Java技术

第四章 Java 异常处理

路　强

luqiang@hfut.edu.cn

合肥工业大学计算机与信息学院

本章学习提示

- **本章我们主要学习Java语言的异常处理机制**
  - Java编程当中的错误类型，如何处理这些错误
  - 学习什么是异常、如何处理异常
  - 如何利用Java所提供的异常类层次
  - 定义自己的异常类

什么是异常

- 软件开发和运行时，可能会发生下面情况：
  - 想要打开的文件不存在
  - 网络连接中断
  - 操作数超出预定范围
  - 正在装载的类文件丢失
  - 访问的数据库打不开
  - ……

“人算不如天算 ！”

这些并不是因为程序的错误引起的！

? 如何处理

异常处理

- **异常** 指程序运行过程中出现的非正常现象，例如用户输入错误、除数为零、需要处理的文件不存在、数组下标越界等。
- 由于异常情况总是难免的，良好的应用程序除了具备用户所要求的功能以外，还应该具备预见并处理可能发生的各种异常的功能。这种对异常情况进行处理的技术成为**异常处理**。
- 计算机系统对异常处理通常由两种方法
  - 第一种是程序直接检测程序中的错误，遇到错误给出错误信息并终止程序的运行。
  - 第二种办法是由程序员在程序中加入异常处理功能

Java的异常处理机制

- 程序设计时必须考虑发生的突发错误，避免程序被中断或破坏，在以前的程序设计语言中，通常通过if-else结构来手工判断，不仅增加的程序员的工作，而且会遗漏隐藏的错误。
- Java提供了功能强大的异常处理机制，可以方便地
  - 在程序中监视可能发生异常的程序块
  - 并将所有异常处理代码集中放置在程序的某处

  使完成正常功能的程序代码与进行异常处理的程序代码分开。
- Java中处理异常两种方法：
  - 在发生异常的地方直接处理
  - 将异常抛给调用者，让调用者处理

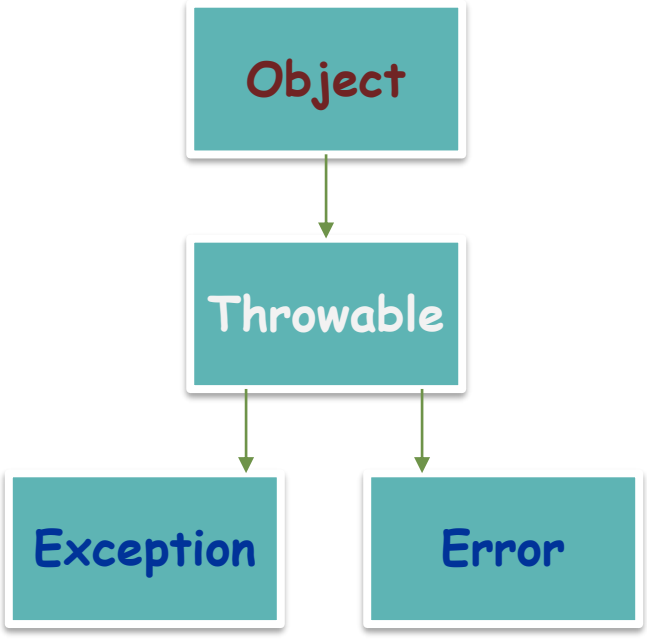
Java的异常处理机制-2

- 在Java的异常处理机制种引进了很多用来描述和处理异常的类，成为异常类。每个异常类反映一类运行错误，类定义中包含了该类异常的信息和对异常进行处理的方法：
  - 程序执行中如出现异常，系统会检测到并自动生成一个相应的异常类对象，然后交给运行时系统。
  - 运行时系统自动寻找相应的异常处理代码处理这一异常。若找不到可以处理该异常的代码，则运行时系统将终止，程序运行将推出。

```
public void WorkEveryday() {
    try {
        工作8个小时; //可能发生各种异常情况
        下班回家;
    }catch (孩子在幼儿园裤子尿湿了异常 e){
        去送干净衣服;
    }
}
```

Java的异常处理机制-3

- Java将异常分为Exception（异常）和Error（错误）两大类。
  - **Exception类** 解决由程序本身及环境所产生的异常，
  - **Error类** 处理内部系统错误。
  - Exception类异常可以被捕获并进行处理，而对Error类异常，程序员通常无能为力，只能在其发生时由用户按照系统提示关闭程序。



