北京交通大学
BEIJING JIAOTONG UNIVERSITY

软件系统分析与设计
System Analysis & Design

M210007B

Monday, May 6, 2024

/-1 软件系统动态建模

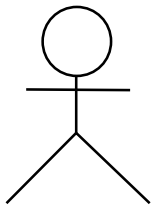
第6章 用例图

6-1 用例图

4-2 参与者

参与者

- 参与者代表与系统交互的人、硬件设备、或另一个系统。
- 参与者并不是软件系统的组成部分，参与者只存在于系统的外部。
- 参与者的UML符号表示是如图所示的“小人”，并可在符号下标出参与者名。

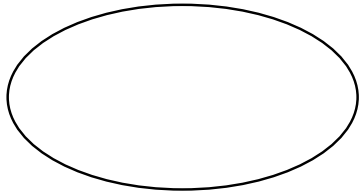


Actor

4-3 用例

用例

- 用例规定了系统或部分系统的行为，它描述了系统所执行的动作序列集，并为执行者产生一个可供观察的结果。
- 用例的UML符号是椭圆，并可在椭圆下标出用例名。

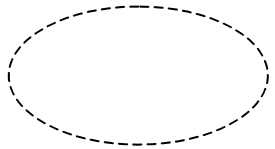


Use Case

4-4 协作

协作

- 协作命名了彼此合作完成某个行为的类、接口和其他元素的群体。
- 协作可以用来定义用例和操作的实现，为系统体系结构上的重要机制建模。
- 协作的UML符号是虚线椭圆，每个协作都有一个名字以与其他协作相区分。



Collaboration

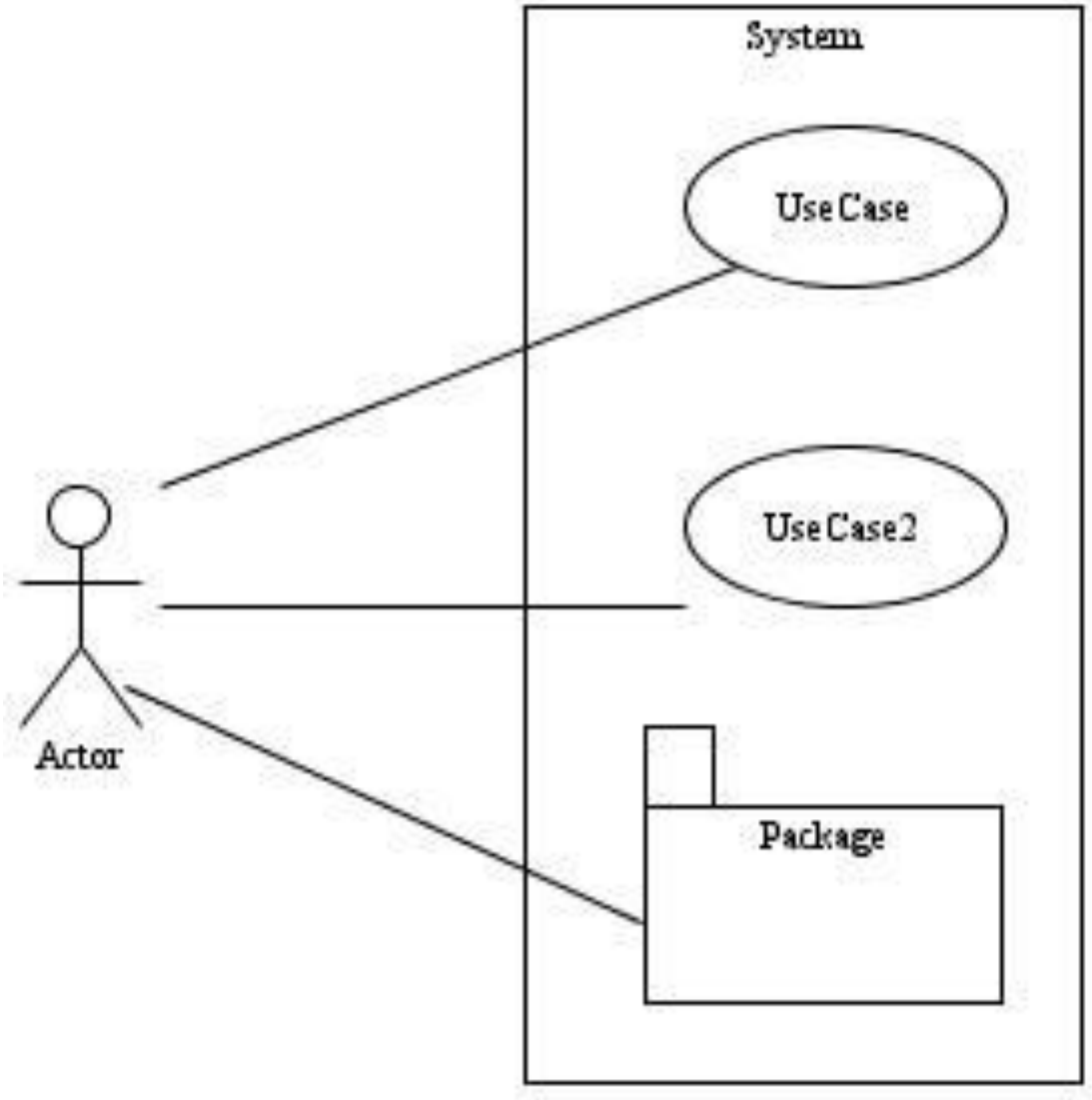
用例图

- 用例图（Use Case Diagrams）是UML中用来对系统的动态方面进行建模的7种图之一（另外6种图是活动图、状态机图、顺序图、通信图、定时图和交互概览图）。
- 用例图描述了用例、参与者以及它们之间的关系。

用例图

- 三种主要建模元素：
  - 用例（Use Case）。
  - 参与者（Actor）。
  - 依赖、类属和关联关系。
- 可选元素：
  - 注释和约束。
  - 包。
  - 系统边界框。

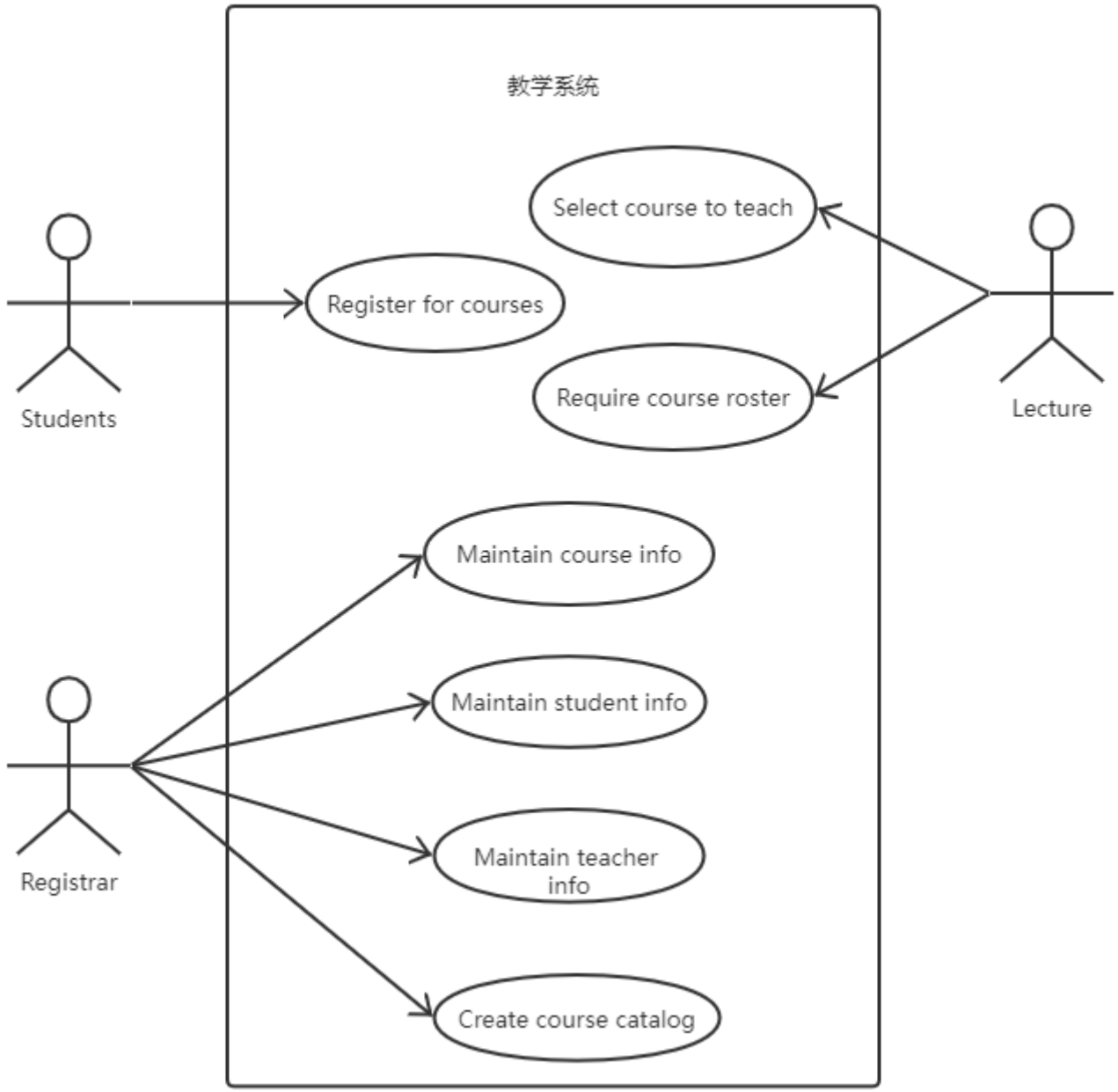
用例图



6-2 参与者

参与者

- 参与者代表与系统接口的事物或人，它是具有某一种特定功能的角色，因此参与者是虚拟的概念，它可以是人，也可以是外部系统或设备。
- 同一个人可能对应多个参与者，因为一个人可能扮演多个角色。
- 参与者不是系统的一部分，它们处于系统的外部。
- 如何识别出参与者?
  - 参与者代表角色。
  - 参与者不是对职位进行建模。


教学系统
Select course to teach
Register for courses
Require course roster
Students
Lecture
Maintain course info
Maintain student info
Maintain teacher info
Create course catalog
Registrar

6-3 用例

用例

- 用例是对系统行为的动态描述，它可以增进设计人员、开发人员与用户的沟通，理解正确的需求；还可以划分系统与外部实体的界限，是系统设计的起点，是类、对象、操作的来源，而通过逻辑视图的设计，可以获得软件的静态结构。
- 如何识别用例？

事件流文档模板

- 事件流文档模板：
  - X. 用例XX（用例名）的事件流
  - X.1 前置条件（Pre-Conditions）
  - X.2 后置条件（Post-Conditions）
  - X.3 扩充点（Extension Points）
  - X.4 事件流
  - X.4.1 基流（Basic Flow）
  - X.4.2 分支流（Subflows） （可选）
  - X.4.3 替代流（Alternative Flows）

