



计算机学院

北華航天工業學院
NORTH CHINA INSTITUTE OF AEROSPACE ENGINEERING

第8章 Java异常

本章目标

- ◆ Java异常概念
- ◆ Java异常分类
- ◆ 异常的处理
- ◆ 自定义异常

Java异常概念

◆ java异常是java提供的用于处理程序中错误的一种机制。

- 所谓错误是指在程序运行的过程中发生的一些异常事件（如：数组下标越界，所要读取的文件不存在等）。
- 设计良好的程序应该在异常发生时提供处理这些错误的方法，使得程序不会因为异常的发生而终止程序的执行。
- 在编译时出现的错误并不是异常，换句话说，异常就是为了解决程序运行时出现的错误。

- `public class Calculator {`
- `public int devide(int num1, int num2) {`
- `//判断除数是否为0`
- `if(num2 == 0) {`
- `throw new IllegalArgumentException("除数不能为零");`
- `}`
- `return num1/num2;`
- `}`
- `}`
- 除数赋值为0, 刚会抛出异常。

Java异常概念 (TestException.java)

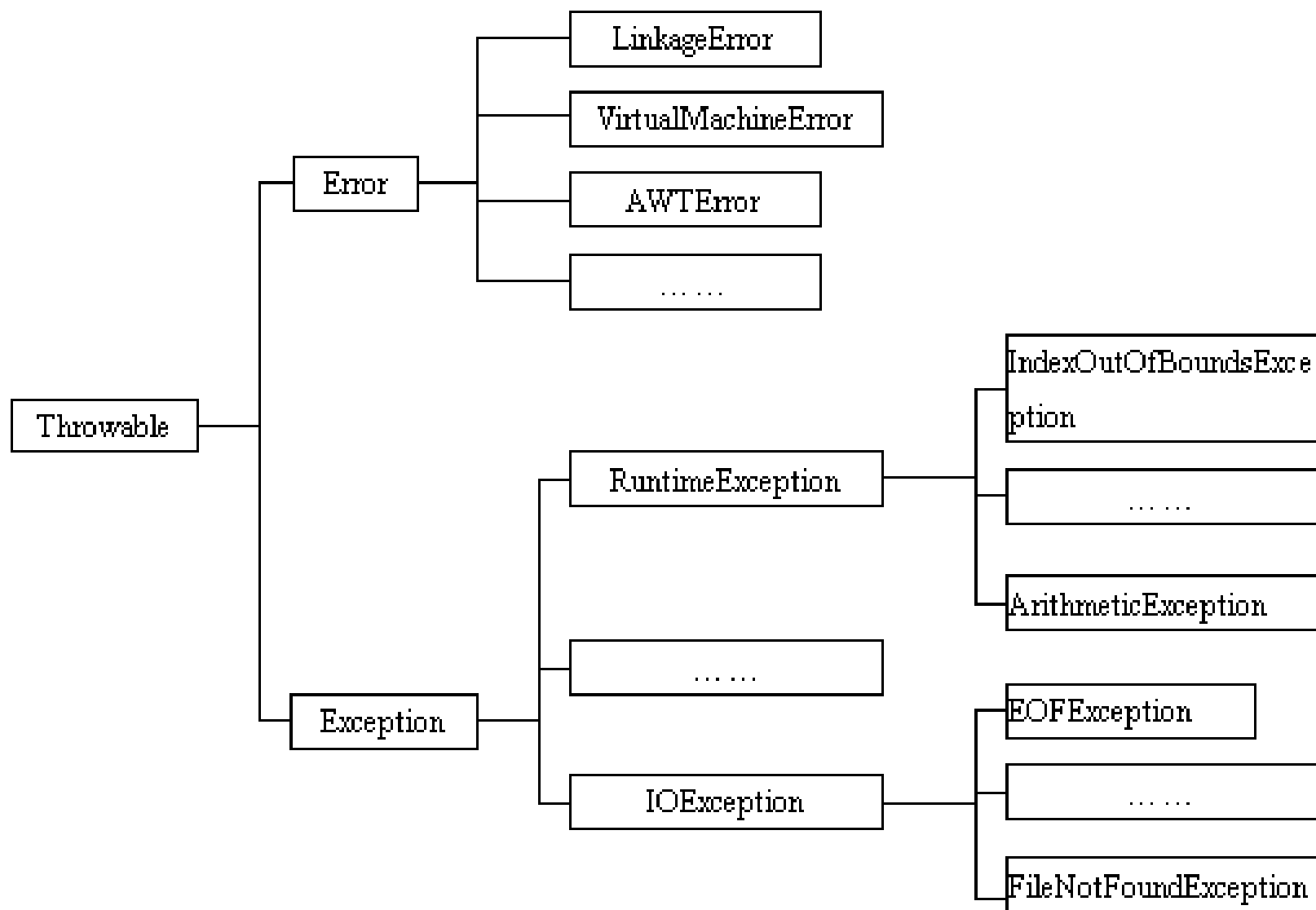


- 数组越界异常
- 空指针异常
- 算数异常
- 类型转换异常

Java异常分类

- ◆ java异常是java提供的用于处理程序中错误的一种机制
- ◆ 异常是一个对象，它继承于Throwable类，所有的Throwable类的子孙类所产生的对象都是例外（异常）。从Throwable直接派生出的异常类有 Exception 和 Error。

Java异常分类



Java异常分类

◆ **Exception** 所有异常类的父类，其子类对应了各种各样可能出现的异常事件，一般需要程序员进行处理。

- **RuntimeException** 称为运行时异常，用户可以不必对其进行处理，可以有系统自动检测并将它们交给缺省的异常处理程序。
- 如果出现 **RuntimeException**，那么一定是程序员的错误。例如，可以通过检查数组下标和数组边界来避免数组越界访问异常。

◆ **Error** 称为错误，由 java 虚拟机生成并抛出。程序对其不做处理

Java异常处理

```
class Exc {  
    public static void main(String args[]) {  
        int d = 0;  
        Int a = 42 / d;  
    }  
}
```

