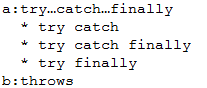
# 处理异常（两种方式）

## 处理异常（两种方式）



## 如何使用异常处理

### 原则

如果该功能内部可以将问题处理，用try；如果处理不了，交由调用者处理,这是用throws，如果JDK没有提供对应的异常，需要自定义异常。

### 区别

后续程序需要继续运行就try

后续程序不需要继续运行就throws

## final,finally和finalize的区别

final可以修饰类,不能被继承

修饰方法,不能被重写

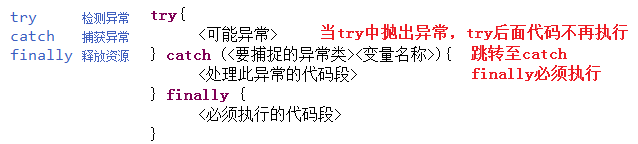
修饰变量,只能赋值一次

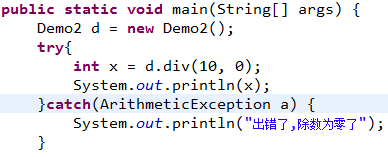
finally是try语句中的一个语句体,不能单独使用,用来释放资源

finalize是一个方法,当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法。

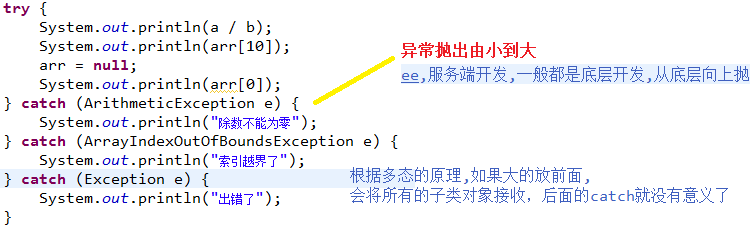
# 检测、捕获

## try，catch





try...catch的方式处理多个异常



## finally

### 特点

无论是否发生异常，被finally控制的语句体一定会执行

特殊情况：在执行到finally之前jvm退出了(比如System.exit(0))

