Chapter 12

O-O TEST

前面我们一直以结构化程序的测试方法介绍为主，是为了将注意力集中在测试上，而不是编程方法上。

面向对象的软件开发技术已经成为软件开发的主流技术，如果仅介绍结构化程序的测试方法而不涉及面向对象的软件测试方法，课程是不全面的。

而事实上，即使前面的黑盒测试、白盒测试，大家也基本都是基于面向对象的软件进行测试的。

从本章起，我们介绍面向对象软件测试的基本概念、特点和方法，对大家今后的学习应该是十分有用的。

1. 面向对象测试的特点
2. 类测试
3. 面向对象的集成测试

1 面向对象测试的特点

1.1 面向对象测试的单元

单元的定义:

- 可以编译和执行的最小软件组件。
- 是不会指派给多个设计人员开发的软件组件。

在结构化程序设计中，单元就是程序的一个函数或过程！

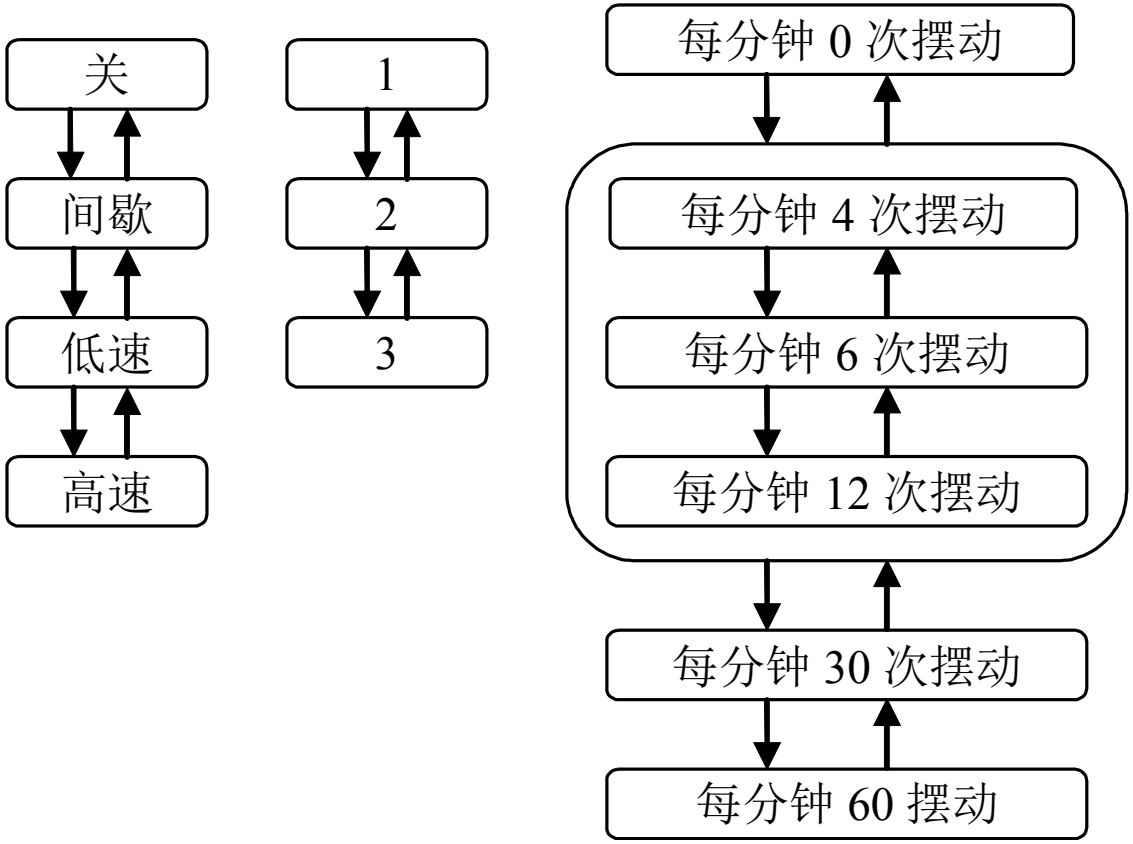
在面向对象程序设计中，类具有封装性，是单元的一种选择；但有时类会很大，具有许多的变量和方法，此时，类中的方法似乎更接近单元的特性。