Java运行时系统检查到被零除的情况，它构造一个新的异常对象然后引发该异常。这导致**Exc**的执行停止，因为一旦一个异常被引发，它必须被一个异常处理程序捕获并且被立即处理。该例中，我们没有提供任何我们自己的异常处理程序，所以异常被**Java**运行时系统的默认处理程序捕获。任何不是被你程序捕获的异常最终都会被该默认处理程序处理。

Java异常处理

◆异常处理语句

异常处理语句有try、catch、finally、throw和throws。

◆异常处理的基本结构

```
try {  
    //包含可能会出现异常的语句  
} catch (SomeException1 ex1) {  
    //捕获并处理异常  
} catch (SomeException2 ex2) {  
    //捕获并处理异常  
} finally {  
    //无论是否发生异常均执行  
}
```

说明：可以有一个或者多个catch代码块，每个catch代码块声明其能处理的一种特定类型的异常并提供处理的方法

finally代码块无论是否发生异常都有执行
允许你修正错误。防止程序自动终止。

Java异常处理

◆ try、catch和finally语句说明

● try

捕获异常的第一步是用try {...} 语句块选定捕获异常的范围。

● catch

在catch语句块中是对异常对象进行处理的代码，每个try语句块可以伴随一个或多个catch语句，用于处理可能产生的不同类型的异常对象。与其它对象一样，可以访问一个异常对象的成员变量或调用它的方法。

- getMessage() 方法，用来得到有关异常事件的信息

- printStackTrace() 用来跟踪异常事件发生时执行堆栈的内容。

● finally

- 捕获异常的最后一步是通过finally语句为异常处理提供一个统一的出口，使得在控制流转到程序的其它部分以前，能够对程序的状态作统一的管理。不论在try代码块中是否发生了异常事件，finally块中的语句都会被执行。

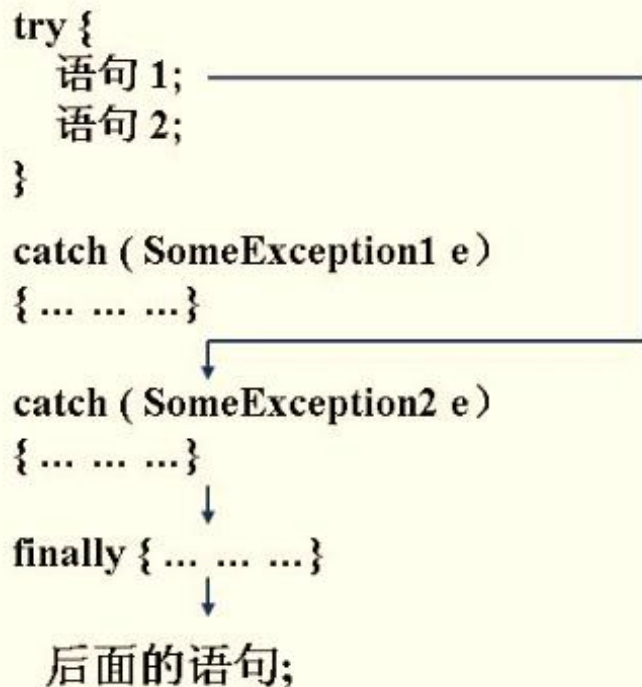
- finally语句是任选的

```
class Exc {  
    public static void main(String args[]) {  
        int d, a;  
        try{  
            d = 0;  
            a = 42 / d;  
            System.out.println("This will not be printed.");  
        } catch (ArithmeticException e) {  
            System.out.println("Division by zero.");  
        }  
        System.out.println("After catch statement.");  
    }  
}
```

