Java7:异常处理

授课教师：邱元杰　　电子邮箱：yuanjiq@126.com，微信电话: 13679081552

第7章 异常处理

异常概念及异常类的层次

异常处理

自定义异常及处理

异常的传递

异常概念

- 异常就是程序执行过程中出现的不正常现象。非预期情况，错误的参数、网络故障。
- 任何一个程序都可能出现异常，Java使用对象表示对打开的文件不存在、内存不够、数组访问超界等非预期情况。
- Java使异常处理标准化，使程序设计思路更清楚，理解更容易。

异常类的层次

- Java定义的异常类有自己的类层次。所有异常类都是Throwable类的子类。
- Throwable属于java.lang包，在程序中不必使用import语句引入即可使用。
- Throwable类有三个最基本的子类Error, Exception和RuntimeException类。

异常类的层次

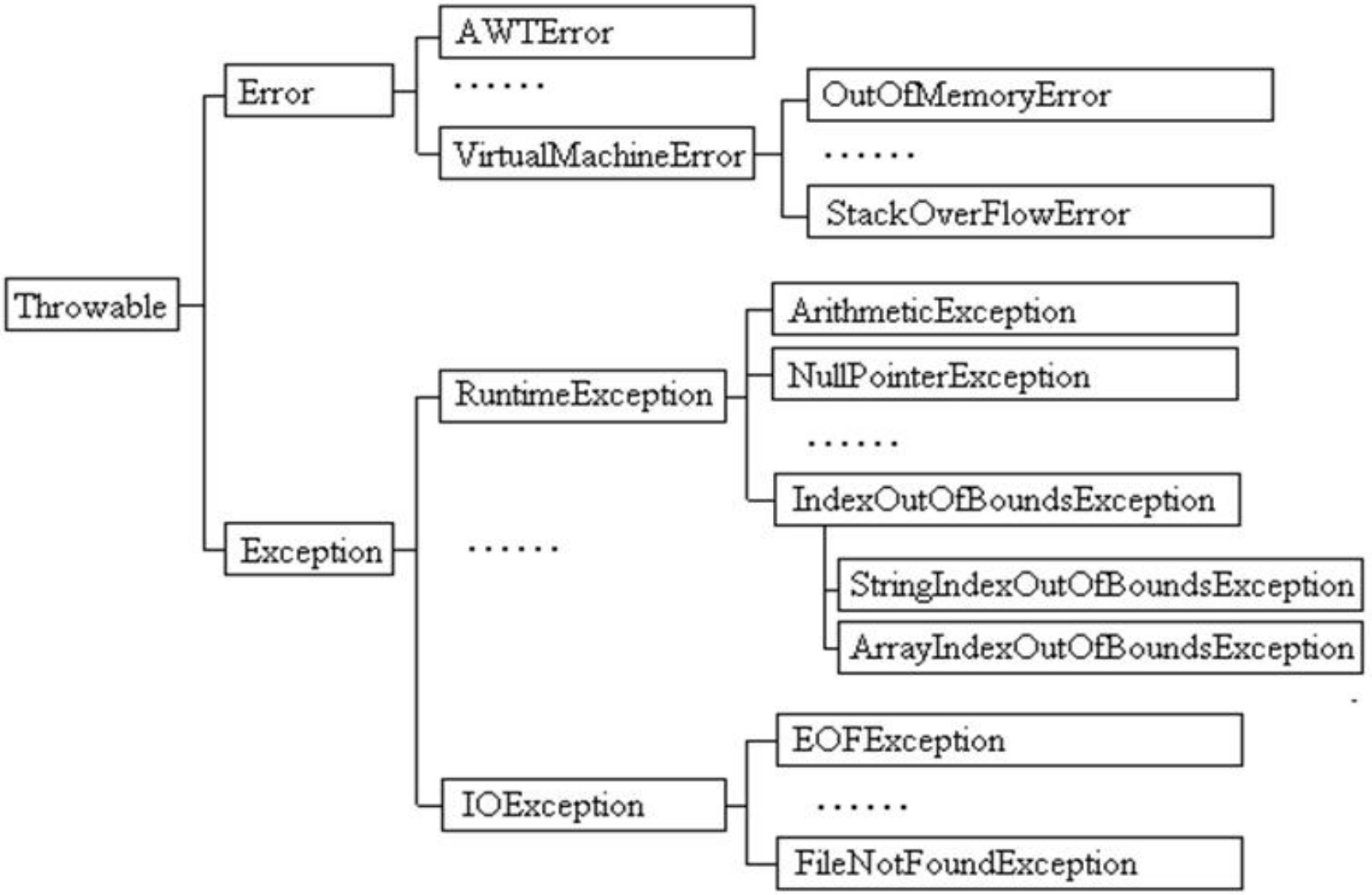

图 11-1　异常类层次结构示意图

Throwable类的定义

```
public class Throwable extends Object implements Serializable{
    public Throwable();
    public Throwable(String message);
    public String getMessage();
    public String getLocalizedMessage();
    public String toString();
    public void printStackTrace();
    public void printStackTrace(PrintStream s);
    public void printStackTrace(PrintWriter s);
    public native Throwable fillInStackTrace();
}
```

Throwable类常用的方法

- getMessage()：
  - 获得更详细的异常信息，但并非每个异常都有详细信息。如果没有详细信息，该方法调用后返回空值。
- toString()：
  - 获得异常的简短描述。
- printStackTrace()：
  - 打印异常发生处堆栈跟踪的信息，包括异常的类名、方法名及所在程序的行数。

三种不同形式的异常

- Error
  - 表示产生了非常严重的问题，即使有可能使程序恢复正常也非常困难，如内存不足等。对于这一类问题，一般不要求应用程序进行异常处理。
