面向对象程序设计Java

江春华

电子科技大学信息与软件工程学院

内　容

第7章　异常处理

1. 异常概念及异常类的层次
2. 异常处理
3. 自定义异常及处理
4. 异常的传递

异常概念

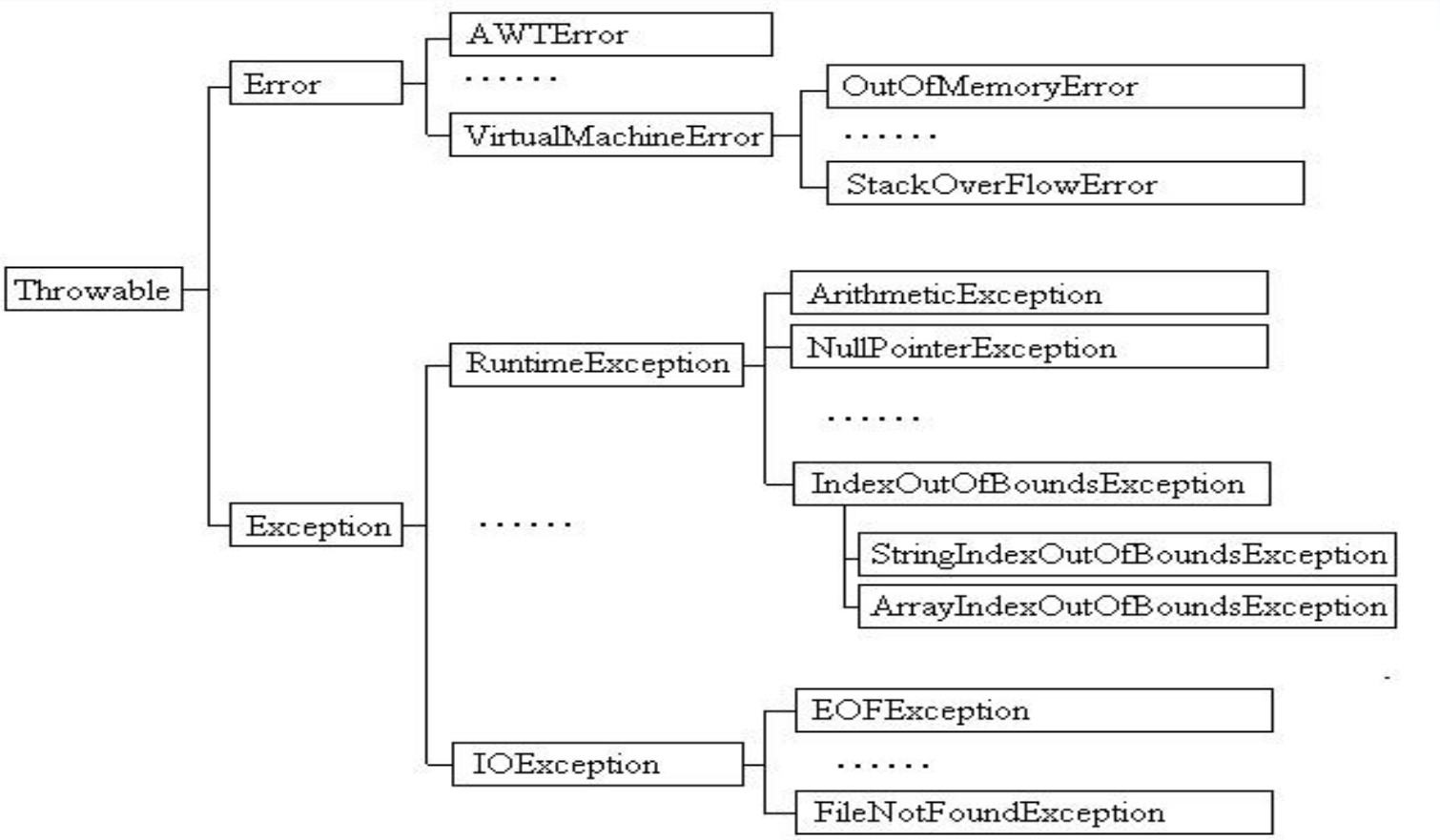
- 异常就是程序执行过程中出现的不正常现象。非预期情况，错误的参数、网络故障。
- 任何一个程序都可能出现异常，Java使用对象表示对打开的文件不存在、内存不够、数组访问超界等非预期情况。
- Java使异常处理标准化，使程序设计思路更清楚，理解更容易。

