

第5章 异常处理

- 5.1 异常处理基础
- 5.2 异常处理措施



第5章 异常处理



内容和要求:

1. 发现各种程序错误，采取不同的手段排除错误。
2. 理解异常处理机制的运行方式，掌握**Java**异常的抛出、捕获及处理方法。
3. 熟悉自定义异常在程序设计中的作用。

重点：掌握**try**语句捕获异常并处理。

难点：捕获并处理异常，抛出异常、使异常在方法间传递。



5.1 异常处理基础

5.1.1 异常处理机制的必要性

1. 面向过程语言错误处理方式的缺陷

- ① 不进行范围检查
- ② 采用**if**语句进行事先判断以防止出现错误。

2. 面向对象语言异常处理的思想

- ① 将程序正常代码与错误处理代码分开。
- ② 使程序具有处理错误的能力。



3. Java语言是安全的

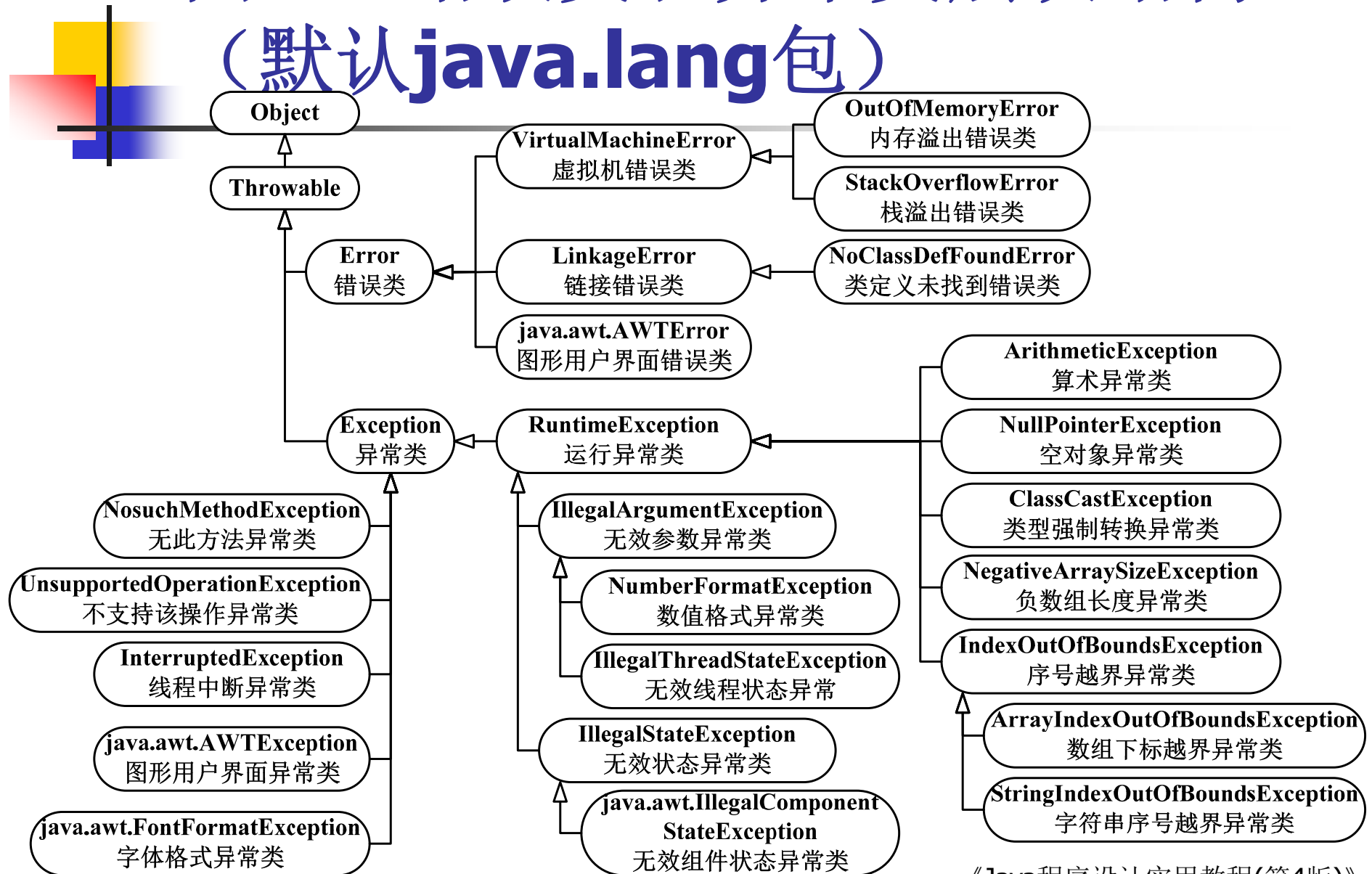
- ① **Java**语言提供严密的语法规则。
- ② **Java**在编译和运行时严格检查错误。
- ③ **Java**提供异常处理机制。
- ④ **Java**提供内存自动管理方式。