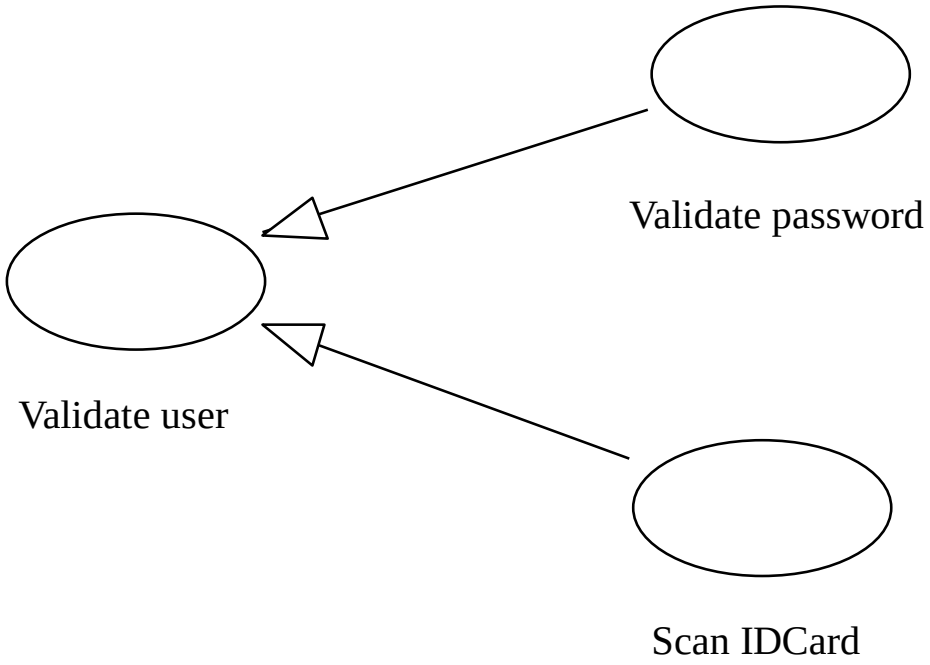
用例与脚本

- 脚本或场景（Scenario）是系统行为的一个特定动作序列。
- 脚本与用例的关系就象实例与类的关系，即脚本是用例的一个实例。

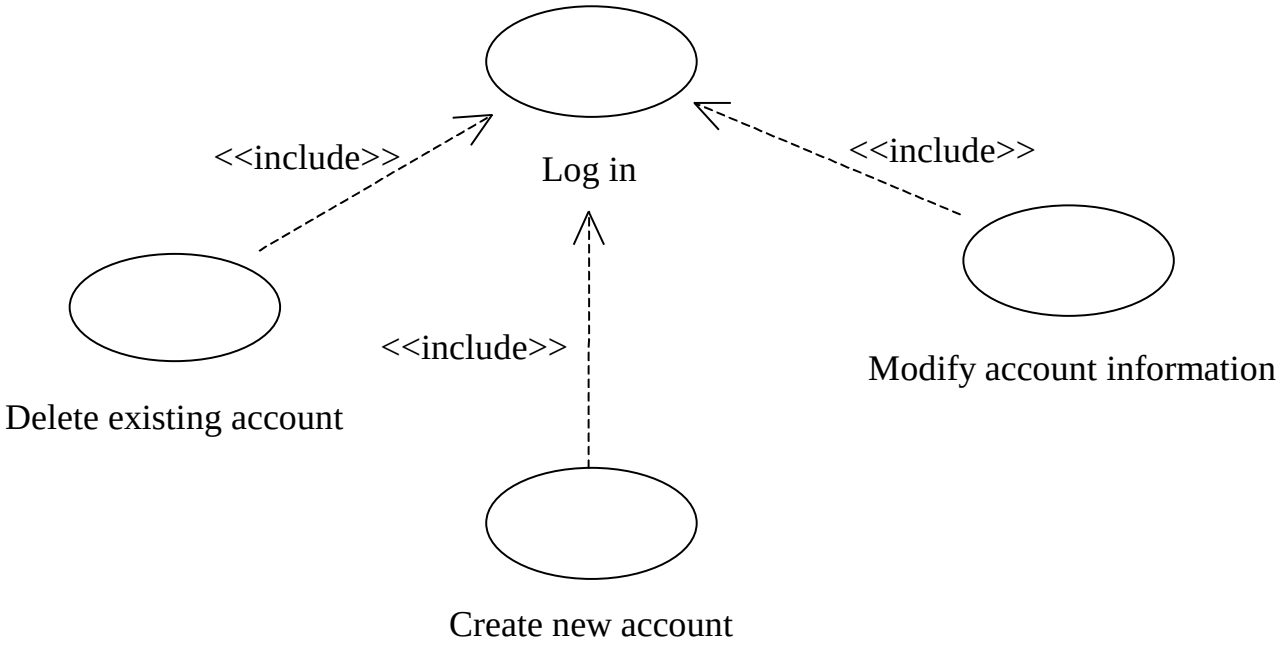
用例间的关系

- 类属关系
  - 如同类间的类属关系。即，子用例继承父用例的行为和含义，子用例可以添加新行为或覆盖父用例的行为。
- Include关系（包含关系）
  - 用例间的包含关系表示在基用例的指定位置，基用例显式地包含另一个用例的行为。
  - 被包含的用例是不能独立存在的，只是包含它的更大用例的一部分。
- Extend关系（扩充关系）
  - 扩充关系用来说明可选的、只在特定条件下运行的行为。
  - 扩充关系用衍型为<<extend>>的依赖关系表示，并在基用例中列出基用例的扩充点，这些扩充点是出现在基用例的流中的标记。

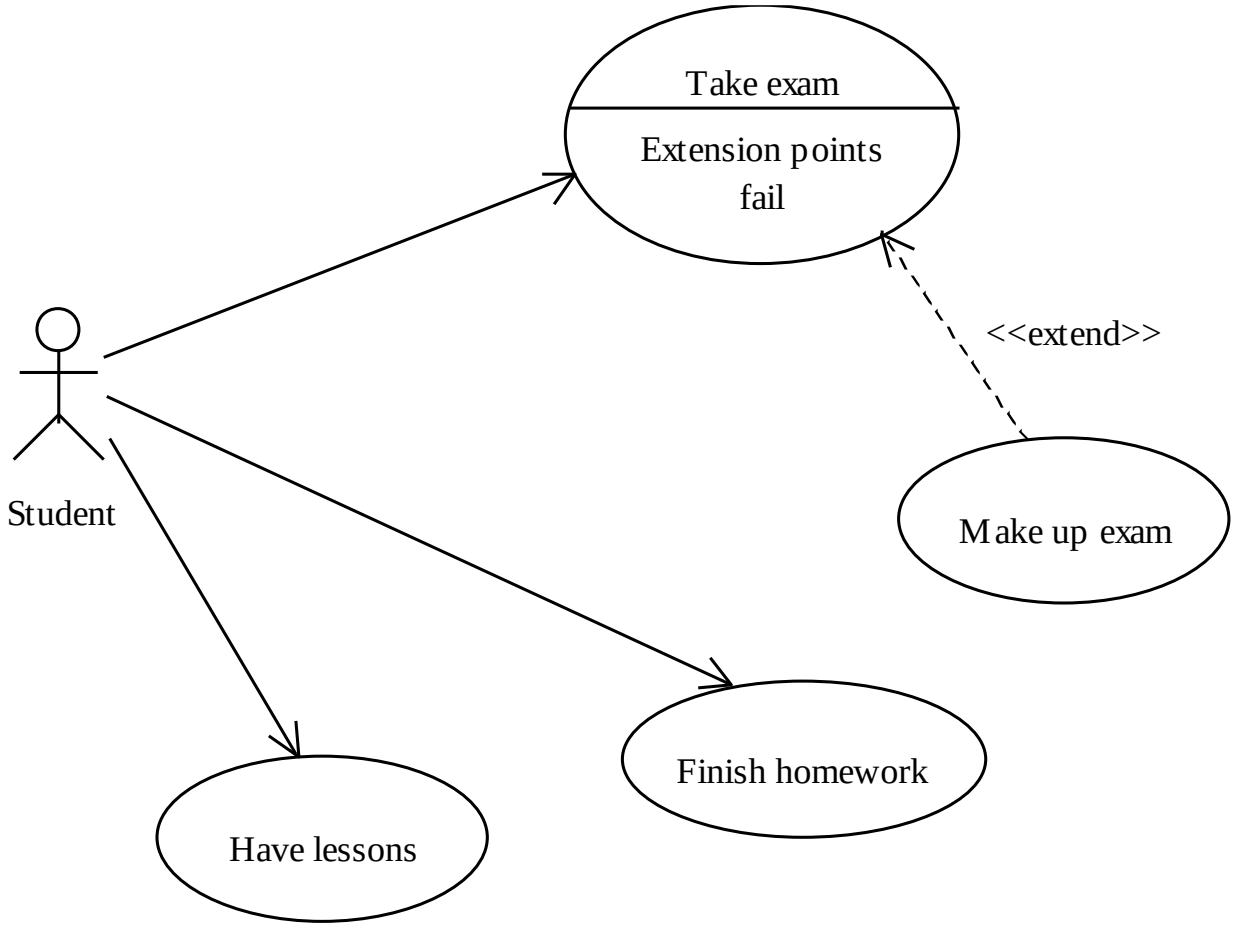
类属关系



Include关系


<<include>>
Log in
<<include>>
<<include>>
Delete existing account
Modify account information
Create new account

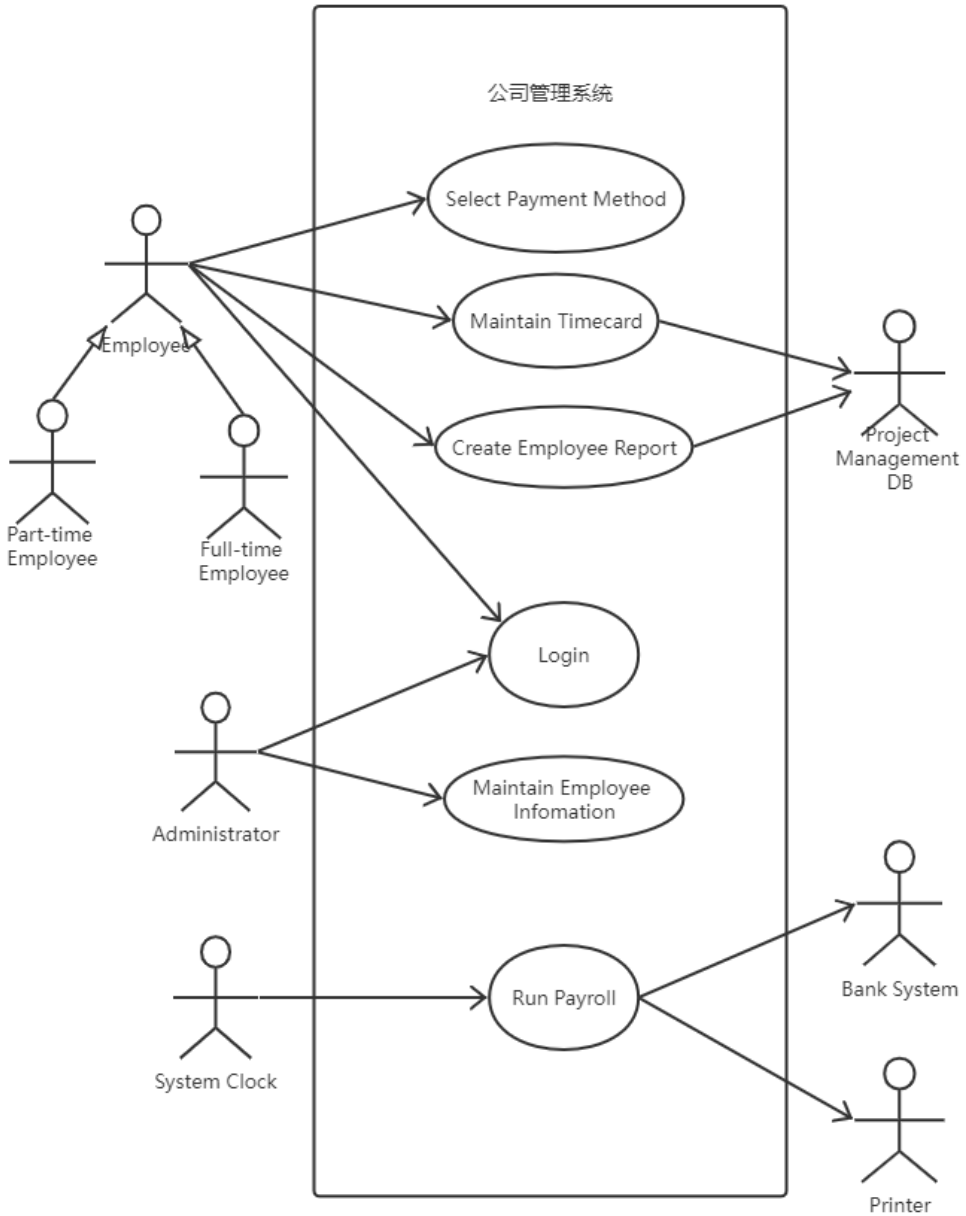
Extend关系


Take exam
Extension points
fail
<<extend>>
Student
Make up exam
Finish homework
Have lessons

