adbc是一个小型的Library，支持contract programming合同编程给AspectJ 以及随后的JAVA和AspectJ

adbc的合同以JAVASCRIPT的形势表示，并用JAVA注释

method可以有前置后置条件，class可以有不变量

@requires指定了前置变量

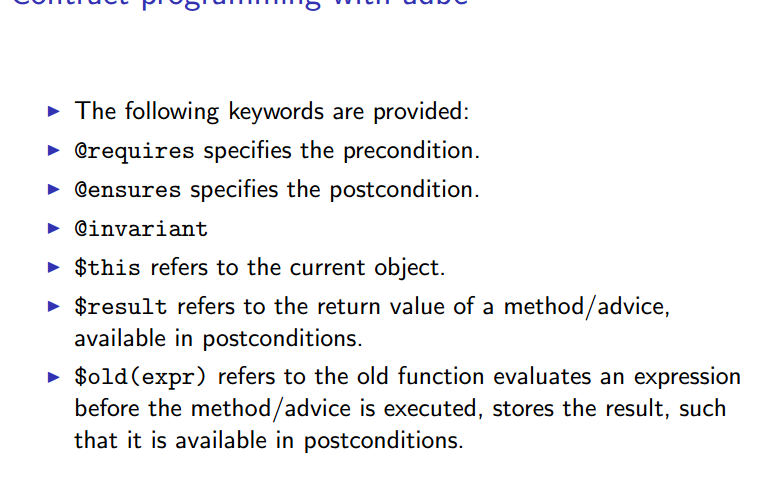
@ensures指定了后置变量

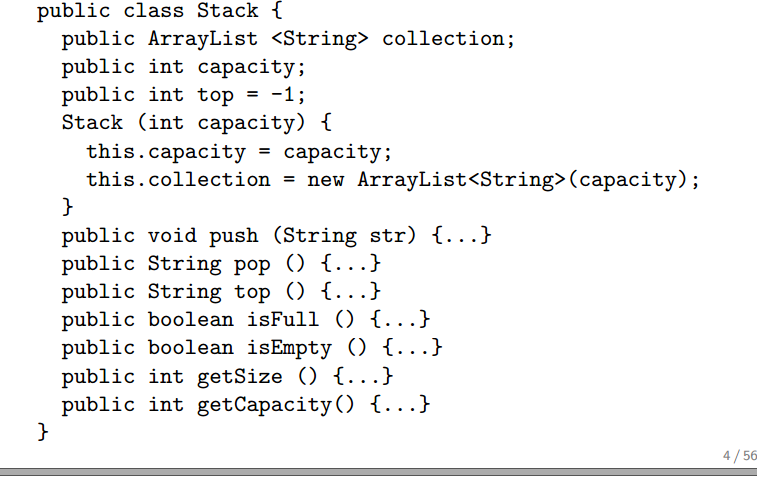
@invariant 不变量

$ this指定当前的object

$ result指定这个method/advice的return value，在postcondition中可以看见

$old 用old函数在方法/通知执行之前来产生一个表达式，并存储结果，以便在postconditions中可用。



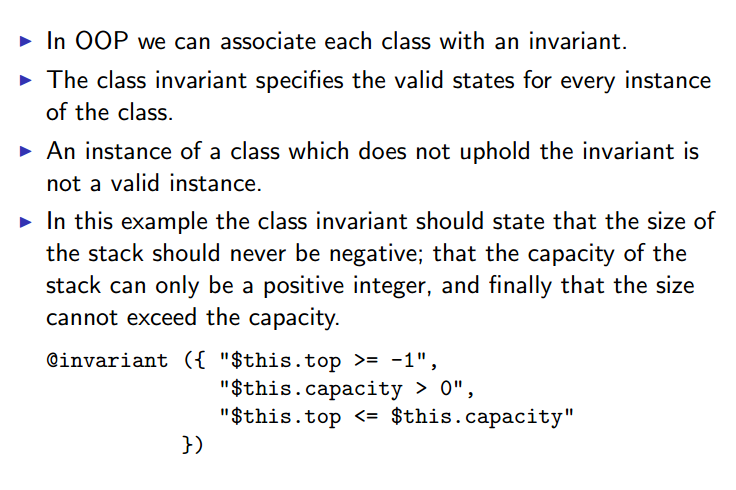


STACK

我们尝试给他一个contract