- RuntimeException
  - 表明产生了一个设计或执行问题，如果程序设计正确应该能够避免发生这类问题，如在访问数组时，数组下标越界等。对于这类问题也不要求进行处理，使该类问题能够暴露出来，从而改正程序。
- 其它Exception
  - 由于执行环境的影响，不可避免地将产生的问题。如用户敲入错误的文件名而导致文件没有找到等。对于这类问题，Java强烈要求应用程序进行完整的异常处理，给用户友好的提示，或者修正后使程序继续执行。

常用异常类

- ArithmeticException
  - 算术运算中，整数被零除，如int i = 12/0;
- ArrayIndexOutOfBoundsException
  - 访问数组超界异常
- ArrayStoreException
  - 进行写数组操作时，对象或数据类型不兼容，导致异常。
- ClassCastException
  - 当试图把对象A转换为对象B时，如果对象A既不是对象B的同类，又非对象B的子类，将产生该异常。
- IllegalArgumentException
  - 在方法的参数表中，如果参数无效，将产生异常。

常用异常类

- IllegalThreadStateException
  - 非法改变线程状态，如启动已执行线程，导致异常。
- IndexOutOfBoundsException
  - 是ArrayIndexOutOfBoundsException类的超类，它是抽象类。
- NegativeArraySizeException
  - 创建数组时，规定数组大小的参数是负数，产生异常
- NullPointerException
  - 试图访问空对象的变量、方法或空数组的元素，产生异常。
- NumberFormatException
  - 试图把一字符串非法转换成数组（或相反），导致该异常

常用异常类

- SecurityException
  - Applet试图执行被WWW浏览器安全设置所禁止的操作，产生异常。
- IncompatibleClassChangeException
  - 有两种情况抛出该异常，一是某成员变量的声明被从静态改变为非静态，但其它引用了这个变量的类却没有重新编译，或者相反。二是删除了类声明中的某一域或方法，但没有重新编译那些引用了这个域或方法的类。
- OutOfMemoryException
  - 表示“内存不足”异常。
- NoClassDefException
  - Java执行时找不到所引用的类，产生该异常。