6-4 用例图的应用

用例图的应用

- 用例图的应用
  - 为系统的上下文建模。
  - 为系统的需求建模。


公司管理系统
Select Payment Method
Maintain Timecard
Create Employee Report
Login
Maintain Employee Infomation
Run Payroll
Employee
Part-time Employee
Full-time Employee
Administrator
System Clock
Project Management DB
Bank System
Printer

第8章 交互作用图

交互作用图

- 交互作用图描述了对象间的交互作用，由对象、对象间的关系组成，并包含在对象间传递的消息。
  - **顺序图**
    - 顺序图强调消息的时间顺序。
  - **通信图**
    - 通信图强调发送和接收消息的对象的组织结构。
- 交互作用图的主要组成元素如下：
  - 对象。
  - 连接。
  - 消息。
  - 像其他的图一样，交互作用图中也可以有注释和约束。

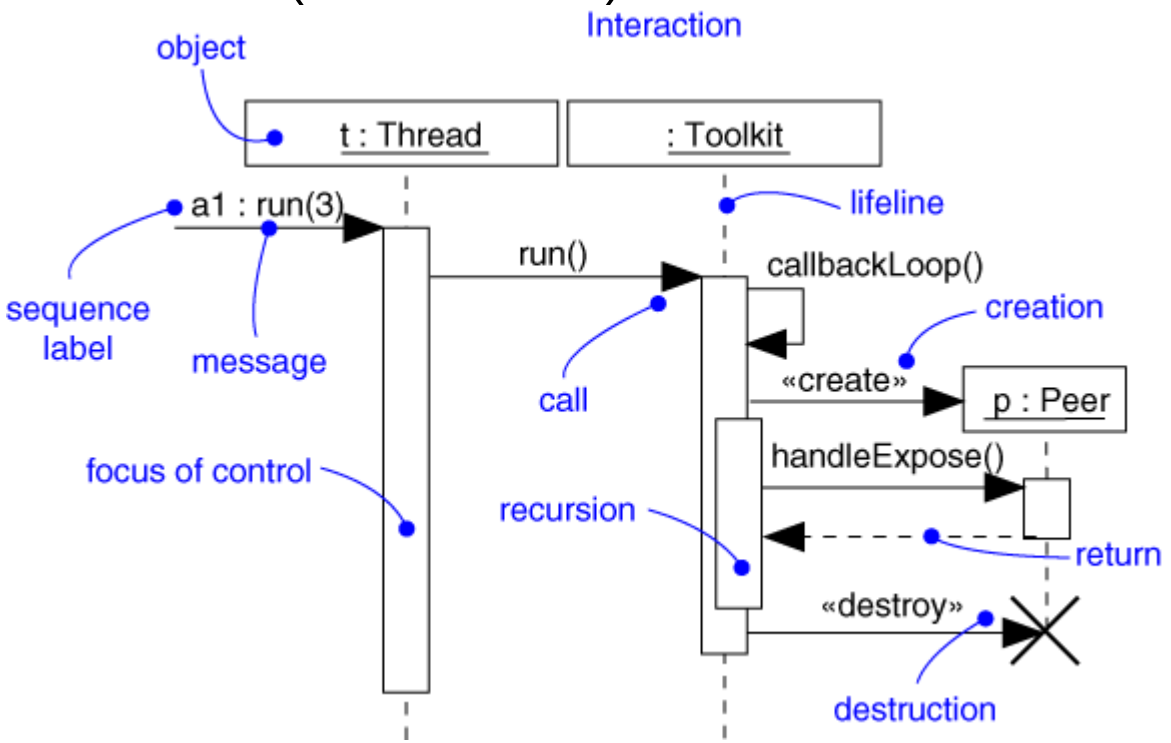
8-1 顺序图
Sequence Diagram

顺序图

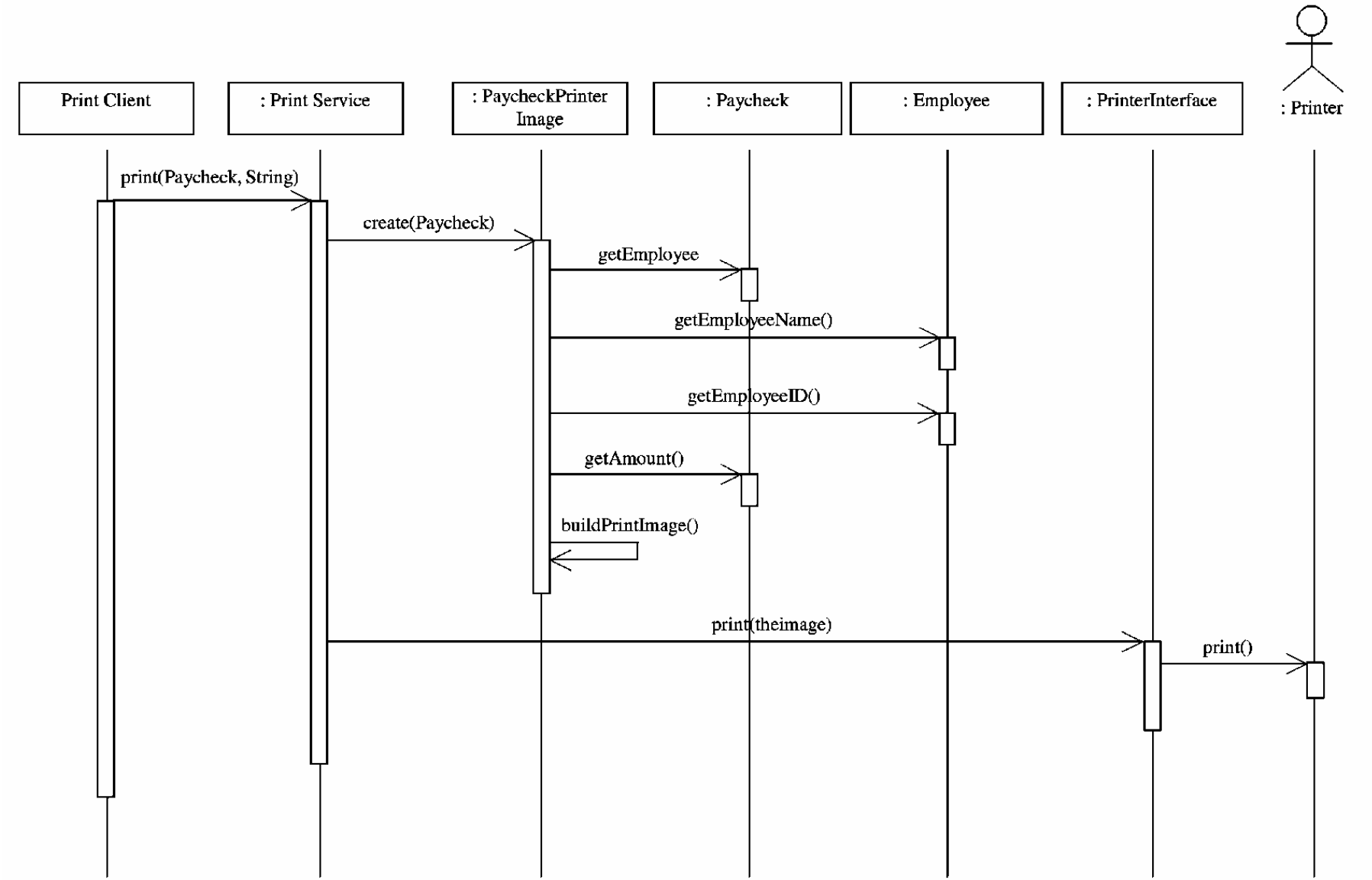
- 顺序图（Sequence Diagram）
  - 存在两个轴，水平轴表示不同的对象；垂直轴表示时间。
  - 顺序图中的对象用一个带有垂直虚线的矩形框表示，并标有对象名和类名。
  - 垂直虚线是对象的生命线，用于表示在某段时间内对象是存在的。
  - 对象间的通信通过在对象的生命线间画消息来表示。
- 与通信图相区别，顺序图具有两个特点：
  - 有对象生命线
  - 有控制中心

顺序图

- 捕捉系统的动态方面 (面向时间)

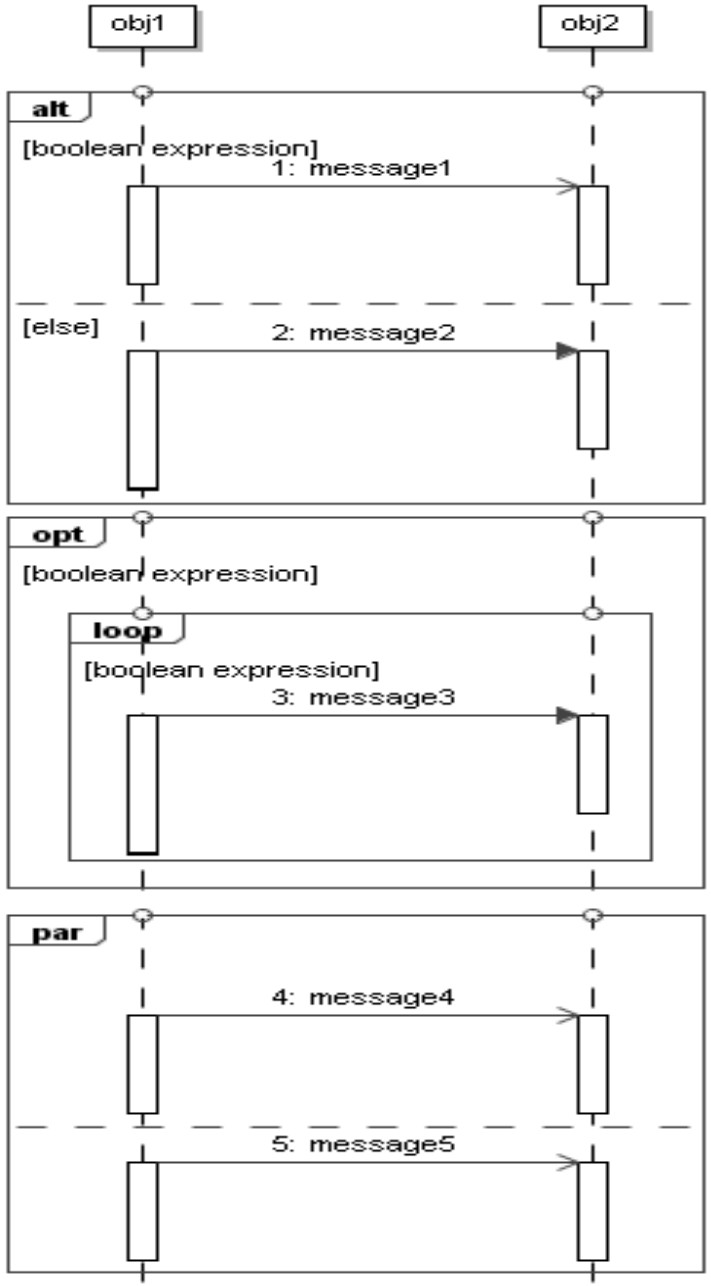


顺序图


Print Client
: Print Service
: PaycheckPrinter Image
: Paycheck
: Employee
: PrinterInterface
: Printer
print(Paycheck, String)
create(Paycheck)
getEmployee
getEmployeeName()
getEmployeeID()
getAmount()
buildPrintImage()
print(theimage)
print()

