第七章

异常处理

東北林業大学
NORTHEAST FORESTRY UNIVERSITY

本章任务

- 异常处理概述
- 异常分类
- 异常的捕获处理
- 重新抛出异常
- 定义新的异常类型

本章目标

- 掌握异常处理定义和原理
- 掌握try-catch-finally结构
- 掌握定义新的异常类型和重新抛出异常

1 程序错误

- 编译错误--因为程序没有遵循语法规则，编译程序能够自己发现并且提示我们错误的原因和位置，这个也是大家在刚接触编程语言最常遇到的问题。

- 运行时错误--因为程序在执行时，运行环境发现了不能执行的操作。

- 逻辑错误--因为程序没有按照预期的顺序和逻辑执行。

2 程序错误实例

```
public class Demo1 {
    public static void main(String[] args) {
        int x,y,z,a=10,b=0;
        y=a*a;
        z=a+b
        x=a/b;
        System.out.println(x+" "+y+" "+z);
    }
} //输出a的立方、a+b、a/b
```

3 异常和异常处理机制

- 异常--程序运行时可能出现一些错误，比如试图打开一个根本不存在的文件等。
- 如果置之不理，程序便会终止或直接导致系统崩溃，显然这不是我们希望看到的结果。

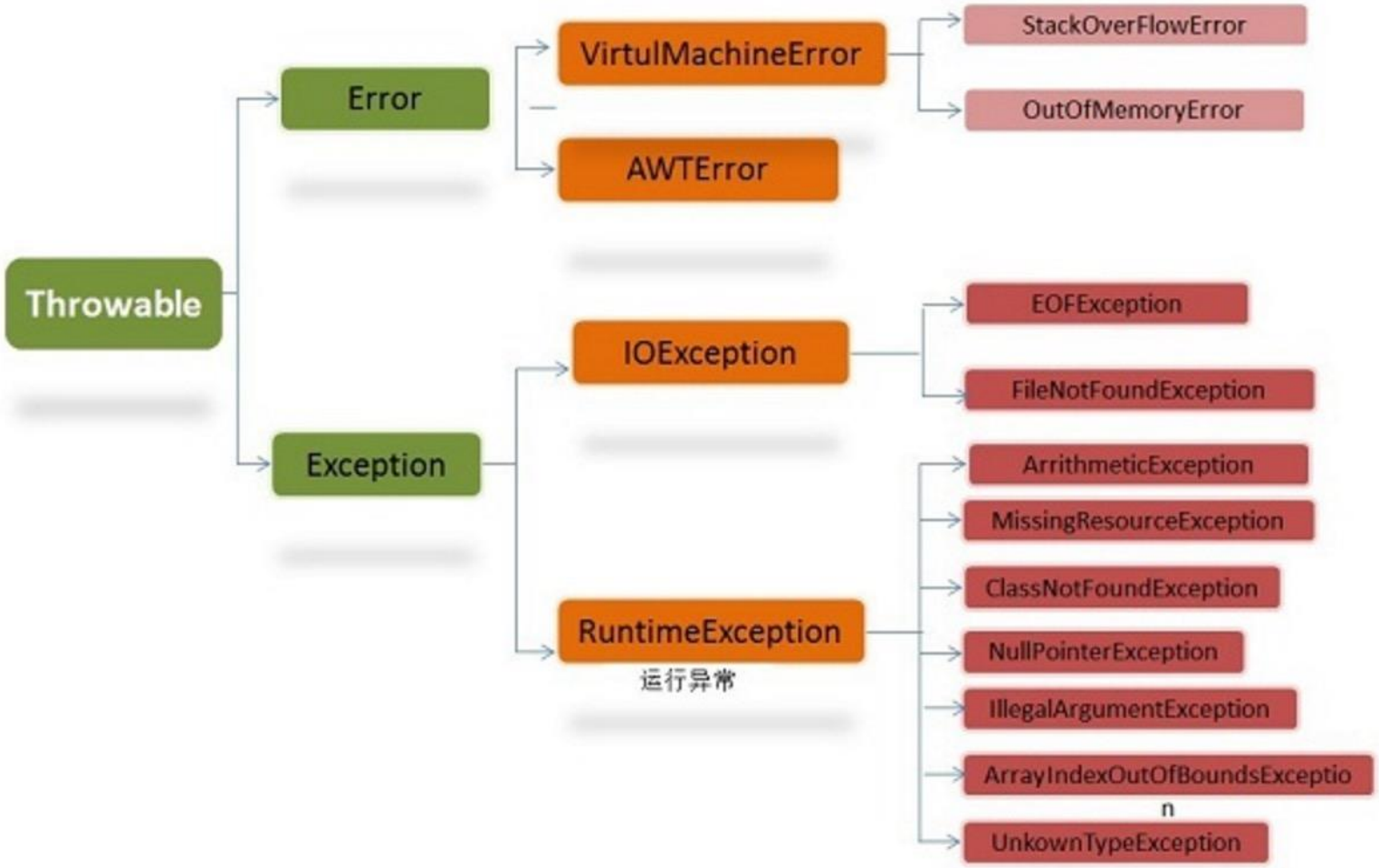
- 异常处理机制--指当程序出现错误后，程序如何处理。
- 具体来说，异常机制提供了程序退出的安全通道。当出现错误后，程序执行的流程发生改变，程序的控制权转移到异常处理器。

