[策略模式 1](#_Toc528593374)

[策略的概念 2](#_Toc528593375)

[策略模式的成员 2](#_Toc528593376)

[策略模式类图 2](#_Toc528593377)

[策略模式的示例 2](#_Toc528593378)

[内部类 4](#_Toc528593379)

[成员内部类 4](#_Toc528593380)

[成员内部类的实例化 5](#_Toc528593381)

[静态内部类(静态成员内部类简称) 5](#_Toc528593382)

[静态变量可以是局部变量吗 6](#_Toc528593383)

[局部内部类 6](#_Toc528593384)

[匿名内部类 6](#_Toc528593385)

[常见类 7](#_Toc528593386)

[String StringBuilder StringBuffer 7](#_Toc528593387)

[SimpleDateFormart 7](#_Toc528593388)

[Java中的日期类 7](#_Toc528593389)

[Calendar 8](#_Toc528593390)

[异常 8](#_Toc528593391)

[积极机制 9](#_Toc528593392)

[当finally邂逅了return怎么办 11](#_Toc528593393)

[消极机制 12](#_Toc528593394)

[Throw和Throws的区别 12](#_Toc528593395)

[自定义异常 12](#_Toc528593396)

[课堂练习 14](#_Toc528593397)

[参考 14](#_Toc528593398)

# 策略模式

在四个月的Java学习中.有几个模式是必须掌握的.(Java至少有23中设计模式)

1. 单例
   1. 涉及到线程: 线程安全的单例
   2. 涉及到枚举: 单例的枚举写法
2. 工厂
   1. 简单工厂/静态工厂
   2. 抽象工厂
      1. 生产工厂的工厂
3. 策略
   1. 比如不同的会员等级不同的折扣率就是策略
4. 模板模式
5. 装饰模式
   1. Java的IO就是采用装饰模式
6. 适配模式
7. 代理模式

## 策略的概念

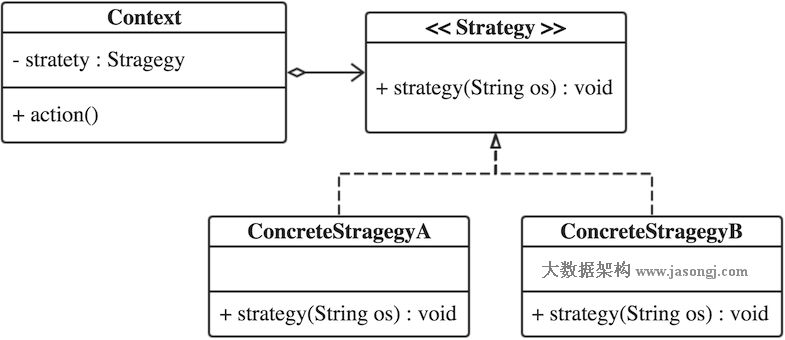
策略模式是一种软件设计模式.指对象拥有某个行为,但是在不同的场景中,该行为有不同的实现算法,.

## 策略模式的成员

策略模式中一共有三种成员:

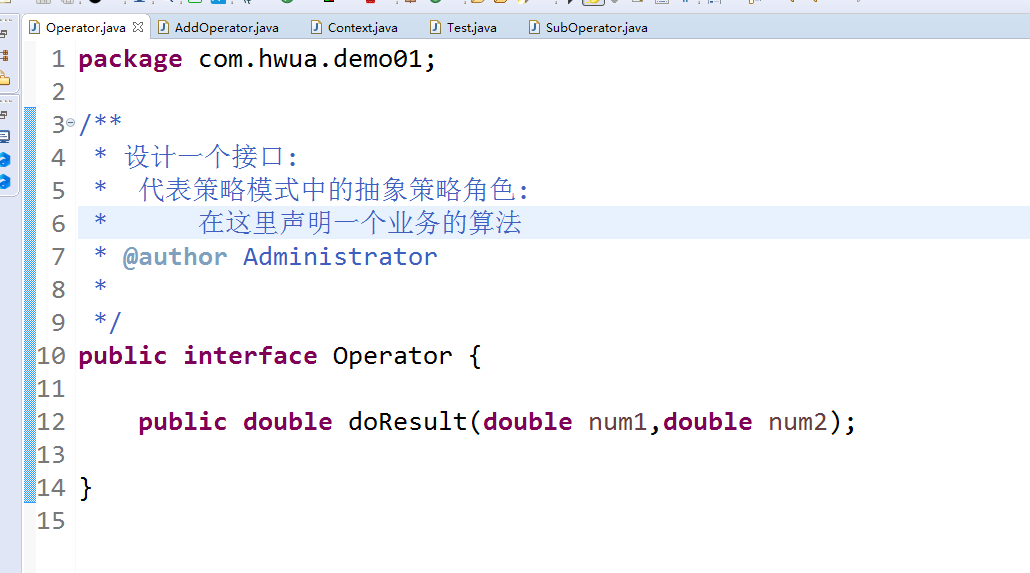
* 抽象策略角色. 代表所有策略的父类或者接口.
* 具体策略角色. 抽象的实现
* 环境角色. 策略的持有者.(上下文)