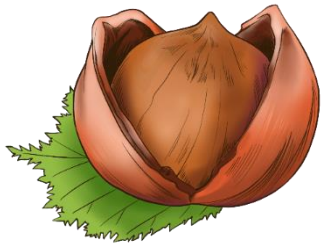
顺序图

- 顺序图的结构控制
  - **可选执行**（Optional Execution）
    - 标记为“opt”。
    - 护卫条件（Boolean Expression）为真时，可选执行部分才被执行。
  - **条件执行**（Conditional Execution）
    - 标记为“alt”。
    - 条件执行部分由水平虚线分割为多个子区域，每个子区域都有一个护卫条件，代表一个条件分支。
  - **并行执行**（Parallel Execution）
    - 标记为“par”。
    - 并行执行部分也由水平虚线分割为多个子区域，每个子区域代表一个并行分支。
  - **循环执行**（Loop/Iterative Execution）
    - 标记为“loop”。
    - 在每次循环之前，若护卫条件为真，循环执行部分就被重复执行；若为假时，循环执行部分被跳过，不再执行。



举个例子

Examples



课堂练习

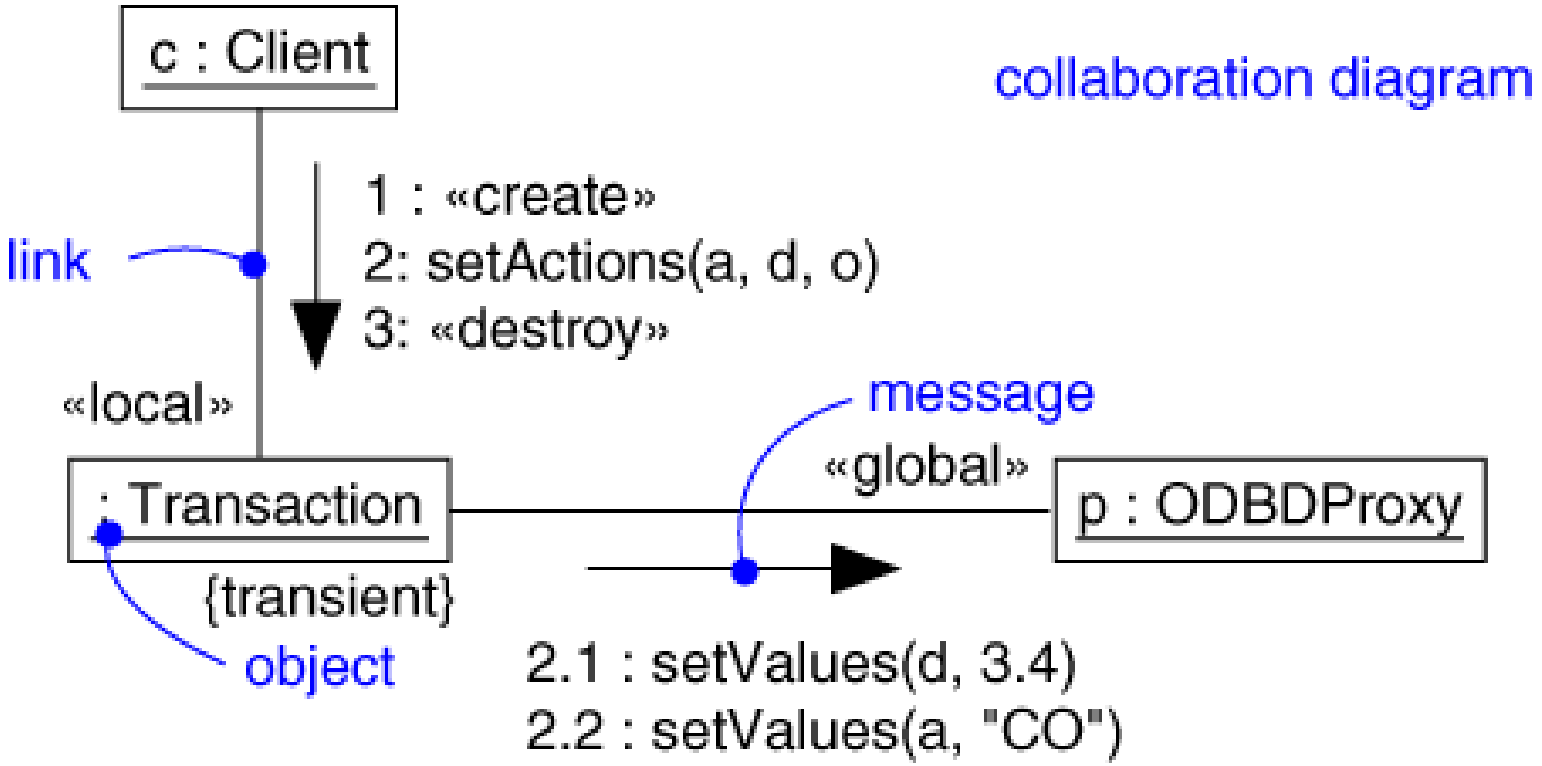
8-2 通信图
Communication Diagram

通信图

- 通信图（Communication Diagram）强调了参与交互作用的对象的组织。
- 通信图描述了两个方面：
  - 第一个方面是对交互作用的对象的静态结构的描述，包括相关的对象的关系、属性和操作
  - 第二个方面是为完成工作在对象间交换的消息的时间顺序的描述。
- 与顺序图区分，通信图有两个特点：
  - 有路径
  - 有序列号

通信图

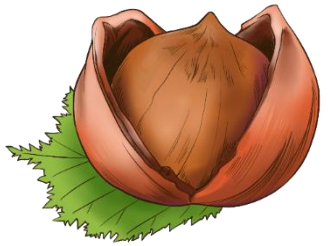
- 捕捉系统的动态方面 (面向消息)

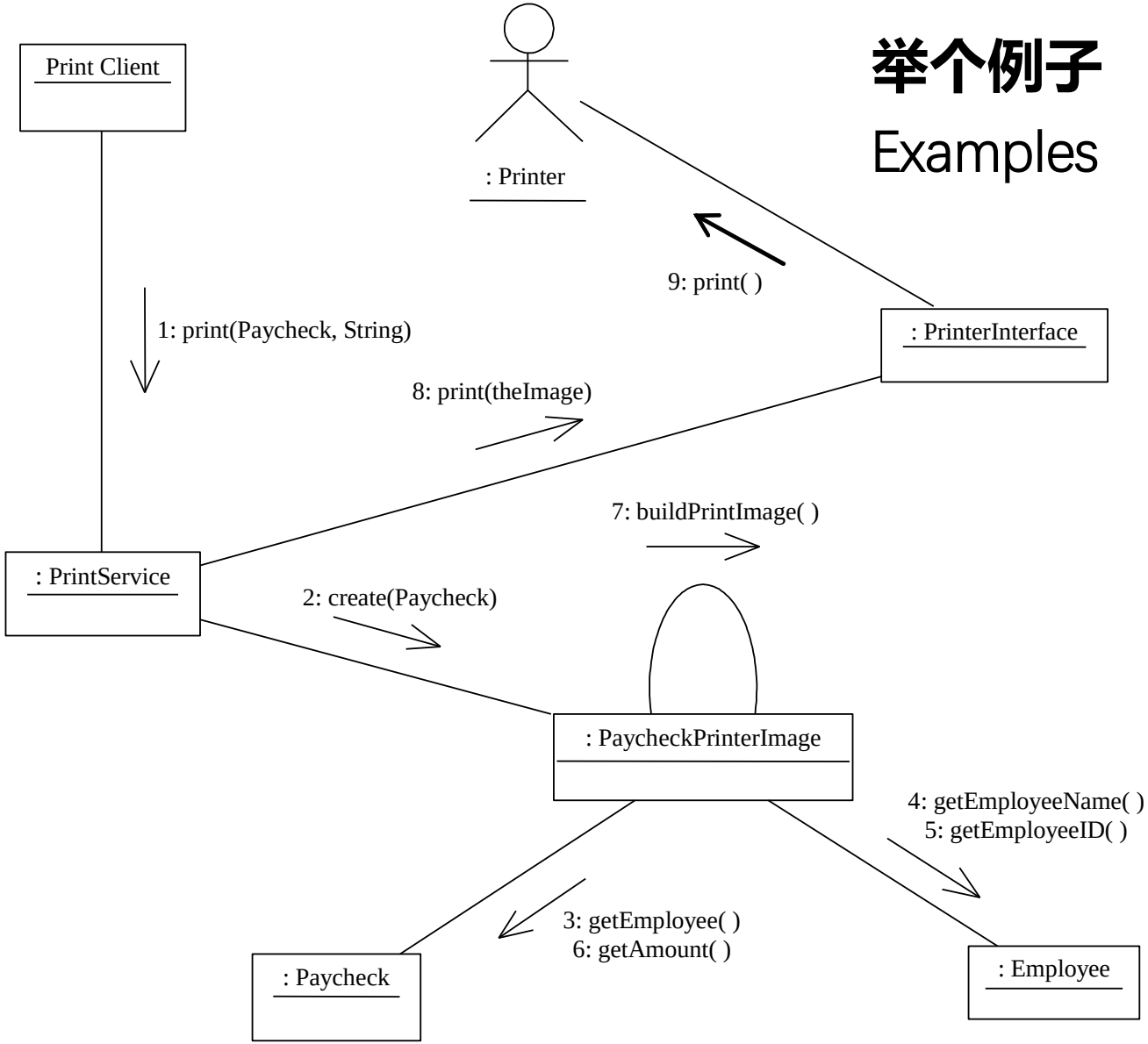


通信图

- 顺序图和通信图在语义上是等价的，都用来描述对象间的交互作用，但侧重点却不一样。
  - 顺序图着重体现交互的时间顺序
  - 通信图则着重体现交互作用的对象间的静态连接关系。

举个例子
Examples





8-3 交互作用图
Interaction Diagram

交互作用图

- 交互作用图的应用
  - ➢按时间顺序为控制流建模
    - ◆确定交互作用的上下文。
    - ◆确定哪些对象参与了交互作用，并将这些对象从左到右放在顺序图中，其中重要的对象放在图左边。
    - ◆ 确定每个对象的生命线。
    - ◆ 从发起交互作用的消息开始，将消息按发生的时间顺序从上到下逐一地标出。
    - ◆ 如果需要规定时间或空间约束，可以为消息附加适当的时间或空间约束。
    - ◆ 如果想更正式地描述一个控制流，可以为流中的每个消息添加前置条件和后置条件。

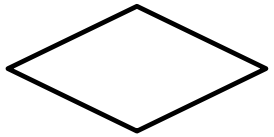
交互作用图

- 按组织结构为控制流建模
  - 确定交互作用的上下文。
  - 确定哪些对象参与了交互作用，并将这些对象放在通信图中，其中重要的对象放在图的中间。
  - 确定每个对象的初始特性。
  - 确定对象间的连接。