4 减少和避免程序错误

- 学习和掌握java的必要知识：java基础知识（1、2章）、面向对象程序设计（3-5章）、JavaSE常用类（6-9章）
- 养成良好的编程习惯：规范各种标识符、语句描述，测试完成的类、对象、域、方法等
- 培养良好的程序员职业能力：细心、创新、逻辑、学习
- 掌握和编写处异常理机制程序：找到异常出现的逻辑、使用完整的数据测试、形成有针对性的处异常理。

7.2 异常分类

1 分类


Throwable
Error
VirtulMachineError
StackOverFlowError
OutOfMemoryError
AWTError
Exception
IOException
EOFException
FileNotFoundException
RuntimeException
运行异常
ArrithmeticException
MissingResourceException
ClassNotFoundException
NullPointerException
IllegalArgumentException
ArrayIndexOutOfBoundsExceptio
n
UnkownTypeException

2 错误和异常

- Throwable是所有异常的共同祖先，异常都是从Throwable继承而来的。Throwable有两个子类，Error和Exception。
- Error是错误，表示运行应用程序中出现了严重的错误，都是通过Error抛出的，一般是程序不能处理的系统错误。
- Exception是异常，表示程序运行时，程序本身可以捕获并且可以处理的错误。
- 异常和错误的区别是，异常是可以被处理的，而错误是没法处理的。

3 错误Error分类

- Error类包括一些不能被程序员处理的严重的系统错误类，共有12个直接子类，一般表示代码运行时 JVM（Java 虚拟机）出现的问题，如内存溢出、虚拟机错误、栈溢出等。
- 这类错误与硬件有关，大多数错误与代码编写者执行的操作无关，通常由系统进行处理，程序本身无法捕获和处理。
- 如：当JVM耗完可用内存时，将出现OutOfMemoryError。如Java虚拟机运行错误，将出现VirtualMachineError、类定义错误将出现NoClassDefFoundError等。
- 这些错误是不可查的，因为它们在应用程序的控制和处理能力之外，而且绝大多数是程序运行时不允许出现的状况。