1.invariant，如果是object-oriented programming，每个class都需要有一个invariant，来记录不变量

invariant的目的是给所有的实例变量（属性）都划定一个范围



一个软件element的合同是由assertion断言组成的，断言又分为preconditions与postconditions

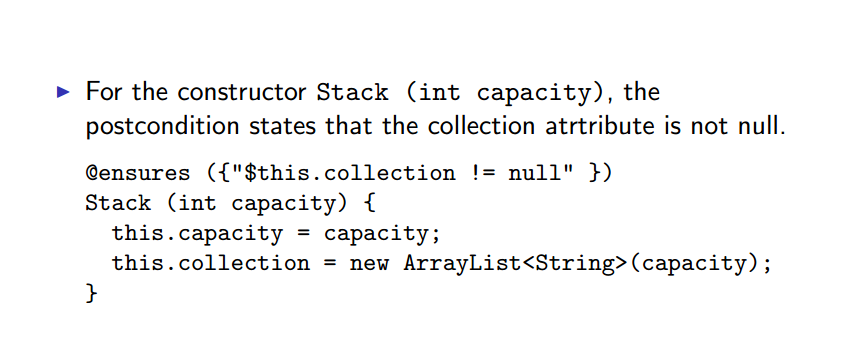
因为contract指定了element与外界的关系，我们应该给所有public/protected method都写一个assertion

Class Stack有的方法push(String)

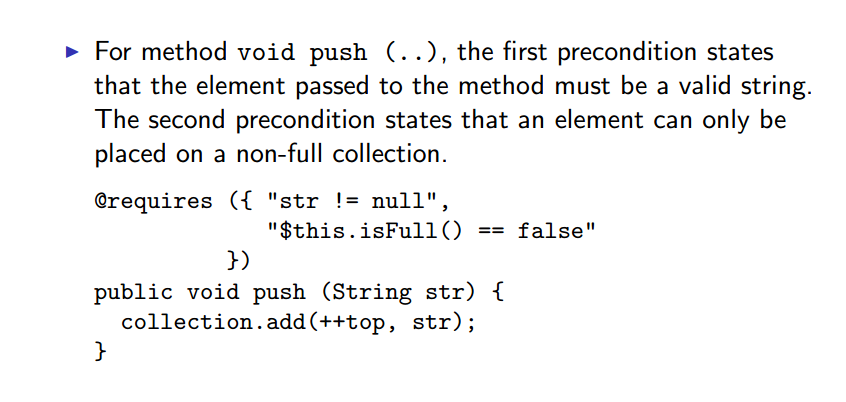
pop()//remove然后return topmost

top//return

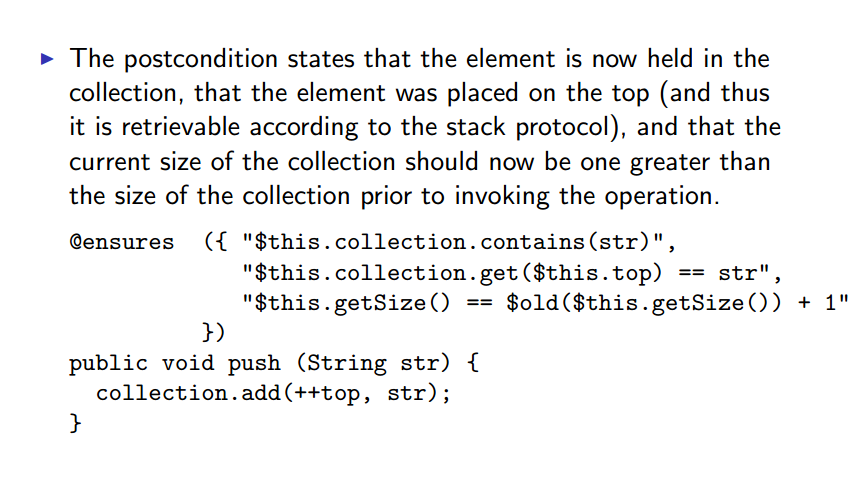
ensure 构造constructor的时候可以用，ensure确认的是postCondition