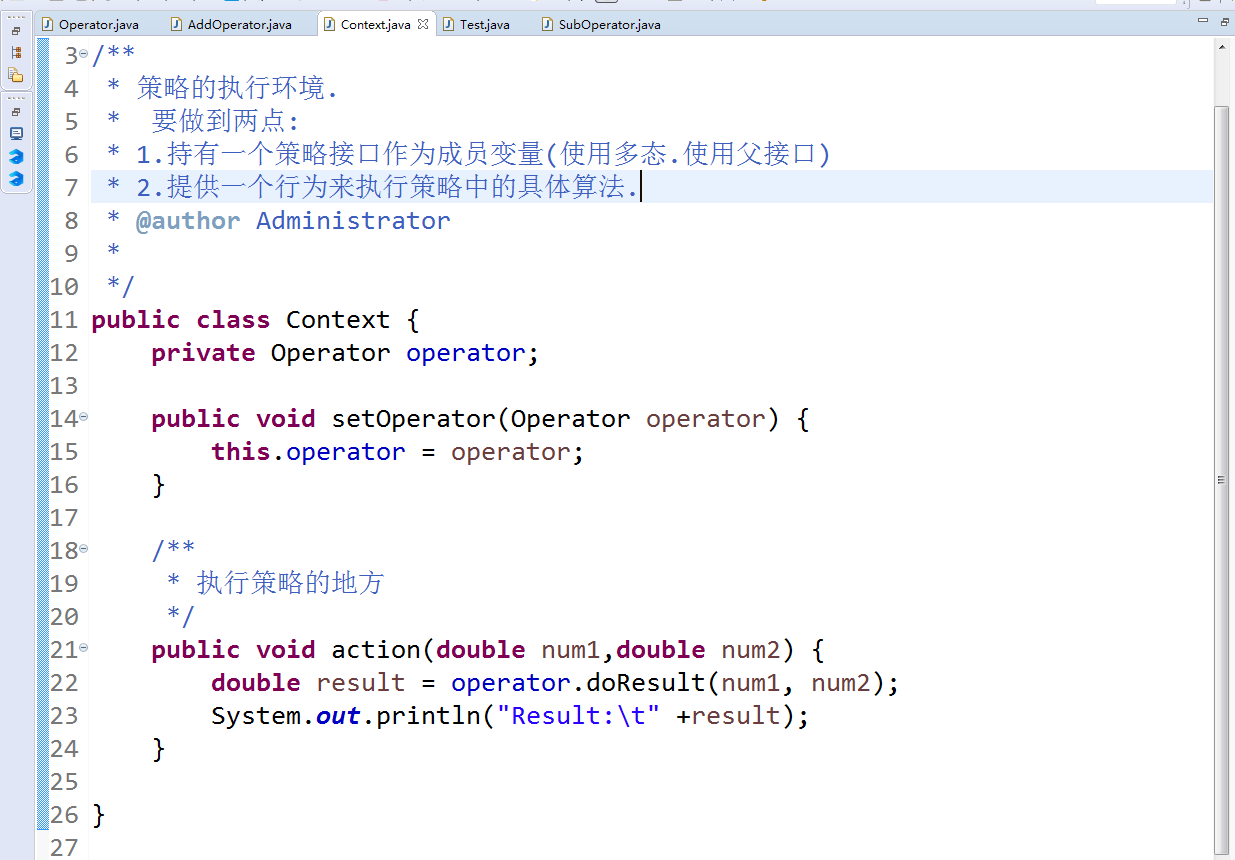
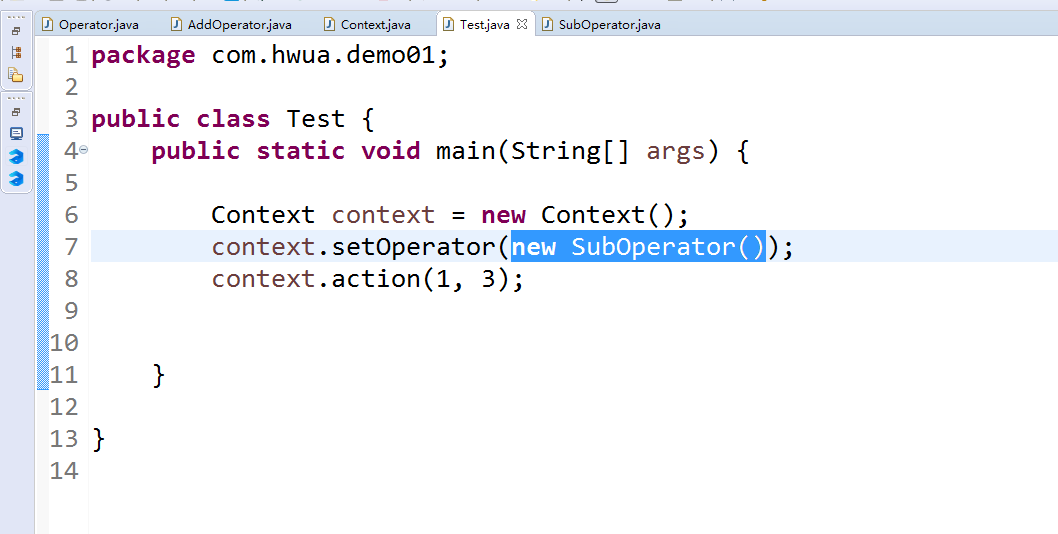
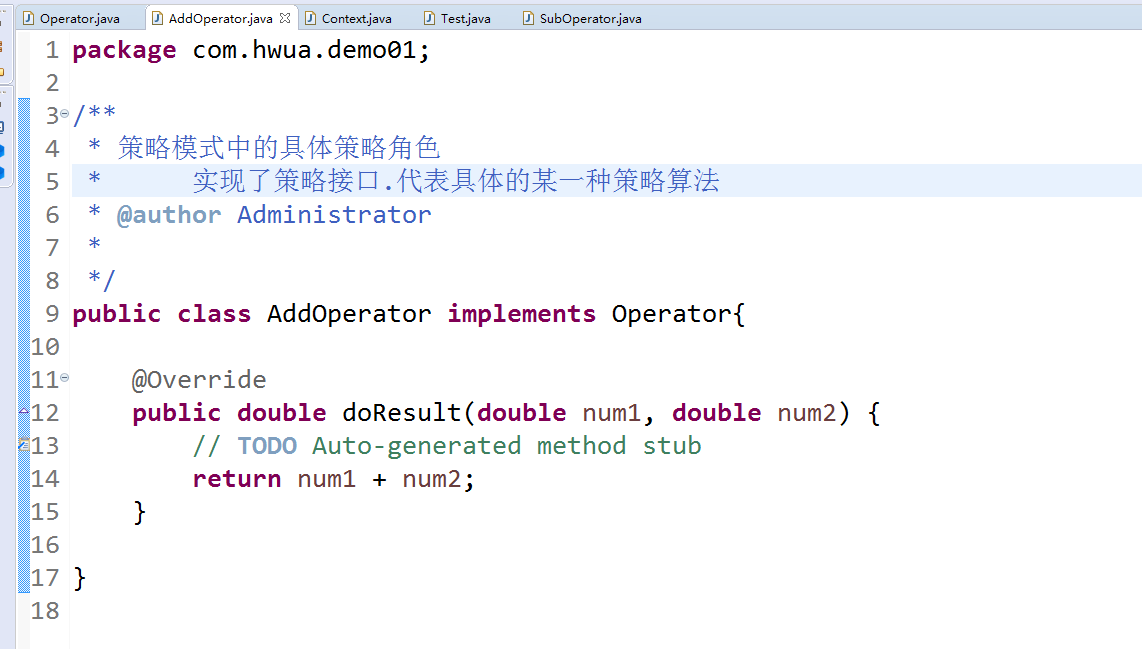
## 策略模式类图



## 策略模式的示例

使用策略来实现一个四个运算.