5.1.2 错误和异常

1. **错误**（**error**）指程序运行时遇到的硬件或操作系统的错误。 **Error**错误类
2. **异常**（**exception**）指在硬件和操作系统正常时，程序遇到的运行错。 **Exception**异常类

图5.1 错误类和异常类层次结构 (默认java.lang包)





异常类说明

```
public class Throwable implements Serializable
{
    public String getMessage()    //获得异常信息
    public String toString()      //获得异常对象的描述信息
    public void printStackTrace() //显示异常栈跟踪信息
}
```

```
public class Exception extends Throwable
{
    public Exception()
    public Exception(String s)
}
```



3. RuntimeException运行异常类

1. **ArithmeticException** 算术异常：除数为0

2. **NullPointerException** 空对象异常

```
int a[] = null;
```

```
a[0] = 1; //对空数组中的元素进行操作
```

```
String str = null;
```

```
str.length() //调用空对象的方法
```

3. **ClassCastException** 类型强制转换异常

```
Object obj = new Object();
```

```
String str = (String) obj;
```

问：什么情况下能够进行类型强制转换？

答：只有当**obj**引用**String**实例时，**Object obj = "abc";**

3. RuntimeException运行异常类

4. NegativeArraySizeException 负数组长度异常

```
int a[] = new int [-1];
```

5. ArrayIndexOutOfBoundsException 数组下标越界异常

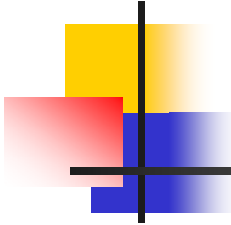
```
int a[] = new int[1];  
a[1] = 1;
```