1.2 合成与封装的涵义

合成是面向对象软件开发的核心策略。将具有高耦合度的数据和操作封装在一起，就是类的概念，而对外是松散耦合的关系（类与类之间）。

如果在类的设计中，将高耦合度的关联设计到了类与类之间，那么即使通过了非常好的单元（类）测试，集成测试的复杂度会大大增高！

挡风玻璃雨刷是一个多次出现的事例



控制杆、刻度盘、雨刷类的行为

控制杆、刻度盘、雨刷类的伪代码：

```
Class lever（leverPosition；
Private senseLeverUp（），
Private senseLeverDown（））

Class dial（dialPosition；
Private senseDialUp（），
Private senseDialDown（））

Class wiper（wiperSpeed；
 setWiperSpeed（））
```

控制杆和刻度盘有感知其物理设备变化的操作。控制杆在位于“间歇”位置时，与刻度盘产生数据交互。这种交互应该在哪里控制？

封装的原则是，类只了解自己的信息，并根据这些信息进行操作。

方法一：控制杆和刻度盘只报告自己的位置，由雨刷类决定做什么。雨刷类成为主程序。

方法二：使控制杆类成为“聪明”对象，当处于“间歇”状态时，获得刻度盘的位置，并告诉雨刷类如何运动。但是，如果控制杆停在“间歇”未动，而刻度盘动了呢？

方法三：建立主程序，轮询控制杆和刻度盘的事件，决定雨刷类如何运动。

三种方法中，哪一种方法更好呢？

为什么？

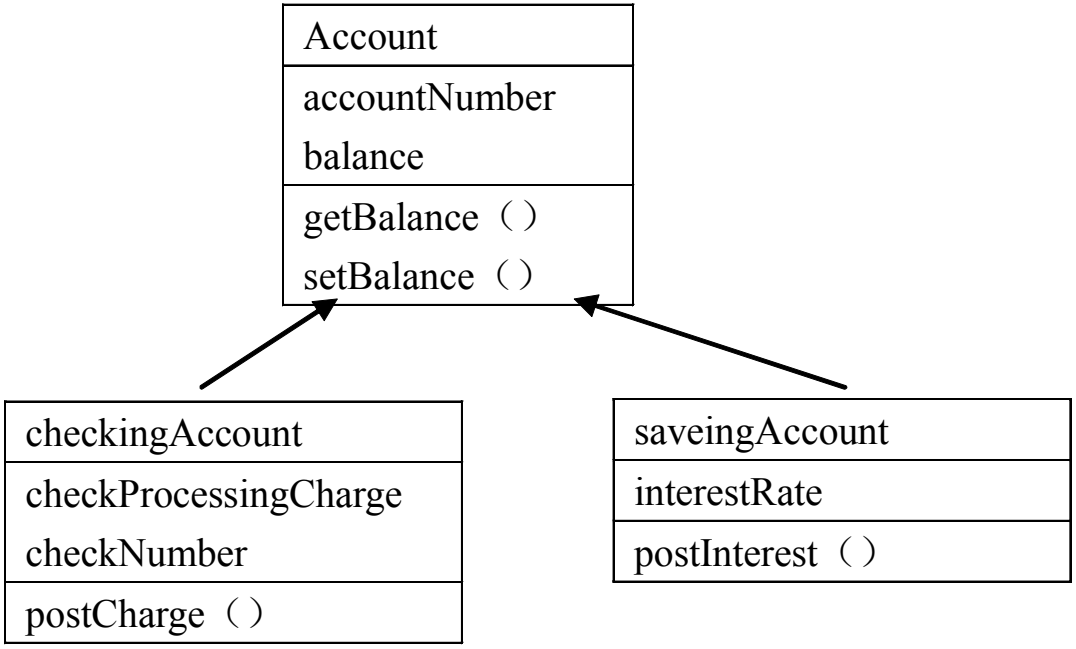
1.3 继承的涵义

类作为测试的单元时，如果存在类的继承性问题，就变成为类间测试问题，解决的方法是：“扁平类”，即对有继承性的类进行扩充，以包含全部所继承的属性和操作。

存在的问题，经过扁平化的测试类不是交付程序，可能导致测试的不真实性。

事例：SATM

每个银行客户都可以通过支票和储蓄卡两种方式进行帐户操作，但计算方式不同。因此，支票帐户类和储蓄卡帐户类都继承了银行帐户类的属性和操作。下图所示。



显然，如果不扁平化，单独的支票帐户类和储蓄卡帐户类都不能访问或修改余额。对单元测试而言，这显然是不合理的。下图给出了扁平化后的支票帐户类和储蓄卡帐户类。

checkingAccount
accountNumber balance checkProcessingCharge checkNumber
getBalance（） setBalance（） postCharge（）

saveingAccount
accountNumber balance interestRate
getBalance（） setBalance（） postInterest（）