第9章 活动图

判定

- 判定（Decision）代表了活动图或状态机图上的一个特殊位置，在这个位置上工作流将根据护卫条件进行分支。
- 判定节点的UML符号是一个空心菱形。



同步条

- 同步条（Synchronization Bars）用来定义活动图中的分叉（Fork）和联结（Join）。
- 同步条的UML符号表示用粗的水平或竖直条表示。

活动

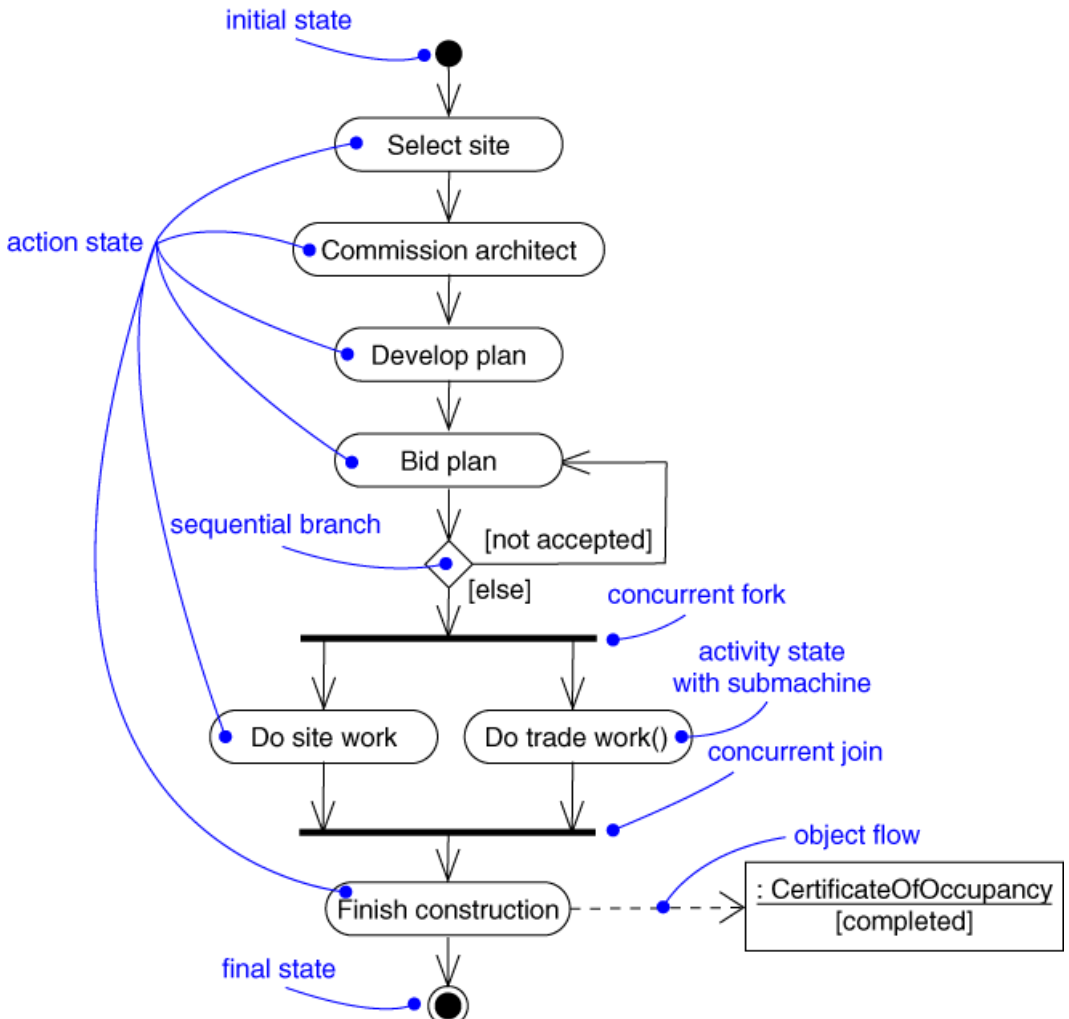
- 活动是在状态机中进行的一个非原子的执行，它由一系列的动作组成。
- 活动的UML符号表示

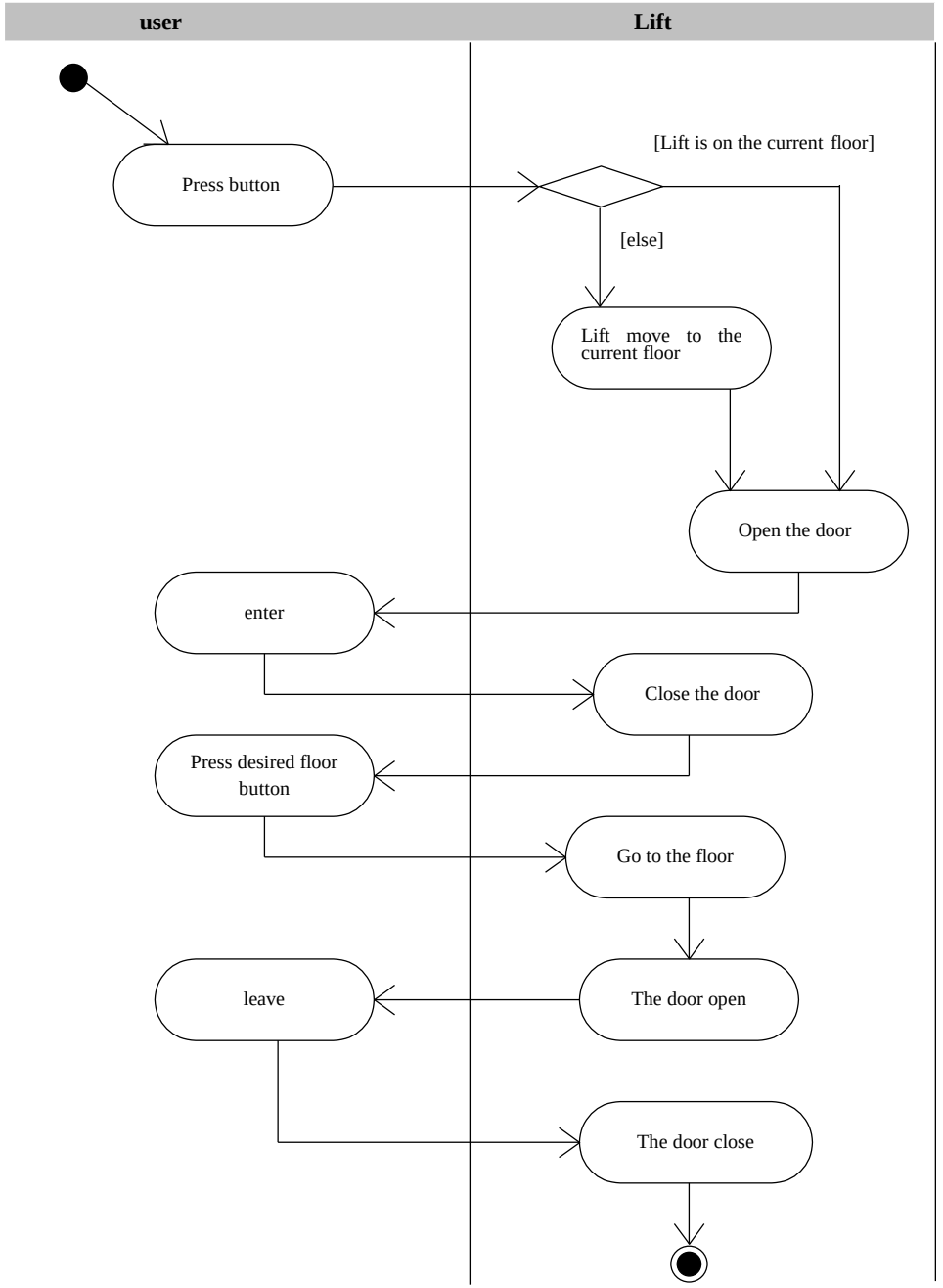
活动图

- 活动图主要是一个流图，描述了从活动到活动的流。
- 活动是在状态机中进行的一个非原子的执行，它由一系列的动作组成。
- 动作是由可执行的不可分的计算组成，这些计算可以引起系统的状态发生变化或者返回一个值。
- 交互作用图强调从对象到对象的控制流。
- 活动图则强调从活动到活动的控制流。
- 活动图主要包含下列元素：
  - 活动状态
  - 动作状态
  - 跃迁
  - 对象
  - 活动图中也可以有注释和约束

活动图

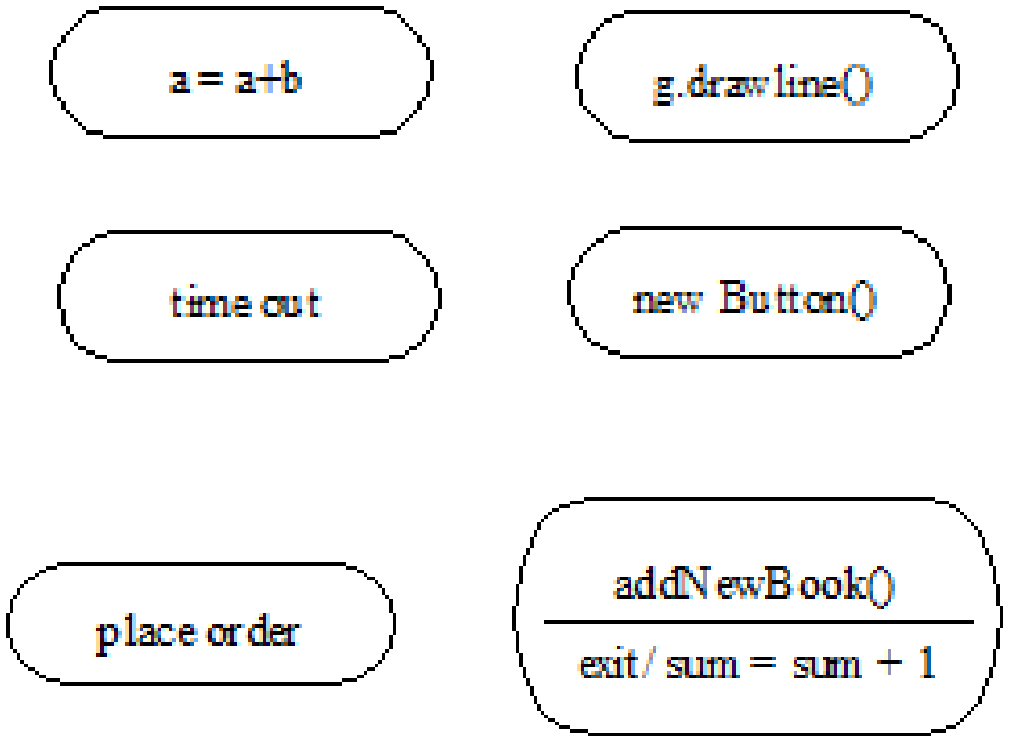
- 捕捉系统的动态方面 (面向活动)




user
Lift
Press button
[Lift is on the current floor]
[else]
Lift move to the current floor
Open the door
enter
Close the door
Press desired floor button
Go to the floor
leave
The door open
The door close