常用异常类

- IncompatibleTypeException
  - 试图实例化一个接口，产生该异常。
- UnsatisfiedLinkException
  - 所调用的方法是C方法，但执行时无法连接这个方法，将产生该异常。
- InternalException
  - 是系统内部故障所导致的异常。

异常处理

- 发生异常时，就会抛出一个异常，通过捕获这个异常，就可以进行相应异常处理。其形式如下：

```
try {
    正常程序段;
}
catch(异常类1    异常变量) {
    与异常类1有关的处理程序段;
}
catch(异常类2    异常变量) {
    与异常类2有关的处理程序段;
}
......
finally {
    退出异常处理程序段;
}
```

异常抛出

异常1被捕获

异常2被捕获

与异常是否抛出无关

异常处理

```
public class TryTest1 {
 public TryTest1() {
   try {
     int a[] = new int[2];
     a[4]= 3;
     System.out.println("在异常处理后，会返回到这吗?");
   }
 public static void main(String[] args){
   new TryTest1();
 }
}
```

结果为:

```
Exception msg: 4
Exception string:java.lang.ArrayIndexOutOfBoundsException:4
java.lang.ArrayIndexOutOfBoundsException: 4
        at TryTest1.<init>(TryTest1.java:5)
        at TryTest1.main(TryTest1.java:19)
-------------
finally
No exception?
```

嵌套的异常处理

- 在try-catch-finally结构中，可以使用嵌套形式，即在捕获异常处理过程中，可以继续抛出异常。
- 在这种嵌套结构中，产生异常后，首先与最内层的try-catch-finally结构中的catch语句进行匹配比较。
- 如果没有相匹配的catch语句，该异常情况可以被抛出，让外层的try-catch-finlly的结构重复进行匹配检查。这样从最内层到最外层，逐一检查匹配，直到找到一个匹配为止。

TryTest2.java

MethNestTry.java

throw语句

- 在实际的应用程序中，除了可能产生Java的标准异常外，还可能产生应用程序的特定异常，这时应用程序应该给用户提供明确的指示，帮助用户正确理解和使用该应用程序。
- throw语句抛出异常格式为：
  - throw 表达式;
  - 其中，“表达式” 为一个异常对象。
  - 例：
  - throw new IOException("Not found the file");

ThrowTest.java

自定义异常类

- 用户可以根据需要定义异常类。则要完成三件事:
  - 生成Throwable类或其子类的一个子类。
  - 在可能发生异常的地方，判断是否发生异常，如果发生异常，则用throw抛出异常。
  - 用try-catch-finally结构来捕获异常，进行处理。
- 例，自定义异常类IllegalMarkException:

```
class IllegelMarkException extends Throwable{
        IllegelMarkException() {}
}
```

throws语句

- 抛出异常的方法并不处理该异常，而是由调用该方法的另一方法来处理，那么这时可以使用throws语句给方法声明一个例外情况，其声明格式为：
- <返回类型> <方法名>(paraList) throws 异常类1,...
- 例：

```
void exam(int mark) throws NegativeMarkException,
                           OutofMarkException {
   if(mark < 0) throw new NegativeMarkException();
   if(mark> 100) throw new OutofMarkException();
}
```

ThrowsTest.java

断言

- 发现错误立刻停止
- Assert booleanExpression
- Assert booleanExpression:message
- 当然booleanExpression为false的时候，程序从断言处停止执行，并输出message
- 启用断言：java –ea ClassName

思考

- Java语言中的异常处理是什么含义?
- 在异常处理的中由哪些部分组成，它们各有什么作用?
- 在异常处理的catch语句中，异常类在安排次序是有什么要注意的?
- 什么情况下必须使用throw语句？举例说明。
- throws子句在什么位置使用，它表示什么含义？举例说明。