# 内部类

顾名思义,就是写在类内部的类.参考变量的位置,内部类可以分为4种

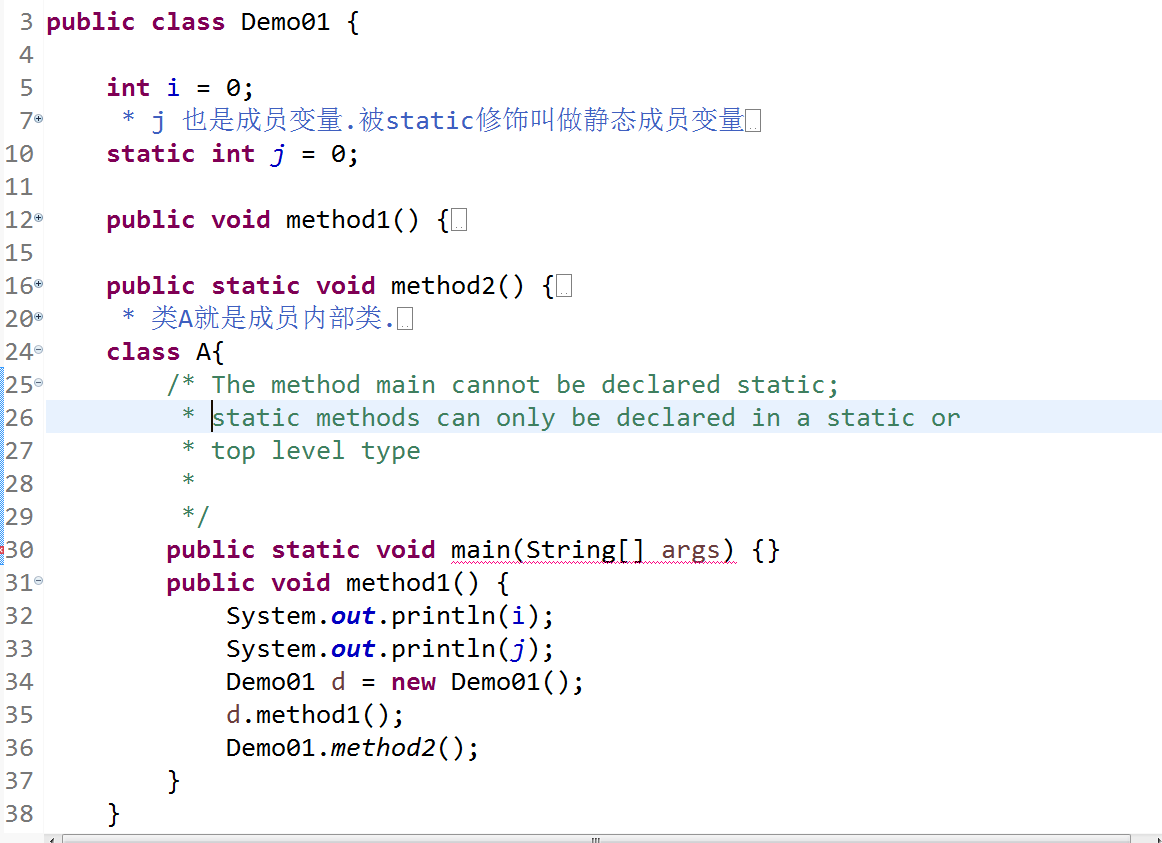
* 成员内部类
* 静态成员内部类
* 局部内部类
* 匿名内部类

## 成员内部类

和成员变量一样.写在一个类的内部,且没有被static修饰.它具备所有成员变量该有的特性.我们可以思考下成员变量所具备的特性

1. 成员是写在方法的外面.所以作用域是整个类
2. 成员变量可以被当前类中所有非static内容直接访问
3. 如果静态内容想要访问成员变量必须要通过对象.来进行访问.
4. 可以final修饰.变成常量.

我们把这几点试验在成员内部类上.

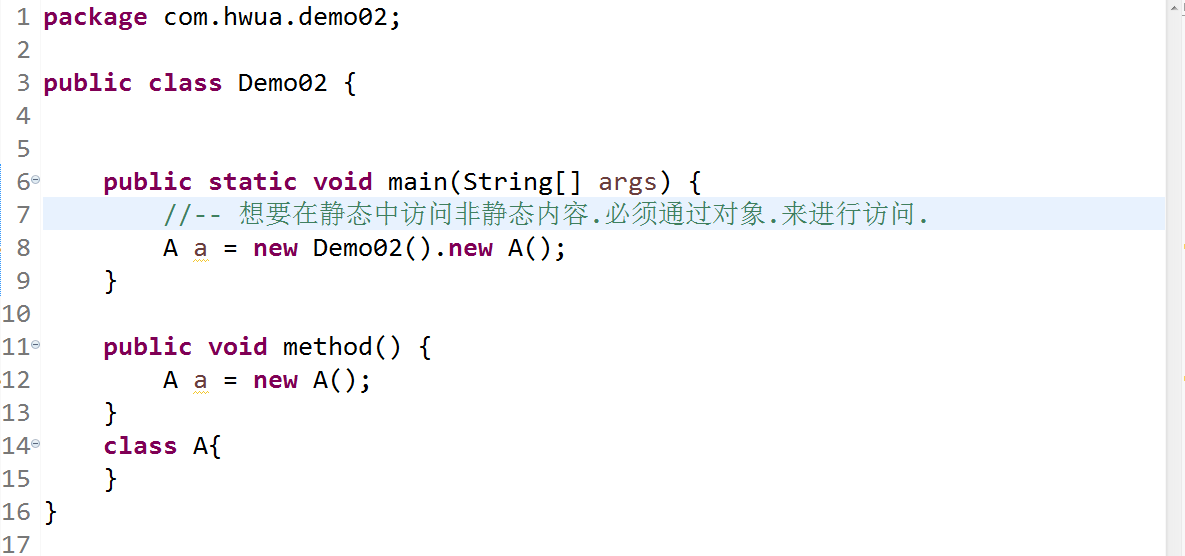


从上面代码我们可以发现.在成员内部类中我们可以直接访问外部类的所有内容:静态和非静态

我们在成员内部类中可以定义非静态的内容(包括变量和方法)

### 成员内部类的实例化

内部类也是一个类.那既然是类就可以创建的对象.成员内部类的创建对象方式



唯一注意的就是在静态访问中访问成员内部类对象.