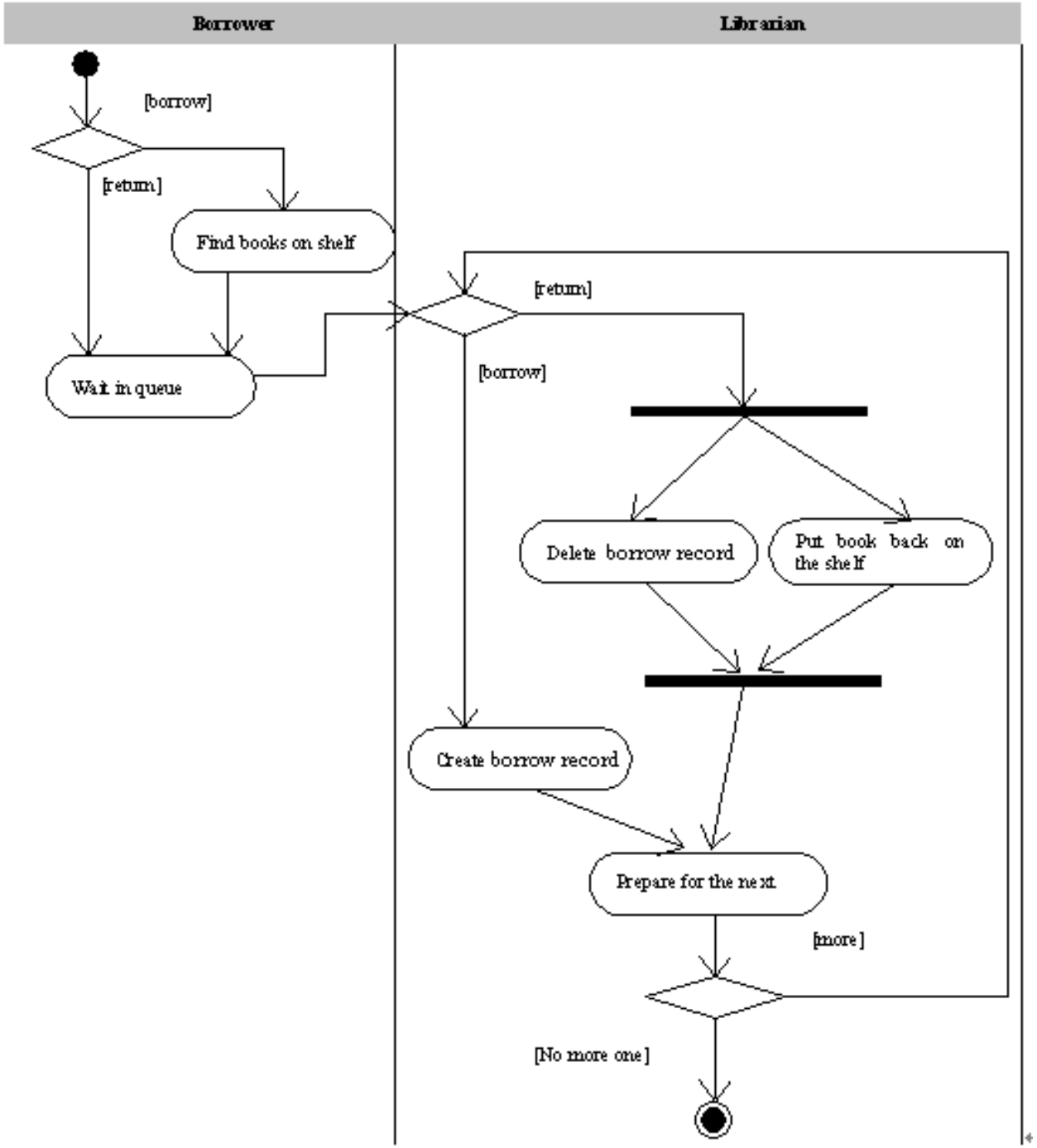
活动图

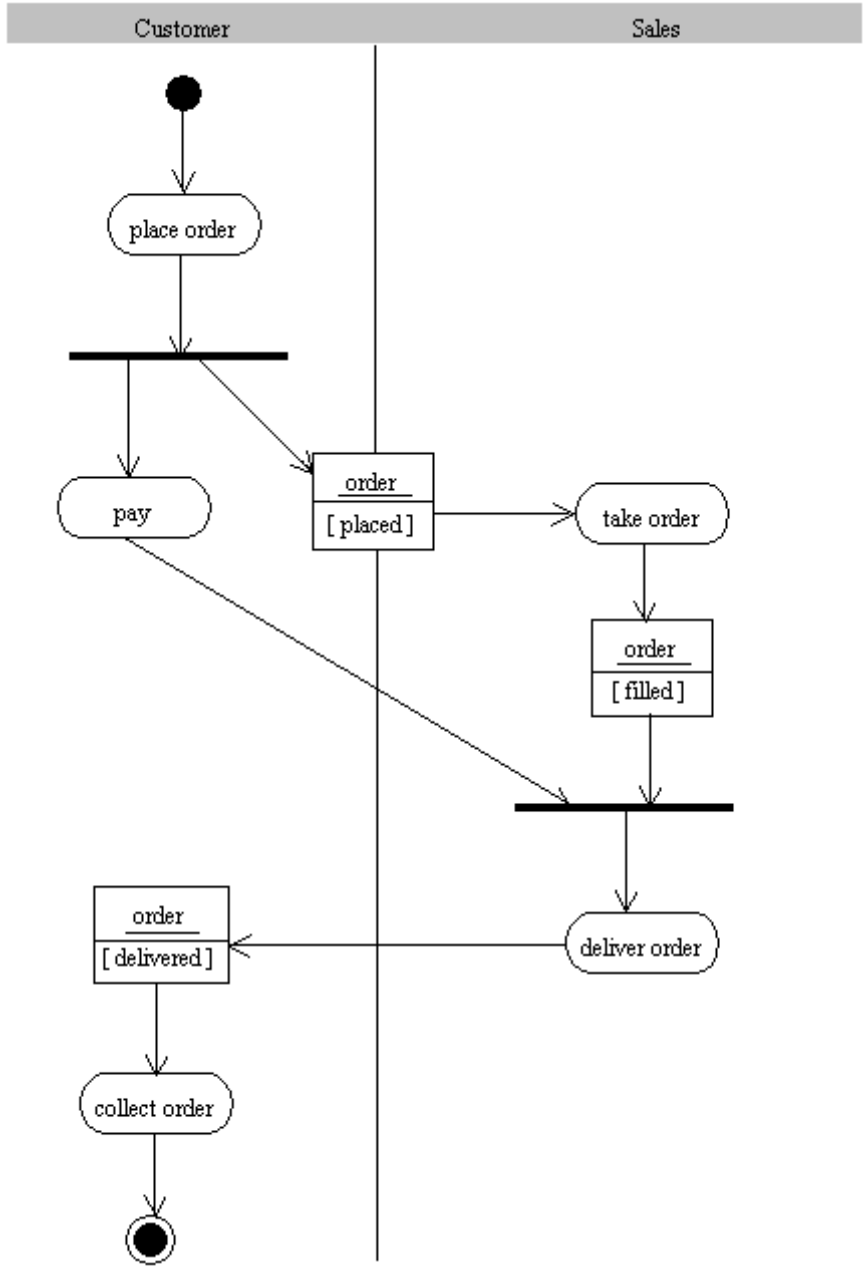
- 动作状态
- 活动状态



活动图

- 分叉和联结
  - 分叉表示将单一的控制流分成两个或多个并发的控制流。
  - 联结代表了两个或多个并发控制流的同步，联结有多个输入跃迁和一个输出跃迁。
- 泳道
  - 泳道用矩形框来表示，将对象名放在矩形框的顶部，将属于某个对象的活动放在该对象的泳道内，泳道中的活动由相应对象负责。


Borrower
Librarian
[borrow]
[return]
Find books on shelf
Wait in queue
[return]
[borrow]
Delete borrow record
Put book back on the shelf
Create borrow record
Prepare for the next
[more]
[No more one]

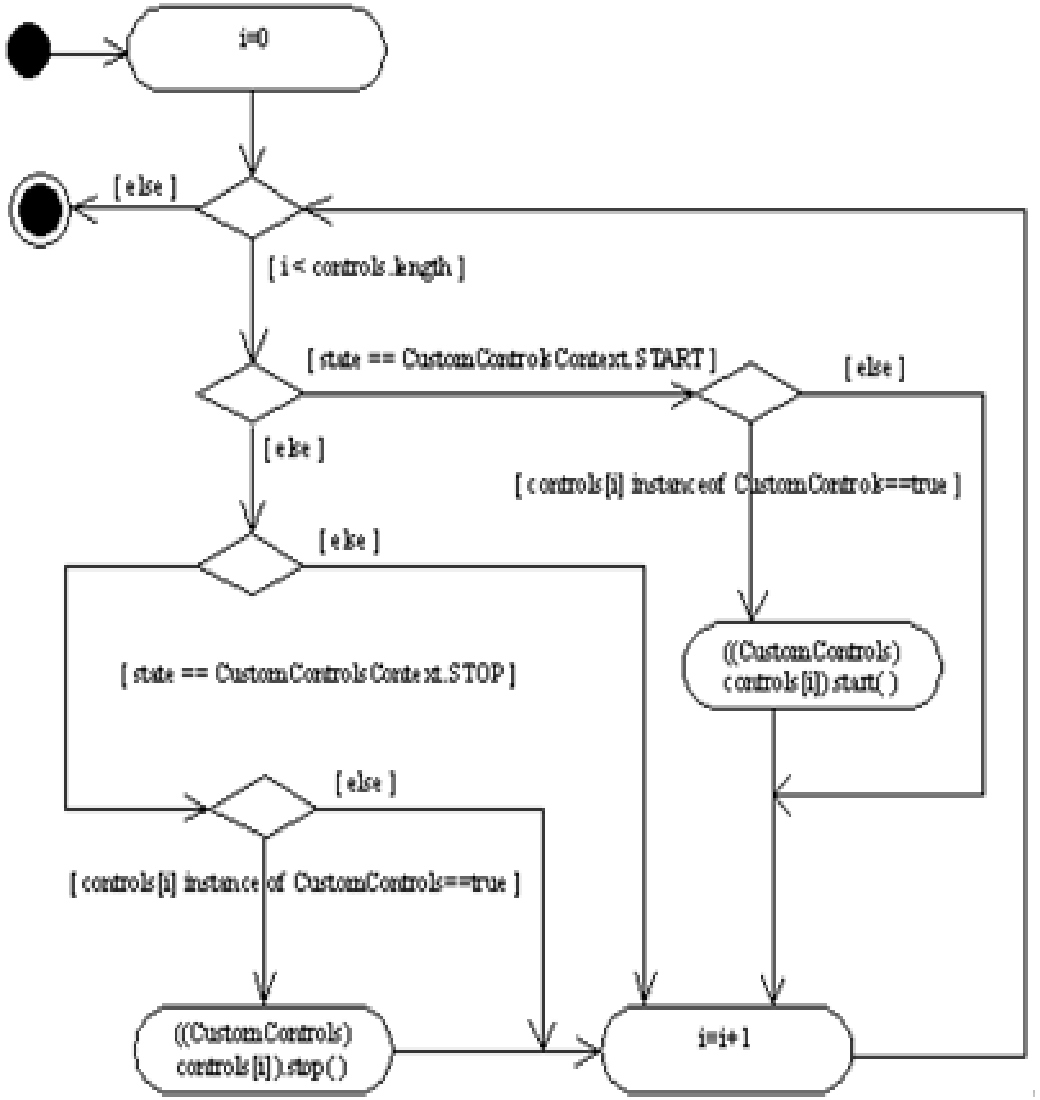

Customer
Sales
place order
pay
order
[placed]
take order
order
[filled]
order
[delivered]
deliver order
collect order

