第5章　异常处理

软件系统应该为用户提供一套完善的服务，系统不仅要满足用户的需求功能，还需要具有可靠性、稳定性和容错性。软件系统不仅自身不能有错误，还要具备较强的抗干扰能力：当用户操作出现错误时，或遇到不可抗拒的干扰时，软件系统也不能放弃，而必须尽最大努力排除错误继续运行；只有具备这样能力的软件系统才会具有更好的应用空间。Java 语言的异常处理机制能够很好地解决以上问题。

本章主要内容：

- 异常处理的基本概念。
- 捕获与处理异常。
- 抛出异常。

5.1　Java 异常处理的基本概念

编译和运行程序时，经常会由于各种各样的原因而导致程序出错。例如，在编译程序时，违反语法规范的错误一般称为语法错，这类错误通常在编译时被发现，又称为编译错，如标识符未声明、变量赋值时的类型与声明时的类型不匹配、括号不匹配、语句末尾缺少分号等。这类错误容易发现，也容易修改。为避免产生语法错误，应严格按照 Java 语言约定的规则编写程序，注意标识符中字母大小写等细节问题。

对程序运行时出现的错误进行处理则要复杂一些。如果程序在语法上正确，但在语义上存在错误，称为语义错，如输入数据格式错，除数为 0 错，给变量赋予超出其范围的值等。语义错不能被编译系统发现，只有到程序运行时才能被系统发现，所以含有语义错的程序能够通过编译。有些语义错能够被程序事先处理，如除数为 0，数组下标越界等，程序中应该设法避免产生这些错误。有些语义错不能被程序事先处理，如待打开的文件不存在，网络连接中断等，这些错误的发生不由程序本身所控制，因此必须进行异常处理。

还有一类错误，程序能够通过编译并且能够运行，但运行结果与期望值不符，这类错误称为逻辑错。如由于循环条件不正确而没有结果，循环次数不对等因素导致计算结果不正确等。由于系统无法找到逻辑错，所以逻辑错最难确定和排除。该类错误需要程序员凭借自身的编程经验找到错误原因和出错位置，改正错误。

Java 应用程序出现错误时，根据错误的性质不同可以分为两类：错误和异常。

5.1.1　错误与异常

1. 错误

错误(error)是指程序遇到非常严重的不正常状态，不能简单地恢复执行，一般是在运行时遇到的硬件或操作系统的错误，如内存溢出、操作系统出错、虚拟机出错等。错误对于程序而言是致命性的，将导致程序无法运行，而且程序本身不能处理它，而只能依靠外界干

预，否则会一直处于非正常状态。例如，没有找到.class 文件，或.class 文件中没有 main()方法等，将导致应用程序不能运行。

2. 异常

异常(exception)指非致命性错误，一般指在运行程序时硬件和操作系统是正常的，而程序遇到了运行错，如整数进行除法运算时除数为 0，操作数超出数据范围，要打开一个文件时发现文件不存在，网络连接中断等。

异常会导致应用程序非正常中止，但 Java 语言提供的异常处理机制使应用程序自身能够捕获异常，并且能够处理异常，由异常处理部分调整应用程序运行状态，使应用程序仍可继续运行。