## 静态内部类(静态成员内部类简称)

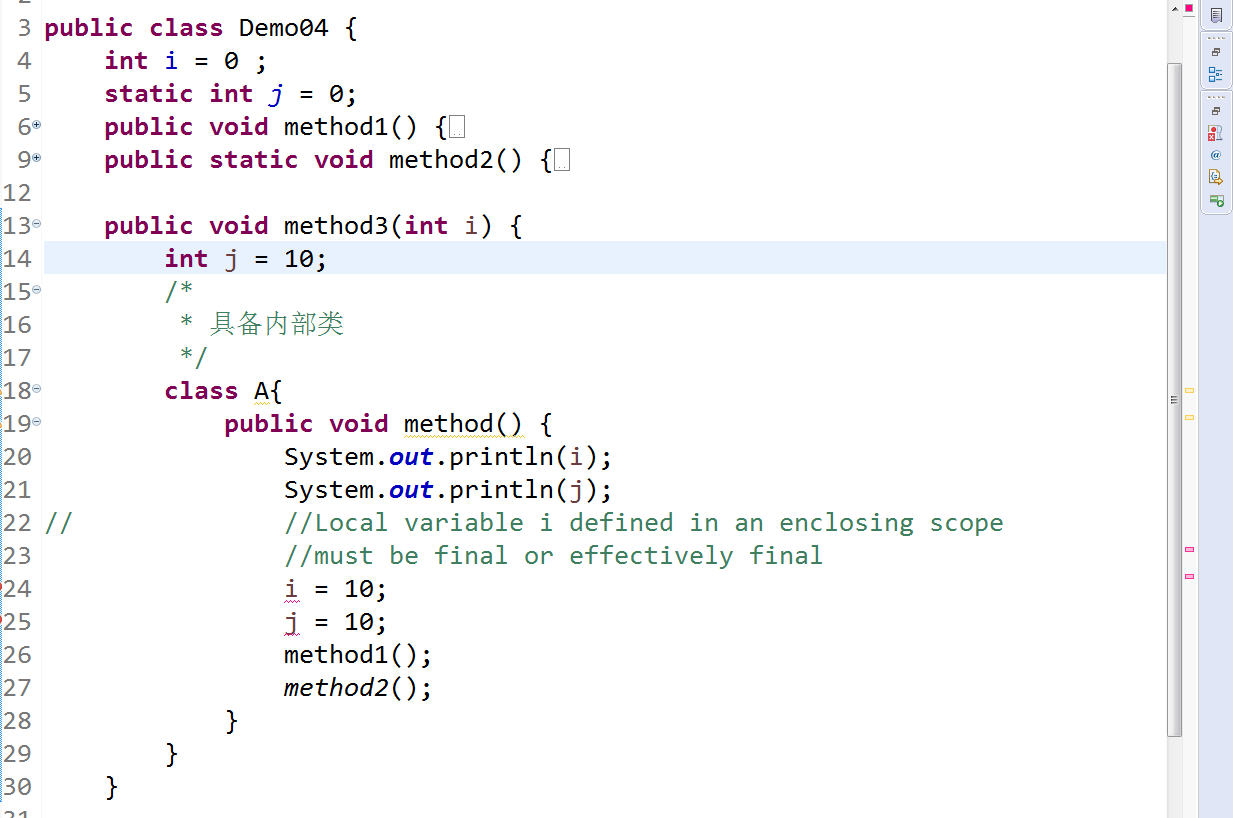
首先它是一个成员内部类.然后再是静态内容.所以具备了静态和成员内部类的所有特点.

### 静态变量可以是局部变量吗

不可以.静态变量是通过类名.变量的形式来进行访问.当把它写到方法内部时.方法不调用变量不存在.也就无法通过类名.变量来进行访问.所有的静态内容都一定是直接属于类的.写在类中的.不会是局部的.

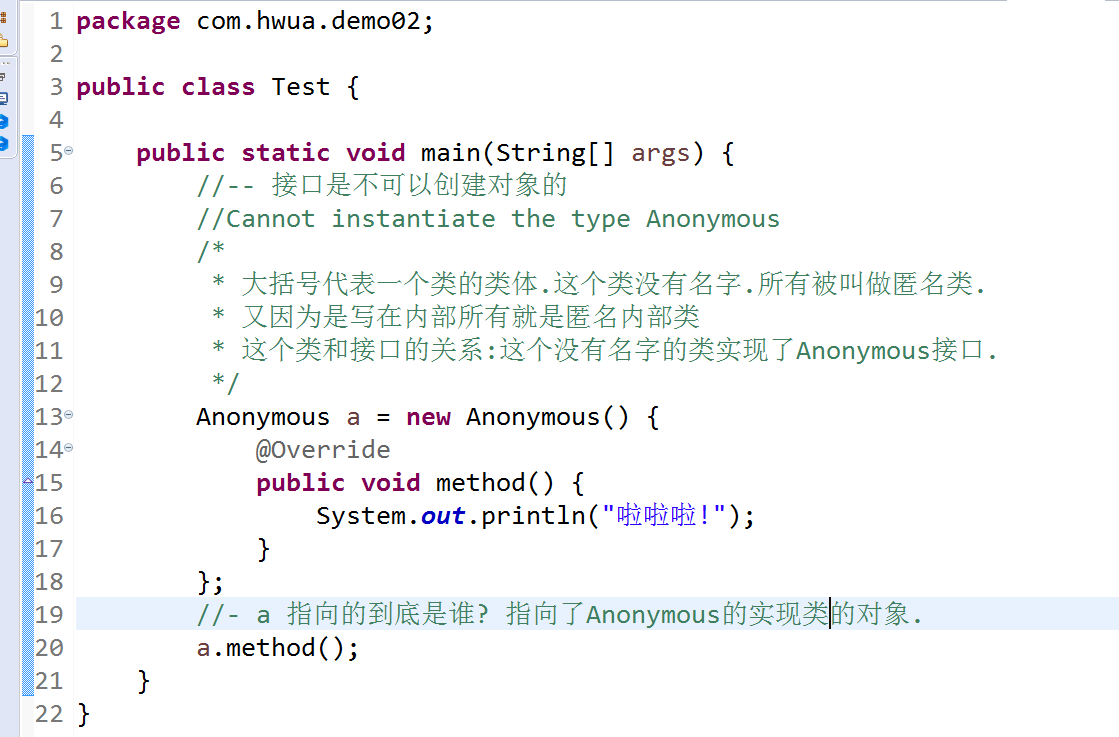
## 局部内部类

局部内部类是使用频率最低的.它写在方法的内部.唯一的注意点:在局部内部类中访问局部变量(方法中的变量).该变量必须是final的.并且注意点在JDK1.7前后写法上有差异.1.7之前是必须写final关键字的.在1.7之后默认就是final.因此可以省略final关键字



## 匿名内部类

这是所有内部类中使用频率最高的.也是大家必须掌握的.匿名就是没有名字的意思,其它的三个内部类会生成class文件.该class文件的名字是以外部类$内部类的形式出现的.而匿名内部类虽然也有class文件生成.但文件名字没有



匿名内部类的目的就是为了快速实现接口或继承父类.来重写方法的.所有说匿名内部类就是为了创建已提供的接口或者抽象类的实现类的对象,.