对象流

活动图的应用

- 为工作流建模
- 为操作建模

```
public void handleThread(int state) {
        for (int i = 0; i < controls.length; i++) {
            if (state == CustomControlsContext.START) {
                if (controls[i] instanceof CustomControls) {
                        ((CustomControls) controls[i]).start();
                }
            } else if (state == CustomControlsContext.STOP) {
                if (controls[i] instanceof CustomControls) {
                        ((CustomControls) controls[i]).stop();
                }
            }
        }
    }
```




```
public void handleThread(int state) {
        for (int i = 0; i < controls.length; i++) {
            if (state ==
CustomControlsContext.START) {
                if (controls[i] instanceof
CustomControls) {
                    ((CustomControls)
controls[i]).start();
                }
            } else if (state ==
CustomControlsContext.STOP) {
                if (controls[i] instanceof
CustomControls) {
                    ((CustomControls)
controls[i]).stop();
                }
            }
        }
    }
```

第10章 状态图

10-1 状态图
Statechart Diagram

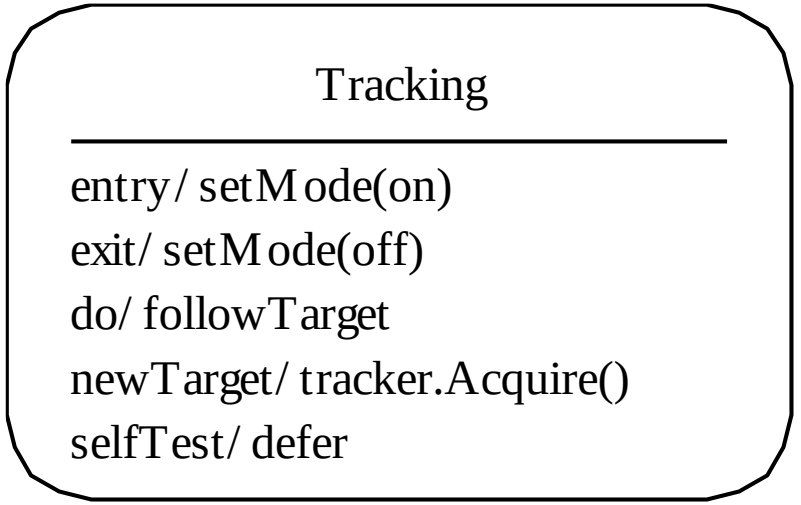
4-11 状态

状态

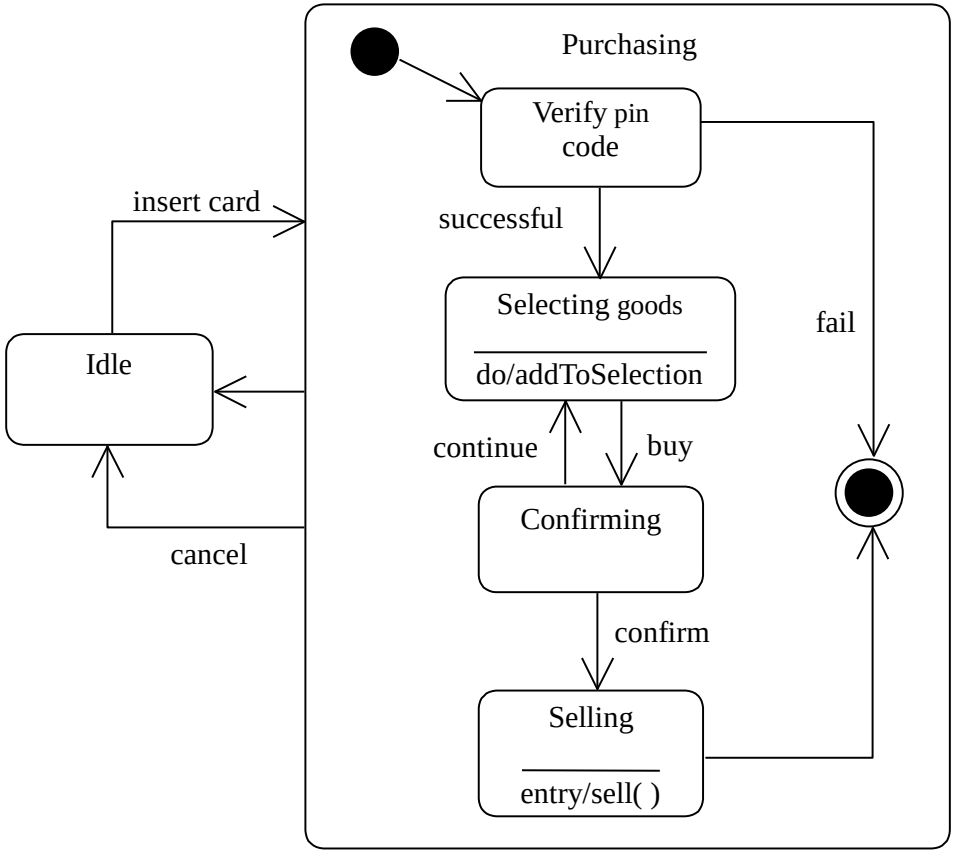
- 状态机（State Machine）描述了对象在生命周期中响应事件所经历的状态的序列以及对象对这些事件的响应。状态机由状态、跃迁、事件、活动、动作等组成。
- 状态描述对象在生命周期中的一种条件或状况，在这种状况下，对象满足某个条件，或执行某个动作、或等待某个事件。
- 一个状态在一个有限的时间段内存在。

状态

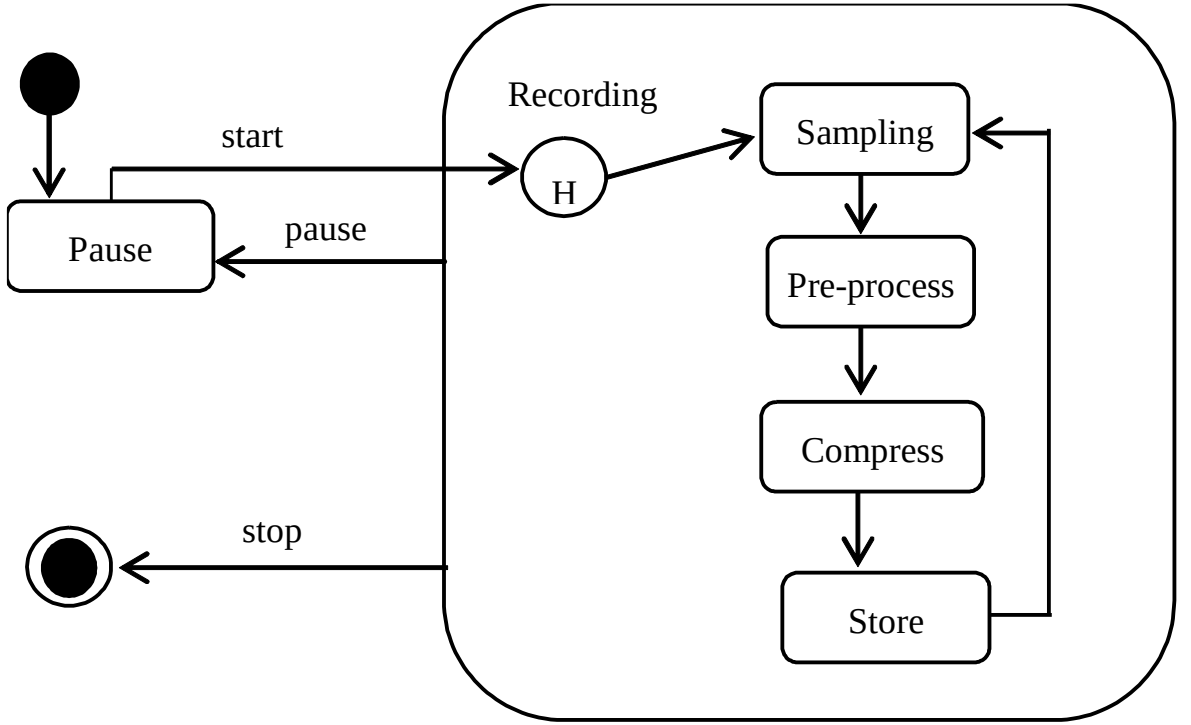
- 状态由以下几部分组成:
  - 名字
  - 入口 / 出口动作
  - 内部跃迁
  - 子状态



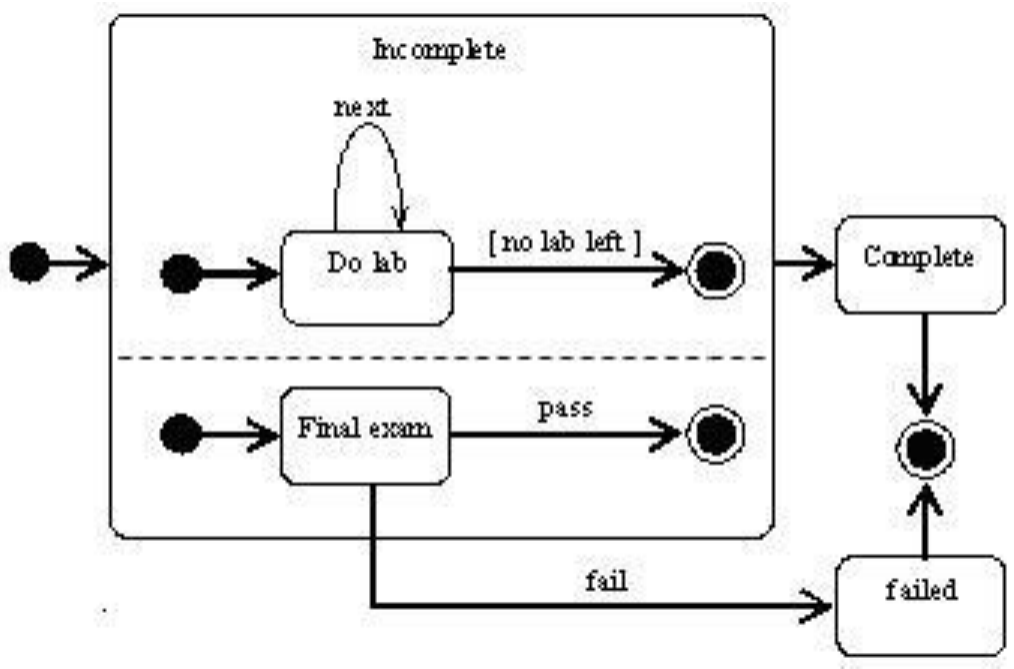
状态：不相交子状态


Purchasing
Verify pin code
successful
Selecting goods
do/addToSelection
fail
insert card
Idle
continue
buy
Confirming
cancel
confirm
Selling
entry/sell( )

状态：历史状态


Recording
start
H
Sampling
Pause
pause
Pre-process
Compress
stop
Store

状态：并发子状态


Incomplete
next
Do lab
[no lab left]
Final exam
pass
fail
Complete
failed

4-12 跃迁

跃迁

- 跃迁是两个状态间的一种关系，它表示对象在第一个状态将执行某些动作，当规定的事件发生或满足规定的条件时，对象进入第二个状态。
- 跃迁表示了从活动（或动作）到活动（或动作）的控制流的传递。
- 跃迁由以下部分组成：
  - 源状态与目标状态
  - 触发事件
  - 护卫条件
  - 动作

状态图

- 状态图（State Diagrams）给出了一个状态机，强调了从状态到状态的控制流。
- 状态机（State Machine）定义了对象在生命周期中响应事件所经历的状态的序列以及对象对这些事件的响应。状态机由状态、跃迁、事件、活动、动作等组成。
- 状态（State）代表对象在生命周期中的一种条件或状况，在这种状况下，对象满足某个条件，或执行某个动作、或等待某个事件。一个状态在一个有限的时间段内存在。

