

第05讲 动态建模-UML动态视图

SUST

2020.4

主要内容

- 动态建模*

- 动态视图

 - 交互图*

 - 序列图（顺序图）
 - 协作图

 - 行为图*

 - 状态图
 - 活动图

系统建模

- 一个完整的系统模型必须描述系统的静态和动态两个方面。

静态建模：描述系统中对象，每个对象包含的数据，以及它们之间存在的链接。产生静态模型

动态建模：描述系统运行时的动态行为。产生动态模型

动态建模---系统的动态行为

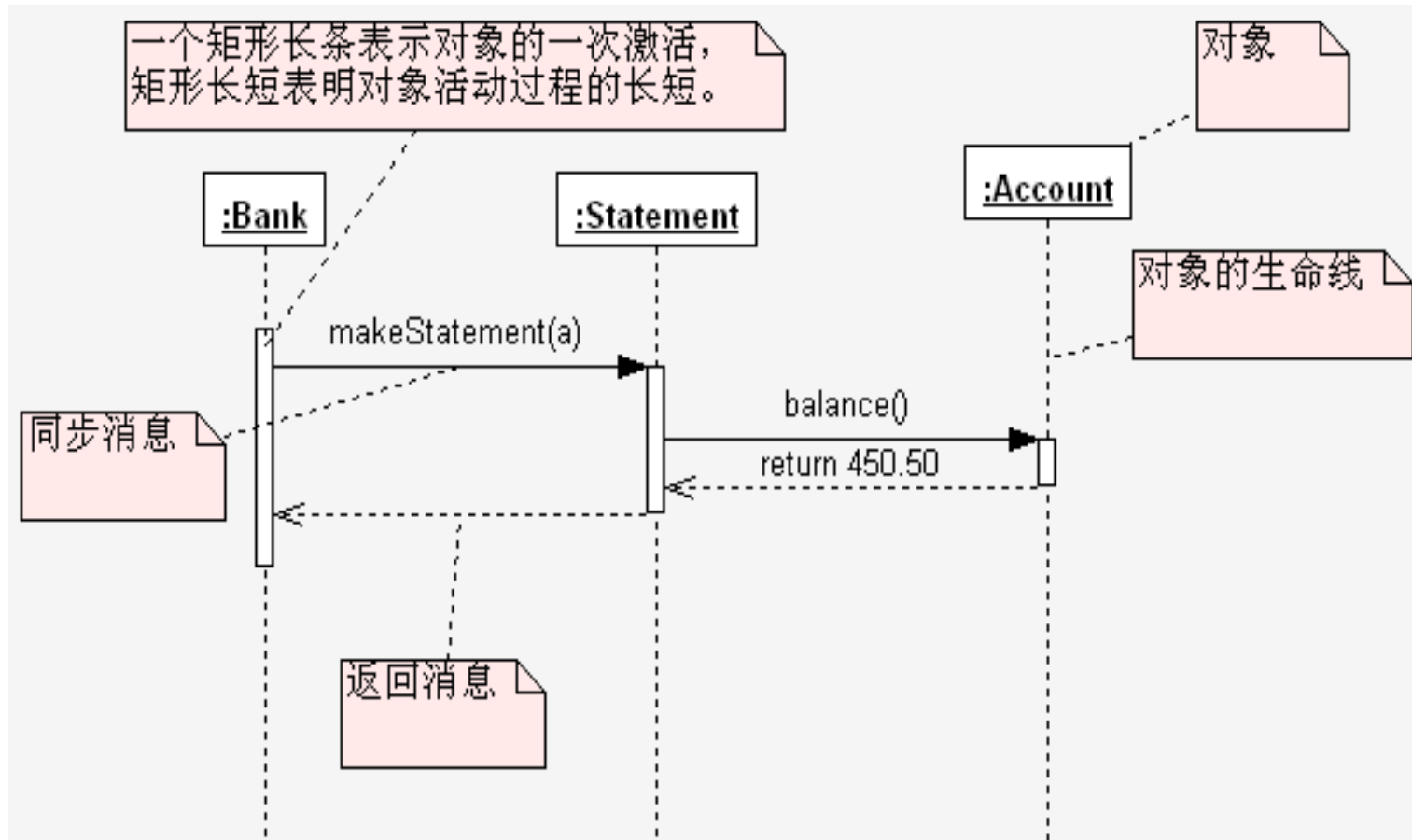
- 系统中的对象是如何进行通信？
- 通信的结果如何？
- **对象通过通信来协作的方式，以及对象在系统的生命周期中改变状态的方式是系统的动态行为**

●案例：

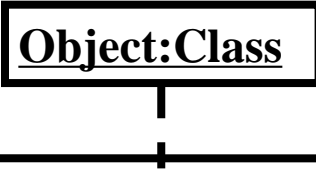




为银行账户打印结算单的过程是：银行对象将相关账户对象传递给另外一个独立对象，该独立对象准备编制所需要的结算单，在编制结算单的过程中查明该账户的当前余额。

顺序图(Sequence Diagram)

- Captures dynamic behavior (time-oriented)



顺序图的可视化图符

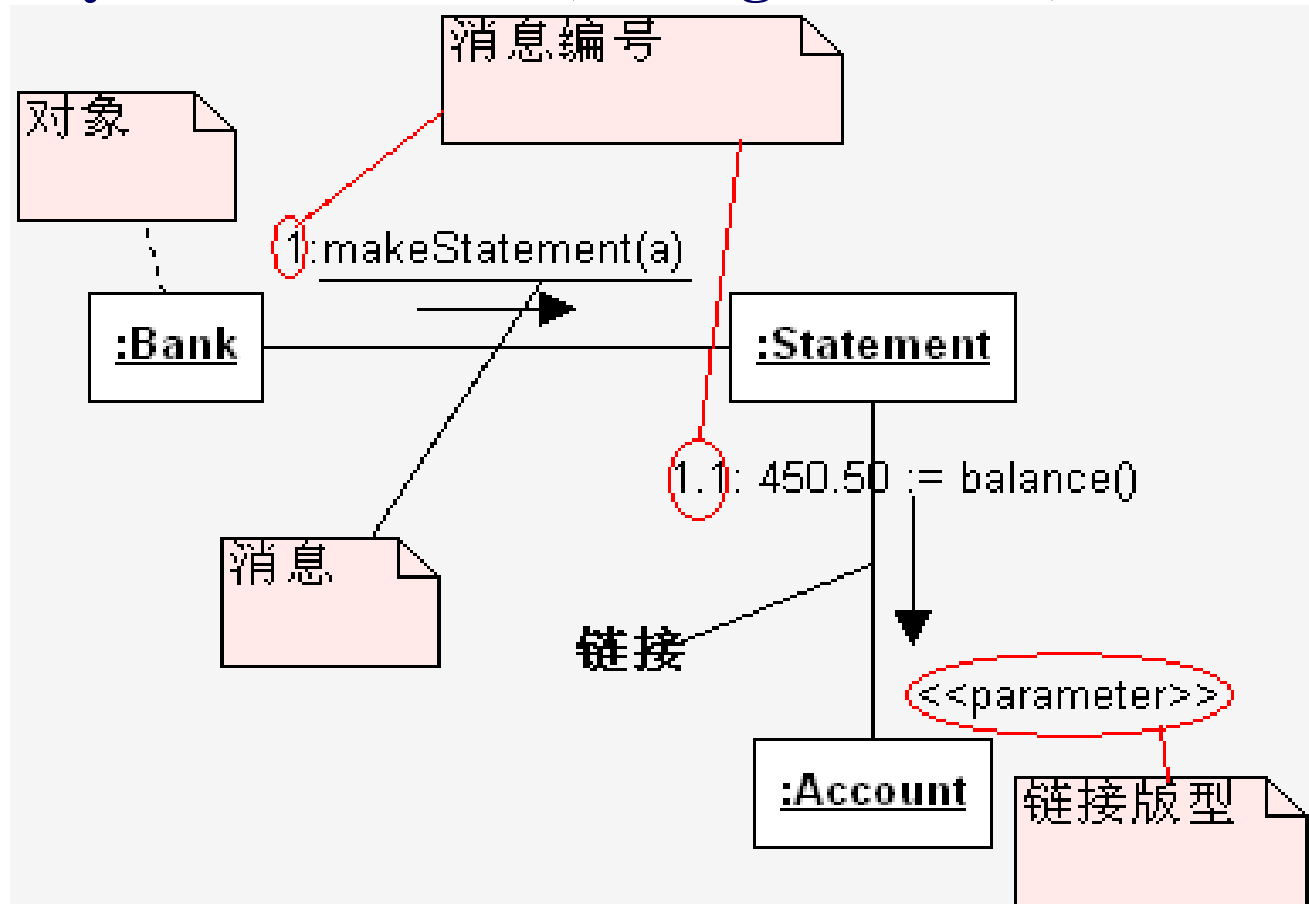
可视化图符	名 称	描 述
	带有生命线的对象	用于表示顺序图中参与交互的对象，每个对象的下方都带有生命线，用于表示该对象在某段时间内是存在的。
	激活的对象	用于表示对象正执行某一动作，在对象的生命线之间发送消息，即激活对象。
	分支生命线	生命线可以劈分成多条生命线，用于表示条件，接收分支消息。
	删除标志	标于生命线或激活上。表示已删除该对象或活动的执行。
	简单消息	表示简单的控制流。用于描述控制如何在对象间进行传递，不考虑通信的细节。

顺序图的可视化图符

可视化图符	名 称	描 述
	同步消息	表示嵌套的控制流。操作的调用是一种典型的同步消息。调用者发出消息后必须等待消息的返回；当处理消息的操作执行完毕，调用者才可继续执行自己的操作。
	异步消息	表示异步控制流。当调用者发出消息后不要等待消息的返回即可继续执行自己的操作。异步消息主要用于描述实时系统中的并发行为。
	返回消息	用于表示从同步消息激活的动作返回到调用者的消息。
	注释体	用于对UML实体进行文字描述。
	注释连接	注释连接将注释体与要描述的实体相连。说明该注释体是对该实体所进行的描述。