### 作用

当异常发生时，程序可能会意外中断，有些被占用的资源就得不到清理（用于释放资源，在IO流操作和数据库操作中会见到）。finally块可以确保执行所有的清理工作；

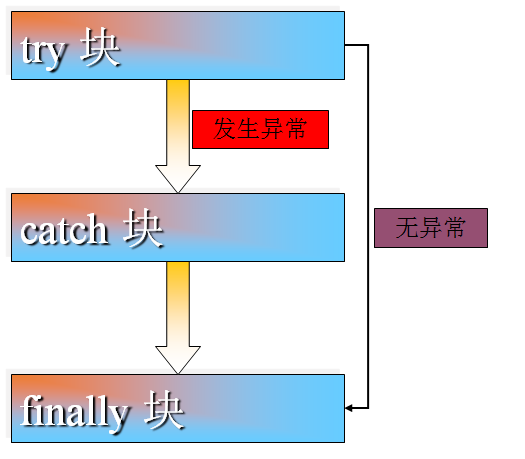
### 注意

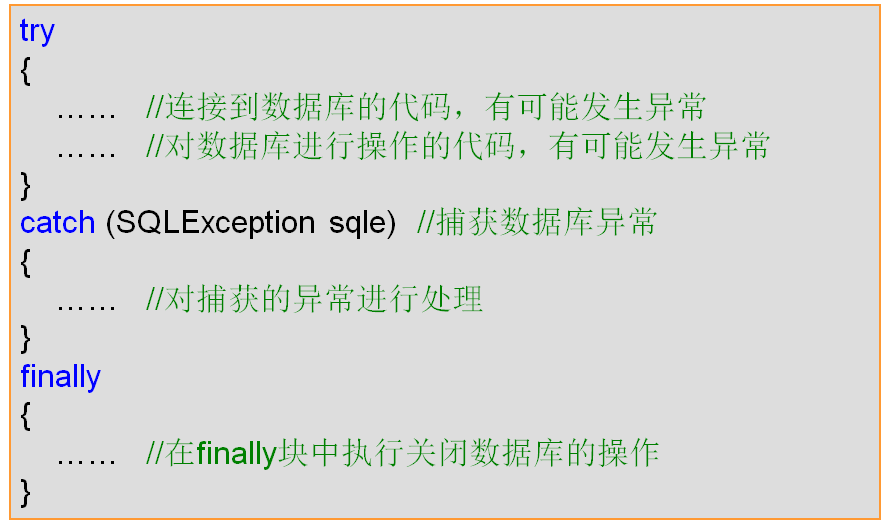
finally块必须和try块一起使用，不能单独存在。是可选的，可视具体情况决定是否添加

千万不要在finally里面写return语句: 如果在这里面写返回语句,那么try和catch的结果都会被改变,所以这么写就是犯罪

## 执行流程

try开始执行，发生异常跳过异常之后的语句，走catch，再走finally；无异常直接走finally





# 多重和嵌套catch

## 多重catch

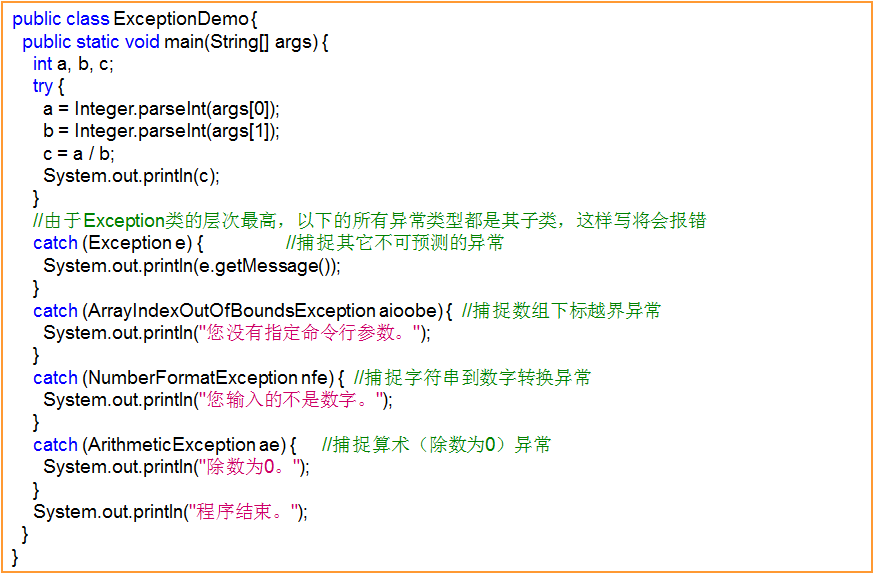
在try块中的代码段将有可能产生多种不同类型的异常，而我们又需要针对不同的异常类型进行不同的处理方式，那么我们就可以使用多重catch块，来分别捕获不同类型的异常。

虽然多重catch块可以同时监视多个不同类型的异常，但是try块中一旦有某个异常产生，程序就会跳转到与之异常类型最匹配的catch块中执行，然后执行finally块（如果有finally块的话）或之后的语句；

也就是说，多重catch块只会捕捉到最先产生的异常，而不是把所有的异常全部捕捉完；

即：不论有多少个catch块，最多只会执行其中的一个；

请注意catch块的书写顺序：类层次越低的越往上写，越高的越往下写。



## 嵌套try/catch块

有时候，整个语句块可以产生异常，而其中的某个部分又可能产生另外的异常，而我们需要分别进行处理；这样，就可以通过嵌套try/catch块来完成；

嵌套try/catch块就是在一个try/catch块中包含有另外的try/catch块。