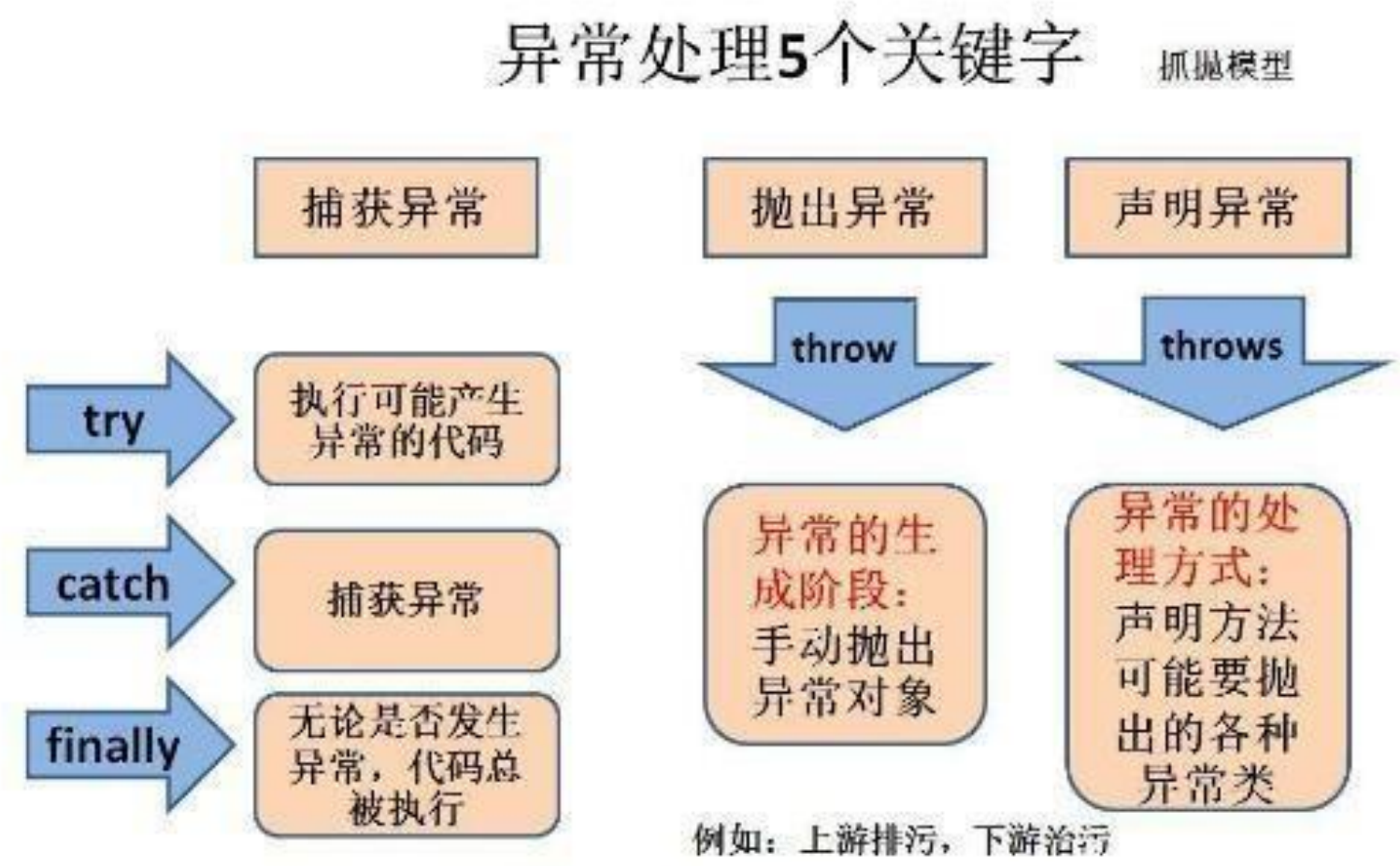
4 异常Exception分类

- 运行时异常(**不受检异常**) -- RuntimeException类及其子类：表示JVM在运行期间可能出现的错误。编译器不会检查此类异常，并且不要求必须处理异常。
- 如：空值对象的引用（NullPointerException）、数组下标越界（ArrayIndexOutBoundException）。此类异常属于不可查异常，一般是由程序逻辑错误引起的，在程序中可以选择捕获处理，也可以不处理。
- 非运行时异常(**受检异常**) --除RuntimeException及其子类之外的异常，如果程序中出现此类异常，如IOException，必须对该异常进行处理，要么使用try-catch捕获，要么使用throws语句抛出，否则编译不通过。

5 常用异常

- 输入输出异常：IOException
- 算术异常类：ArithmeticExecption
- 空指针异常类：NullPointerException
- 类型强制转换异常：ClassCastException
- 操作数据库异常：SQLException
- 文件未找到异常：FileNotFoundException
- 数组负下标异常：NegativeArrayException
- 数组下标越界异常：ArrayIndexOutOfBoundsException
- 违背安全原则异常：SecturityException
- 文件已结束异常：EOFException
- 字符串转换为数字异常：NumberFormatException
- 方法未找到异常：NoSuchMethodException

6 处理异常关键字


异常处理5个关键字
抓抛模型
捕获异常
抛出异常
声明异常
throw
throws
try
执行可能产生异常的代码
catch
捕获异常
finally
无论是否发生异常，代码总被执行
异常的生成阶段：手动抛出异常对象
异常的处理方式：声明方法可能要抛出的各种异常类
例如：上游排污，下游治污

7.3 异常的捕获处理

1、try-catch-finally结构

```
try {   ...
//监视代码执行过程，一旦返现异常则直接跳转至catch，
// 如果没有异常则直接跳转至finally}
catch (ExceptionType ExceptionObject) {
 //可选执行的代码块，如果没有任何异常发生则不会执行；
 //如果发现异常则进行处理或向上抛出。}
finally {
 //可选执行的代码块，不管是否有异常发生，
 //即使内存溢出异常也执行，通常用于处理善后清理工作。}
```