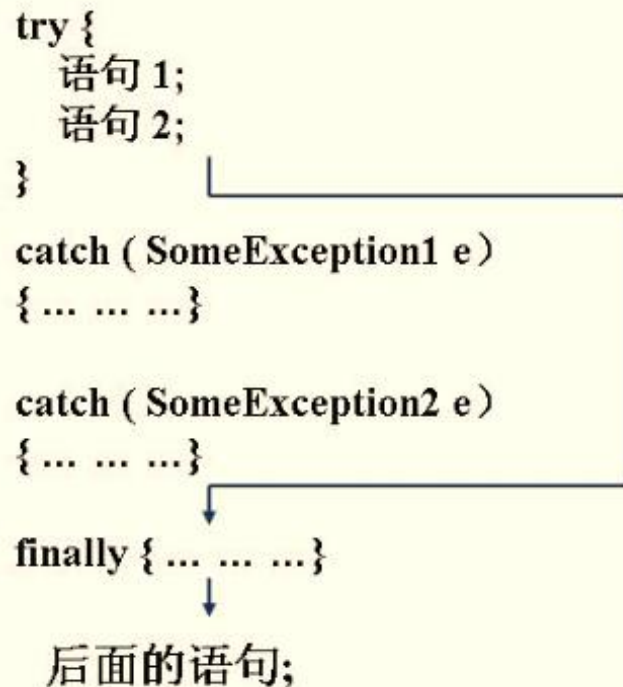
Java异常处理

◆异常处理的执行过程

➤捕获SomeException2时:



➤没有捕获到异常时:



Java异常处理



◆异常处理举例

```
public class Test9_2{
    public static void main(String[] args)    {
        String friends[]={"lisa","bily","kessy"};
        try {
            for(int i=0;i<5;i++) {
                System.out.println(friends[i]);
            }
        } catch (java.lang.ArrayIndexOutOfBoundsException e)  {
            System.out.println("index err");
        }
        System.out.println("\nthis is the end");
    }
}
```

Java异常处理



◆ 异常处理举例

程序Test8_2运行结果: `java Test8_2`

Java异常处理

◆ IOException异常处理举例（未添加捕获）



```
import java.io.*;

public class Test1{
    public static void main(String[] args)    {
        try{
            FileInputStream in=new FileInputStream("myfile.txt");
            int b;
            b = in.read();
            while(b!= -1)        {
                                System.out.print((char)b);
                                b = in.read();
                            }
        } catch(FileNotFoundException e){
            System.out.println( " " );
        } catch(IOException ex){
        }

        in.close();
    }
}
```


Java异常处理



◆ IOException异常处理结果

- 未报告的异常 `java.io.FileNotFoundException` ; 必须被捕获或被声明抛出

```
FileInputStream in=new FileInputStream("myfile.txt");
```

- 未报告的异常 `java.io.IOException` ; 必须被捕获或被声明抛出

```
b = in.read();
```

- `java.io.IOException` ; 必须被捕获或被声明抛出

```
b = in.read();
```

- 未报告的异常 `java.io.IOException` ; 必须被捕获或被声明抛出

```
in.close();
```

4 个错误

Java异常处理



◆ IOException异常处理举例（添加捕获）

```
import java.io.*;
public class Test2 {
    public static void main(String[] args) {
        try {
            FileInputStream in = new FileInputStream("myfile.txt");
            int b;    b = in.read();
            while (b != -1) {
                System.out.print((char)b);
                b = in.read();
            }
            in.close();
        } catch (IOException e) {
            System.out.println(e);
        } finally {
            System.out.println(" It' s ok!");
        }
    }
}
```