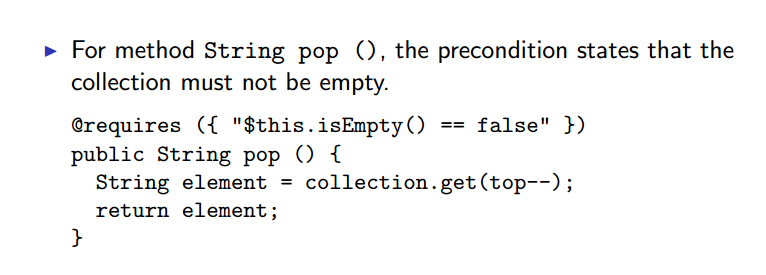
在我们构造完以后，collection必须存在



require他的Push,输入的string是有效的，然后this(这个object)还没full

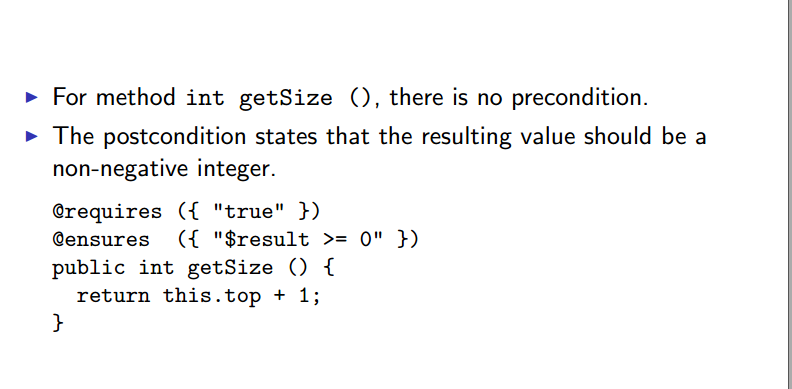


ensure，push完以后，这个collection里包括str，而且就是top，并且size加1，old是用在post condition里的，用来表达前后，就相当于之前的那个’