协作图

- Captures dynamic behavior (message-oriented)



协作图

- 描述协作对象间的交互和链接，侧重于空间的协作
- 链接显示对象之间是如何联系在一起的
- 协作图显示对象间的链接以及链接对象间如何发送消息
- 协作图从初始化整个交互或协作的消息开始
- 协作图没有显示的返回消息表示，是为了简化协作图。从消息返回的数据可以放在消息名前面，中间用“:=“隔开。

关联(链接)进一步理解

- 两个类之间的关联暗示了这两个类的实例可以链接，并且可以在它们之间发送消息。这些链接主要有：

一个对象作为消息参数传递给另一个对象

<<parameter>>

操作可以创建类的局部实例，然后在操作执行期间向这些实例发送消息。

<<local>>

存在全局对象，其他对象可以和这个全局对象发送消息。

<<global>>

对象向自己发送消息。

<<self>>

对象B是A的属性，则A可以向B发送消息。 **<<Associate>>**

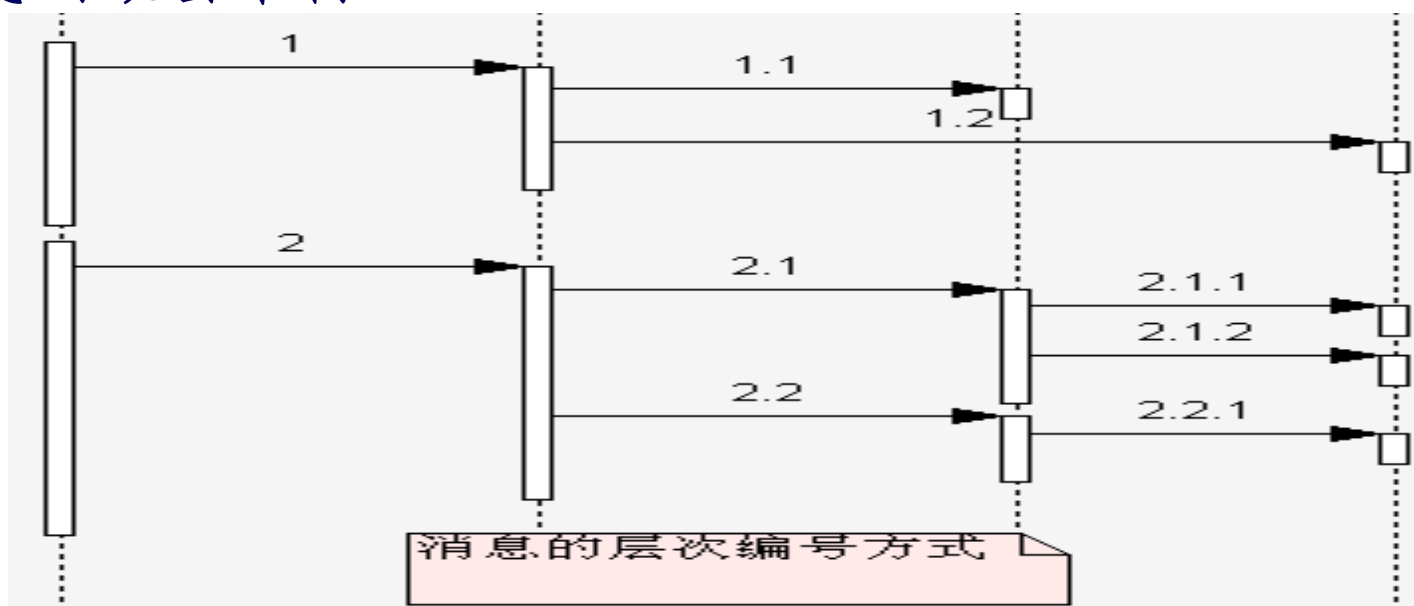
- 以上这些是对象间不同的链接，常常成为关联(链接)的版型（泛型、构造型）。
- 利用协作图可以很好地表示出上述的几类链接以及相应的消息

协作图中消息的编号方法

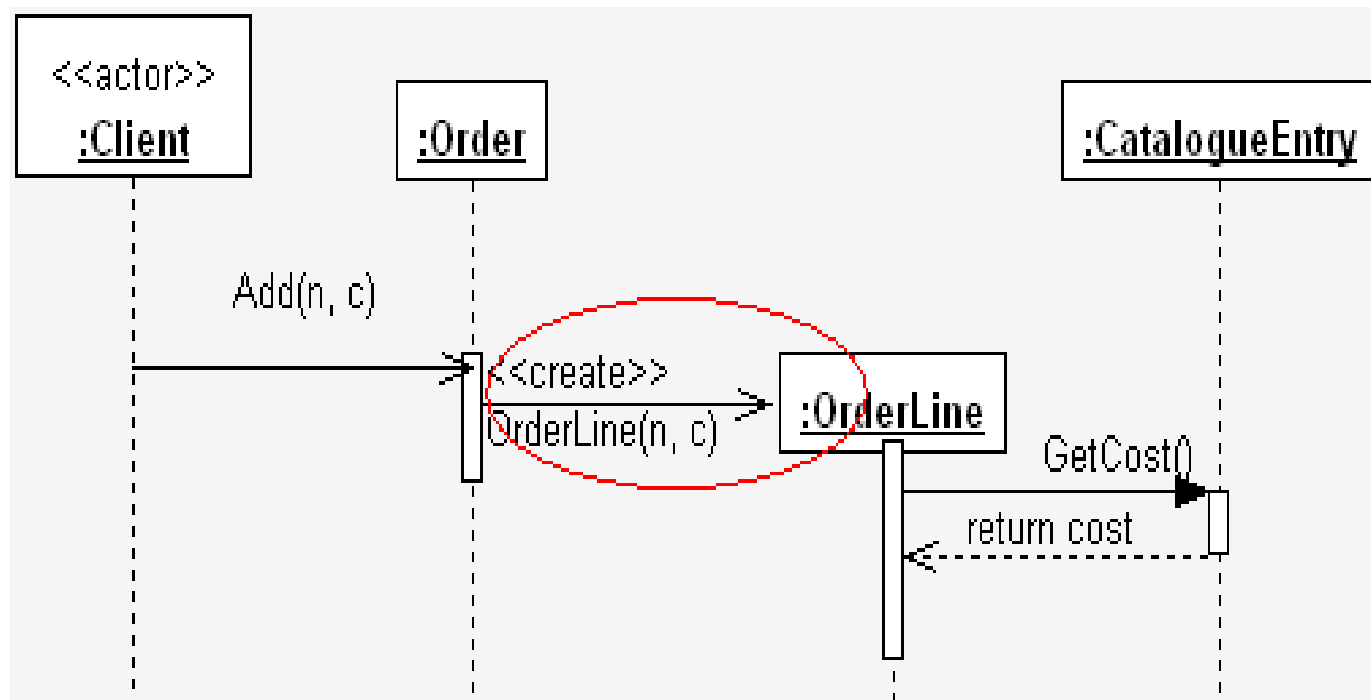
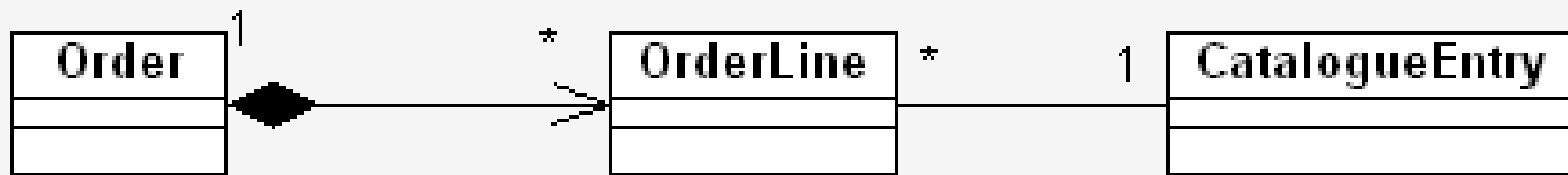
- 和序列图不同，协作图中消息先后顺序不能图形化地显示，而是通过给消息编号来指出它们的发送次序。