一、编译、运行应用程序**Test1.java/Test2**，体会java语言中异常的捕捉和处理机制。

◆相关知识：**FileInputStream**类的成员方法**read()**的功能是每次从相应的(本地为ASCII码编码格式)文件中读取一个字节，并转换成0~255之间的int型整数返回，到达文件末尾时则返回-1。

Java异常处理—抛出异常

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。多个异常可使用逗号分割。

throws语句的语法格式为：

```
methodname throws Exception1,Exception2,...,Exception  
N  
{  
}
```

方法名后的**throws**

Exception1,Exception2,...,ExceptionN 为声明要抛出的异常列表。当方法抛出异常列表的异常时，方法将不对这些类型及其子类类型的异常作处理，而抛向调用该方法的方法，由他来处理。

Java异常处理

◆声明抛出异常是Java中处理异常的第二种方式

- 指明方法中可能要产生的异常类型，由调用者进行异常处理。

```
public class Excep {
    public int division(int a,int b) throws Exception
    {
        return a/b;
    }

    public int fn1(int a,int b) throws Exception
    {
        return division(a,b);
    }
}
```

告诉方法调用者会发生异常

```
public class ExcepTest7 {
    public static void main(String[] args) {
        .....
        try {
            excep.fn1(a, b);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }
}
```

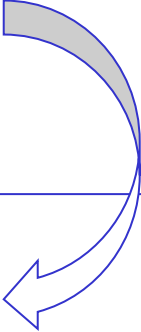
由调用者捕获异常

Java异常处理

- ◆异常是抛给调用者的，向上一级抛出，如果还有上一级再往上抛出
- ◆还可以将异常通过main函数抛给java运行时系统处理

```
public class ExceptTest7 {
    public static void main(String[] args) throws Exception{
        .....
        except.fn1(a, b);
    }
}
```

通过main函数
抛给java运行
时系统处理



```
<terminated> ExceptTest7 [Java Application] C:\Program Files\Java\jdk1.6.0_10\bin\javaw.exe (2011)
请输入第一个数:
5
请输入第二个数:
0
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at ch13.javaoo.example7.Excep.division(Excep.java:6)
    at ch13.javaoo.example7.Excep.fn1(Excep.java:11)
    at ch13.javaoo.example7.ExceptTest7.main(ExceptTest7.java:13)
```

Java异常处理

`throw`总是出现在函数体中，用来抛出一个`Throwable`类型的异常。**程序会在`throw`语句后立即终止，它后面的语句执行不到**，然后在包含它的所有`try`块中（可能在上层调用函数中）从里向外寻找含有与其匹配的`catch`子句的`try`块。

我们知道，异常是异常类的实例对象，我们可以创建异常类的实例对象通过`throw`语句抛出。该语句的语法格式为：

```
throw new exceptionname;
```

例如抛出一个`IOException`类的异常对象：

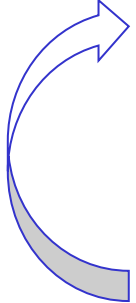
```
throw new IOException;
```

也可直接抛出异常：`throw e;`

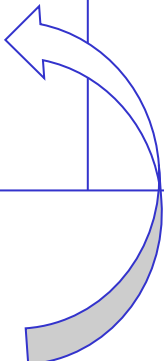
Java异常处理

◆利用throw关键字抛出异常对象

```
public class Excep {  
    public int division(int a, int b) throws Exception  
    {  
        try {  
            return a / b;  
        } catch (Exception ex) {  
            throw ex;  
        }  
    }  
    public int fn1(int a, int b) throws Exception {  
        return division(a, b);  
    }  
}
```



```
public class ExcepTest8 {  
    public static void main(String[] args) {  
        .....  
        try {  
            excep.fn1(a, b);  
        } catch (Exception ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

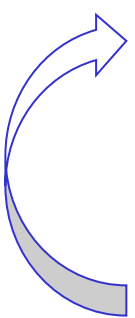


Java异常处理

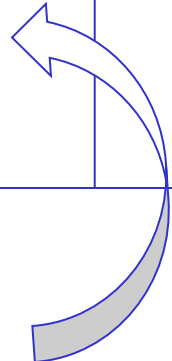
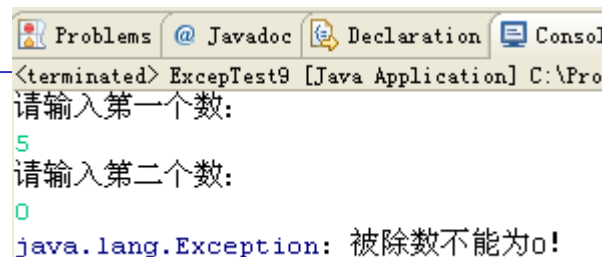
◆可以抛出异常时new一个新的异常