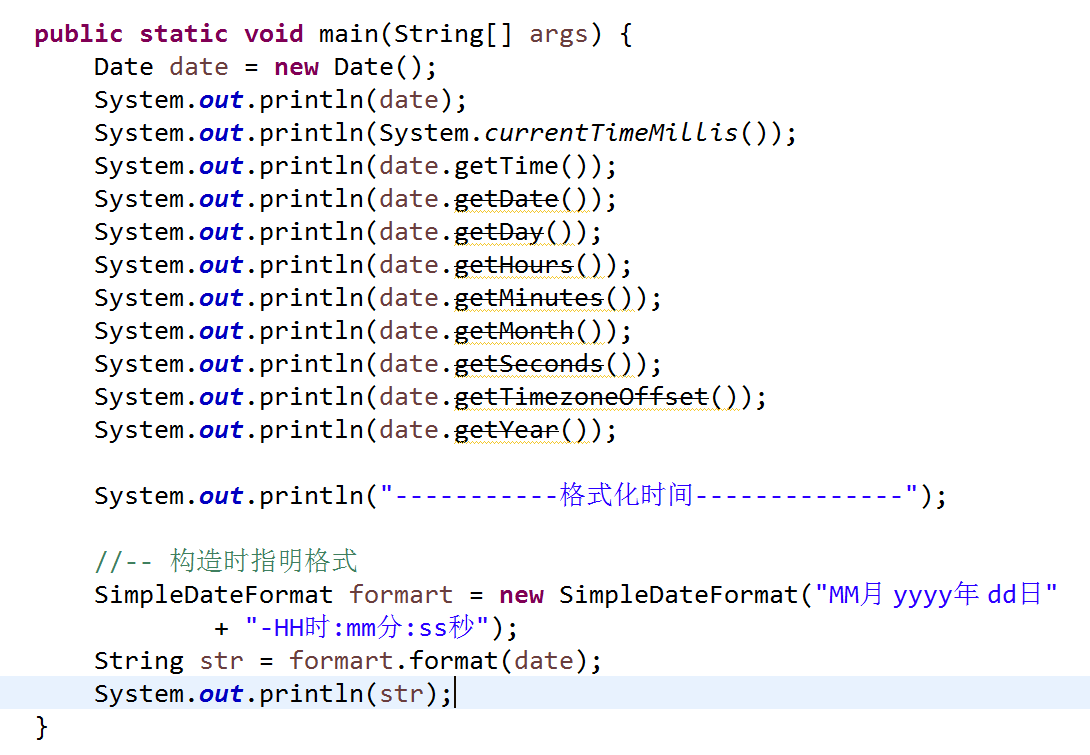
# 常见类

## String StringBuilder StringBuffer

## SimpleDateFormart

### Java中的日期类

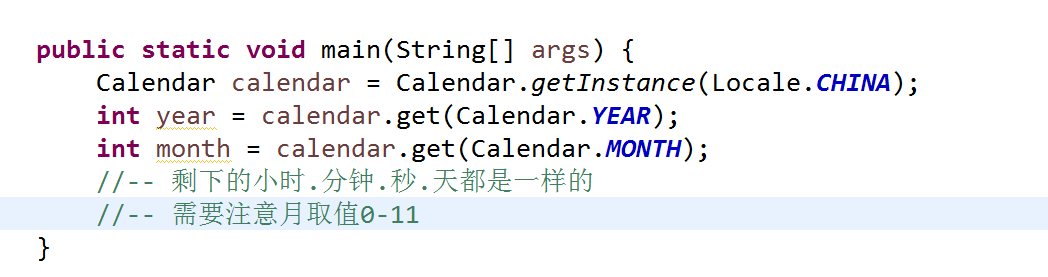
Java中提供了一个类:java.util.Date.专门用来封装日期和时间.这个类中的很多方法都已经过时了,被Calendar代替了.所以在Java中如果要获取时间主流的使用方式是:Calendar



## Calendar

Calendar的使用比较简单.通过静态方法来直接获取Calendar的实例.再通过实例获取指定的时间因素.

因为Calendar是一个抽象类.是不可以通过new来进行实例化的.只能通过内部提供的静态方法getInstance来进行实例化.至于到底实例化的是Calendar的那个实现类.我们并不关心.因为遵循多态的原则.以父类为类型.所创建的使用方法时,只能用在父类中声明过的方法.



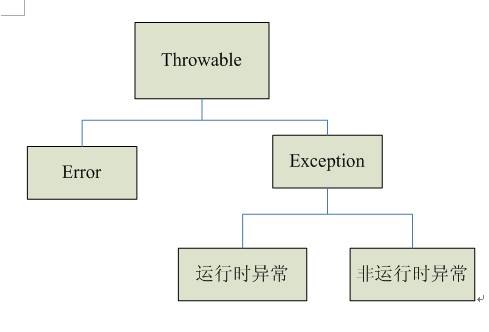
# 异常

任何一款应用,它的代码都不可能是完美的.在运行期间或多或少都会有问题产生.甚至在编译器(代码的书写)期间都会有问题产生.Java是面向对象的,Java设计就想设计一个类型可以代表所有的不正常.这个类叫做Throwable,它是Java异常的根类.Java针对异常的处理代码基本上都是在该类中写好的.

在Throwable的下方有两个子类:Error和Exception.分别是错误和异常.错误一般指JVM产生的.在代码中不处理,我们主要处理的是Exception.

根据异常的产生时间段.Exception又可以分为两类:RuntimeException和非RuntimeException.

所以整个Java的异常体系如下



Java不仅提供了这样的异常体系还提供了两套异常处理机制(必须掌握的)

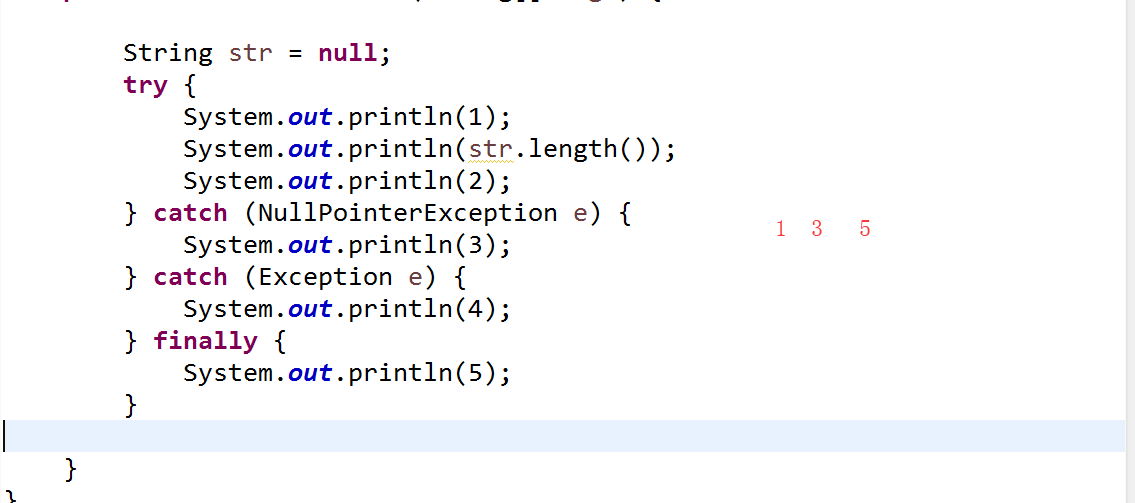
1. 消极机制: 利用throw和throws把异常扔给上层..
2. 积极机制: 利用try-catch-finally把异常处理掉.