顺序法：从1开始，由小到大，顺序对消息进行编号。

层次法：类似于章、节、点的组织方式，能够很好地反映消息的嵌套结构。

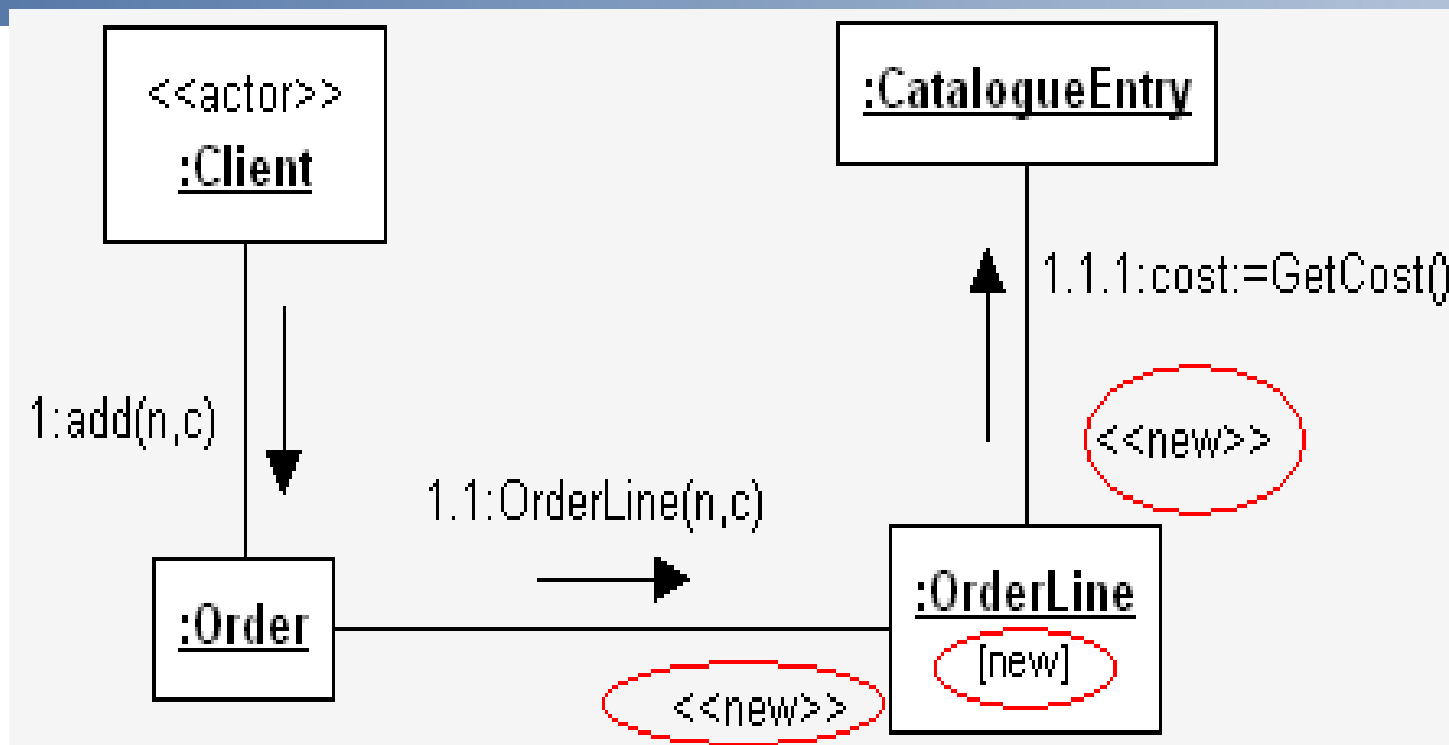


序列图中对象创建的表示



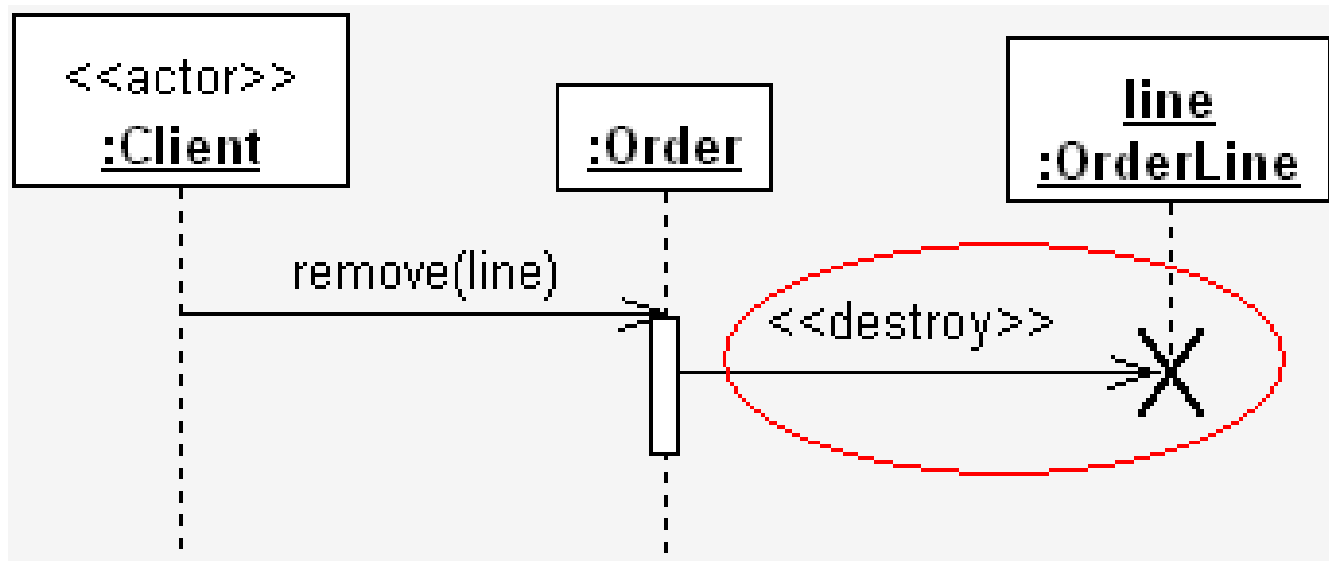
- 在序列图中，创建的新对象画在被创建时的相应位置，创建消息终止于该对象图符号，而不像一般消息终止于一个生命线。