在编译和运行应用程序时，发现 Java 应用程序中的错误和异常并进行处理的流程如图 5-1所示。

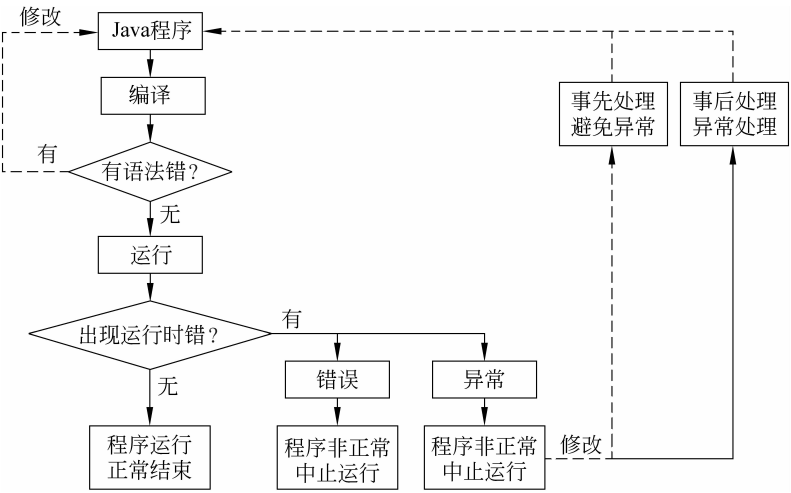

图 5-1　Java 应用程序中发现错误和异常并进行处理

5.1.2　错误和异常的分类

Java 类库提供了许多处理错误和异常的类，主要分为两大部分：Error 类和 Exception 类。

Error 类是错误类，该类由 Java 虚拟机生成并抛给系统，如内存溢出错误、栈溢出错误、动态链接错误等。当运行某一个类时如果没有 main()方法，则产生错误 NoClassDefFoundError；当使用 new 分配内存空间时，如果没有可用内存，则产生内存溢出错误 OutOfMemoryError。

Exception 类是异常类，是 Java 应用程序捕获和处理的对象。每一种异常对应于 Exception 类的一个子类，异常对象中包含错误的位置和特征信息。Java 预定义了多种通用的异常类。Java 语言预定义的错误类和异常类以及子类的层次结构如图 5-2 所示。

【例 5-1】　异常程序示例(ExceptionByZero.java)

```
/*
    功能简介：除数为 0 的异常。
*/
public class ExceptionByZero{
```