6. StringIndexOutOfBoundsException 字符串序号越界异常

```
"abc".charAt(-1)
```

7. NumberFormatException 数值格式异常

```
int j = Integer.parseInt("abc");
```



4. 程序对错误与异常的三种处理方式

1. 程序不能处理错误
2. 程序应避免而不捕获的异常，如除数为**0**、数组下标越界等。
3. 必须捕获的异常



5.2 异常处理措施

1. 5.2.1 异常处理语句
2. 5.2.2 抛出异常
3. 5.2.3 自定义异常类



5.2.1 异常处理语句

1. 异常处理语句语法

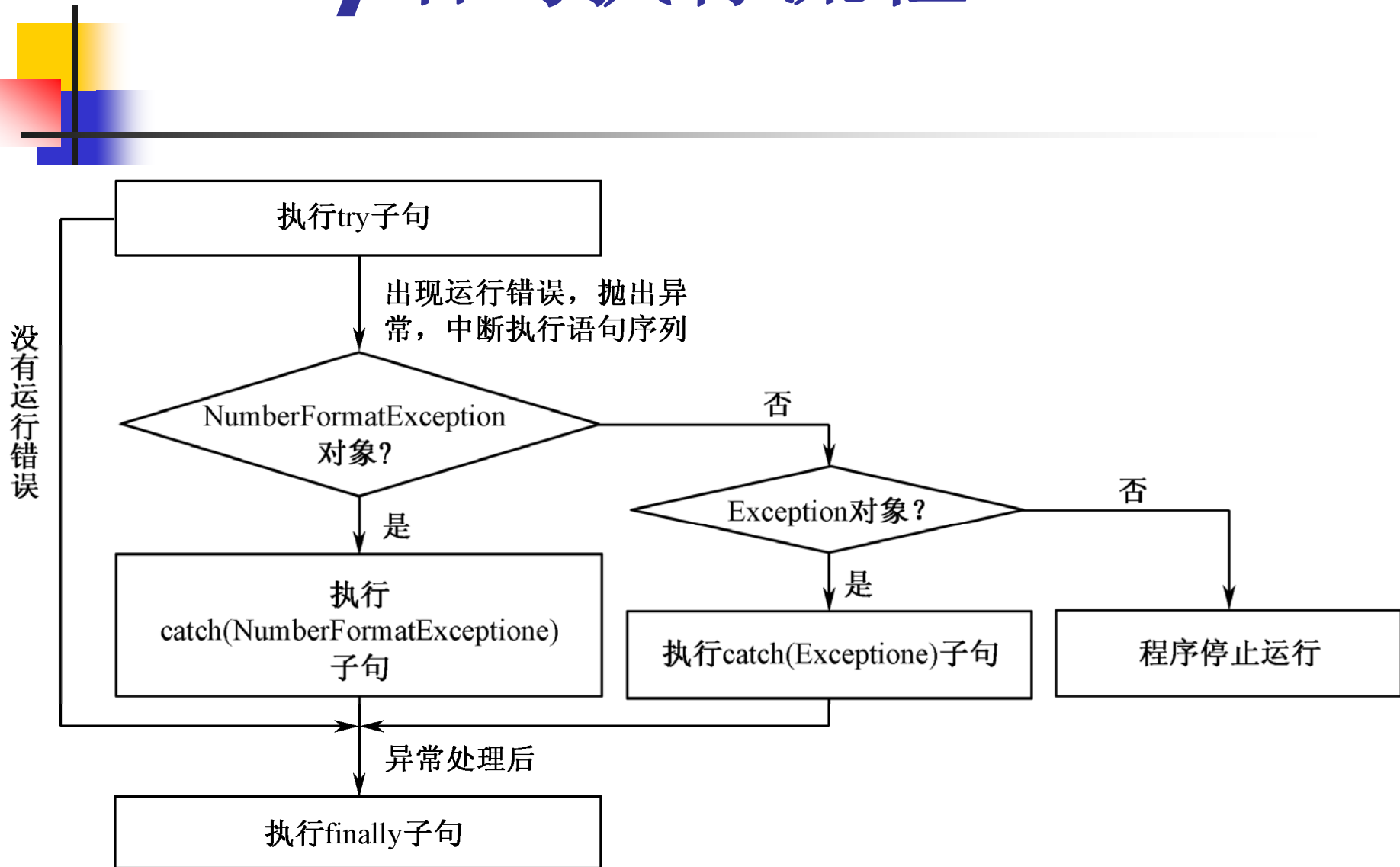
```
try
{
    语句1                //存在潜在异常的代码
}
catch (异常类 异常对象)
{
    语句2                //捕获到异常并处理的代码
}
finally
{
    语句3                //最后必须执行的代码，无论是否捕获到异常
}
```



调用**parseInt()**方法的异常处理语句

```
String str = "123a";  
try  
{  
    int i = Integer.parseInt(str);    //调用声明抛出异常的方法  
}  
catch(NumberFormatException ex) //捕获异常对象  
{  
    System.out.println(str+"字符串不能转换为整数");  
}  
catch (Exception ex)                //捕获所有异常对象  
{  
    e.printStackTrace();            //显示异常栈跟踪信息  
}
```

2. try语句执行流程





【例5.1】 求数组元素的平均值。

1. 存在空对象异常和除数为**0**的潜在错误。
2. 程序改写,
 - ① **weightedAverage(value[], weight[])**方法求**value**数组元素的加权平均值, 采用**if**语句对可能出现的运行时错误进行事先处理, 避免除数为**0**的运行错误。
 - ② **toIntArray(str[])**方法获得字符串数组中的整数。采用异常处理语句对产生的运行时错误进行事后处理。先按十进制形式转换成整数, 不能转换时, 再按十六进制形式转换成整数, **try**语句嵌套。



5.2.2 抛出异常

1. 抛出自定义异常对象的**throw**语句

throw 异常对象

```
public void set(int year, int month, int  
    day)
```

```
{
```

```
    if (month<1 || month>12)
```

```
        throw new Exception("月份错误");
```

```
}
```




2. 方法声明抛出异常的**throws**子句

[修饰符] 返回值类型 方法([参数列表]) [**throws** 异常类]

```
public static int parseInt(String s) throws  
    NumberFormatException
```

日期类声明抛出异常的方法与方法调用者处理异常。

```
public void set(int year, int month, int day) throws Exception  
public MyDate(int year, int month, int day) throws Exception  
{  
    this.set(year, month, day);  
}  
public static void main(String args[]) throws Exception
```



5.2.3 自定义异常类

```
catch(Exception ex)  
{  
    if (e.toString().equals("月份错误"))  
}
```



实验5 异常的抛出、捕获并处理

- 目的：理解异常处理机制。
- 要求：发现各种程序错误，采取不同的手段排除错误。
- 重点：掌握**try**语句捕获异常并处理。
- 难点：捕获并处理异常，抛出异常、使异常在方法间传递。