异常根类：java.lang.Throwable

子类：java.lang.Error

java.lang.Exception,

平常所说的异常指java.lang.Exception

Exception:编译期异常，进行编译写代码java程序出现的问题

RuntimeException运行期异常，java程序运行过程中出现的问题

异常发生：

1、jvm根据异常产生的原因创建一个异常对象，包含异常产生的内容原因位置

2、出错方法没有异常的处理逻辑，jvm将异常对象抛出给方法的调用者

3、main方法接收到异常对象，没有异常处理逻辑，将对象抛出给main方法的调用者jvm处理

4、jvm接收到异常对象，将异常对象以红色字体打印在控制台

5、jvm终止正在执行的java程序

异常处理：

关键字throw：

throw new xxxException（"异常产生的原因"）

注意：

throw关键字必须写在方法的内部

throw关键字后边new的对象必须是Exception或者Exception的子类对象

throw关键字抛出指定的异常对象，必须处理这个异常对象

throw关键字后边创建的是RuntimeException或者是RuntimeException的子类对象，可以不处理，默认交给jvm处理

throw关键字后边创建的是编译异常，必须处理异常，要么throws，要么try catch

工作中必须对方法传递过来的参数进行合法性校验

如果参数不合法，那么必须使用抛出异常的方式，告知方法的调用者，传递的参数有问题

**public class** Demo01Exception {

**public static void** main(String[] args) {

**int**[] arr=**null**;

**int** e=*getElement*(arr,2);

}

**public static int** getElement(**int**[] arr,**int** index){

**if** (arr==**null**){

**throw new** NullPointerException(**"传递的数组是空"**);

}

*/\*数组索引越界异常\*/*

**if** (index<0||index>arr.**length**-1){

**throw new** ArrayIndexOutOfBoundsException(**"越界"**);

}

**int** ele=arr[index];

**return** ele;

}

}

Objects非空判断

**import** java.util.Objects;

**public class** Demo01Exception {

*/\**

*Objects类中的静态方法*

*public static <T> T requireNull(T obj)查看指定引用对象不是null*

*源码*

*public static <T> T requireNonNull(T obj)*

*if(obj==null)*

*throws new NullPointerException();*

*return obj;*

*\* \*/*

**public static void** main(String[] args) {

*method*(**null**);

}

**public static void** method(Object obj){

Objects.*requireNonNull*(obj);

Objects.*requireNonNull*(obj,**"传递的对象的值是null"**);

}

}

声明异常throws

将问题标识出来，报告给调用者，如果方法内通过throw抛出编译时异常而没有捕获处理，那么必须通过throws进行声明，让调用者去处理

声明异常格式：

修饰符+返回值类型+方法名（参数）+throws 异常类名1，异常类名2{}

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.Objects;

*/\*throws关键字：异常处理的第一种方式，交给别人处理*

*作用：*

*方法内部抛出异常对象的时候，必须处理异常对象*

*可以使用throws关键字处理异常对象，会把异常对象声明抛出方法的调用者处理，最终交给jvm处理*

*throws必须写在方法声明处*

*throws关键字声明的异常必须是exception或者是Excep的子类*

*方法内部如果抛出多个异常对象，throws也必须声明多个异常*

*如果抛出的多个异常对象有子父类关系，直接声明父类异常即可*

*调用一个声明抛出异常的方法，就必须处理声明的异常*

*要么继续使用throws声明抛出，交给方法调用者处理，最终交给jvm*

*要么try catch自己处理*

*\* \*/*

**public class** Demo01Exception {

*/\*FileNotFoundException extends IOException,可以省略filenot异常\*/*

**public static void** main(String[] args)**throws** IOException{

*readLine*(**"c:\\a.text"**);;

}

**public static void** readLine(String filename) **throws** IOException {

**if** (!filename.equals(**"c:\\a.txt"**)){

**throw new** FileNotFoundException(**"传递的文件路径不对"**);*//编译异常*

}

**if** (filename.endsWith(**".txt"**)){

**throw new** IOException(**"文件的后缀名不对"**);

}

System.***out***.println(**"路径没有问题，读取文件"**);

}

}

**import** java.io.IOException;

**public class** Demo01Exception {

*/\*try catch:异常处理的第二种方式，自己处理异常*

*try{*

*可能产生异常的代码*

*}catch（定义一个异常变量，用来接收try中抛出的异常对象）{*

*异常处理的逻辑，异常对象之后怎么处理异常对象*

*}*

*\* 一般在工作中，会把异常的信息记录到一个日志中*

*\* 注意：try中可能抛出多个异常对象，就可以使用多个catch处理这些异常毒性*

*\* 如果try中产生了异常，就会执行catch中的异常处理逻辑执行完毕catch中的处理逻辑，继续执行try catch之后的代码块*

*\* 如果try没有产生异常，就不好执行catch中的处理逻辑*

*\* \*/*

**public static void** main(String[] args) {

**try**{

*readLine*(**"d:\\a.txt"**);

}**catch** (IOException e){

System.***out***.println();

}

}

**public static void** readLine(String fileName)**throws** IOException{

**if** (!fileName.endsWith(**".txt"**)){

**throw new** IOException(**"文件后缀名"**);

}

}

}

Throwable类中定义了一些查看方法：

public String getMessage():获取异常的描述信息，原因，简短描述

public String toString()：获取异常的类型和异常描述信息，详细信息字符串

public void printStackTrace()打印异常的跟踪栈信息并输出到控制台，异常信息最全面，jvm打印异常对象默认此方法

e.toString()