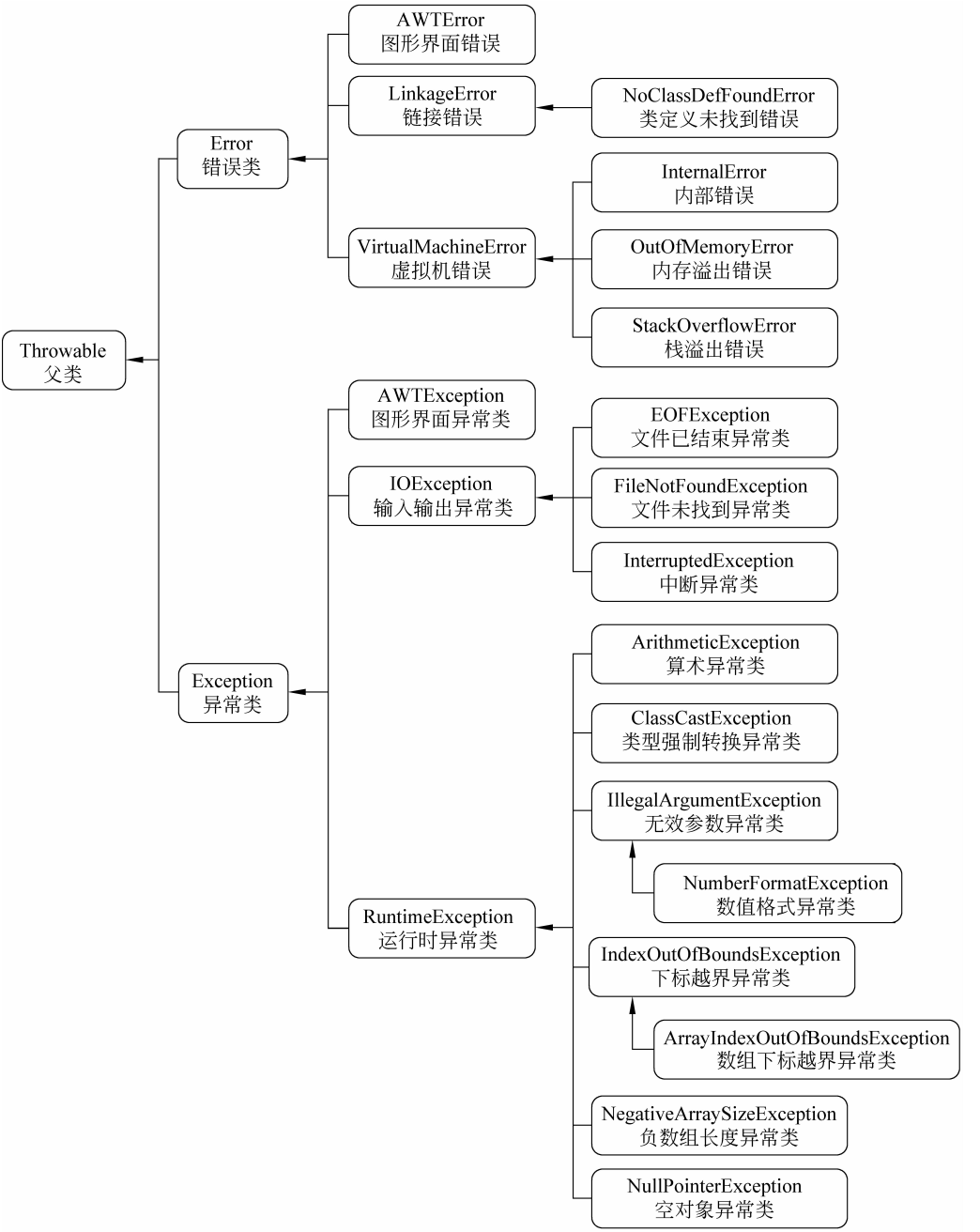

图 5-2　错误和异常类以及子类层次图

```
public static void main(String args[ ]){
      int x=6;
      int y=0;
      System.out.println("x="+x);
      System.out.println("y="+y);
      System.out.println("x/y="+x/y);
```

```
    }
}
```

编译、运行结果如图 5-3 所示。

图 5-3　ExceptionByZero.java 编译、运行结果

5.2　异 常 处 理

编译时若出现语法错误异常，程序员必须处理这些异常，否则程序无法进行编译。Java 语言提供了异常处理机制。处理异常的方式有两种：捕获异常并处理和抛出异常。

5.2.1　捕获异常并处理

捕获异常处理方式是通过 try-catch-finally 来捕获和处理异常的。语法格式如下：

```
try
{
    ⋮       //可能会产生异常的语句序列
}
catch (ExceptionType1  e1)
{
    ⋮       //异常处理代码,捕获到该类型异常后进行处理
}
    ⋮
catch (ExceptionTypeN  eN)
{
    ⋮       //异常处理代码,捕获到该类型异常后进行处理
}
finally
{
    ⋮       //语句序列,无论是否捕获到异常都必须执行的语句
}
```

其中：

try 引导的语句是应用程序中有可能出现异常的代码段，一旦 try 捕获到异常就由 catch 子句处理异常。

ExceptionType 是异常类。

e1,…,eN 表示不同异常类型对应的对象。

catch 语句块用于处理 try 捕获到的异常,catch 可以有多个子句,每个子句对应的异常类型不同。一旦捕获到异常,将自动匹配 catch 子句中的异常类型,找到相应的异常类型后执行该 catch 语句。如果没有捕获到异常,则所有 catch 语句将不执行。

在异常处理过程中,finally 语句块总是会被执行到,无论有没有异常发生,也无论有没有异常被捕捉到。finally 语句块是可选项,通常位于 catch 语句块的后面。可以用来释放 try 语句块中获得的资源,如关闭在 try 语句块中打开的文件等。

catch 和 finally 可以只有其中一项或者两项都有,但是必须至少有一项。

【例 5-2】 异常处理程序(TryCatchFinally.java)

```
/*
    功能简介:使用 try-catch-finally 语句进行异常处理。
*/
public class TryCatchFinally{
    public static void main (String args[]){
          int i=0;
          int a[]={1,2,3,4,5};
          for(i=0;i<6;i++){
              try{
                   System.out.print("a["+i+"]/"+i+"="+(a[i]/i));
              }
              catch(ArrayIndexOutOfBoundsException e){
                  System.out.print("捕获到数组下标越界异常!");
              }
              catch(ArithmeticException e){
                  System.out.print("捕获到算术异常!");
              }
              catch(Exception e){
                  System.out.print("捕获"+e.getMessage()+"异常!");    //输出异常信息
              }
              finally{
                  System.out.println("i="+i);
              }
          }//for 结束
    }
}
```