协作图中对象创建的表示

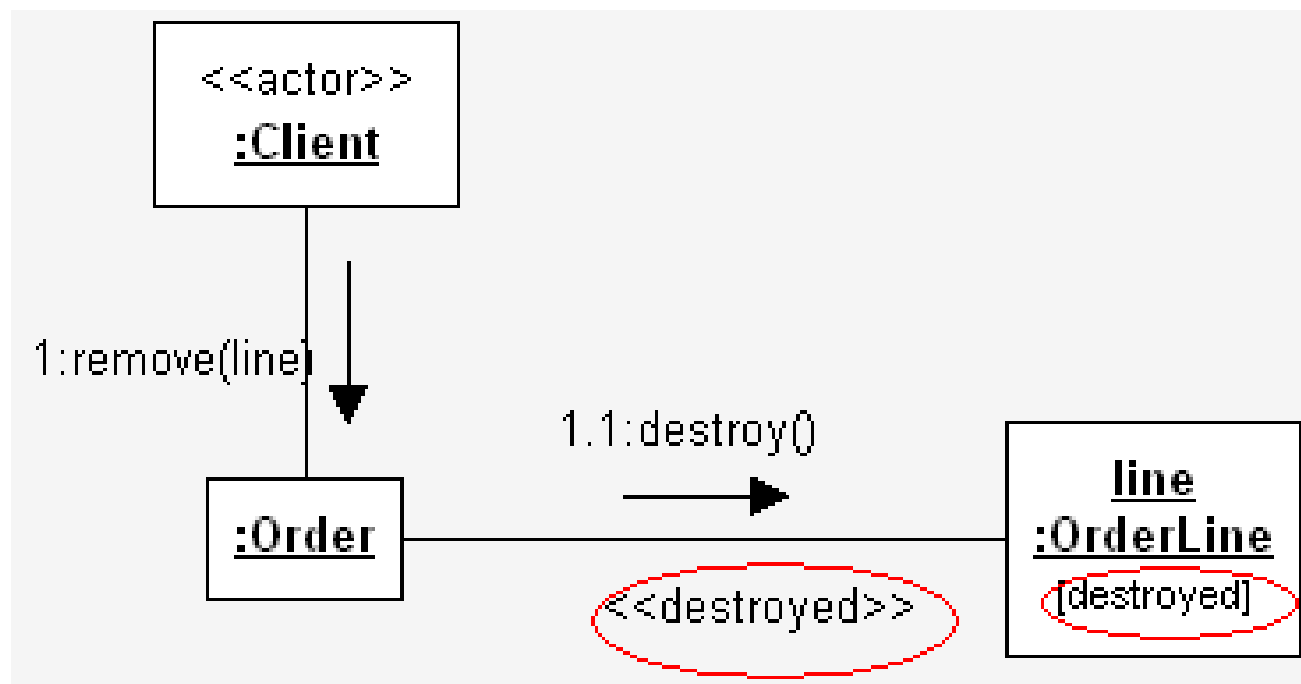


- 协作图不能显示地表明新对象创建的时间。为了区分在交互过程中创建的元素和交互开始就存在的元素，对应于新对象和新链接用特性`new`标注。

序列图中对象删除的表示



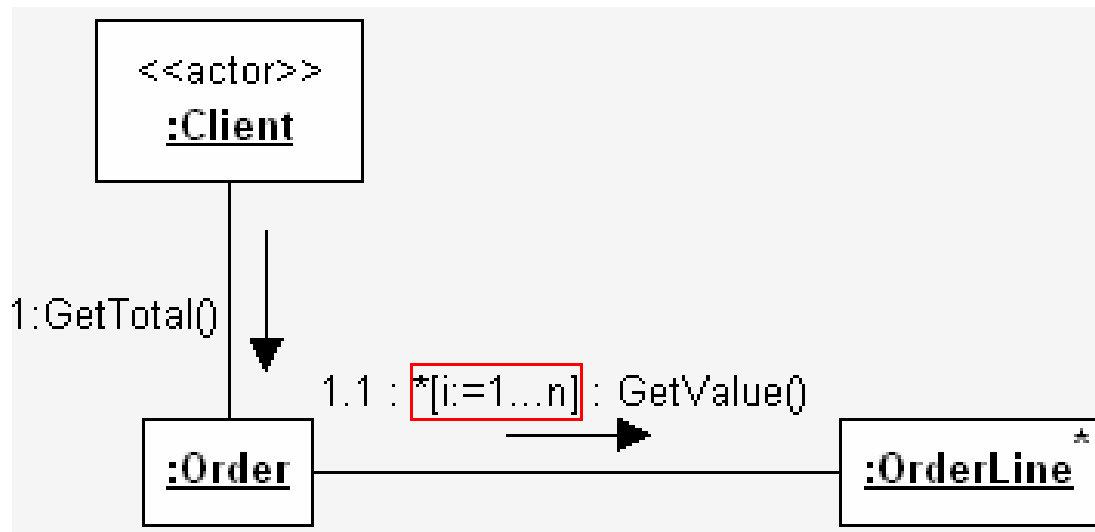
协作图中对象删除的表示



- 如同对象创建一样，协作图也不能显示表明对象被销毁的时间。也需要对被销毁的对象和链接用 `destroyed` 予以标注。

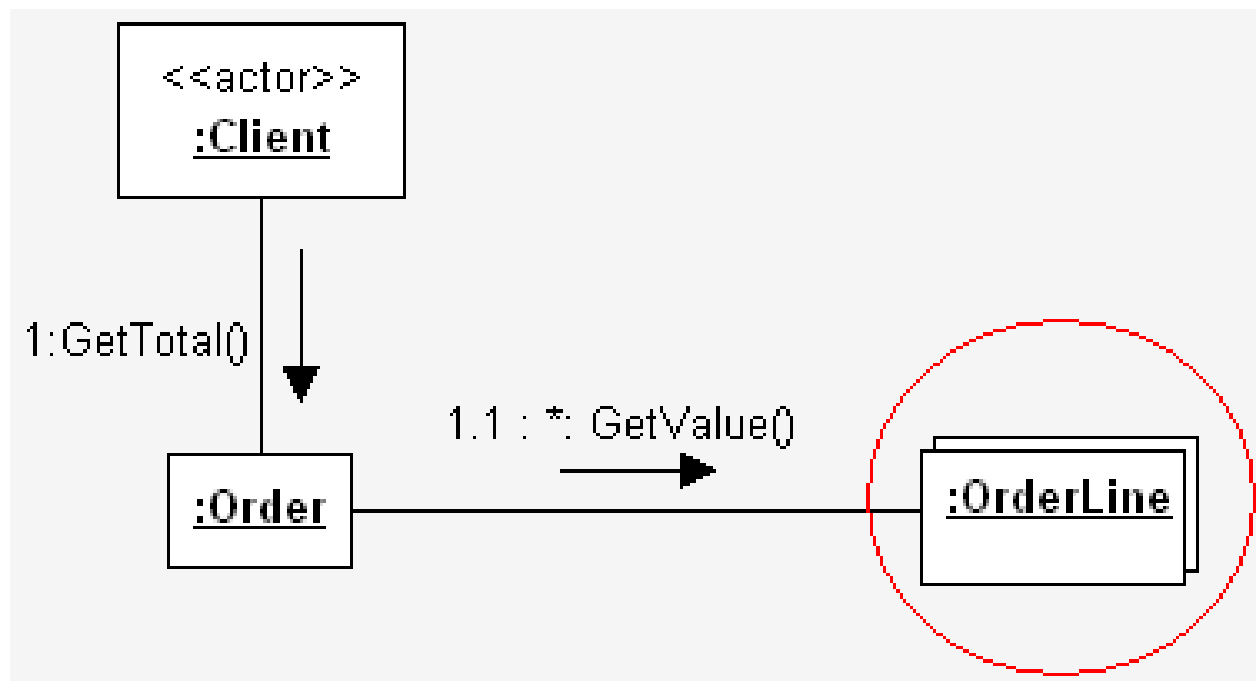
协作图中迭代消息的表示

- 迭代消息表示一个消息可能在一次交互中被多次发送。在消息上加一个递归(recurrence)来表示。
 - 递归：包括一个写在消息编号后的星号,可能还跟上一个迭代子句。
 - 迭代子句：指定了重复的详细信息.如果只是表示消息可能重复,可以省略迭代子句。



协作图中对象重数—多对象的表示

- 在许多情况下，充当特定角色的对象的数目可以因交互的不同而不同。需要在交互中体现重数（把静态关系中的重数、聚合、组合表示出来）



两类特殊消息

请同学们查询资料补充完整以下概念：

- 条件消息：
- 自返消息：

顺序图和协作图的联系

●区别与联系:

序列图和协作图统称为交互图

虽然序列图和协作图都用来描述对象间的交互关系,但侧重点不一样。序列图着重体现交互的时间顺序,协作图则着重体现交互对象间的消息连接关系,进而体现之间的静态链接关系。

序列图和协作图之间是可逆的。即可由序列图画协作图,反之亦可。

动态建模

行为图*

- 状态图
- 活动图

引言

●动态行为描述

交互图显示了在较短的一段时间内系统中的对象之间传递消息的情况，描述的是很多对象之间的动态交互关系。

如何详细地描述一个对象在其生命周期内能够接收的所有可能的消息序列，以及它对这些消息的响应，还有该对象在不同时期所经历的不同状态 呢？

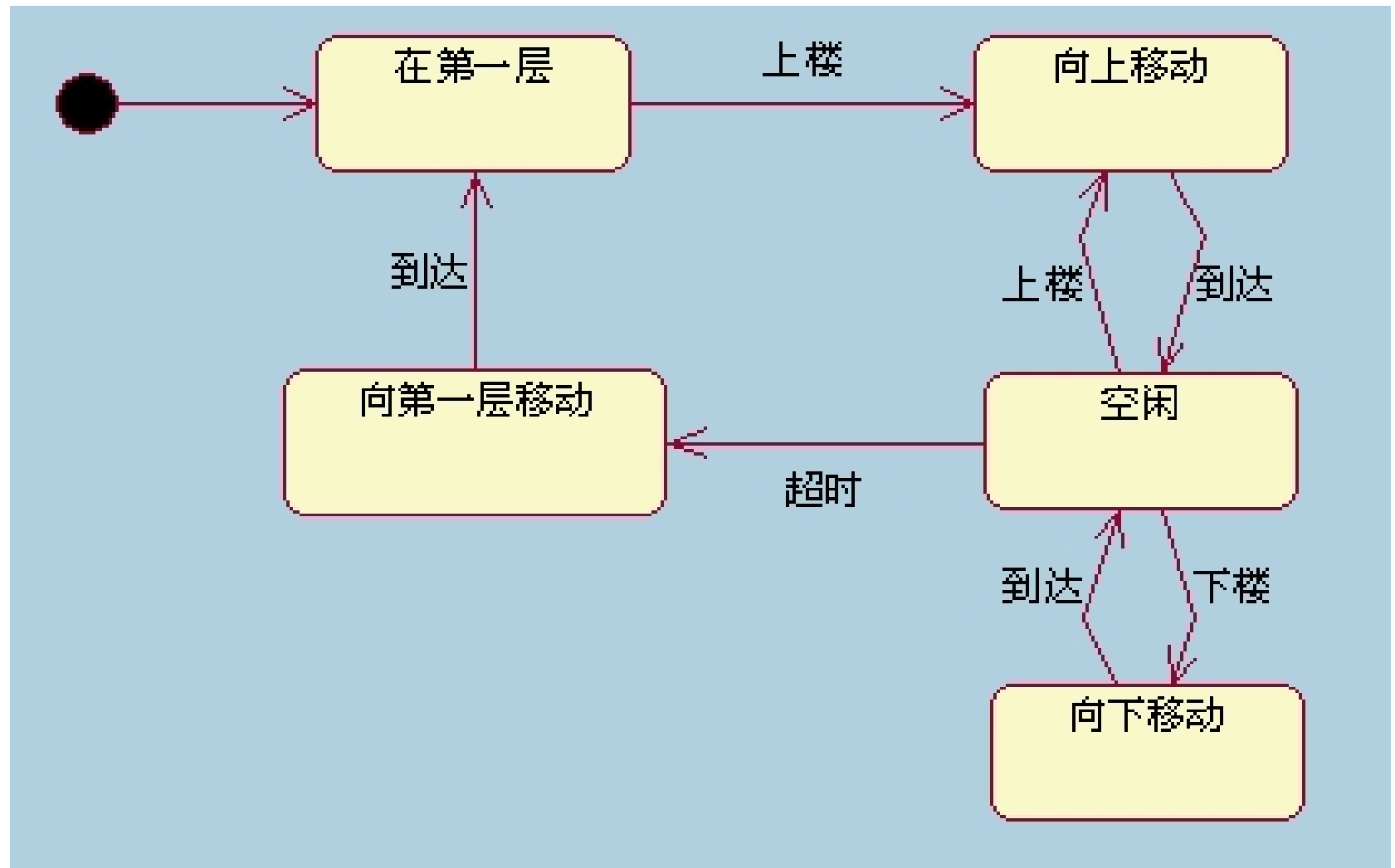
状态图

Statechart Diagram

举例说明

- 支票(对象)已付(状态)
- 汽车(对象)停在那儿(状态)
- 发动机(对象)正在运行(状态)
- 小王(对象)已婚(状态)
-

状态图(Statechart Diagram)



状态图

- 状态图描述对象在生命周期中可能的状态s以及每一种状态的重要行为；描述它所检测到的事件以及什么样的事件引起对象状态发生改变。

通过状态图可以了解到一个对象所能到达的所有状态以及对象检测到的事件（收到消息）对对象状态的影响等

所有的类，只要它有可标记的状态和复杂的行为都应该有一个状态图

- 为了构造一个对象的状态图，必须首先至少暂时地确立对象能够处于什么状态以及他能够检测什么事件。

状态、事件和迁移

- 状态

对象操作执行的结果

由对象的属性值以及指向其它对象的链来决定的



- 状态的重要特征

一个对象有若干个可能的状态，并且在任何给定时间恰好处于这些状态中的一个

对象可以改变状态

不同时间，一个对象可能依赖其状态对同一刺激做出不同的响应

- 区分状态的基本原则

处于一个特定状态的对象对至少一个事件的响应和它处于其他状态时对该事件的响应不同。

状态、事件和迁移

- 事件(event)

当某些事情发生时，对象的状态发生改变，称改变对象状态的事情为“事件”
表现为发送给对象的消息。

例如：**CD**播放机对象可能检测到的三个事件：**Load**、**Play**、**Stop**。

例如：付了支票、开始启动汽车

- 迁移(Transition)