当然，这会导致getBalance()、setBalance()被多次测试！

1.4 面向对象测试的层次

一般有四层:

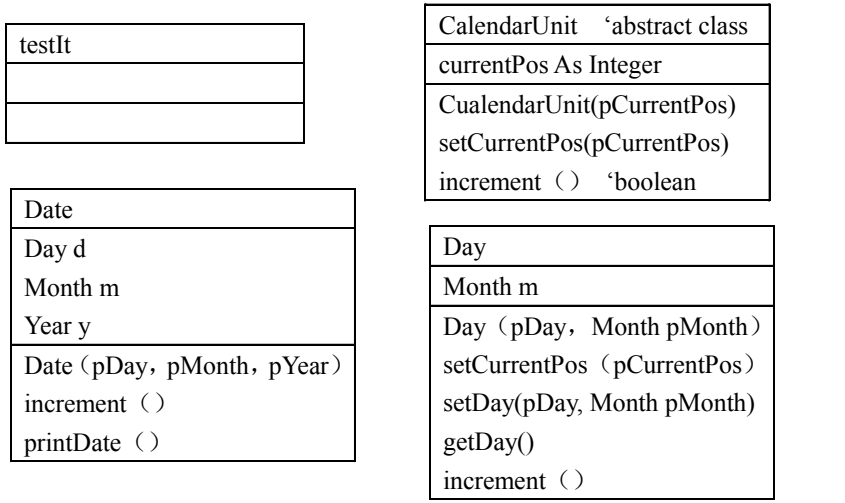
操作/方法测试	单元测试(方法相同)
类测试	单元/交互测试
集成测试	集成测试
系统测试	系统测试(方法相同)

1.5 GUI测试(略)

1.6面向对象软件的数据流测试(略)

1.7 例子的说明

面向对象的日历算法



日、月和年的类，都继承抽象类calendarUnit的属性和方法。

date对象由日、月、年的实例组成。具体伪代码实现在下一节说明。

2.1 以方法为单元

以方法为单元的测试，等价于结构化程序设计中的单元（函数/过程）测试，可以使用所有传统黑盒测试和白盒测试技术。

还记得McCabe圈数V（G）的计算方法吧，

V（G）=e-n+2p

e：边数；

n：节点数

p：连接区域数

对于类中的方法，都不会有复杂的逻辑，我们也不再做特别的说明。

书上的例子：o-oCalendar的介绍基本上是一个程序阅读和理解的过程，对于测试用例的生成并无新的知识点，给大家留10分钟阅读时间，看看有什么问题？

P212~P275

2.2 以类为单元

以类为单元进行测试，重点在于类的消息序列（或事件组合），而对于类中方法的单独测试，前面已经做过说明，不再重复。

为了以实例进行说明，对挡风玻璃雨刷的例子进行改造，将控制杆、刻度盘和雨刷类放在一个类中，进行类单元测试的说明。

这样，方法（getxxxx）感知控制杆、刻度盘，并在leverPosition和dialPosition变量中记录（setxxxx）控制杆和刻度盘的位置。sensexxxx方法会向setWiperSpeed方法发送一个消息，由setWiperSpeed方法设置相应的wiperSpeed。

Class windshieldWiper

Private wiperSpeed

Private leverPosition

Private dialPosition

windshieldWiper(wiperSpeed, leverPosition, dialPosition)

getWiperSpeed()

setWiperSpeed()

getLeverPosition()

setleverPosition()

getDialPosition()

setDialPosition()