5.2.2 抛出异常

Java 语言中要求捕获到的异常必须得到处理。如果在某个成员方法体中可能会发生异常,则该成员方法必须采用 try-catch-finally 语句处理这些异常,或者将这些可能发生的异常转移到上一层调用该成员方法的方法中,这就是抛出异常,也就是说本方法不会处理该异常,而是由上一层处理产生的异常。

1. throw

throw 是 Java 语言提供的主动抛出异常的关键字。throw 的语法格式如下：

throw 异常对象；

其中：

throw 是关键字，用于抛出异常，由 try 语句捕获并处理。

异常对象是程序创建的指定异常类对象。

```
public void set(int age)
{
    if (age>0 && age<160)
          this.age=age;
    else
          throw new Exception("IllegalAgeData");     //抛出异常
}
```

【例 5-3】 抛出异常程序(Person.java)

```
/*
    功能简介：使用抛出异常，构造一个 Person 类，封装了姓名和年龄，可以比较两个人年龄的
    大小。
*/
public class Person{
    private String name;                                  //姓名
    private int age;                                      //年龄
    public Person(String name,int age)                    //构造方法
    {
          this.setName(name);
          this.setAge(age);
    }
    public void setName(String name){
          if (name==null || name=="")
              this.name="姓名未知";
          else
              this.name=name;
    }
    public String getName() {
          return this.name;
    }
    public void setAge(int age){
          try{
              if (age>0 && age<100)
                  this.age=age;
              else
                  throw new Exception("年龄无效");
          }
```

```
            catch(Exception e){
                System.out.println(e.toString());
            }
        }
        public int getAge(){
            return this.age;
        }
        public String toString(){
            return getName()+","+getAge()+"岁";
        }
        public int olderThen(Person p2)                    //比较两个人的年龄
        {
            return this.getAge()-p2.getAge();
        }
        public static void main(String args[]) {
            Person p1=new Person("小李子",36);
            System.out.println(p1.toString());
            Person p2=new Person("小贾",26);
            System.out.println(p2.toString());
            System.out.println(p1.getName()+"比"+p2.getName()+"大"+
               p1.olderThen(p2)+"岁");
        }
    }
```

2. throws

假如一个方法的方法体会产生异常，而该方法体中不想处理或不能处理该异常，则可以在方法声明时采用 throws 子句声明该方法，将异常抛出。

throws 的语法格式如下：

```
[修饰符] 返回类型  方法名(参数列表) throws 异常类 1, 异常类 2,…
{
    ⋮                                                  //方法体
}
```

这样在本方法内就可以不处理这些异常，而调用该方法的方法就必须处理这些异常。

其中，throws 是关键字，用于指定声明的方法向上层抛出异常。

异常类是方法要抛出的异常类，一个方法可以抛出多个异常类，各异常类之间用逗号隔开。

例如：

```
public void set(int age) throws Exception
{
    if (age>0 && age<160)
        this.age=age;
    else
        throw new Exception("年龄无效"+age);
}
```

如果一个方法调用一个抛出异常的方法(含有 throws 的方法),则该方法必须捕捉含有 throws 的方法的异常;含有 throws 的方法本身不处理异常,而是“谁用谁处理”。

5.3 自定义异常类

在进行 Java 应用程序编程时,可以使用类库中已经定义好的异常类。系统预定义的类库有时候不能满足用户的需要,则程序员编写应用程序时可以自定义需要的异常类。自定义异常类必须继承已有的异常类,即用户自定义的异常类都必须直接或间接地是 Exception 类的子类。

【例 5-4】 自定义年龄异常类(AgeException.java)

```
/*
    功能简介:声明自定义异常类。
*/
public class AgeException extends Exception     //无效年龄异常类
{
    public AgeException(String s){
          super(s);                             //调用父类的构造方法
    }
    public AgeException(){
          this("");
    }
}
```