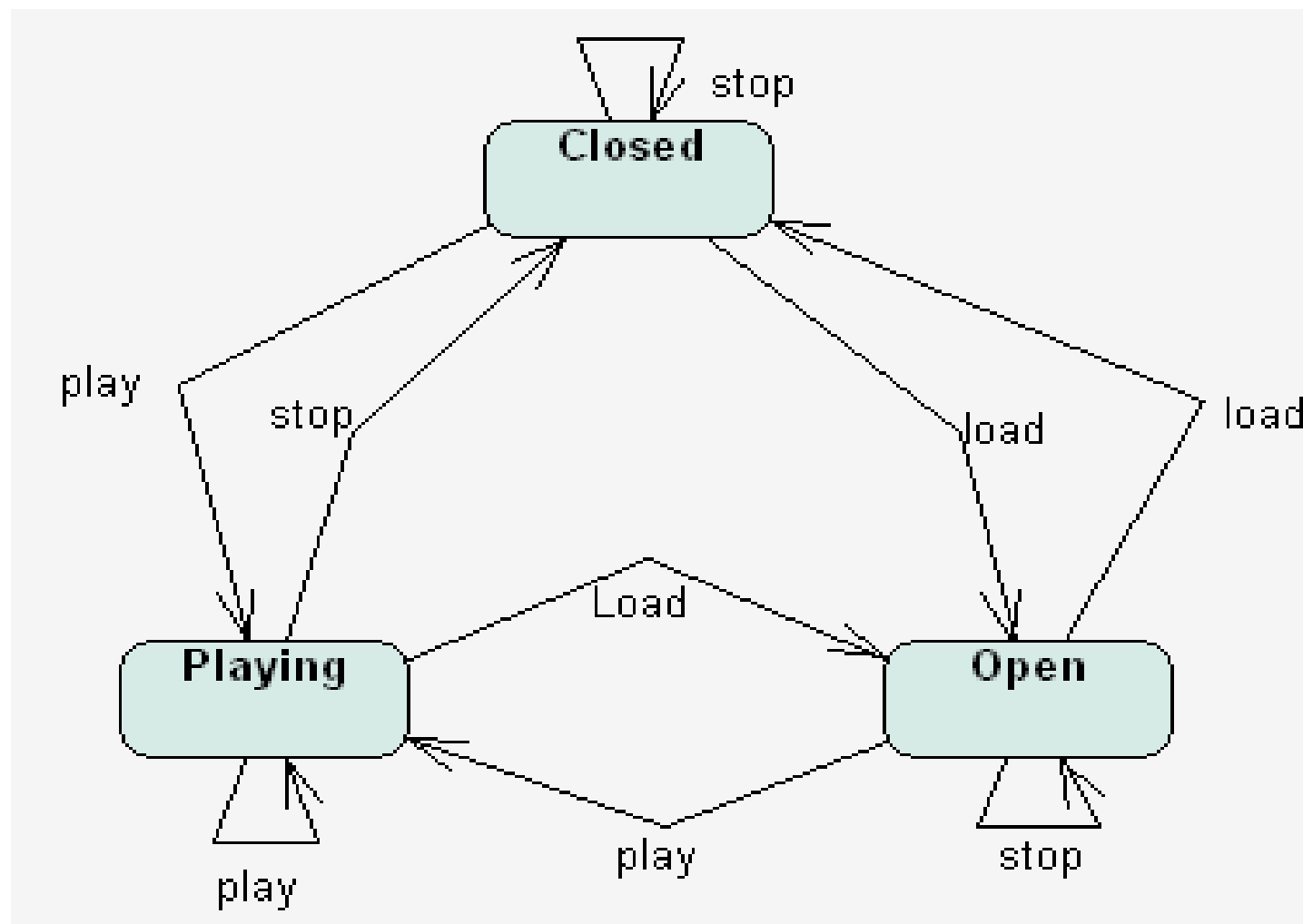
检测到一个事件可能导致对象从一个状态移动到另一个状态，这样的移动成为迁移或转换

状态迁移用连接两个状态的箭头表示。箭头必须标注一个事件的名字。

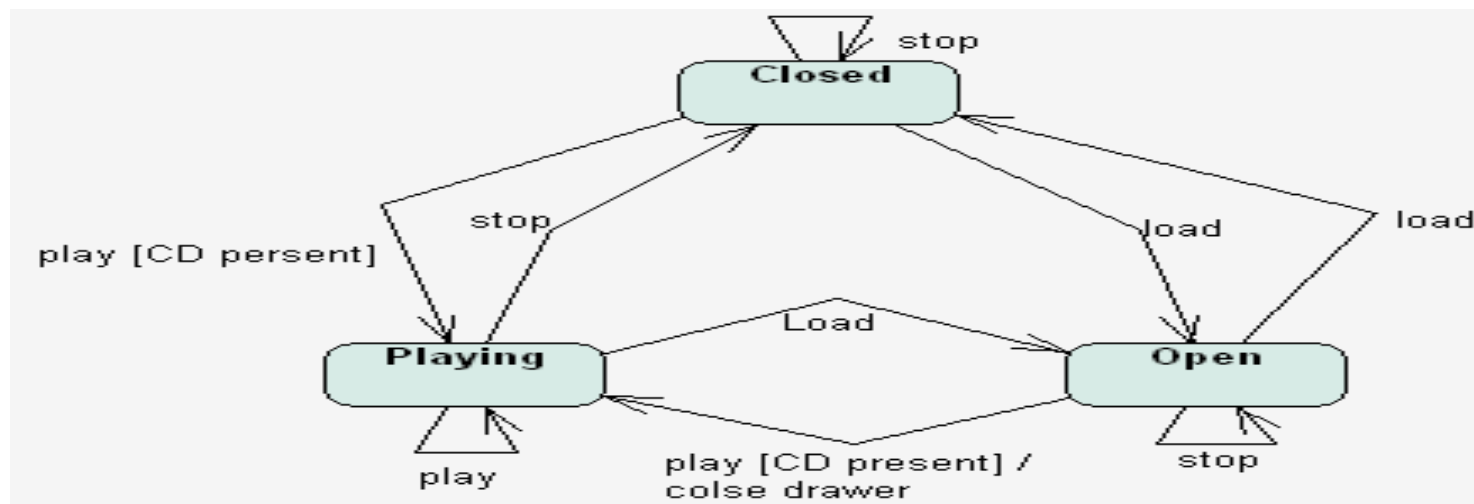
箭头的含义：如果对象在处于箭头尾的状态时接收到事件，对象将迁移到箭头所指向的状态。



状态图的表示



状态迁移的详细表示



- 状态之间的迁移可带有标注，由三部分组成(每一部分都可省略)，其语法为：

事件(名)[守卫条件]/动作(名)

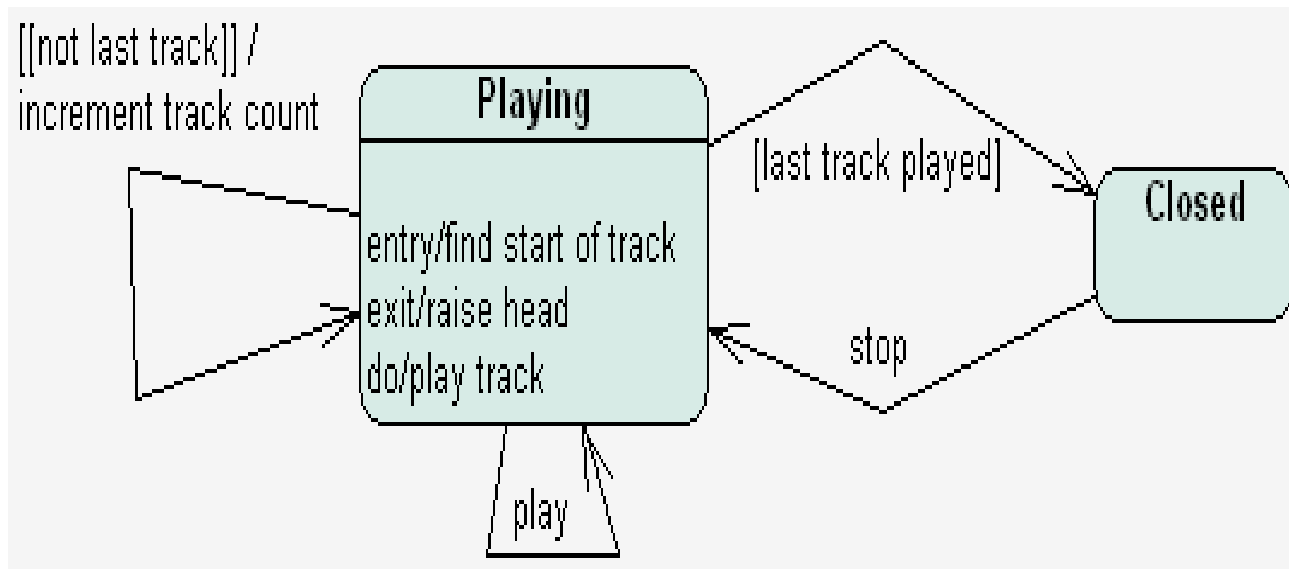
守卫条件：迁移说明的一部分。表示在满足守卫条件的情况下迁移才被激发。

动作：表示对象响应特定事件（比如迁移事件）时做什么。动作被看成是简短的一段处理，所花费的完成时间可以忽略。

特殊迁移

●完成转换(completion transition)

对象某个状态的活动没有被中断自动结束了，引起对象状态的迁移，这种迁移称为完成迁移。

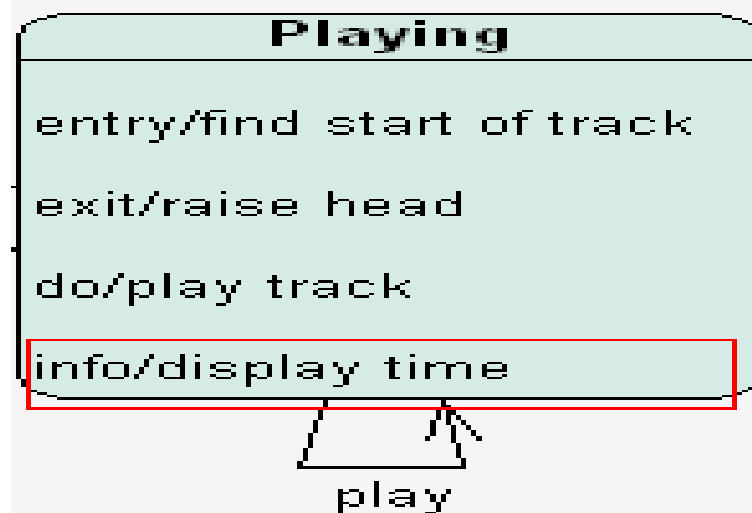


特殊迁移

- 内部迁移

表示这样的事件：事件不触发状态的改变，让对象停在同一状态上，同时也不触发入口和出口动作的执行。

内部迁移写在状态之中，标注为引起该转换的事件的名字。



状态图详细表示

- 状态的名称: **Playing, Stop, Closed**
- 入口和出口动作: 状态图中的内部动作

入口动作(Entry): 指定进入一个状态的(第一个)动作, 例如给属性赋值或发送一条消息。语法如下:

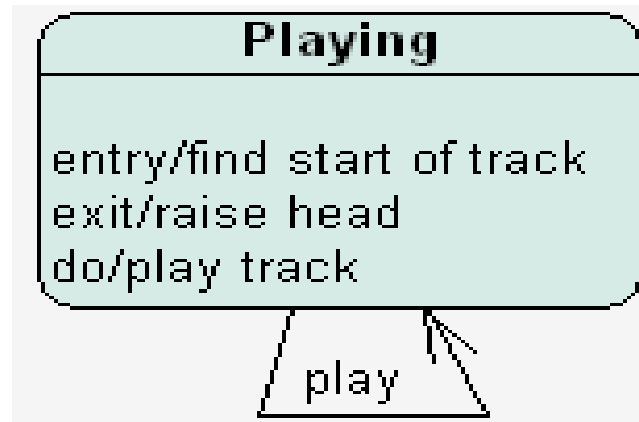
entry/动作名

出口动作(Exit): 指定退出一个状态的动作。语法如下:

exit/动作名

活动(Activity): 状态中要花费时间完成的具有延续性的操作。状态图中的包含一个活动。语法如下:

do / 活动名

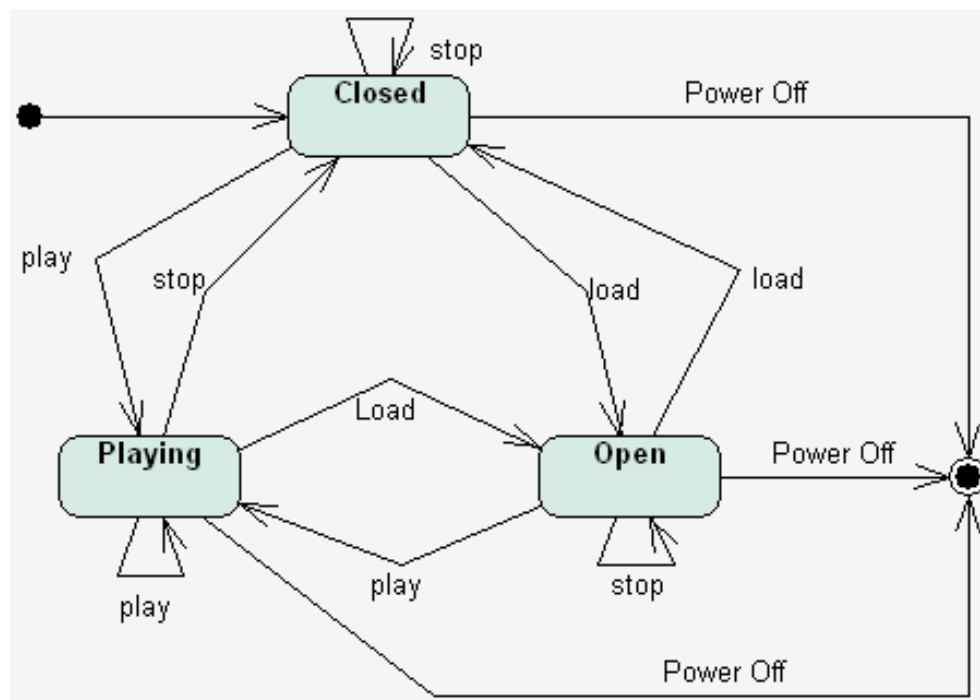


动作和活动的理解

- 动作和活动都是一种过程，都由对象中的方法来实现，但动作与迁移关联，处理较快且不会被中断；活动与状态关联，处理时间较长且可以被事件中中断。

特殊状态---初始状态和终止状态

- **初始状态**：一个黑色小圆点表示。从初始状态出发的迁移表示创建或初始化对象时进入的状态。从初始状态出发的转换不写任何事件。
- **终止状态**：用大圆圈中加一个黑色小圆点表示。终止的状态代表对象在响应撤销、关闭或其他终止事件时到达的状态。



特殊状态---组合状态(composite state)

- 允许一个状态包含若干子状态。这些子状态都具有一些共有的特性。这些特性组合成一组放入一个状态中，这个状态即是组合状态。

如果组合状态是激活的，那么它的子状态中有且只有一个子状态是激活的。

可用于简化复杂的状态图。从组合（超）状态出发的迁移适用于所有嵌套的子状态。

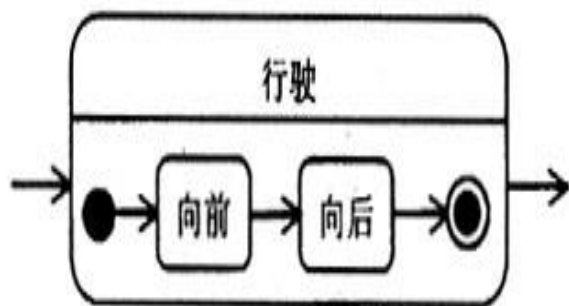


图4 “或关系”的子状态

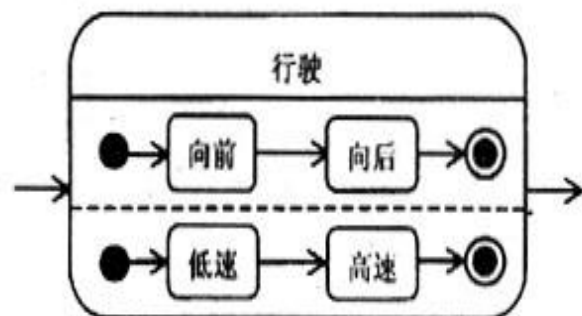
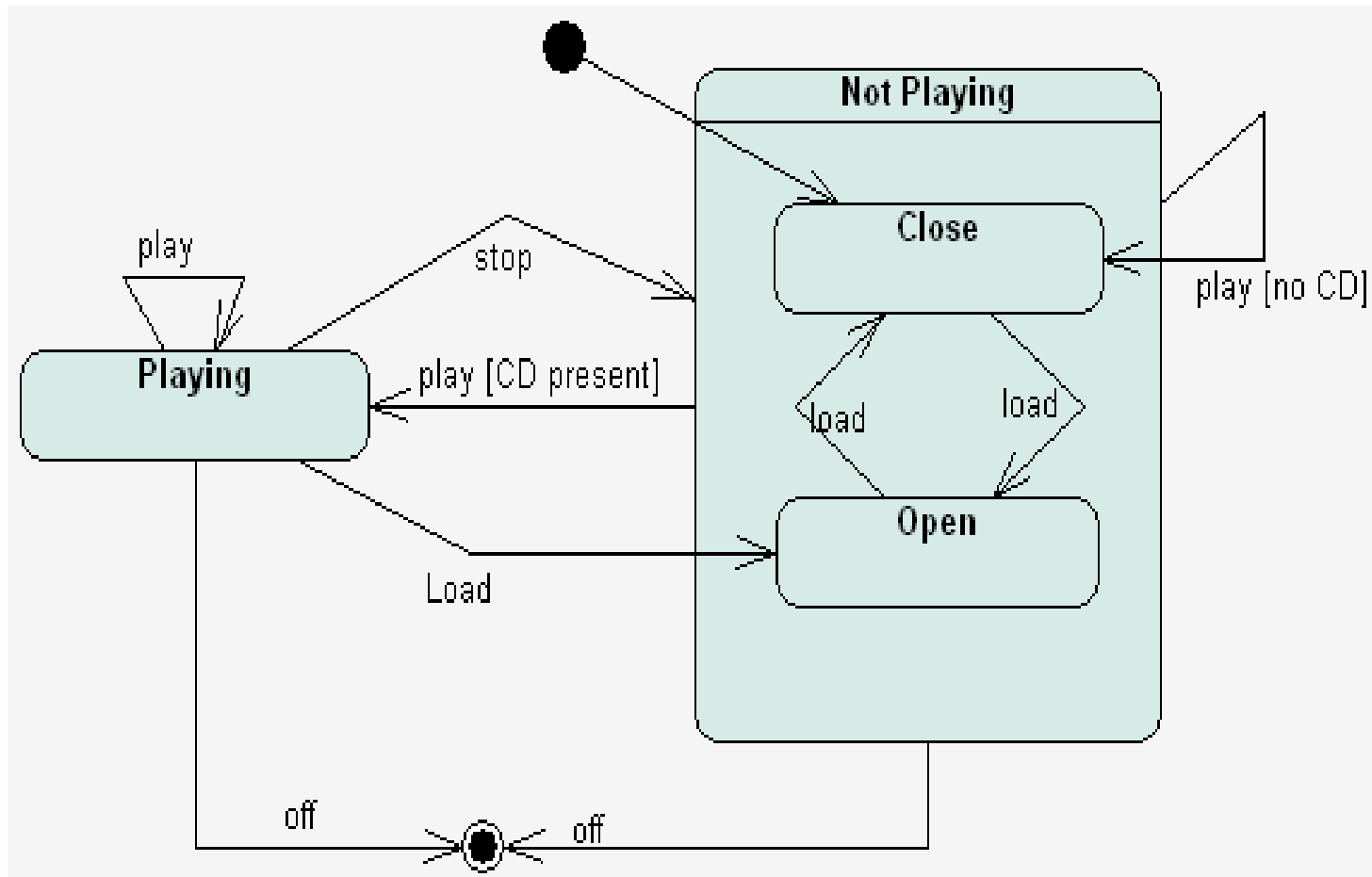