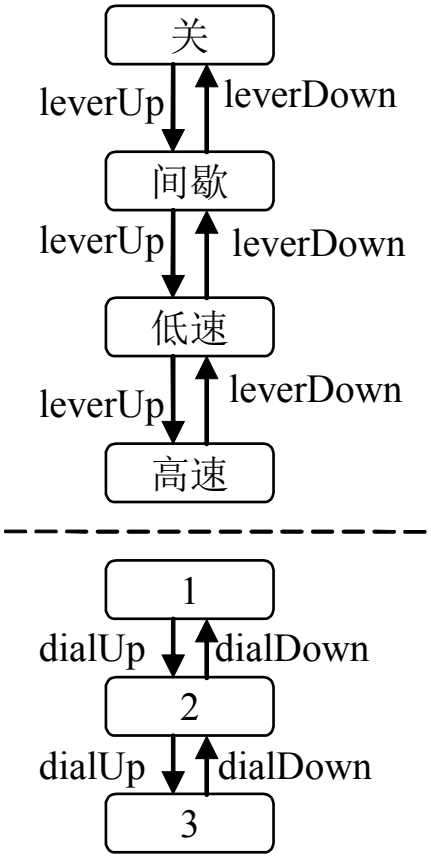
senseLeverUp()

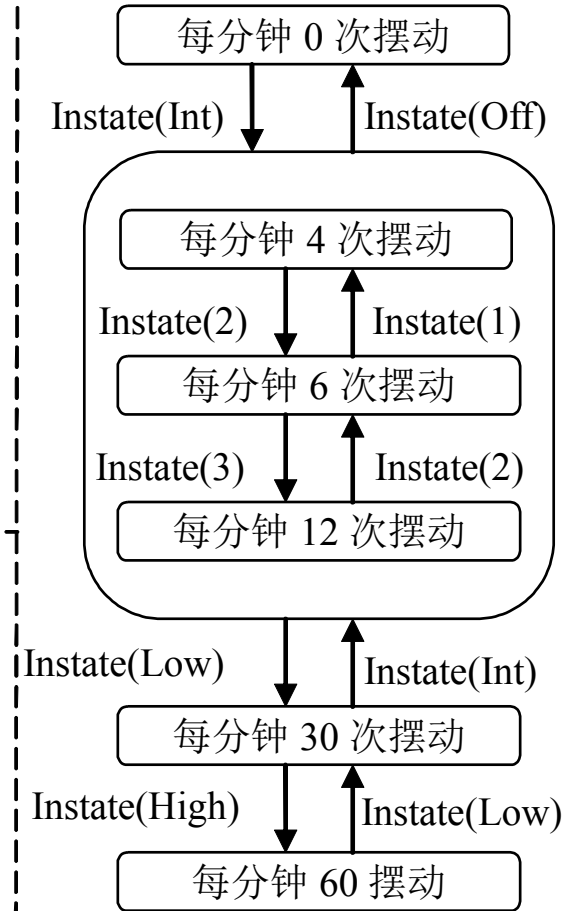
senseleverDown()

sensedialUp()

sensedialDown()

End class windshieldWiper





状态图

senseLeverUp方法的伪代码

```
senseLeverUp()
  Case leverPosition Of
       Case 1: leverPosition=Int
                  Case dialPosition of
                     Case 1: wiperSpeed=4
                     Case 2: wiperSpeed=6
                     Case 3: wiperSpeed=12
                   Endcase
       Case 2:leverPosition=low
                  wiperSpeed=30
       Case 3:leverPosition=high
                  wiperSpeed=60
       case 4:( impossible )
 End case
```

测试驱动器类的伪代码:

```
Class testSenseLeverUp
wiperSpeed
leverpos
dialpos
testResult
Main()
    testCase=instantiate windshieldWiper(0,Off,1)
    windshieldWiper.senseLeverUp()
    leverPos=windshieldWiper.getleverPosition()
    If leverPos=Int
       Then testResult=Pass
       Else  testResult=Fail
End main
```

从上例我们可以看到,对SenseLeverUp方法的测试,已经包含了对其它方法的调用,而不是桩函数或驱动函数。因为它们在一个单元中，不会再设计新的桩函数或驱动函数。

此时，我们对测试的关心是状态的变化，我们知道，控制杆有4个状态，有6个状态变化过程。

如果再加上刻度盘的状态，还有4个状态变化。书上的表15-1给出了控制杆的6个状态变化。

而对于刻度盘的考虑，则采用场景的方法进行用例设计。

关于这些内容的详细描述，见P219~220，这里不再多讲。

3 面向对象的集成测试

3.1 集成测试的UML支持

在后面的介绍和讲解中,将使用UML进行说明，UML是面向对象设计中十分重要和有效的技术，用例图、类图、对象图、包图、状态图、顺序图、协作图、活动图是UML设计的主要模型图。而其中的序列图、协作图、状态图是我们进行软件测试的有利工具。

✓协作图?

