# 1.2 PIE模型

## 关于bug的三个概念

（在软件运行的不同时期有不同的含义）

### Fault：

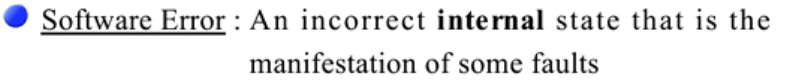
**静态**存在于软件当中的缺陷（由于在coding过程当中犯了个错误写错了，通常称之为缺陷）



（manifestation：表示 显示）

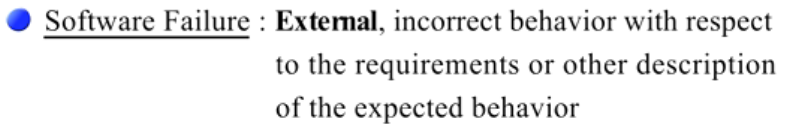
### Error：

在运行过程当中，运行到fault触发了一个错误的中间状态，称之为error



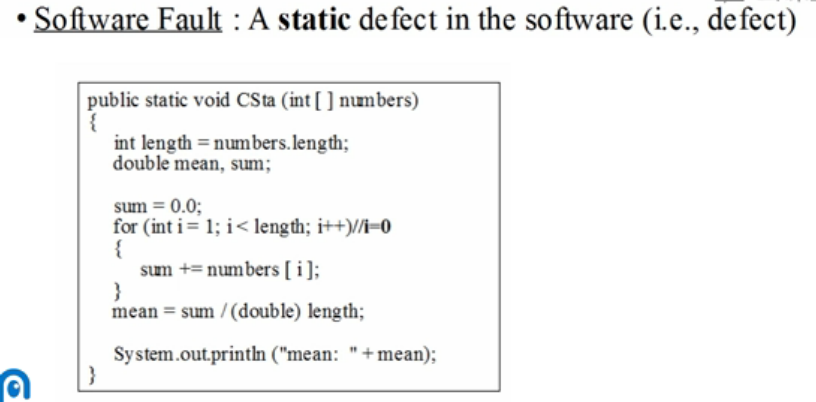
### Failure：

（通常称为失效）是指刚才的error一直传播到软件的外面，使得测试人员或用户能够观测到这个失效的行为

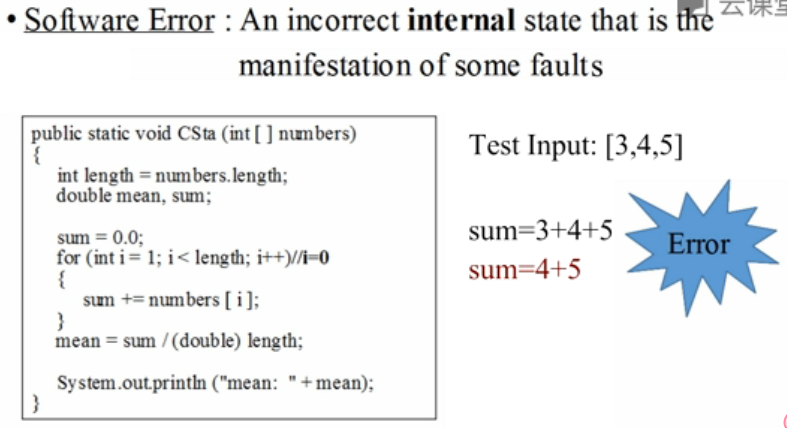


## 用于理解的例子

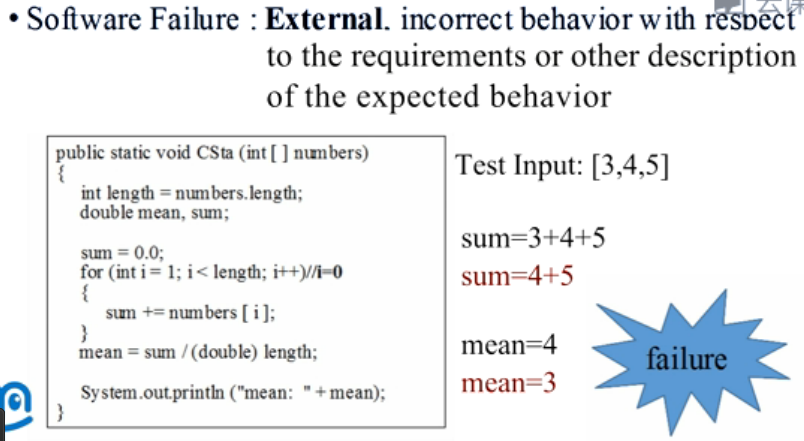
在编码的时候，把循环变量初始值误写成了1——fault



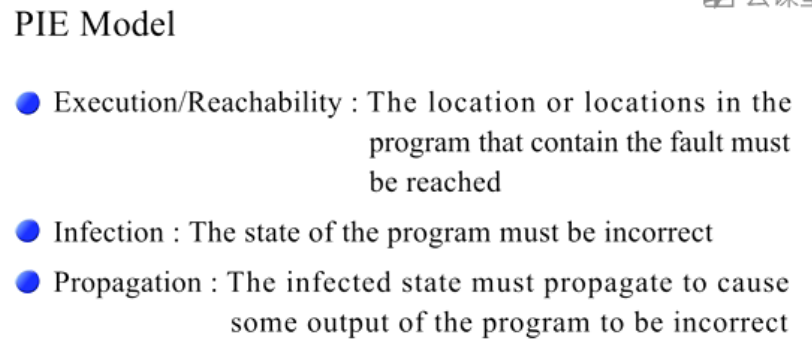
运行程序时，输入[3,4,5]，当执行到循环体时，由于初始条件的错误只累加到了4，5，导致sum现在为9——错误的中间状态