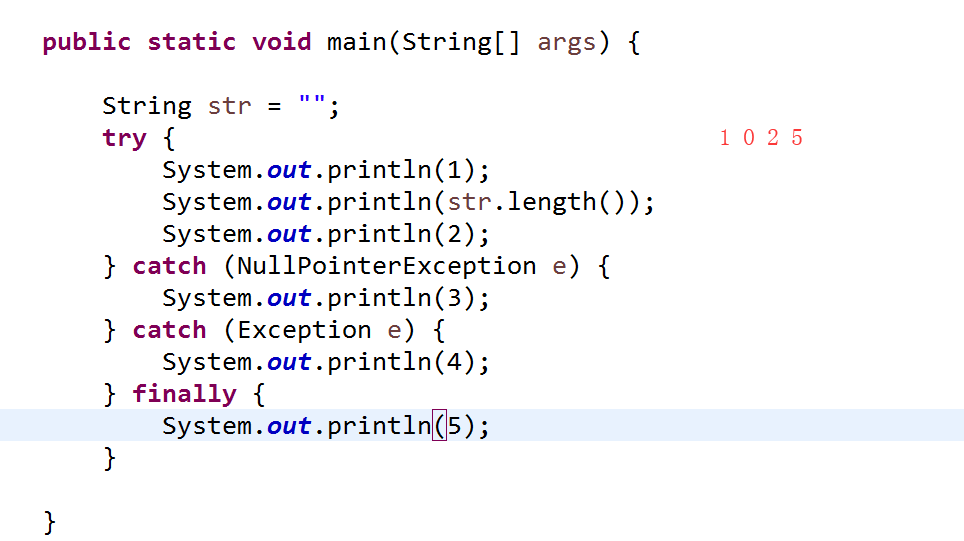
## 积极机制

尝试去执行可能会产生异常的代码.如果产生了异常交给下面的捕获器来捕捉异常.

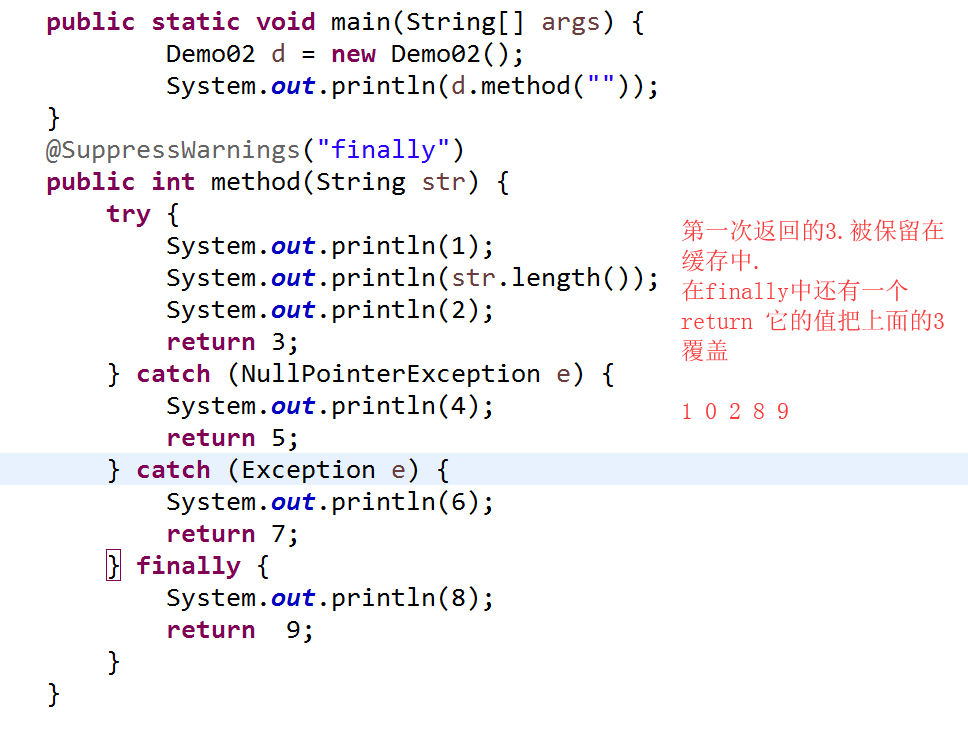


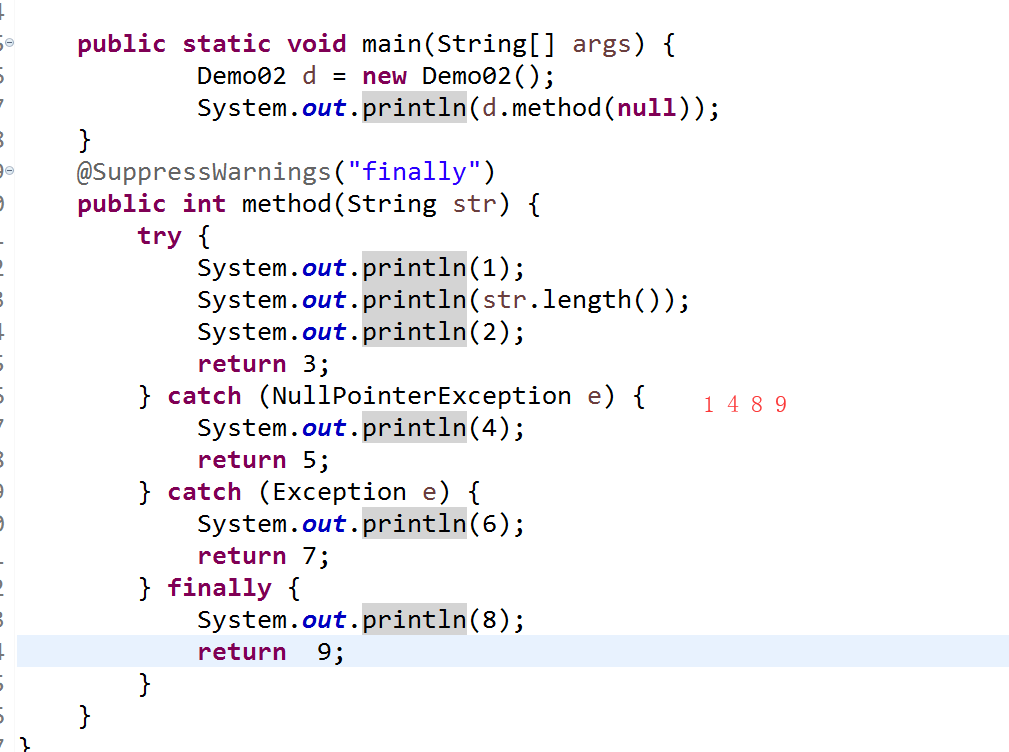
那什么是finally.它代表最终一定会被执行.即无论try中是否会产生异常.也无论catch执行还是不执行.finally都会被执行.