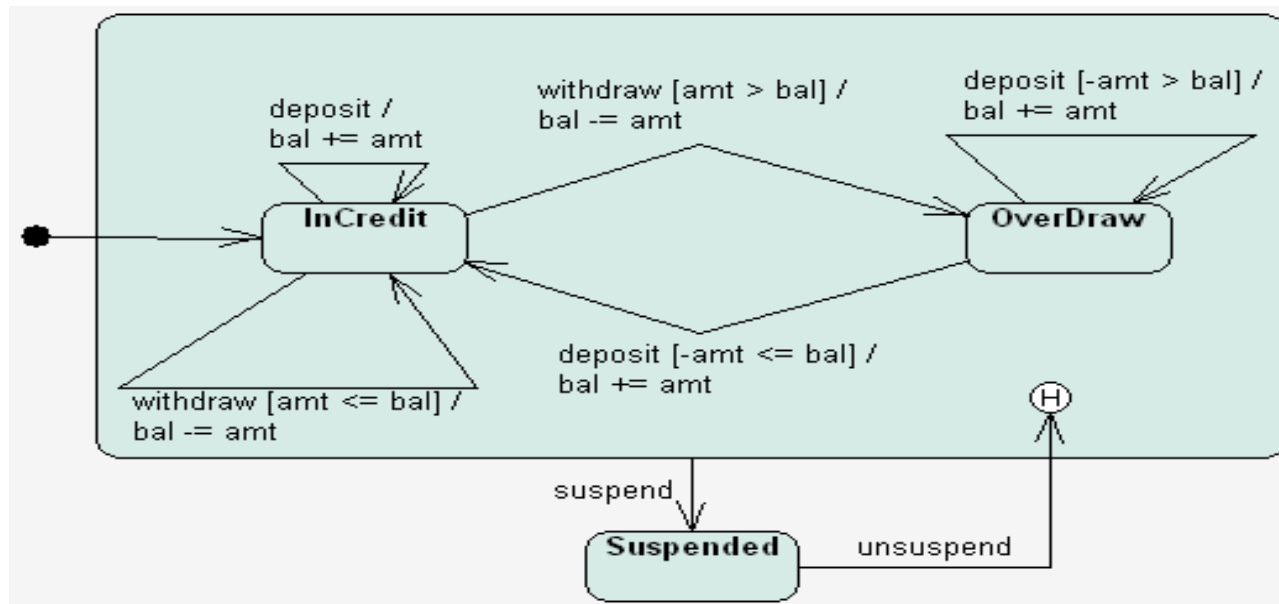
图5 一个具有并发子状态的状态图

特殊状态---组合状态



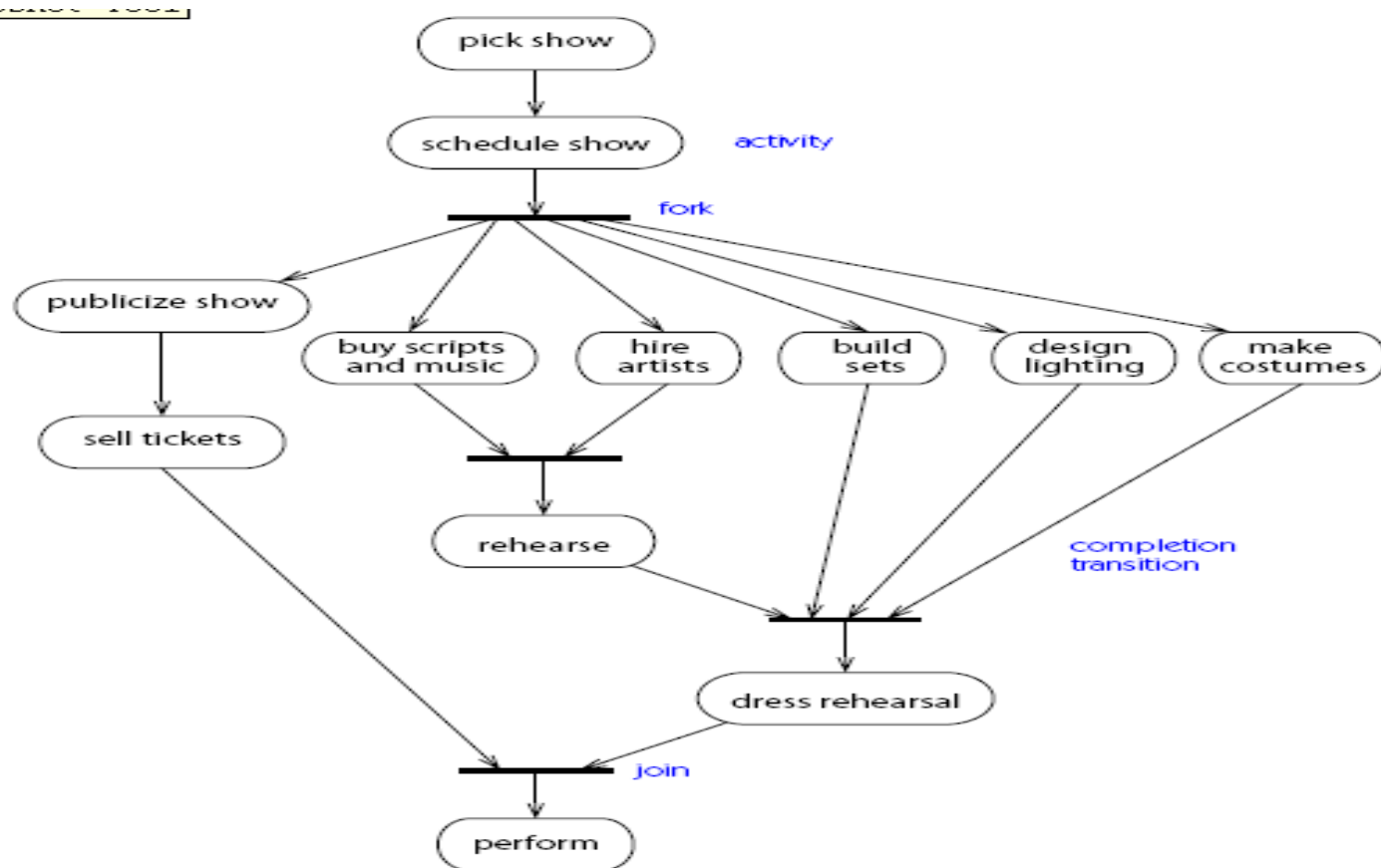
状态图的实现

- 下图是一个银行账户的两状态模型，它表明银行账户不是处于借记状态就是处于透支状态。假定这个示例仅有的两个操作是向该账户存款或取款，守卫条件和动作是根据在交易中涉及的存取款金额 amt 和该账户的当前余额 bal 确定支持哪个操作执行。当账户透支时，不能进行取款。



活动图(Activity Diagram)

- Captures dynamic behavior (activity-oriented)



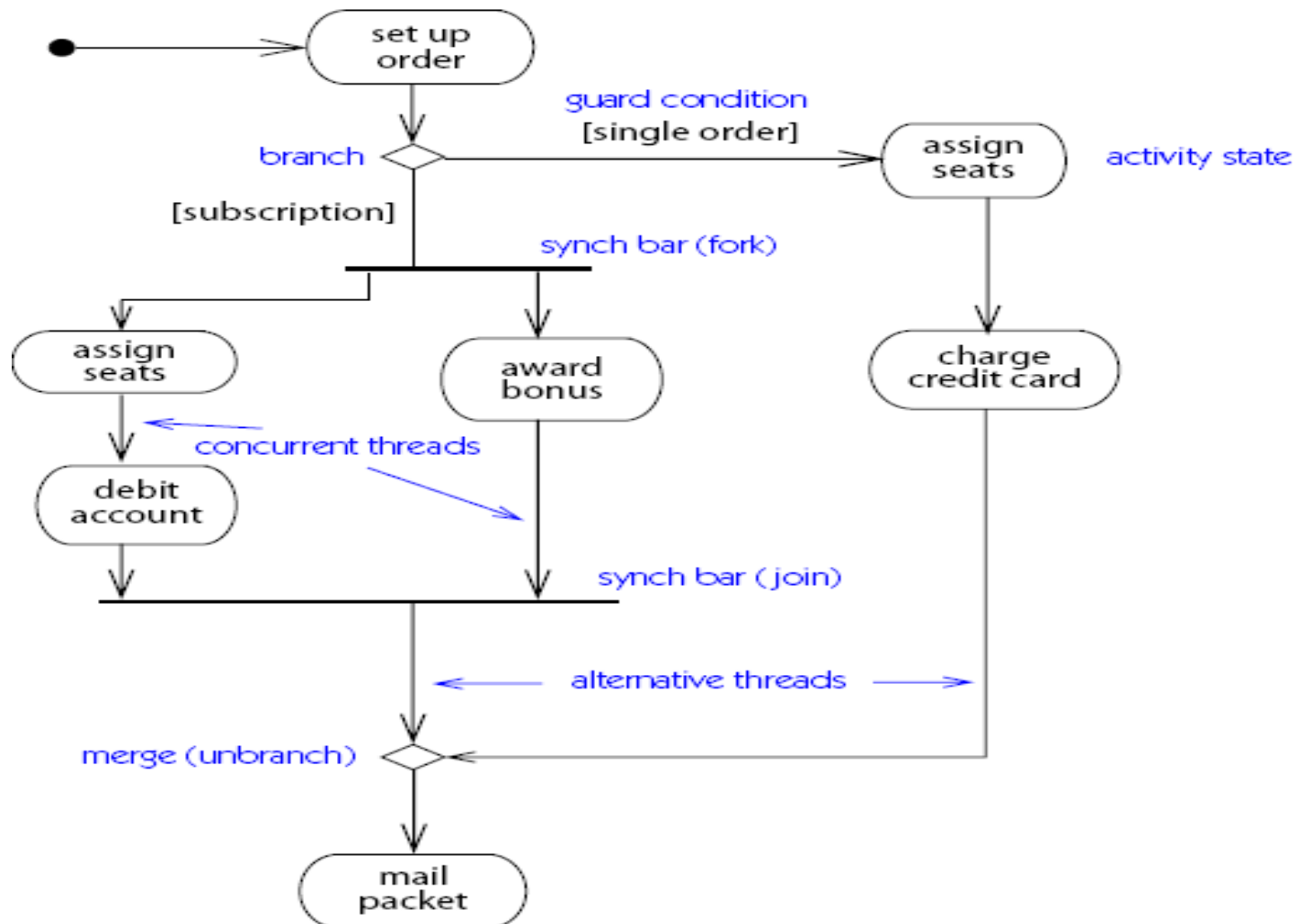
活动图

- 捕获动态行为（面向活动的）
- 目的

给商业 workflow 建模

给操作建模

BoxOffice::ProcessOrder



活动图的要素--活动和迁移

- 活动和迁移

活动仅有一个起始点；

可以有多个结束点；

活动间的迁移允许带有**guard-condition**和**action-expression**，其语法与状态图中定义的相同。

顺序关系：一个活动可以顺序地跟在另一个活动之后，这是简单的顺序关系。

条件关系：如果在活动图中使用一个菱形的判断标志，则可以表达条件关系，判断标志可以有多个输入和输出迁移，但在活动的运作中仅触发其中的一个输出迁移。

并发行为：使用一个称为同步条的水平粗线可以将一条迁移分为多个并发执行的分支，或将多个迁移合为一条迁移。此时，只有输入的迁移全部有效，同步条才会触发，进而执行后面的活动。