```
public class Excep {
    public int division(int a, int b) throws Exception
    {
        try {
            return a / b;
        } catch (Exception ex) {
            throw new Exception("被除数不能为0! ");
        }
    }

    public int fn1(int a, int b) throws Exception {
        return division(a, b);
    }
}
```



```
public class ExcepTest9 {
    public static void main(String[] args) {
        .....
        try {
            excep.fn1(a, b);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }
}
```

```
Problems @ Javadoc Declaration Console
<terminated> ExcepTest9 [Java Application] C:\Pro
请输入第一个数:
5
请输入第二个数:
0
java.lang.Exception: 被除数不能为0!
```

自定义异常

◆ Java异常类对象除在程序执行过程中出现异常时由系统自动生成并抛出，也可根据需要人工创建并抛出

- 首先要生成异常对象，然后通过throw语句实现抛出操作（提交给Java运行环境）。

```
IOException e =new IOException();  
throw e;
```

- 可以抛出的异常必须是Throwable或其子类的实例。下面的语句在编译时将会产生语法错误：

```
throw new String("want to throw");
```

自定义异常

- ◆用户自定义异常类MyException，用于描述数据取值范围错误信息：

```
class MyException extends Exception {  
    private int idnumber;  
    public MyException(String message, int id) {  
        super(message);  
        this.idnumber = id;  
    }  
    public int getId() {  
        return idnumber;  
    }  
}
```

自定义异常

◆使用用户自定义异常类MyException :

```
public class Lab8_5{
    public void regist(int num) throws MyException {
        if (num < 0) {
            throw new MyException("人数为负值，不合理", 3);
        }
        System.out.println("登记人数" + num);
    }
    public void manager() {
        try {
            regist(100);
        } catch (MyException e) {
            System.out.print("登记失败，出错种类"+e.getId());
        }
        System.out.print("本次登记操作结束");
    }
    public static void main(String args[]) {
        Lab8_5 t = new Lab8_5();
        t.manager();
    }
}
```

Java常见异常

runtimeException子类:

1、java.lang.ArrayIndexOutOfBoundsException

数组索引越界异常。当对数组的索引值为负数或大于等于数组大小时抛出。

2、java.lang.ArithmeticException

算术条件异常。譬如：整数除零等。

3、java.lang.NullPointerException

空指针异常。当应用试图在要求使用对象的地方使用了null时，抛出该异常。譬如：调用null对象的实例方法、访问null对象的属性、计算null对象的长度、使用throw语句抛出null等等

4、java.lang.ClassNotFoundException

找不到类异常。当应用试图根据字符串形式的类名构造类，而在遍历CLASSPATH之后找不到对应名称的class文件时，抛出该异常。

5、java.lang.NegativeArraySizeException 数组长度为负异常

6、java.lang.ArrayStoreException 数组中包含不兼容的值抛出的异常

7、java.lang.SecurityException 安全性异常

8、java.lang.IllegalArgumentException 非法参数异常

Java常见异常

IOException

IOException: 操作输入流和输出流时可能出现的异常。

EOFException 文件已结束异常

FileNotFoundException 文件未找到异常

其他

ClassCastException 类型转换异常类

ArrayStoreException 数组中包含不兼容的值抛出的异常

SQLException 操作数据库异常类

NoSuchFieldException 字段未找到异常

NoSuchMethodException 方法未找到抛出的异常

NumberFormatException 字符串转换为数字抛出的异常

StringIndexOutOfBoundsException 字符串索引超出范围抛出的异常

IllegalAccessException 不允许访问某类异常

InstantiationException 当应用程序试图使用Class类中的newInstance()方法创建一个类的实例，而指定的类对象无法被实例化时，抛出该异常



一、编程：接收从键盘输入的工资信息如果接受工资为负数的话，就抛出异常（自己的异常类SalaryException）。

总结

- ◆ Java异常概念
- ◆ Java异常分类
- ◆ 异常的处理
- ◆ 自定义异常



计算机学院



谢谢！