### 当finally邂逅了return怎么办

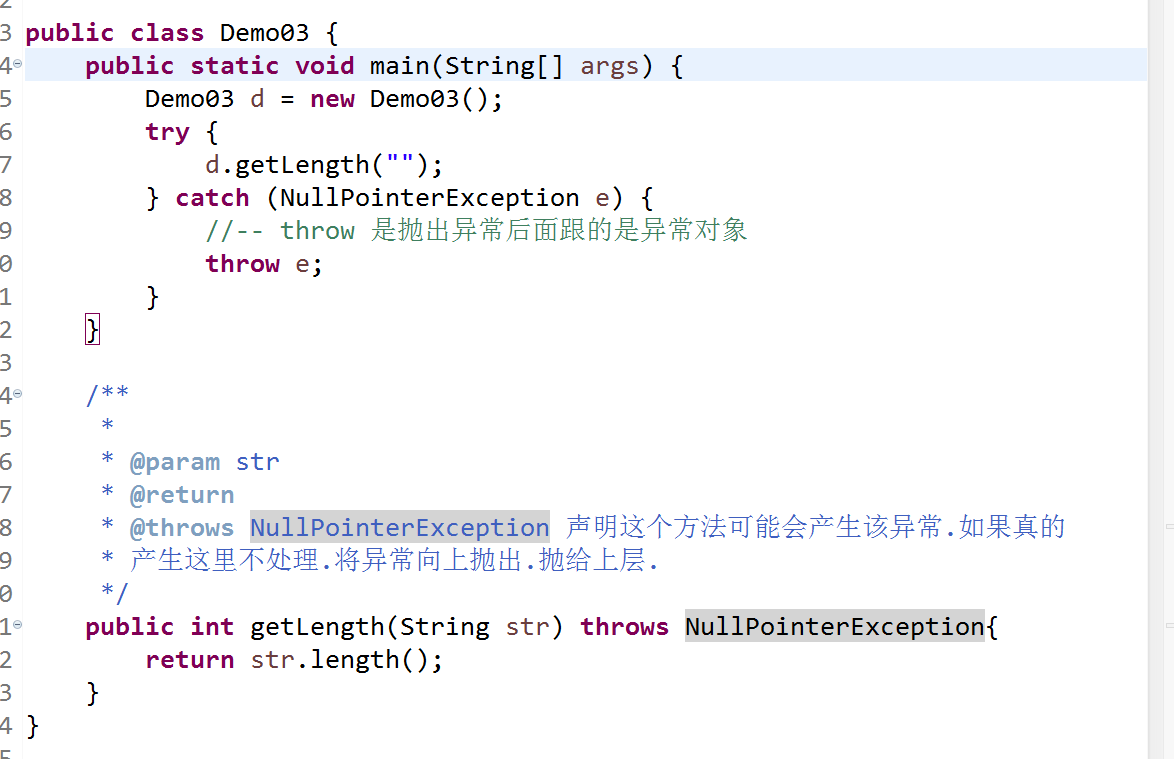




Return会先执行.但是程序没有结束.而是把所要返回的数据保存在缓存中.等待finally执行结束后在进行返回.如果在finally中还有return.最后的一个return会把之前的return覆盖掉.

## 消极机制

消极机制只能用在方法上.



### Throw和Throws的区别

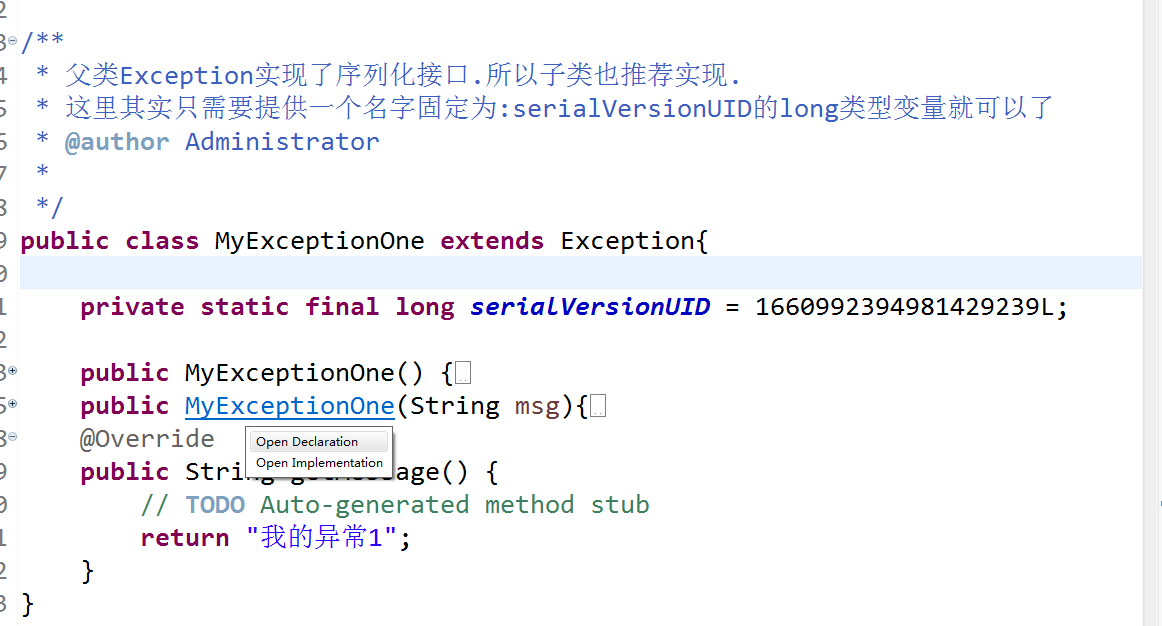
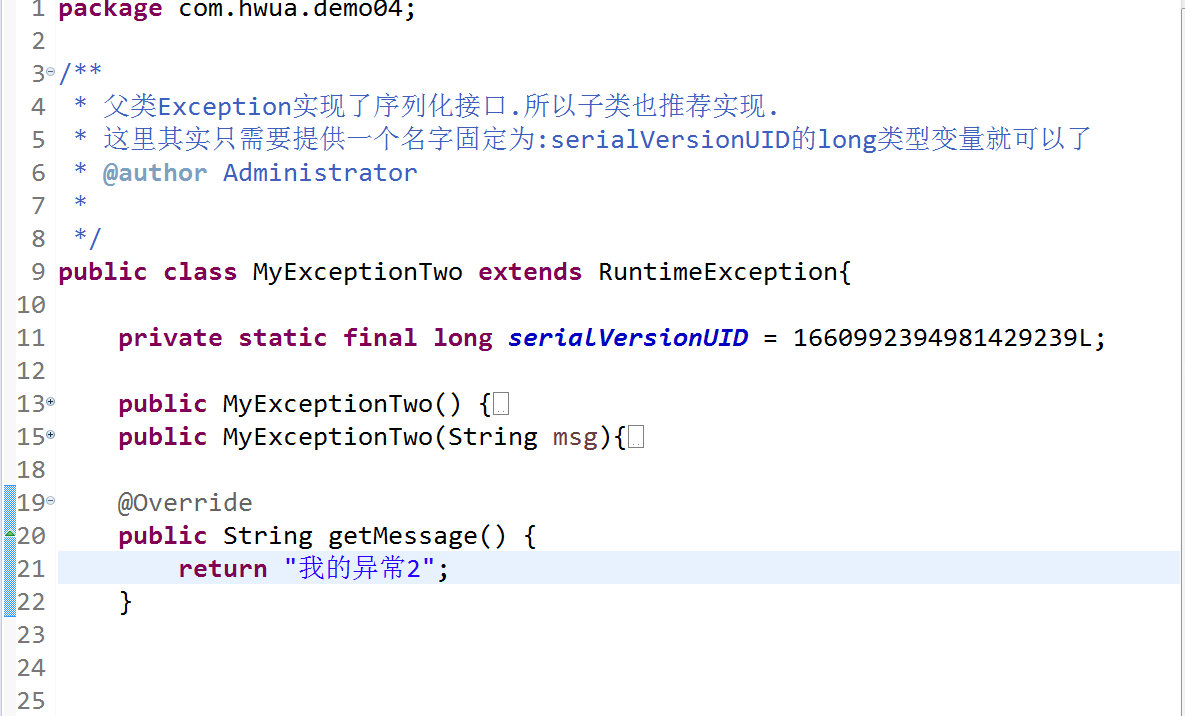
Throw是实际抛出一个异常对象.后面跟的是对象.而Throws是用在方法的声明上.表名这个方法可能会产生什么异常.