最后这个error传播出去，原来平均值应为4，现在是3，观测到失效的行为，称为failure。



## PIE Model



Propagate：繁衍 扩散 蔓延

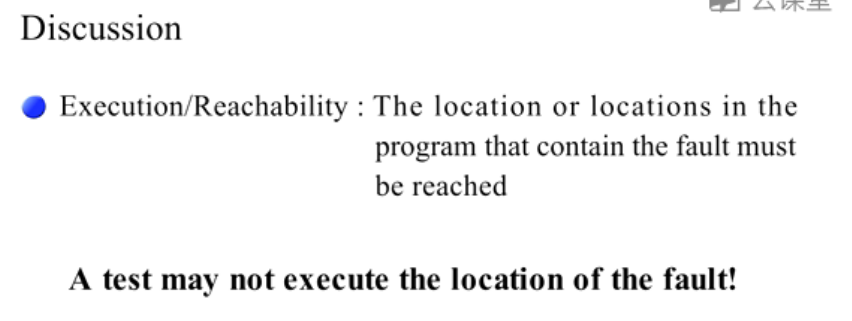
整个运行过程是通过PIE模型来刻画的。

即我们需要观测到一个failure，需要三个必要的条件

1. 执行必须通过错误的代码
2. 感染——在执行这个错误代码的时候，必须触发出一个错误的中间状态
3. 传播——这个错误的中间状态必须传播到最后输出，使得我们能观测到它跟预期的输出不一致，即失效。

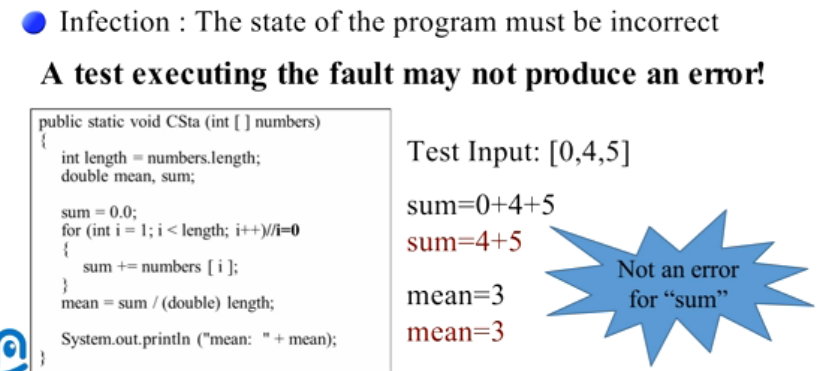
## 对于PIE Model的理解

1. 在很多时候，测试未必能执行到fault的位置。尤其是一些程序的控制流和调用比较复杂的时候。



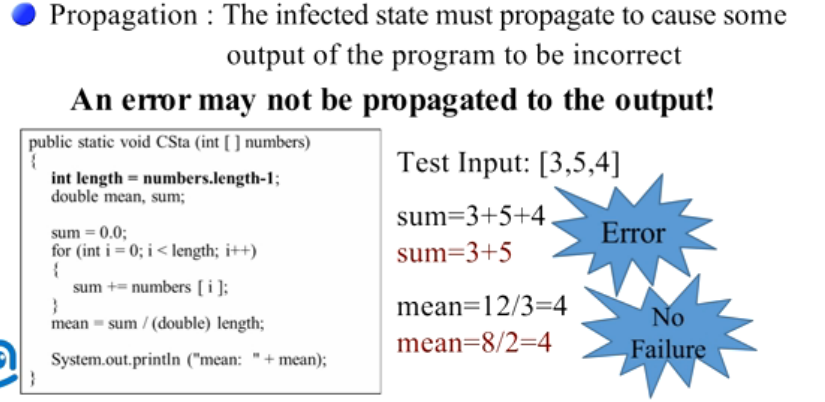
1. 即使测试执行到了错误的代码，也不一定触发error。

这个例子，虽然执行了fault代码，但是对于sum来讲，没有触发error，最终的结果也是正确的。



1. 一个测试，执行到了fault，触发了error，最终也可能没有failure。

被操作掩盖了当初sum的error，最终没有failure，对于测试人员来讲没有观察到失效的行为，会误判为是对的。



通过执行fault触发error产生失效是一个很复杂的过程。

## 总结

Bug分为fault error和failure。整个能够传播出去是靠PIE模型及执行到fault 感染产生error并且传播出去观测到失效的行为。