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

- 异常类的层次示意图



异常类的层次

- Throwable类常用的方法有：
  - 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。

异常类

- 三种不同形式的异常：

1. 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。

2. 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。

异常类

- 三种不同形式的异常：

3．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。

异常类

❖常用异常类：

1.ArithmeticException：

算术运算中，整数被零除，如`int i = 12/0;`

2.ArrayIndexOutOfBoundsException：

访问数组超界异常。

3.IllegalArgumentException：

在方法的参数表中，如果参数无效，将产生异常。

4.NegativeArraySizeException：

创建数组时，规定数组大小的参数是负数，产生异常。

5.NullPointerException：

试图访问空对象的变量、方法或空数组的元素，产生异常。

异常处理

- 发生异常时，就会抛出一个异[illegible]，就可以进行相应异常处理。其形[illegible]：

```
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

- 在对捕获异常类时需要注意的是：如果在catch所要捕获异常类存在继承关系时，则必须将子类放在超类的前面，否则子类异常被抛出时永不会被子类捕获，因为在前的超类就已经匹配。

例：异常类1是异常类2的子类，则其捕获异常的顺序为：

```
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;
}
```

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?");
   }catch(IndexOutOfBoundsException e) {
     System.err.println("Exception msg:"+e.getMessage());
     System.err.println("Exception string:"+e.toString());
```

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。

嵌套的异常示例

```
public class TryTest2 {
   public TryTest2() {
     for(int i=0; i<2; i++) {
        int k;
        try {
          System.out.println("\nOuter try block; Test Case #"+i);
          try {
             System.out.println("\tInner try block");
             switch(i) {
                case 0:
                      int zero = 0;
                      k = 8/zero; break;
                case 1:
                      int[] c = new int[3];
                      k = c[5];   break;
             }
          }catch(ArithmeticException e){
              System.out.println("\tInner> " + e);
          }catch(IndexOutOfBoundsException e){
              System.out.println("\tInner> " + e);
              throw e;
          }
```

嵌套的异常示例

```
结果为:
Outer try block; Test Case #0
        Inner try block
        Inner> java.lang.ArithmeticException:/ by zero
        Inner finally block
Outer finally block

Outer try block; Test Case #1
        Inner try block
        Inner> java.lang.ArrayIndexOutOfBoundsException:5
        Inner finally block
Outer> java.lang.ArrayIndexOutOfBoundsException:5
Outer finally block
```

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");
```

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();
}
```

思考问题

1. Java是如何处理异常，异常类层次是怎样的？
2. 异常的抛出有哪两种方式？
3. 如何在异常中传递信息？

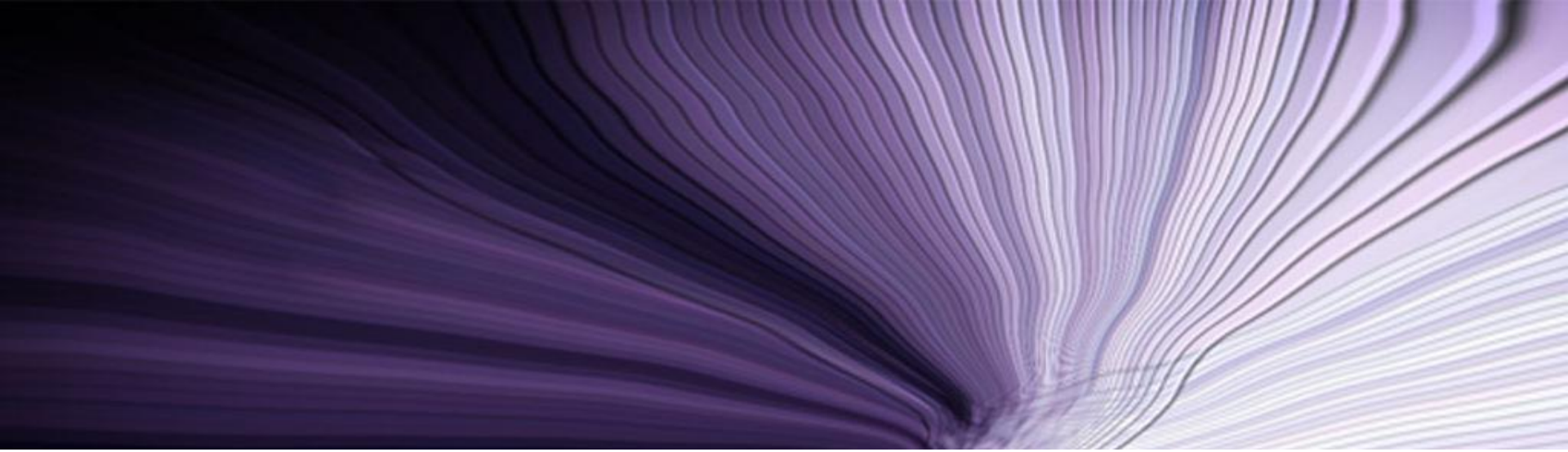
第7章作业

本章习题

习题1-6题

1-5题必做

6题选做



Q&A

电子科技大学信息与软件工程学院