异常类的继承结构

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
   |  |
   |  +--java.io.IOException
   |  |
   |  +--java.lang.RuntimeException
   |  |  |
   |  |  +--java.lang.ArithmeticException
   |  |  |
   |  |  +-- java.lang.IndexOutOfBoundsException
   |  |  |  |
   |  |  |  +--java.lang.ArrayIndexOutOfBoundsException
   |  |  |
   |  |  +-- java.util.NoSuchElementException
   |  |  |   |
   |  |  |   +--java.util.InputMismatchException
   |  |  |
   |  |  +--Others..
   |  |
   |  +--Others..
   |  |
   |  +--java.lang.Error
   |     |
   |     +-- java.lang.VirtualMachineError
   |        |
   |        +--java.lang.OutOfMemoryError
   |        |
   |        +--java.lang.InternalError
   |        |
   |        +--Others...
```

Exception 子类的继承关系

- Exception
  - ClassNotFoundException
  - ClassNotSupporteException
  - IllegalAccessException
  - InstantiationException
  - InterruptedException
  - NoSuchMethodException
  - RuntimeException
  - ArithmeticException
  - ArrayStoreException
  - ClassCastException
  - IllegalArgumenException
  - **IllegalThreadStateException**
  - **NumberFormatException**
  - ArrayIndexOutOfBoundsException
  - StringIndexOutOfBoundsException

内容详见
Java.lang.Exception

异常(错误)处理的“新旧”之分

```
call methodA()
if methodA() worked
{
   call methodB()
   if methodB() worked
   {
      call methodC()
      if methodC() worked
         everything's okay, so display finalResult
      else
         set errorCode to 'C'
   }
   else
      set errorCode to 'B'
}
else
   set errorCode to 'A'
```

传统错误检查的伪代码

面向对象异常处理的伪代码

```
try
{
   call methodA() and maybe throw an exception
   call methodB() and maybe throw an exception
   call methodC() and maybe throw an exception
   everything's okay, so display finalResult
}
catch(methodA()'s error)
{
   set errorCode to "A"
}
catch(methodB()'s error)
{
   set errorCode to "B"
}
catch(methodC()'s error)
{
   set errorCode to "C"
}
```

异常处理方法

- 异常处理有两种方法
  - 使用try…catch…finally结构
  - 通过throws 和 throw 抛出异常
- try…catch…finally一般形式

```
try  {
        可能出现异常的程序代码
} catch(异常类名 1  异常对象名1) {
        异常类名1 对应的异常处理代码
} catch(异常类名2  异常对象名2)  {
        异常类名2对应的异常处理代码
}
   …
  finally {
        无论是否发生异常，一定会执行的代码
}
```

程序执行过程中，如果没有出现异常，将正常执行，catch块不起作用，若出现异常将终止try块代码的执行，自动跳转到所发生的异常类对应的代码块中，执行该块中的代码。

Finally块是可选项，无论是否发生异常，该块代码必定执行!!

异常处理方法

- 语法是依据下列的顺序来处理异常
  - try 程序块若是有异常发生时，程序的运行便中断，并抛出“异常类所产生的对象”
  - 抛出的对象如果属于catch()括号内欲捕获的异常类，则catch会捕捉此异常，然后进到catch的块里继续运行
  - 无论try 程序块是否有捕捉到异常，或者捕捉到的异常是否与catch()括号里的异常相同，最后一定会运行finally块里的程序代码

异常例1 - 数组越界


```
1.  //捕获数组下界越界异常，当i<3改为i<=3时执行正常
2.  public class ExceptionDemo01 {
3.      public static void main(String args[ ])  {
4.          try {
5.                int a[ ]={1,2,3},sum=0;
6.                for(int i=0; i<=3; i++)  sum += a[i];
7.                System.out.println("sum=“ + sum + "Successfully!");
8.          }
9.            catch ( ArrayIndexOutOfBoundsException e ) {
10.               System.out.println("ArrayIndexOutOfBoundsException detected");
11.                // e.printStackTrace();
12.               System.out.println(" 当i<=3改为i<3时执行正常!");
13.         }
14.           finally {
15.               System.out.println("Program Finished !");
16.         }
17.     }
18. }
```

由于使用未定义的a[3]出现数组越界

将i<=3改为 i<3

异常例2 - 算术异常