7.3 异常的捕获处理

1、try-catch-finally结构

```
try {
        // 可能会发生异常的程序代码          }
catch (Type1 id1) {
        // 捕获并处理try抛出的异常类型Type1    }
catch (Type2 id2) {
        // 捕获并处理try抛出的异常类型Type2    }
finally {
        // 无论是否发生异常，都将执行的语句块
```

2关键词try、catch、finally

- try：将可能出现错误的程序代码放在try块中，对try块中的程序代码进行检查，可能会抛出一个或多个异常。因此，try后面可跟一个或多个catch。
- catch：其功能是捕获异常，参数ExceptionObject是由try语句生成的。ExceptionType是Throwable类中的子类，它指出catch语句中所处理的异常类型。catch捕获异常的过程中，要将Throwable类中的异常类型和try语句抛出的异常类型进行比较，若相同，则在catch中进行处理。
- Finally：是这个组合语句的统一出口，一般用来进行一些“善后”操作，例如释放资源、关闭文件等。它是可选的部分。

3 运行时异常处理

运行时异常只有运行时才会出现，将出现异常的语句使用try-catch-finally结构包裹即可。

运行时异常处理。　　判断语句替代

```
int a=10,b=0,x=0;
try {
        x=a/b;
        System.out.println(x);
} catch (ArithmeticException
e) {
        e.printStackTrace();
}
```

```
int a=10,b=0,x=0;
if (b!=0) {
    x = a / b;
    System.out.println(x);
}
else

System.out.println("b=0");
```

4 非运行时异常处理

非运行时异常，是必须进行处理的异常，要么使用try-catch捕获，要么使用throws语句抛出，否则编译不通过。

```
File file=new
File("hello.txt");
char b[]="欢迎welcome北京
".toCharArray();
try{  //捕获
        FileWriter out=new
FileWriter(file);
        out.write(b);
        out.close();
}catch(IOException e)     {
System.out.println(e);}
```

```
public static void
main(String[] args) throws
IOException{//抛出
File file= new
File("hello.txt");
   char b[]="欢迎welcome北
京".toCharArray();
       FileWriter out=new
FileWriter(file);
       out.write(b);
       out.close();    }
```

5 异常处理次序

存在多个异常时，异常的产生是由于语句中的异常顺序决定，还是由catch语句顺序决定。

```
int a=10,b=0,x=0;
try {
        int y[]=new int[-2];
        x=a/b;
        System.out.println(x);
}catch (ArithmeticException e) {
        e.printStackTrace();
}catch (NegativeArraySizeException e) {
        e.printStackTrace();
}
```

6 try-catch-finally语句嵌套

Java语言的try-catch-finally语句可以嵌套，即在try块中可以包含另外的try-catch-finally语句。

```
try{
    嵌套外层代码
    try{ 嵌套内层代码 }
    catch(ExceptionType1 ExceptionObject1){
          处理内层异常   }
    finally {内层出口}
 }
catch(ExceptionType2 ExceptionObject2)
    {处理外层异常 }
finally {外层出口}
```

7 try-catch-finally语句获得

- 运行时异常处理

选择语句块->右键->Surround With->try/catch Block

catch (Exception e) {e.printStackTrace();}

选择语句块->右键->Surround With->try_catch(try catch Block)

catch (Exception e) { }

选择语句块->右键->Surround With->try_finally(try finally Block)

finally { }

- 非运行时异常处理

选择语句块->右键->Surround With->try/catch Block

catch (IOException e) {e.printStackTrace();}

选择语句块->右键->Surround With->try_catch(try catch Block)

catch (Exception e) { }

8 catch语句中异常类匹配

- 匹配的原则是：如果抛出的异常对象属于catch子句的异常类，或者属于该异常类的子类，则认为生成的异常对象与catch块捕获的异常类型相匹配。
- 对于有多个catch子句的异常程序而言，应该尽量将捕获底层异常类（子类）的catch子句放在前面，同时尽量将捕获相对高层的异常类的catch子句放在后面。否则，捕获底层异常类的catch子句将可能会被屏蔽。
- 底层异常类和高层的异常类区别：高层的异常类覆盖底层异常类；底层异常类处理更加精准明确。

9 小结

try块：用于捕获异常。其后可接零个或多个catch块，如果没有catch块，则必须跟一个finally块。
catch 块：用于处理try捕获到的异常。
finally 块：无论是否捕获或处理异常，finally块里的语句都会被执行。当在try块或catch块中遇到return语句时，finally语句块将在方法返回之前被执行。在以下4种特殊情况下，finally块不会被执行：
1）在finally语句块中发生了异常。
2）在前面的代码中用了System.exit()退出程序。
3）程序所在的线程死亡。
4）关闭CPU。