✓主要建模对象之间的交互和链接关系（一条链接就是类图中一个关联的实例化），图15-9。

✓顺序图?

✓描述了对象之间动态的交互关系，着重体现对象之间消息传递的时间顺序，图15-10。

✓状态图?

✓可被用来描述一个类或整个应用系统的外部可见行为。

从上面的定义我们可以知道，状态图是系统的外部可见行为，因此从状态图可以生成系统的事件测试用例。

顺序图（协同图）更适合生成消息序列，描述类之间的联系。

顺序图是事件发生的时序，几乎就是一个完整的MM-路径。

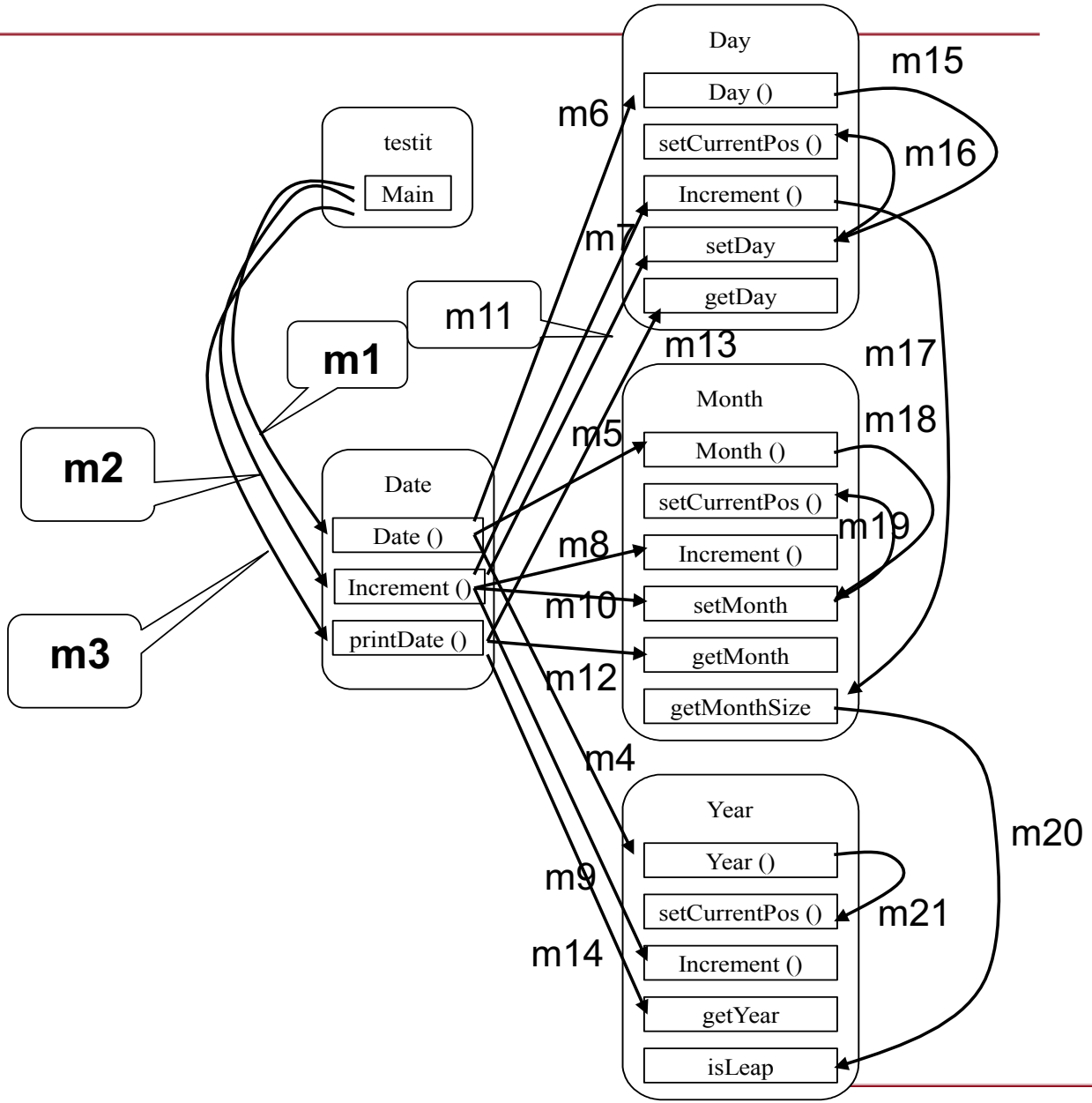
因此，在下面的介绍中，我们以O-OCalendar的测试为例，研究面向对象软件的MM-路径测试方法。