【例 5-5】 自定义异常类的使用(Person1.java)

```
/*
    功能简介:自定义异常类的使用。
*/
public class Person1{
    private String name;                        //姓名
    private int age;                            //年龄
    public Person1(String name,int age) throws AgeException{
          this.setName(name);
          this.setAge(age);
    }
    public void setName(String name){
          if (name==null||name=="")
              this.name="姓名未知";
          else
              this.name=name;
    }
    public String getName() {
          return this.name;
    }
```

```
    public void setAge(int age) throws AgeException{
        if (age>=0&&age<160)
            this.age=age;
        else
            throw new AgeException(""+age);
    }
    public int getAge(){
        return this.age;
    }
    public String toString(){
        return getName()+","+getAge()+"岁";
    }
    public void print(){
        System.out.println(this.toString());
    }
    public static void main(String args[]){
        Person1 p1=null;
        try{
            //调用声明抛出异常的方法,必须写在 try 语句中,否则编译不通过
            p1=new Person1("小李子",36);
            p1.setAge(161);
        }
        catch(AgeException e){                    //捕获自定义异常类,而非 Exception 类
            e.printStackTrace();                  //显示异常栈跟踪信息
        }
        finally{
            p1.print();
        }
    }
}
```

5.4 常见问题及解决方案

(1) 异常信息提示如图 5-4 所示。

图 5-4 异常信息提示(1)

解决方案：在使用 try-catch-finally 语句时，catch 或 finally 子句至少要有一个。

(2) 异常信息提示如图 5-5 所示。

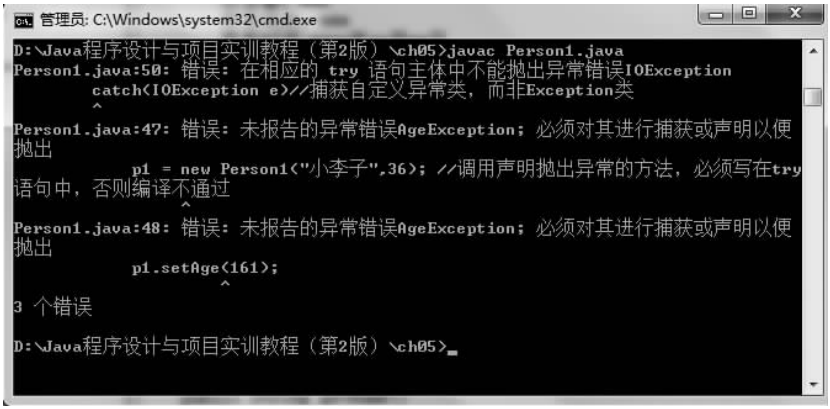

图 5-5　异常信息提示(2)

解决方案：在 Person1 类中，try 中抛出的异常是 AgeException 类，所以 catch 中的异常类应是 AgeException，而非 IOException 类。

5.5　本章小结

本章主要介绍 Java 语言中异常处理机制的基本知识，包括 Java 应用程序在编译和运行时处理异常的常用方法和思路。通过本章学习可以为开发友好的应用程序打下基础，并有利于实现项目的健壮性。

本章主要介绍异常处理机制，学完本章后应该了解和掌握以下内容。

- 异常处理的概念。
- 异常类。
- 异常处理的方法。
- 自定义异常。

总之，本章内容是进行异常处理的关键技术，只有在掌握本章知识的基础上才能够开发出友好的、健壮的、安全的 Java 应用程序。

5.6　习　　题

一、选择题

1. 程序能够通过编译并能运行，但结果与期望值不符的错误是(　　)。

A. 语法错　　B. 语义错　　C. 逻辑错　　D. 编译错

2. Java 语言中所有异常情况的类都会继承的类是(　　)。

A. Error　　B. Throwable　　C. Exception　　D. IOException

3. Java 语言中抛出异常的语句是(　　)。

A. try　　B. catch　　C. finally　　D. throws