# 1.异常的概述

异常:异常出现后,出现异常的语句以及之后的语句将不再执行

我们要做的事情是:即使出现了异常,那么也应该让程序正确执行完毕

# 2.异常的处理(try,catch,finally)

使用格式:

**try{**

**//有可能出现异常的语句**

**}catch(异常类型 对象){**

**//异常处理语句**

**} catch(异常类型 对象){**

**//异常处理语句**

**} catch(异常类型 对象){**

**//异常处理语句**

**}**

**…**

**finall{**

**不管是否出现异常都会执行的代码**

**}**

**以上组合为:try..catch,try..catch...finally,try..finally(不建议)**

**注意:**

1. **try不能单独使用,必须和catch或者finally一起使用**
2. **在编写多个catch的时候,捕捉范围小的要在捕获范围大的前面,不然编译报错**

**范例:**

|  |
| --- |
| **public class Main{**  **public static void main(String args[]){**  **System.out.println("1.除法开始计算");**  **try {**  **System.out.println("除法:" + 10 / 0);**  **System.out.println("###"); //如果发生异常这一行不会被执行,不发生异常会被执行**  **}catch (ArithmeticException e){**  **e.printStackTrace();**  **}finally {**  **System.out.println("finally");**  **}**  **System.out.println("3.除法计算结束");//如果没有异常或者异常被catch捕获到**  **//这一行是会被执行的,如果没有被catch捕获到是不会被执行的**  **}**  **}** |

# 3.异常继承结构

观察两个异常继承结构(java.lang)

|  |  |
| --- | --- |
| ArithmeticException | NumberFormatException |
| java.lang.Object  java.lang.Throwable  java.lang.Exception  java.lang.RuntimeException  java.lang.ArithmeticException | java.lang.Object  java.lang.Throwable  java.lang.Exception  java.lang.RuntimeException  java.lang.IllegalArgumentException  java.lang.NumberFormatException |

所有的异常都是Throwable的子类,Throwable有两个直接的子类(Error,Exception)

|  |
| --- |
| **[root@rt command]# java dsfadfafdsa**  **Error: Could not find or load main class dsfadfafdsa** |

**Error:**指的是JVM错误,即此时的程序还没有执行,如果没有执行用户无法处理

**Exception:**指的是程序运行中产生的异常,用户可以处理

异常处理指的是Exception及其子类异常

注意:异常实例由JVM实例化而来

# 4.throws关键字

一**个异常要么在函数内部try catch处理,要么通过throws关键字交给函数被调用处处理**

throws:用于方法声明上,指的是方法中出现异常后交由被调用处来进行处理

|  |
| --- |
| public class Main{  public static void main(String args[]) throws Exception{ //要么不处理,继续向上抛出,交给被调用者处理  System.out.println(MyMath.div(10,2));  }  public void fun(){ //要么自己处理通过该try catch  try {  MyMath.div(10, 2);  }catch (Exception e) {  e.printStackTrace();  }  }  }  class MyMath{  //由于有throws,表示此方法里产生的异常交友被调用处处理  public static int div(int x , int y) throws Exception{  return x / y;  }  } |

主方法抛出异常,这个异常将由JVM来处理,即打印错误信息,然后结束程序,所以main函数不要加throws

# 5.throw关键字

异常可以由JVM抛出(实例化),也可以有用户手动抛出(实例化)

|  |
| --- |
| public class test{  public static void main(String args[]){  try {  throw new Exception("hahah");  }catch (Exception e){  e.printStackTrace();  }  }  } |

对于抛出的异常,要么处理try..catch,要么向上继续抛出throws Exception

# 6.异常处理标准模型

|  |
| --- |
| public class test{  public static void main(String args[]){  try {  System.out.print(MyMath.div(10,0));  } catch (Exception e) {  e.printStackTrace();  }  }  }  class MyMath{  public static int div(int x ,int y) throws Exception{  int result = 0;  System.out.println("1.除法计算开始"); //这行要执行  try {  result = x / y;  }catch (Exception e) { //其实这里还可以简化,将catch去掉,不建议这么用  throw e; //继续抛出异常  }  finally {  System.out.println("2.除法计算结束");//这行要执行  }  return result;  }  } |

# 7.RuntimeException类

java.lang.Object

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException 运行时异常

在java里是为了方便用户代码编写,这种异常在编译的时候不会强制要求

用户去处理异常,用户可以选择性的去处理,如果没有处理,但是发生了异常,

将交给JVM默认处理

# 8.assert断言(了解)

断言:程序执行到某行代码处一定是预期的结果

|  |
| --- |
| public class test{  public static void main(String args[]){  int num = 10;  //中间可能经过很多行代码来操作num  //期望num的内容应该是20  assert num==20:"num的内容不是20";  System.out.print("num=" + num);  }  } |

Assert在java中默认是不起作用的,要加-ea 参数

# 9.自定义异常

|  |
| --- |
| public class test{  public static void main(String args[]){  int num = 11;  try {  if (num > 10) {  throw new AddException("数值传递过大");  }  }catch (Exception e){  e.printStackTrace();  }  }  }  class AddException extends Exception{  public AddException(String msg){  super(msg);  }  } |