3.2 面向对象软件的MM-路径

定义: 面向对象的MM-路径是由消息连接起来的方法执行序列。

要注意不同类之间的消息调用

图15-11给出了消息视图


testit
Main
m1
m2
m3
Date
Date ()
Increment ()
printDate ()
Day
Day ()
setCurrentPos ()
Increment ()
setDay
getDay
Month
Month ()
setCurrentPos ()
Increment ()
setMonth
getMonth
getMonthSize
Year
Year ()
setCurrentPos ()
Increment ()
getYear
isLeap
m4
m5
m6
m7
m8
m9
m10
m11
m12
m13
m14
m15
m16
m17
m18
m19
m20
m21

以2013年1月3日为测试数据，其MM-路径和伪代码的执行过程如P225 和图15-12所示。

5分钟阅读。

3.3 面向对象数据流集成测试框架

① 事件驱动和消息驱动的Petri网 —EMDPN

(Event Message Drive Petri Net)

 端口输入事件；

 端口输出事件；

 数据点；

 转移汇聚点；

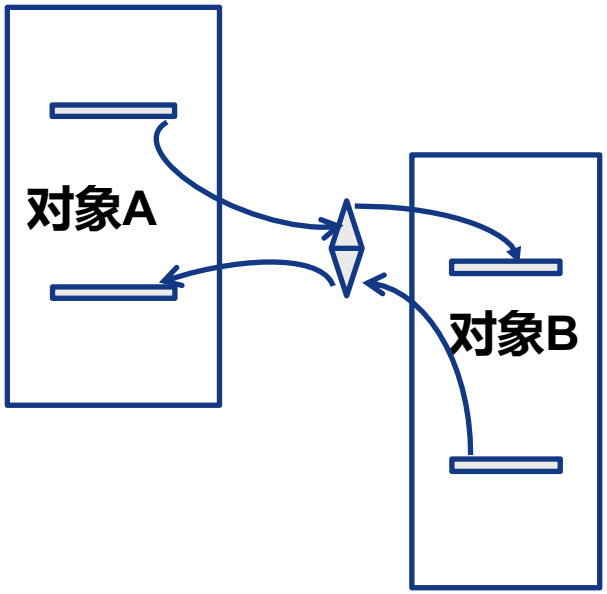