```
1.  public class ExceptionDemo02{
2.    public static void main(String args[]) {
3.                        //尽量不要写throws ArithmeticException {
4.      try {
5.         int  x=15, y=0;
6.         System.out.println("x/y="+x/y);
7.         System.out.println(" Computing  successfully!");
8.      }
9.      catch( ArithmeticException e ) {
10.        System.out.println(“ 捕获  ArithmeticException !");
11.        System.out.println(“ 异常信息:  "+e.toString());
12.     }
13.     finally {
14.        System.out.println(“  善后处理 !");
15.      }
16.   }
17. }
```

```
捕获 ArithmeticException !
异常信息: Java.arithmeticException : /by zero
善后处理!
```

抛出异常

- 程序员可以在程序中通过throw语句抛出异常，其格式为：

  throw new 异常类名（信息）

  - 异常类名可以选择系统异常类名，也可以使用自定义异常类名
  - 信息项可选，若有，该信息增加在toString()方法的返回值中

- 如果类中定义的方法本身不捕获某种异常，将该种异常的捕获和处理交给调用它的方法，这是需要在声明本方法时，使用throws关键字抛出**多个异常**，其方法定义具体格式为：

  [修饰符] 返回值类型 方法名 [（参数表）] throws 异常类型名 {
  　　声明部分
  　　语句部分
  }

  - 这里指给出异常类型名，而不列追加信息
  - 通常直接由Java虚拟机处理

异常例3 – 主动抛出异常


```
1.  //在自己编写的方法中主动抛出异常
2.  // 方法本身不对异常捕获和处理，而由调用它的方法去处理
3.  class MyMath{
4.          public int div (int i, int j)  throws Exception {   //或ArithmeticException
5.                  return (i / j) ;
6.          }
7.  }
8.  //主类
9.  public class ExceptionDemo03{
10.         public static void main(String args[]){
11.                 MyMath m = new MyMath() ;
12.                 try{
13.                         int temp = m.div(10, 0) ;
14.                         System.out.println(temp) ;
15.                 }catch (Exception e){
16.                         System.out.println("捕获 除数为零异常") ;
17.                         //e.printStackTrace(); // 打印异常
18.                 }
19.                 //throw new Exception("抛着玩的。") ;       // 人为抛出
20.         }
21. }
```

调用者捕　获

自定义异常类

- 自定义异常类可以通过继承Exception类来实现，一般格式为：

```
class 自定义异常类名 extends Exception {
    异常类体
}
```

自定义异常类例1


```
1.  // 自定义异常类将继承Exception 类的所有方法
2.  // 除此之外，还可以在类体中定义其他处理方法
3.  class OverFlowException extends Exception  {
4.      OverFlowException()  {
5.          System.out.println(" 此处数据有溢出，溢出类是OverFlowException"  );
6.  } }
7.  public class ExceptionDemo04{
8.      public static int x=100000;
9.      public static int multi() throws OverFlowException{
10.         int aim;
11.         aim=x*x*x;
12.         if(aim>2.15E9||aim<0){
13.           throw new OverFlowException();}
14.         else
15.         return x*x;
16.     }
17.     public static void main(String args[]){
18.         int y;
19.         try{
20.           y=multi();
21.           System.out.println(" y="+y);
22.       }
23.         catch( OverFlowException e){
24.           System.out.println(e); }
25. } }
```

抛出的异常类被捕获后，先执行它的构造方法，然后执行catch块的方法

自定义异常类例2

ExceptionDemo05.Java

mathException

- **mathException**是一个自定义的异常类，其中定义了构造方法，功能是输出信息：输入数据不正确。
  - 当输入工资为负值时，由main() 方法捕获该异常，先执行该类的构造方法，输出“输入数据不正确”，再输出该异常类信息mathException后，结束程序运行。
  - 若输入姓名为空字符串，将抛出Exception类异常，在捕获块中输出异常信息，然后结束程序运行
- 只有当用户给名字输入了非空串，又为工资输入了非负值，程序才能正常运行，并输出计算结果。
  - 若为名字输入空串，系统显示Java.long.Exception
  - 如果为工资输入了负值，系统显示输入数据不正确 mathException
  - 若为名字输入wang 为工资输入1200，则计算结果为Wang 的年薪是14400

本章总结

我们主要学习Java语言的异常处理机制

- Java编程当中的错误类型，如何处理这些错误
- 学习什么是异常、如何处理异常
- 如何利用Java所提供的异常类层次
- 定义自己的异常类

作业与练习

作业2：见作业文档。

Thank You !