一、异常捕获与处理

（一）异常处理

- 处理异常需要知道3件事：
  - 哪里发生了异常？
  - 发生了什么异常？
  - 如何处理异常？

1.异常处理机制

①一旦产生异常，首先会产生一个异常类的实例化对象；

②在try语句中对此异常对象进行捕捉；

③捕捉后将产生的异常对象与catch语句中的各个异常类型进行匹配（try部分剩下的语句就不执行了），如果匹配成功，则可执行响应catch语句中的代码

④最后执行finally语句（异常统一出口）

⑤然后执行try ....catch后面的正常程序部分（而异常处理不能影响正常的执行路径 ）

2.程序异常处理语句

```
try{
    有可能出现异常的语句
}catch(异常类 异常对象){
    异常处理语句
}catch(异常类 异常对象){
    异常处理语句
}finally{
    一定要运行的程序代码
}
```

```
public class job{
    public static void main(String[] args){
        int i = 0;
        int j = 0;
        try{
            String str1 = args[0];
            String str2 = args[1];
            i = integer.parselnt(str1);//将字符串转换成整型
            j = integer.parselnt(str2);
            int temp =i/j; //此处产生异常
            System.out.println("两个数字相除结果："+temp);  //该代码不再执行

        }catch(ArithmeticException e){
            System.out.println("算术异常："+e);
        }catch(NumberFormatException e){
            System.out.println("数字转换异常："+e);
        }catch(ArrayIndexOutOfBoundsException e){  //小范围异常
```

```
            System.out.println("数组越界异常："+e);
        }catch(Exception e){ //大范围异常（这是是上溯，将子类对象赋给父类句柄，任何子类错误都会被判断出来
            System.out.println("其他异常："+e);
        }finally{
            System.out.println("异常处理完成");//即使没有异常也会运行的程序
        }
        System.out.println("计算结束");
    }
}
```

注意：

看起来"异常处理完成"与 "计算结束"语句都是在捕捉到异常后执行的，但是**如果在catch中包含return返回语句，那么finally中的语句还是会被执行，而"计算结束"语句则不会被执行 。**

捕捉范围小的程序必须放在捕捉范围大的异常处理程序后面，否则会报错

Exception表示最大范围的捕捉，可以放在最后面囊括所有，但是建议使用多个异常处理程序分别进行捕获，不采用眉毛胡子一把抓的笼统捕获方式

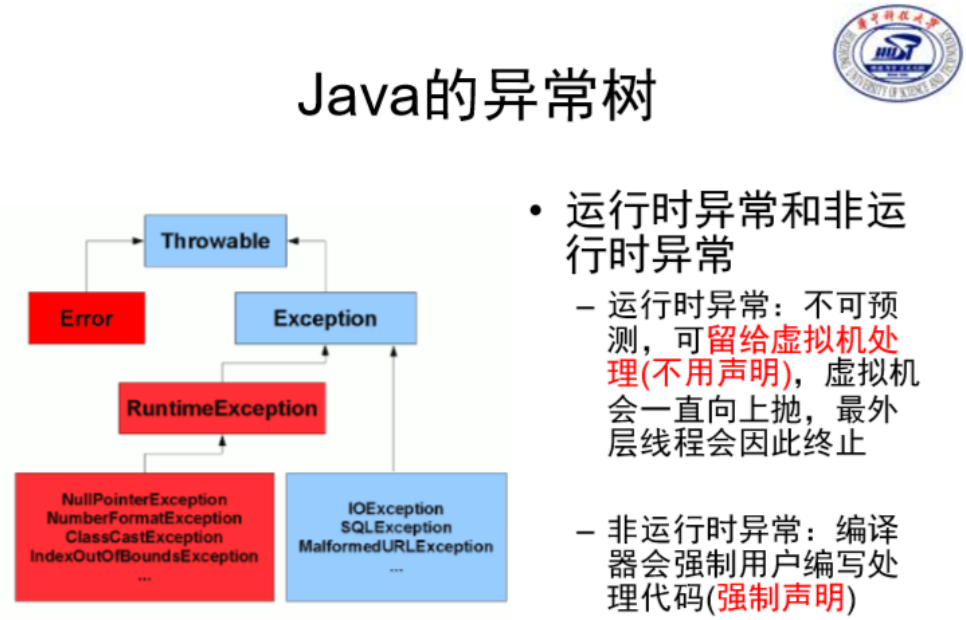
3.异常类的继承结构

java异常结构中常用Exception与Error类，它们都是Throwable的子类。区别是

①Exception一般表示的是程序中出现的问题（比如算术异常、数学格式化异常等），可以直接使用try...catch处理；(程序中的问题,可预知的): Java编译器要求Java 程序必须捕获或声明所有的**非运行时** 异常

②Runtime Exception一般指系统检测，例如被0除等系统错误，数组下标超范围等，用户的java程序可不做处理，系统将它们交给缺省的异常处理程序；

③Error由Java虚拟机生成并抛出,Java程序不做处理 。



在打印异常对象时，除了使用System.out.println()，还可以使用**e.printStackTrace()** ;后者输出信息更加详细

4.异常捕捉的序列

如果方法之间、类之间存在调用关系，在任何一个方法中发生异常，整个异常传递序列都会被保存在异常堆栈中，而通过**e.printStackTrace()** 可以将异常堆栈打印下来

```
E:\学习PPT\大三上课件\java\code5>java Exp3
捕获异常：java.lang.NullPointerException
java.lang.NullPointerException
        at Exp3.method1(Exp3.java:6)
        at Exp3.method2(Exp3.java:9)
        at Exp3.method3(Exp3.java:12)
        at Exp3.main(Exp3.java:17)
执行扫尾工作
```

其中method1是最内层（发生错误的层），main是最外层

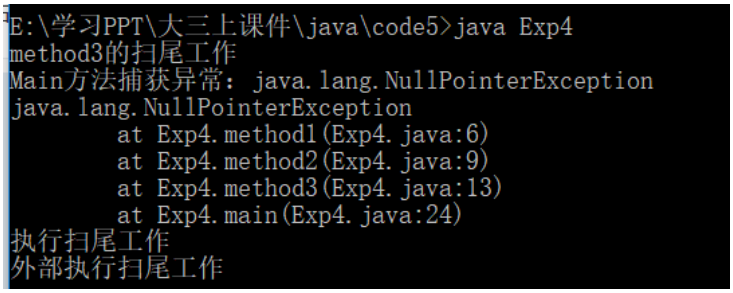
(二) 异常抛出

1.throws与throw关键字

抛出异常分两步：创建某个异常类的事例throws声明异常，然后用throw语句抛出

```
import java.io.*;

public class Exp4{
    String str;
    public void method1(){
        str.length();
    }
    public void method2(){
        method1();
    }
    public void method3() throws Exception{
        try{
            method2();
        }catch(Exception e){
            //System.out.println("Method3方法捕获异常："+e.toString());
            throw e;
        }finally{
            System.out.println("method3的扫尾工作");
        }
    }
    public static void main(String[] args){
        Exp4 exp = new Exp4();
        try{
            exp.method3();
        }catch(Exception ex){
            System.out.println("Main方法捕获异常："+ex.toString());
            ex.printStackTrace();
            //return;
        }finally{
            System.out.println("执行扫尾工作");
        }
        System.out.println("外部执行扫尾工作");
    }
}
```



当抛出异常时！！！！

如果需要追踪到一个完整的异常栈，就把原始的Exception实例传进新生成的Exception。