的 catch 块。它已由 Exception 的 catch 块处理
```

异常处理
finally子句的用法

finally子句需要与try…catch语句一同使用，不管程序中有无异常发生，并且不管之前的try…catch是否顺利执行完毕，最终都会执行finally语句块中的代码，这使得一些不管在任何情况下都必须执行的步骤被执行，从而保证了程序的健壮性。

【例7-2】 下面这段代码虽然发生了异常，但是finally子句中的代码依然执行。

异常处理

使用throws关键字抛出异常

若某个方法可能会发生异常，但不想在当前方法中来处理这个异常，那么可以将该异常抛出，然后在调用该方法的代码中捕获该异常并进行处理。

将异常抛出，可通过throws关键字来实现。throws关键字通常被应用在声明方法时，用来指定方法可能抛出的异常，多个异常可用逗号分隔。

【例7-3】 下面这段代码的dofile() 方法声明抛出一个IOException异常，所以在该方法的调用者main() 方法中需要捕获该异常并进行处理。

异常处理
使用throw关键字

使用throw关键字也可抛出异常，与throws不同的是，throw用于方法体内，并且抛出一个异常类对象，而throws用在方法声明中来指明方法可能抛出的多个异常。

通过throw抛出异常后，如果想由上一级代码来捕获并处理异常，则同样需要在抛异常的方法中使用throws关键字在方法的声明中指明要抛出的异常；如果想在当前的方法中捕获并处理throw抛出的异常，则必须使用try…catch语句。上述两种情况，若throw抛出的异常是Error、RuntimeException或它们的子类，则无须使用throws关键字或try…catch语句。

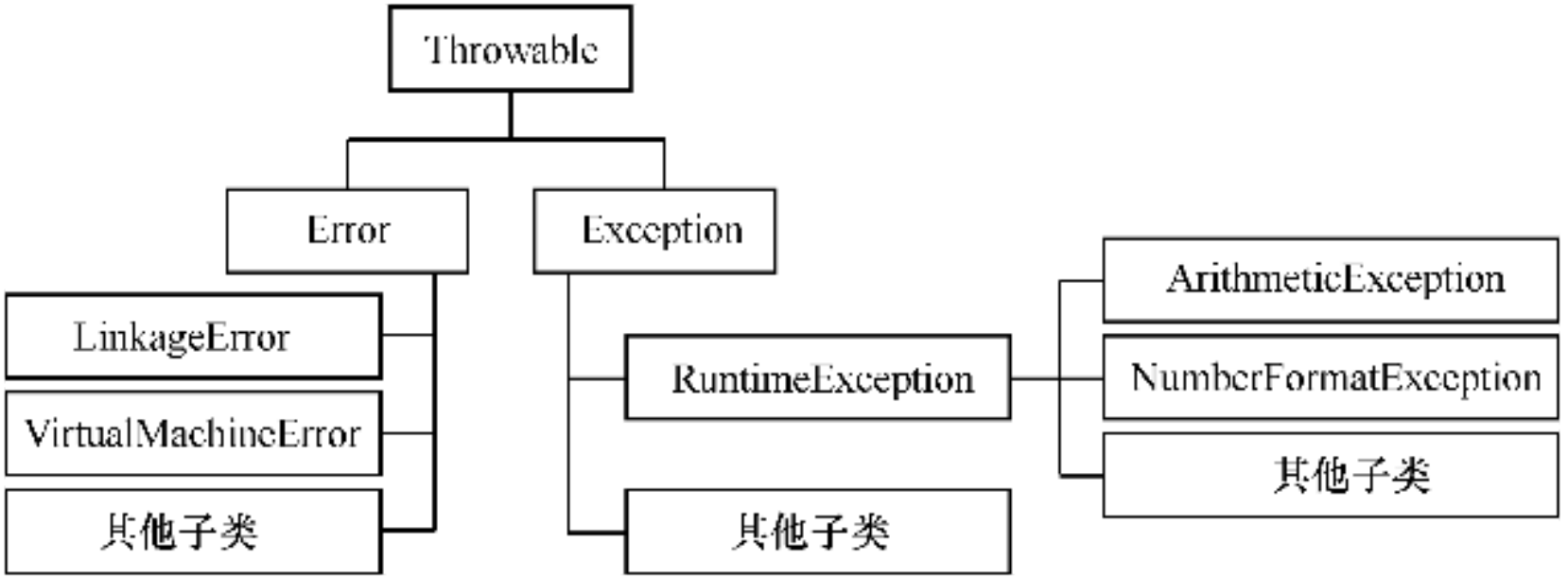
异常处理
使用throw关键字

当输入的年龄为负数时，Java虚拟机当然不会认为这是一个错误，但实际上年龄是不能为负数的，可通过异常的方式来处理这种情况。

【例7-4】 创建People类，该类中的check()方法首先将传递进来的String型参数转换为int型，然后判断该int型整数是否为负数，若为负数则抛出异常；然后在该类的main()方法中捕获异常并处理。

7.3 异常类

在Java语言中提供了一些内置的异常类来描述经常较容易发生的错误，这些类都继承自java.lang.Throwable类。Throwable类有两个子类：Error和Exception，它们分别表示两种异常类型。



7.4 Error类

Error类及其子类通常用来描述Java运行系统中的内部错误以及资源耗尽的错误。

Error表示的异常是比较严重，仅靠修改程序本身是不能恢复执行的，被称为致命异常类。

举一个现实中的例子，例如，因施工时偷工减料，导致学校教学楼坍塌，此时就相当于发生了一个Error异常。在大多数情况下，发生该异常时，建议终止程序。

7.5 Exception类

Exception类可称为非致命异常类，它代表了另一种异常。发生该异常的程序，通过捕获处理后可正常运行，保持程序的可读性及可靠性。在开发Java程序过程中进行的异常处理，主要就是针对该类及其子类的异常处理。对程序中可能发生的该类异常，应该尽可能进行处理，以保证程序在运行时，能够顺利执行，而不应该在异常发生后终止程序。

Exception类又分为两种异常类型：

RuntimeException异常

检查异常。

RuntimeException异常

RuntimeException是运行时异常，也称为不检查异常（unchecked exception），是程序员编写的程序中的错误导致的，修改了该错误后，程序就可继续执行。

例如，学校制定校规，若有学生违反了校规，就相当发生了一个RuntimeException异常。

在程序中发生该异常的情况如除数为0的运算、数组下标越界、对没有初始化的对象进行操作等。