pop，需要不是empty

换句话来说，任何原来需要try catch的error，如今用require来表示



也可以同时有require ensure，require没有的时候，直接括号里面写true

我们将模拟许多场景，假装在前置条件，后置条件和不变量invariant中失败。

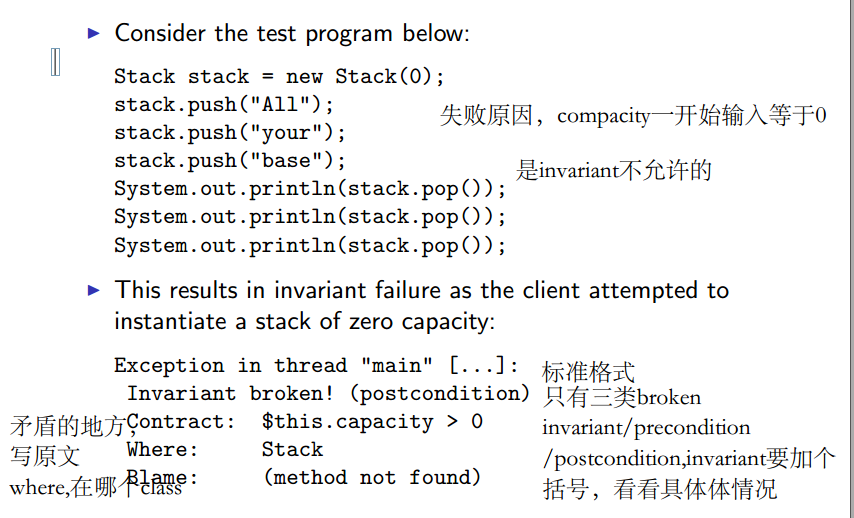
对于每个场景，我们将陈述

a)程序失败的原因和

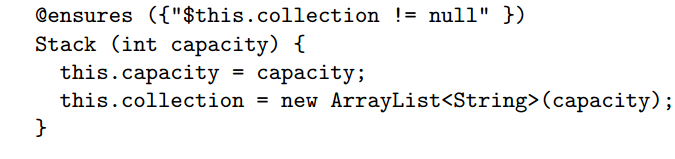
b)程序到底在哪里失败了，（where，具体位置）

c)谁应该负责失败。

invariant failure

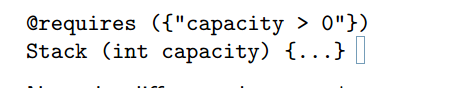


constructor

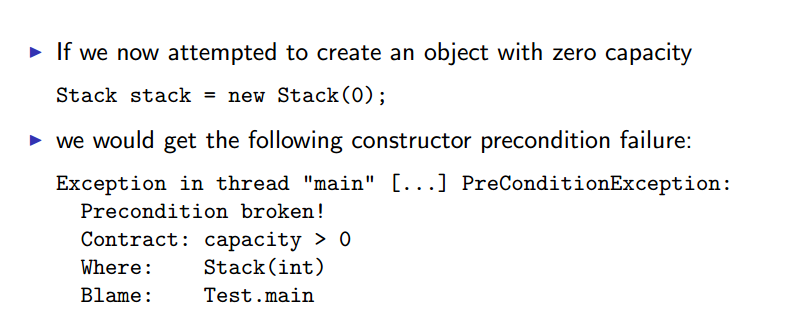


注意我们构建他的时候没有给他preCondition,任何方法或constructor，我们都应该给他preCondition

然后failure实际上是在constructor那里产生的，所以我们给constructor一个prediction

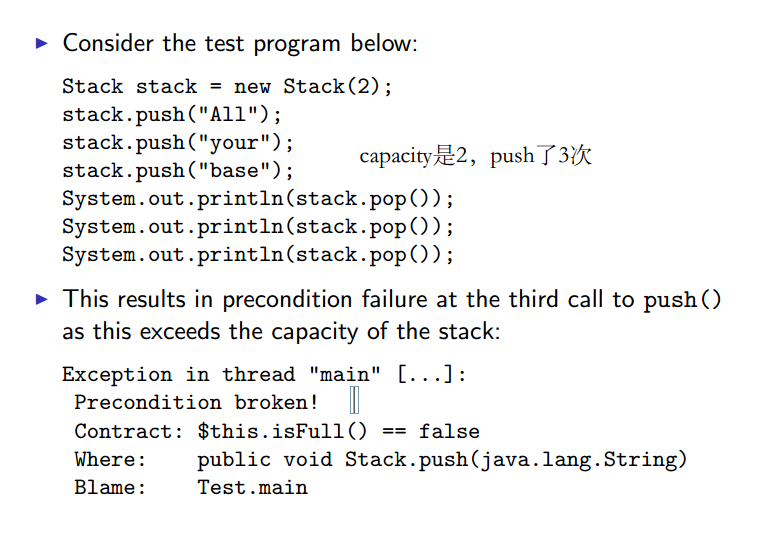


所以现在invariant只是单纯的声明变量，只有constructor才会产生具体值



这才应该是标答，注意这里的where，如果是constructor，就直接CLASS NAME加参数

blame永远是Test.main

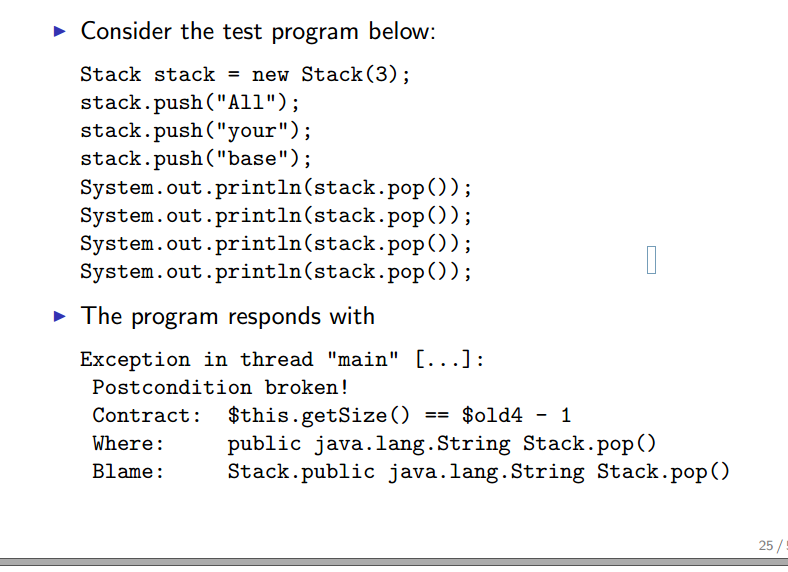


where就是原来的method，但是要注明是哪个class的method，类型是String的话要加，java.lang

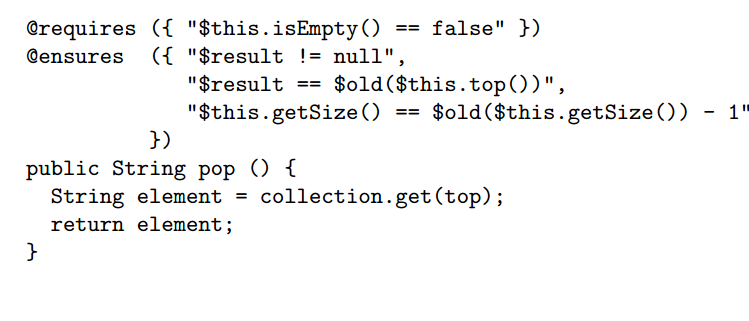
这些是没有满足precondition的，contract直接写违反了哪条原文就好