## 自定义异常

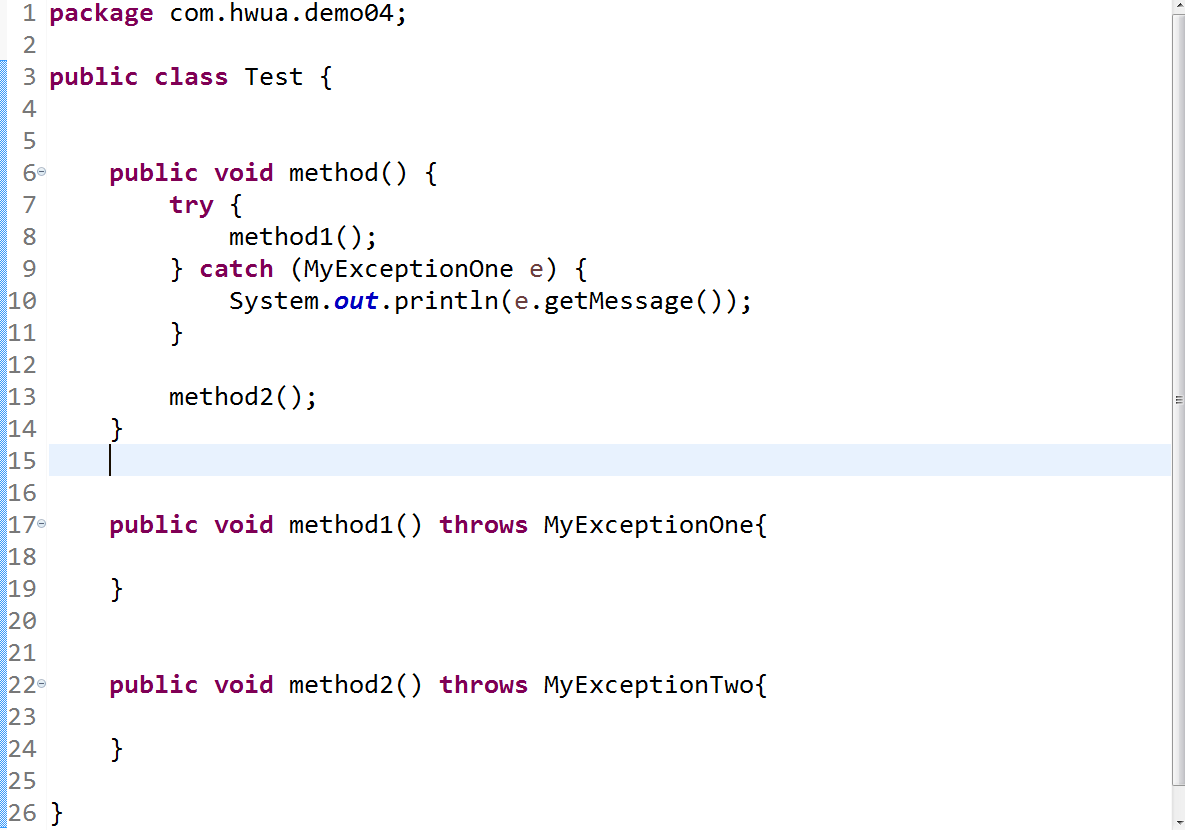
自定义异常不是用来处理语法的.而是用来处理业务的.指在某些情况下.语法可能是允许的但是实际的业务逻辑中是不允许的.

自定义异常和系统提供的那些运行时异常一样都是空壳类.里面没有什么实质的内容.只有构造方法.因为所有对异常处理的内容都已经在Throwable中写好了.所以我们自定义异常就很简单.遵循以下几个步骤

1. 创建一个类
2. 让该类去继承Exception或者RuntimeException
   1. 继承RuntimeExcception 被叫做运行时异常
      1. 运行时异常是不推荐使用try-catch进行捕获处理的.要通过逻辑进行回避
   2. 继承Exception被叫做非运行时异常.也叫做受查异常.
      1. 是必须要通过try-catch进行处理的.
3. 如果想可以提供下构造方法.如果不想啥也不写也行.



getMessage方法可以返回我们希望传递到外面的异常信息.



# 课堂练习

1. 使用策略来实现不同的会员等级.拥有不同的打折优惠

# 家庭作业

1. 课堂练习中异常的第3题和第5题
2. 回家练习中的第2题

# 预习

1. 集合List
   1. Math类中的四舍五入和取值两个函数
   2. 自动装箱和拆箱
   3. String StringBuilder 和 StringBuffer
   4. 泛型<>
      1. 上限
      2. 下限
   5. List接口的两个实现类
      1. ArrayList
      2. LinkedList

# 参考

1. [Java时间](https://www.cnblogs.com/blackheartinsunshine/p/6019408.html)