当RuntimeExeption类或其子类所描述的异常发生后，可以不通过try…catch、throws捕获或抛出，在编译时是可以通过的，只是在运行时由Java虚拟机来抛出。

常见的RuntimeException异常如下表所示：

异常类名称	异常类含义
ArithmeticException	算术异常类
ArrayIndexOutOfBoundsException	数组下标越界异常类
ArrayStoreException	将与数组类型不兼容的值赋值给数组元素时抛出的异常
ClassCastException	类型强制转换异常类
IndexOutOfBoundsException	当某对象(如数组或字符串)的索引超出范围时抛出该异常
NegativeArraySizeException	建立元素个数为负数的数组异常类
NullPointerException	空指针异常类
NumberFormatException	字符串转换为数字异常类
SecurityException	小应用程序(Applet)执行浏览器的安全设置禁止的动作时抛出的异常
StringIndexOutOfBoundsException	字符串索引超出范围异常

检查异常

如果一个记者根据上级指定的地址去采访一个重要人物，这可能会抛出异常，例如根据指定地址没有找到被采访的人或采访被拒绝。该类异常被称为检查异常（check exception），要求必须通过try…catch捕获或由throws抛出，否则编译出错。

Java语言中常见的检查异常如下表所示

异常类名称	异常类含义
ClassNotFoundException	未找到相应类异常
EOFException	文件已结束异常类
FileNotFoundException	文件未找到异常类
IllegalAccessException	访问某类被拒绝时抛出的异常
InstantiationException	试图通过newInstance() 方法创建一个抽象类或抽象接口的实例时抛出该异常
IOException	输入输出异常类
NoSuchFieldException	字段未找到异常
NoSuchMethodException	方法未找到异常
SQLException	操作数据库异常类

7.6 自定义异常

通常使用Java内置的异常类就可以描述在编写程序时出现的大部分异常情况，但根据需要，有时要创建自己的异常类，并将它们用于程序中来描述Java内置异常类所不能描述的一些特殊情况。下面就来介绍如何创建和使用自定义异常。

自定义的异常类必须继承自Throwable类，才能被视为异常类，通常是继承Throwable的子类Exception或Exception类的子孙类。除此之外，与创建一个普通类的语法相同。

自定义异常

自定义异常类大体可分为以下几个步骤。

(1) 创建自定义异常类。

(2) 在方法中通过throw抛出异常对象。

(3) 若在当前抛出异常的方法中处理异常，可使用try…catch语句捕获并处理；否则在方法的声明处通过throws指明要抛出给方法调用者的异常，继续进行下一步操作。

(4) 在出现异常的方法调用代码中捕获并处理异常。

如果自定义的异常类继承自RuntimeExeption异常类，在步骤（3）中，可以不通过throws指明要抛出的异常。

下面通过一个实例来讲解自定义异常类的创建及使用。

【例7-5】 在编写程序过程中，如果希望一个字符串的内容全部是英文字母，若其中包含其他的字符，则抛出一个异常。因为在Java内置的异常类中不存在描述该情况的异常，所以需要自定义该异常类。

7.7 异常的使用原则

编写代码时处理某个方法可能出现的异常，可遵循以下几条原则：

1、在当前方法声明中使用try-catch语句捕获异常。

2、一个方法被覆盖时，覆盖它的方法必须抛出相同的异常或异常的子类。

3、如果父类抛出多个异常，则覆盖方法必须抛出那些异常的一个子集，不能抛出新异常。