Java的异常机制主要依赖于try、catch、finally、throw和throws五个关键字

Try：放置可能引发异常的代码

Catch：用于处理这种类型的代码块

Finally：用于回收在try块里打开的物理资源，异常机制会保证finally总被执行

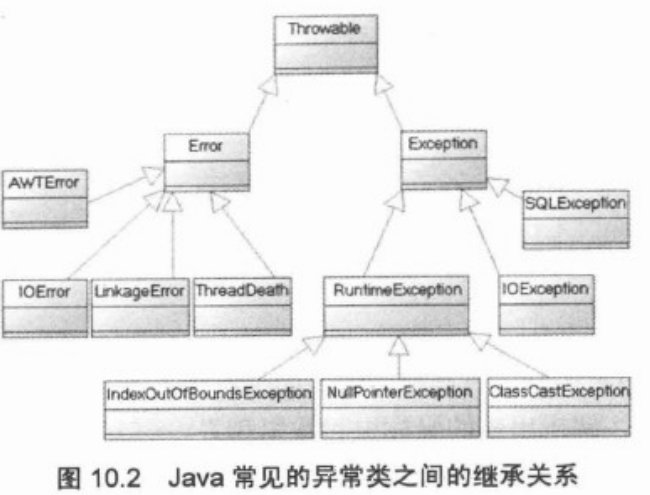
Throws：在方法签名中使用，用于声明该方法可能抛出的异常

Throw：抛出一个实际的异常，可以单独使用，抛出一个具体的异常对象

如果执行try块里的业务逻辑代码出现异常，系统自动生成一个异常对象，该异常对象呗提交给java运行时环境，这个过程被称为抛出（throw）异常

注意：不管程序代码块是否处于try块中，升职包括catch块中的代码，只要执行该代码块时出现了异常，系统总会自动生成一个异常对象，如果程序没有为这段代码定义任何的catch块，则java运行时环境无法找到处理该异常的catch块，程序就会在此退出

在try块里声明的变量是代码块内局部变量，它只在try块内有效，在catch块中不能访问该变量



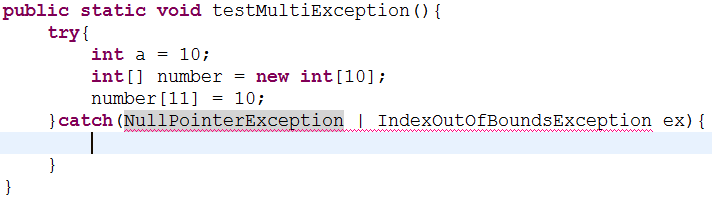
Java把所有的非正常情况分成两种：异常（Exception）和错误（Error），它们都继承Throwable父类

Error错误：一般是指与虚拟机相关的问题，如系统崩溃、虚拟机错误、动态链接失败等，这种错误无法恢复或不可能捕获，将导致应用程序中断。通常应用程序无法处理这些错误，因此引用程序不应该视图使用catch块来捕获Error对象

**Java7提供的多异常处理**

在java7以前，每个catch块只能捕获一种类型的异常，但从java7开始，一个catch块可以捕获多种类型的异常。使用一个catch块捕获多种类型异常需要注意：