 消息发送/返回；

EMDPN是5元组，（P , D, M, S, In, Out）

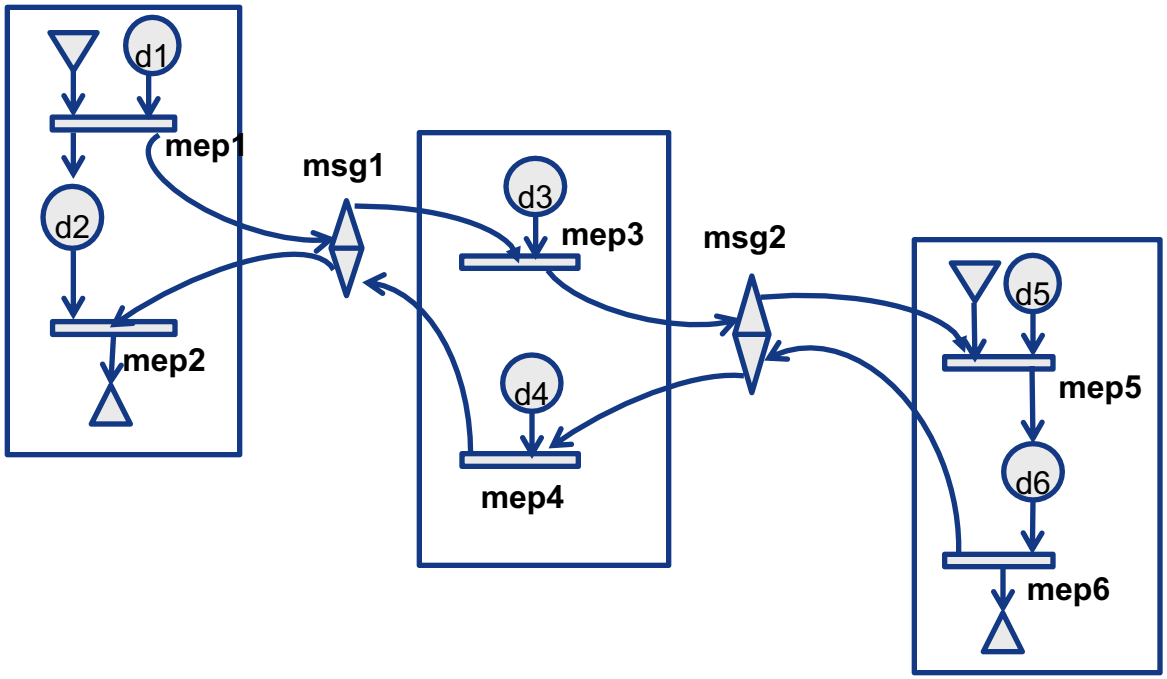
P：端口输入事件集合； D：端口输出事件集合；

M：消息点集合； S：转移点集合；

In：（P∪D∪M）×S Out：S×（P∪D∪M）



- 上图是消息点在EMDPN中的惟一出现方式。
- 借鉴过程代码数据流分析的方法，数据由数据点表示；而数据的使用和定义在类的方法中。



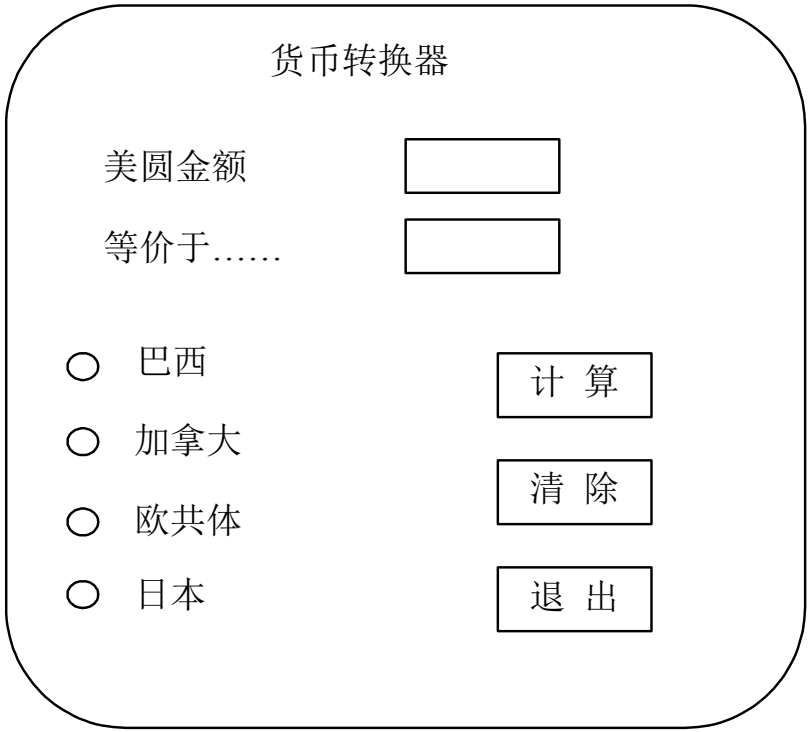
选择2条路径如下：

- Path1=<mep3,msg2,mep5,d6,mep6,return(msg2),mep4,return(msg1),mep2>;
- Path2= <mep6,return(msg2),mep4,return(msg1), mep2>

补充说明：mep3定义d3,mep5使用d5、定义d6，mep6返回d6到d3；mep4使用d4,mep2使用d2.

面向对象的集成测试

货币转换应用程序



正常流程:

1. **输入美圆金额;**
2. **选择货币种类**
3. **选择计算**
4. **输出与美圆金额等价的货币金额**

基于UML的用例分析（阅读P229-233）

基于UML的系统测试（阅读P233-P234）

Q&A