7.4 重新抛出异常

1. Throw语句

- Throw语句用来明确地抛出一个异常。Throw语句的作用是改变程序的执行流程，使程序跳到相应的异常处理语句中执行，它后面的语句执行不到。Throw语句格式如下：

  throw new 异常类名(参数);

- 在异常处理中，try语句要捕获的是一个异常对象，那么此异常对象也可以自己抛出。
- 要注意的是，throw 抛出的只能够是可抛出类Throwable或者其子类的实例对象。下面抛出异常的语句是错误的。

  throw new String("exception");

2 什么时候才需要抛异常

- 异常的设计是为了方便开发者使用的，但不可乱用，随便抛出异常。

- 什么时候才需要抛异常？其实这个问题很简单，如果你的某些”问题”确实解决不了了，那么你就可以抛出异常了。比如，你在写一个service，其中在写到某段代码处，你发现可能会产生问题，那么就请抛出异常吧，可以相信，你此时抛出异常将是一个最佳时机。

3 Throw语句跑出异常

```
try {
    throw new NullPointerException("抛出空指针异常");
}
catch(NullPointerException e) {
    System.out.println("exception:"+e);
}
```

运行结果：

Exception: java.lang.NullPointerException: 抛出空指针异常

4 Throws语句

- 在有些情况下，一个方法不需要或本身没有能力来处理异常，就把异常向上移交给调用这个方法的方法来处理，在这种情况下则需要通过Throws语句来实现。Throws语句的格式如下：

  returnType methodName(para1,para2,...)Throws exceptionList

- exceptionList：异常列表，多个异常时使用“,”间隔
- throws语句用在方法定义时声明该方法要抛出的异常类型，如果抛出的是Exception异常类型，则该方法被声明为抛出所有的异常。

5 Throws抛出异常的规则

- 如果是不可查异常（unchecked exception），即Error、RuntimeException或它们的子类，那么可以不使用throws关键字来声明要抛出的异常，编译仍能顺利通过，但在运行时会被系统抛出。
- 必须声明方法可抛出的任何可查异常（checked exception）。即如果一个方法可能出现受可查异常，要么用try-catch语句捕获，要么用throws子句声明将它抛出，否则会导致编译错误
- 仅当抛出了异常，该方法的调用者才必须处理或者重新抛出该异常。当方法的调用者无力处理该异常的时候，应该继续抛出异。
- 调用方法必须遵循任何可查异常的处理和声明规则。若覆盖一个方法，则不能声明与覆盖方法不同的异常。声明的任何异常必须是被覆盖方法所声明异常的同类或子类。

6 Throw语句

```
static void throwOne( )  throws IllegalAccessException{
        throw new IllegalAccessException("安全权限异常");
}
public static void main(String[] args) {
        try{
                throwOne( );
        }
        catch(IllegalAccessException e) {
                System.out.println(e);
        }
}
```

7.5 定义新的异常类型

1Throwable类中的常用方法

- getCause()：返回抛出异常的原因。如果 cause 不存在或未知，则返回 null。
- getMeage()：返回异常的消息信息。
- printStackTrace()：获取异常类名和异常信息，以及异常出现在程序中的位置。
- toString()：获取异常类名和异常信息(即简单的异常信息描述)，

2 自定义异常类的步骤

- 使用Java内置的异常类可以描述在编程时出现的大部分异常情况。除此之外，用户还可以自定义异常。用户自定义异常类，只需继承Exception类即可。
- 在程序中使用自定义异常类，大体可分为以下几个步骤。

（1）创建自定义异常类。

（2）在方法中通过throw关键字抛出异常对象。

（3）如果在当前抛出异常的方法中处理异常，可以使用try-catch语句捕获并处理；否则在方法的声明处通过throws，指明要抛出给方法调用者的异常，继续进行下一步操作。

（4）在出现异常方法的调用者中捕获并处理异常。

3 银行结算

```
public class BankException extends Exception {
   String message;
   public BankException(int m,int n) {
      message="入账资金"+m+"是负数或支出"+n+"是正数，不符合系统要求.";
   }
   public String warnMess() {
      return message;
   }
}
```

4 货船装载

```
public class DangerException extends Exception {
    final String message = "超重";
    public String warnMess() {
        return message;
    }
}
```

try和catch

東北林業大學
NORTHEAST FORESTRY UNIVERSITY