1. 捕获多种类型的异常时，多种异常类型之间用竖线（|）隔开
2. 捕获多种类型异常时，异常变量有隐式的final修饰，因此程序不能对异常变量重新赋值

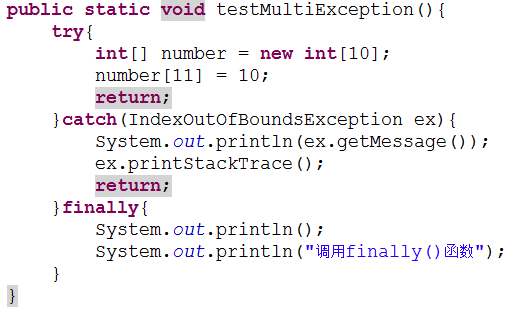


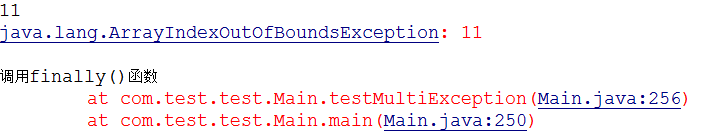
在该eclipse中会提示多种异常在java1.7之后才能使用的提示（已测试）

**访问异常情况**

所有的异常对象都包含了如下几个常用方法：

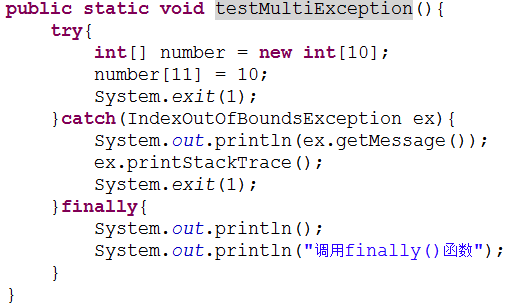
1. getMessage：返回该异常的详细描述字符串
2. printStackTrace：将该异常的跟踪栈信息输出到标准错误输出
3. printStackTrace(PrintStram s)：将该异常的跟踪栈信息出处到指定输出流
4. getStackTrace：返回该异常的跟踪栈信息

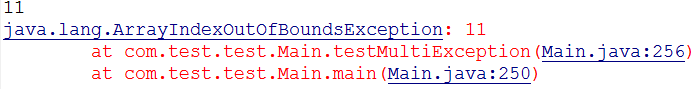




在上面的例子中可以发现：即使调用了return语句，finally也会执行

如果将上面的代码修改为：

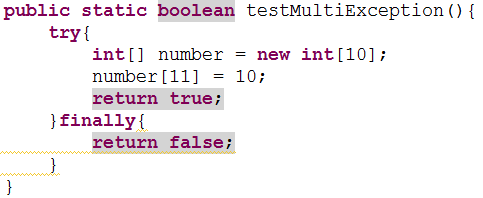




System.exit(1)表示退出虚拟机则不会执行finally代码块

注意：除非在try块、catch块中调用了退出虚拟机的方法，否则不管在try块、catch块中执行怎样的代码，出现怎样的情况，异常处理的finally块总会被执行。

在通常情况下，不要在finally块中使用如return或throw等导致方法终止饿语句，一旦在finally块中使用了return或throw鱼的，将会导致try块、catch块中的return、throw语句失效。





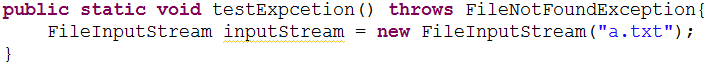


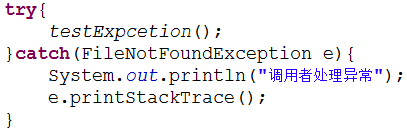
当java程序执行try块、catch块时遇到了return或throw语句，这两个语句都会导致该方法立即结束，但是系统执行这两个语句并不会结束该方法，而是去寻找该异常处理流程中是否包含finally块，如果没有finally块，程序立即执行return或throw语句，方法终止；如果有finally块，系统立即开始执行finally块---只有当finally块执行完成后，系统才会再次跳回来执行try块、catch块里的return或throw语句；如果finally块里也使用了return或throw等导致方法终止的语句，finally块已经终止了方法，系统将不会跳回去执行try块、catch块里的任何代码。

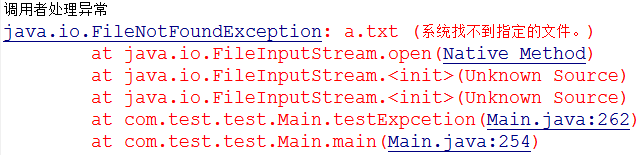
**使用throws声明抛出异常**

使用throws声明抛出异常的思路是，当前方法不知道如何处理这种类型的异常，该异常应该由上一级调用者处理：如果main方法也不知道如何处理这种类型的异常，也可以使用throws声明抛出异常，该异常将交给jvm处理。Jvm对异常的处理方法是，打印异常的跟踪栈信息，并终止程序运行，这就是前面程序在遇到异常后自动结束的原因。

Throws声明抛出只能在方法签名中使用，一旦使用throws语句声明抛出该异常，程序就无须使用try-catch来捕获该异常了。







注意：使用throws声明抛出异常时有一个限制，就是方法重写时“两小”中的一条规则：子类方法声明排除的异常类型应该是父类方法声明抛出的异常类型的子类或相同，子类方法声明抛出的异常不允许比父类方法声明抛出的异常多。

**使用throw抛出异常**

Throw语句抛出的不是异常类，而是一个异常实例，而且每次只能抛出一个异常实例。自己抛出异常后，然后由catch来进行处理