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

postcondition，我们要把error插入到implementation的过程中，然后通过插入，让结果fail，来证明这个error是fail的

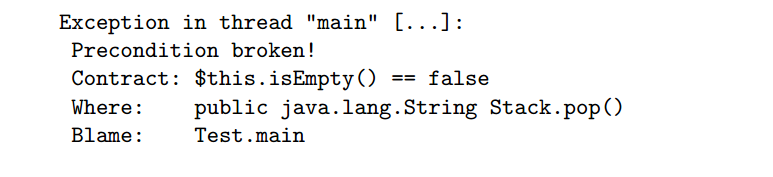


连续pop了四次



违反了post condition

注意他也违反了precondition



所以我们两个都要描述。

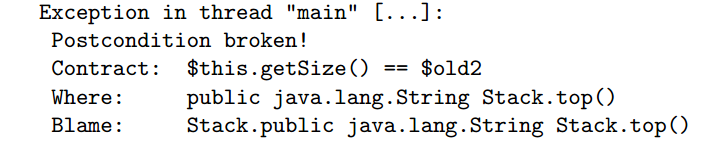




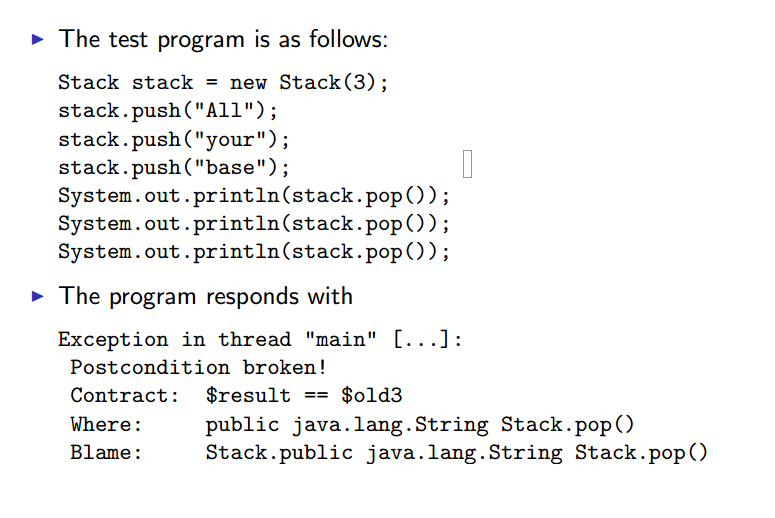
post的contract不仅要抄原文，还要把错误代入，old后面直接写

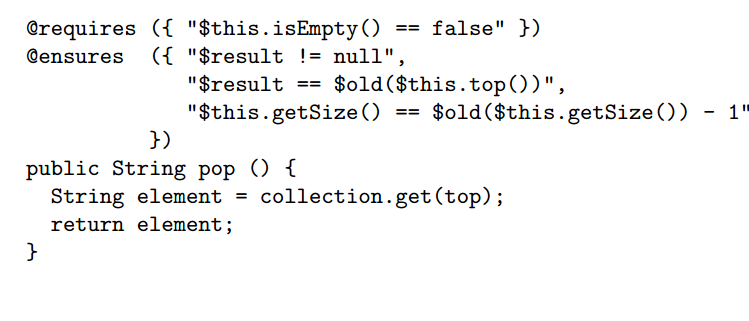
where就是哪里出问题，与pre一样

Blame则不同，应该是Class+ 具体Method



two cautionary test cases

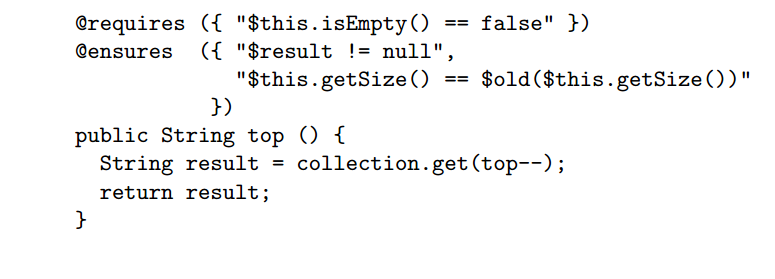




看起来implementation是对的，

但是Pop还是没有满足postcondition

因为pop虽然是对的，但是top是错的，我们call pop的时候，abdc会默认top是对的，所以abdc错误的把top的错误归咎到了pop



他本质上就是错误的，他让top—了，所以改变了size

abdc继承不要