活动图的要素--泳道

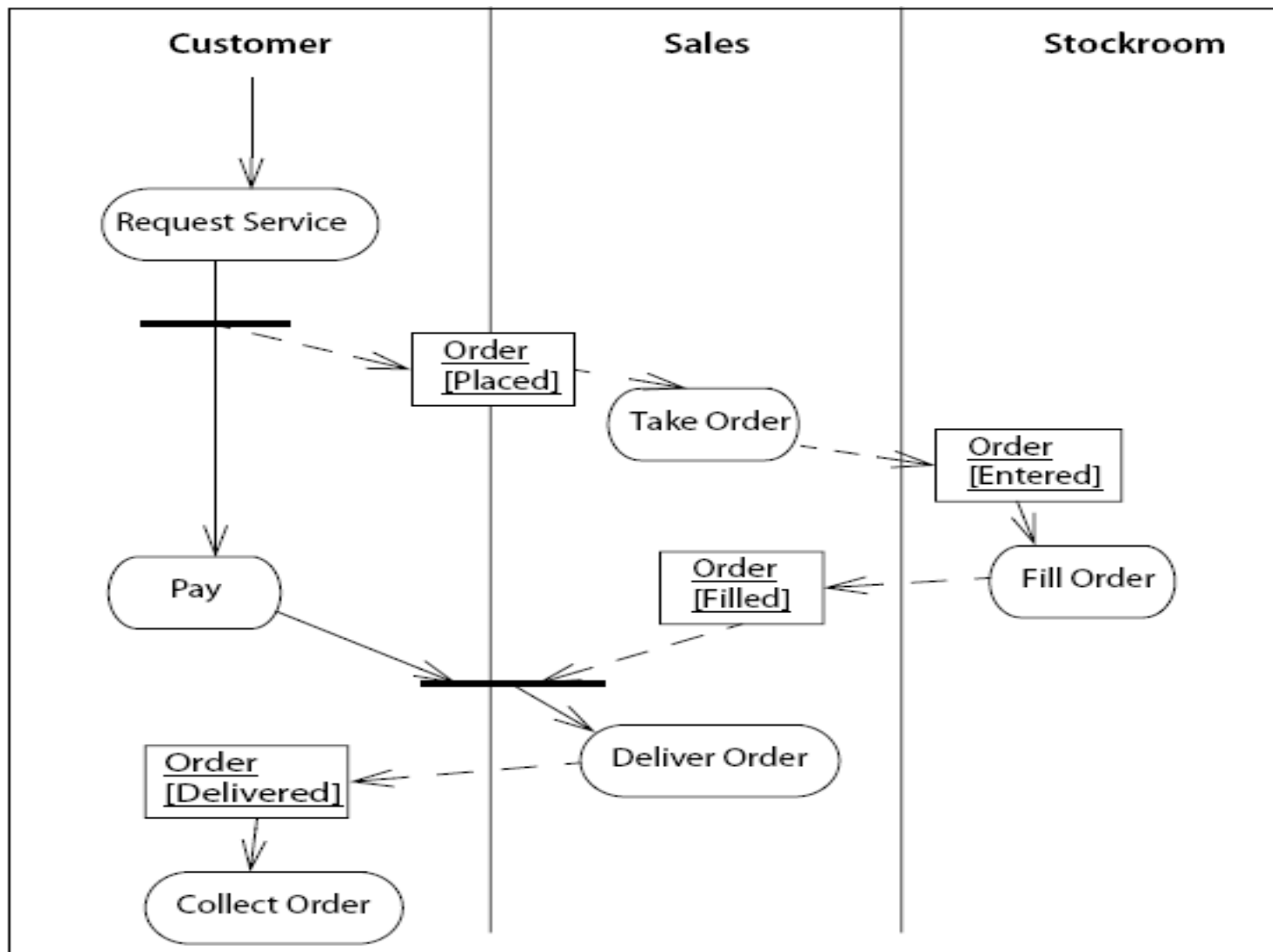
● 泳道

活动图告诉你发生了什么，但没有告诉你该项活动由谁来完成。在程序设计中，这意味着活动图没有描述出各个活动由哪个类来完成。泳道解决了这一问题。它将活动图的逻辑描述与序列图的责任描述结合起来。

泳道用矩形框来表示，属于某个泳道的活动放在该矩形框内，将对象名放在矩形框的顶部，表示泳道中的活动由该对象负责。

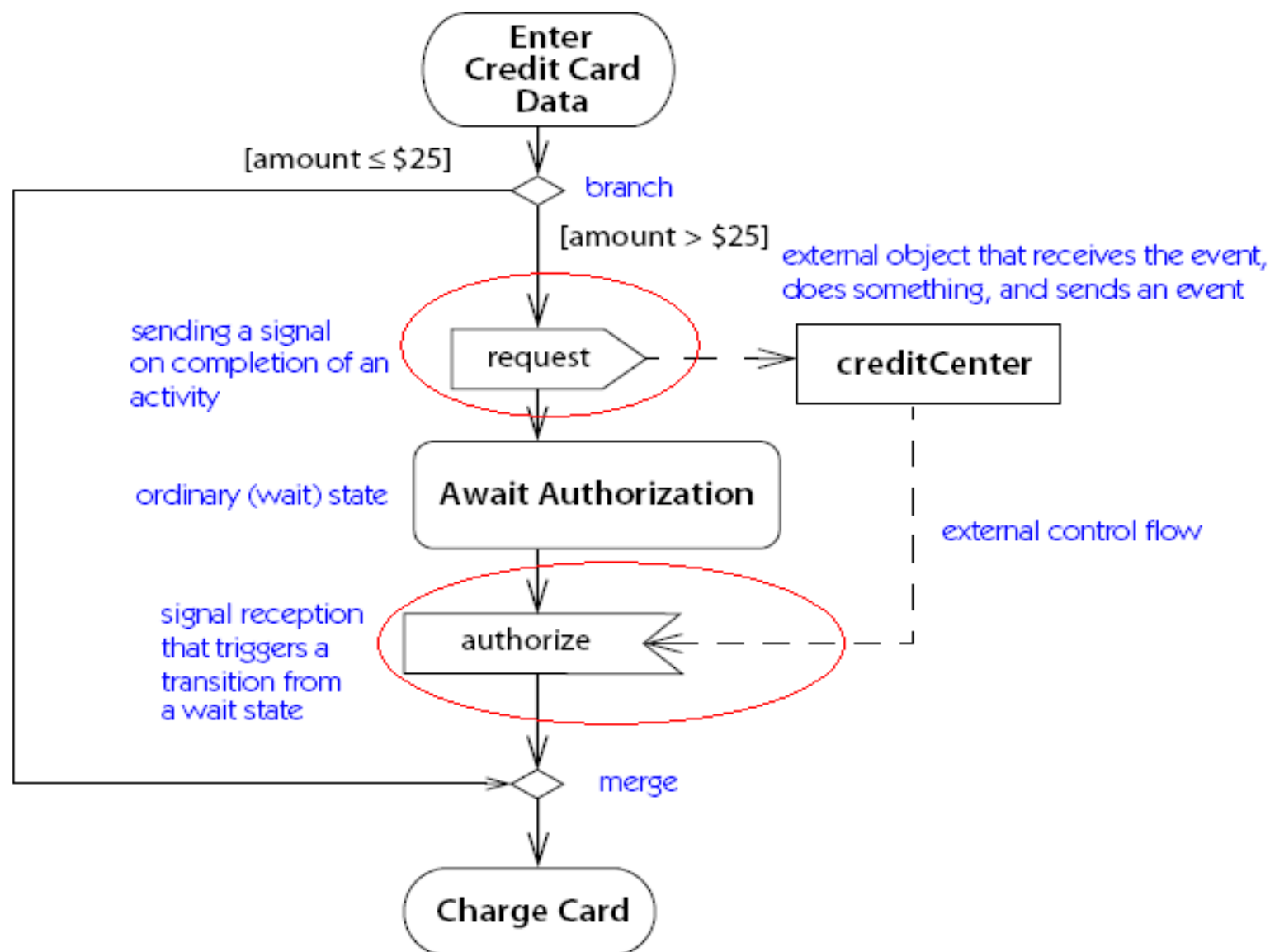
● 对象

在活动图中可以出现对象。对象可以作为活动的输入或输出，对象与活动间的输入/输出关系由虚线箭头来表示。



活动图的要素--信号

- 在活动图中可以表示信号的发送与接收，分别用发送和接收标志来表示。发送和接收标志也可与对象相连，用于表示消息的发送者和接收者。



活动图的使用

- 活动图最适合描述并行行为
- 活动图最大的缺点是很难清楚地描述动作与对象之间的关系。
- 以下情况可以使用活动图：
 - 分析用例
 - 理解牵涉多个用例的工作流
 - 处理多线程应用
- 下列情况一般不要使用活动图：
 - 显示对象间合作
 - 显示对象在其生命周期内的运转情况

总结

- 序列图、协作图和活动图都是用来描述对象交互的。
- 所以在具体描述一个交互时，就需要为此作出选择。具体如何选择取决于你需要着重描述交互的哪个方面？
- 时间？空间？还是活动？
- 不要对系统中的每个类都画状态图