状态图

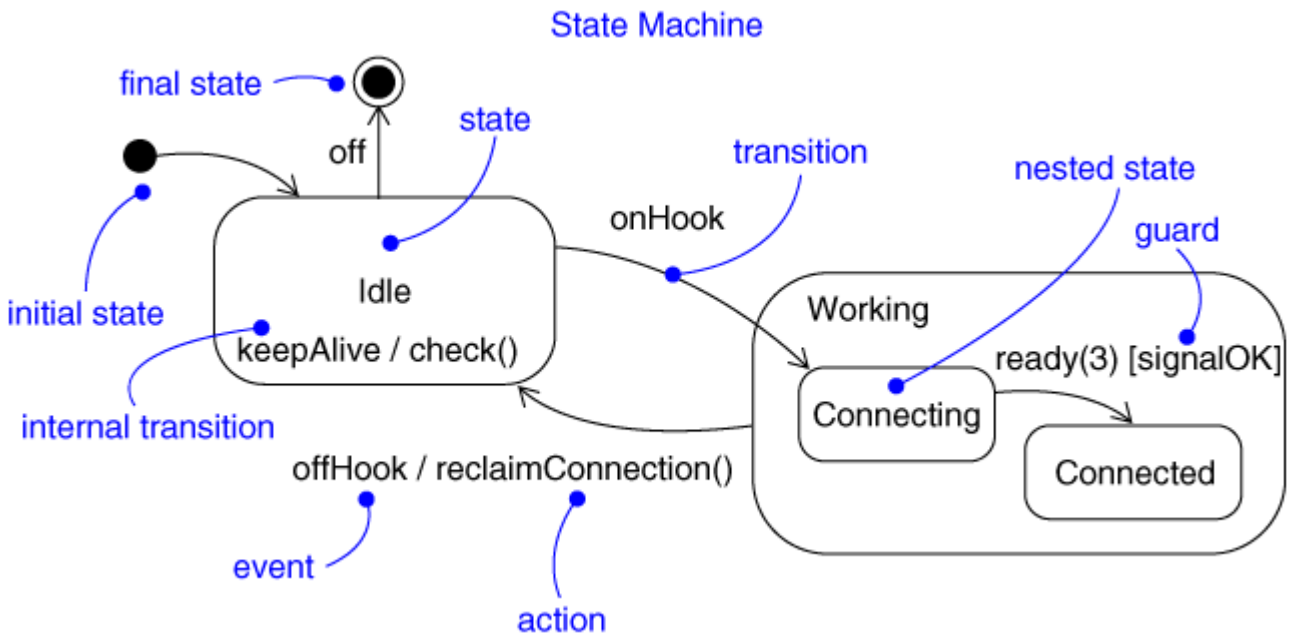
- UML 状态图可以捕获对象、子系统和系统的生命周期，可以告知一个对象可以拥有的状态，并且事件(如消息的接收，时间的流逝、错误、条件为真等)会怎样随着时间的推移来影响这些状态。
- 一个状态图应该连接到所有具有清晰的可标志状态和复杂行为的类；该图可以确定类的行为以及该行为如何根据当前的状态而变化，也可以展示哪些事件将会改变类的对象的状态。
- 状态图主要是为了模拟响应系统。
- 使用状态图的主要目的：
    - 为了模拟系统的动态环节。
    - 反应系统模型生命周期。
    - 一个对象来描述不同的状态，在其生命周期的时间。
    - 定义一个状态机模型状态的对象。

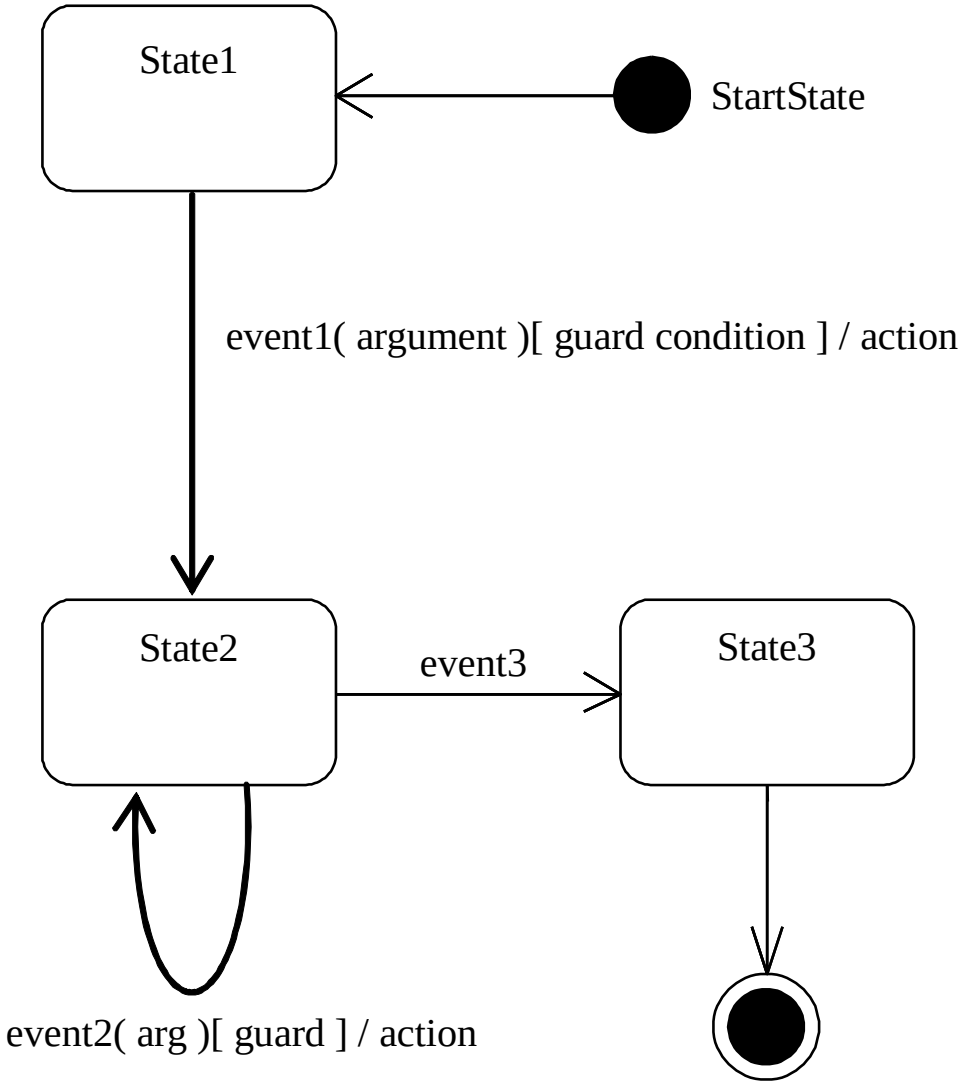
状态图

- 事件（Event）是一个重要事件的规范，该事件在时间和空间域中有一个位置。
- 跃迁（Transition）是两个状态之间的关系，它表示第一个状态的对象将执行某个动作，如果规定的事件发生或规定的条件被满足，则对象进入第二个状态。
- 活动（Activity）是状态机中正在执行的可分解的计算。
- 动作（Action）是可执行的、不可分的计算，该计算造成了模型的状态变化或者值的返回。

状态图

- 捕捉系统的动态行为 (面向事件)




State1
StartState
event1(argument)[guard condition] / action
State2
event3
State3
event2(arg)[guard] / action

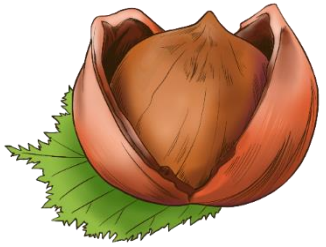
10-2 状态图的应用

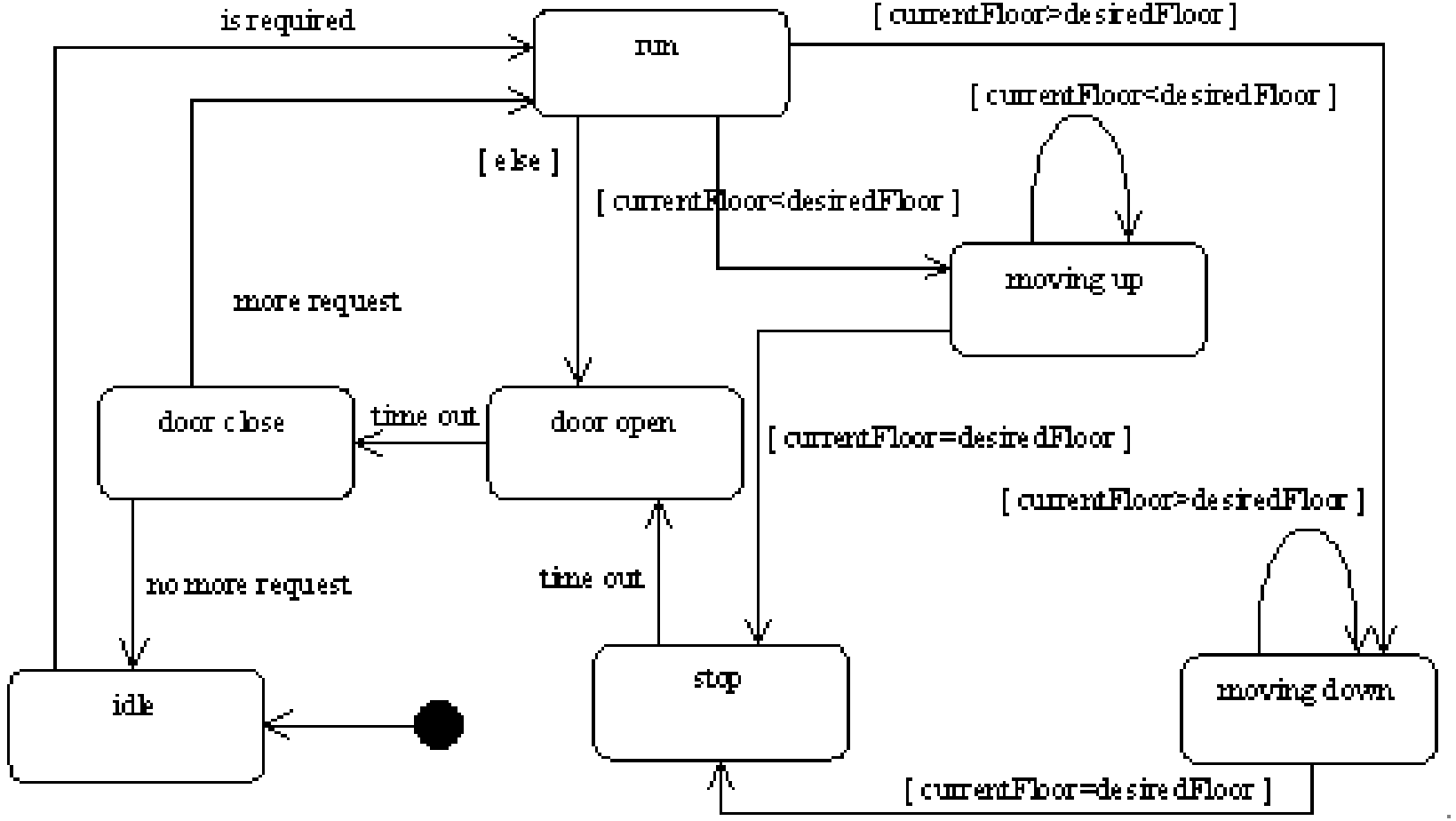
状态图的应用

- 交互作用图描述了多个对象间的交互作用，而状态机图描述单个对象在它的整个生命周期的行为。
- 活动图描述了从活动到活动的控制流，状态机图描述了从事件到事件的控制流。
- 状态机图可以用来描述整个系统、子系统、或类的动态方面，还可以描述用例的一个脚本。

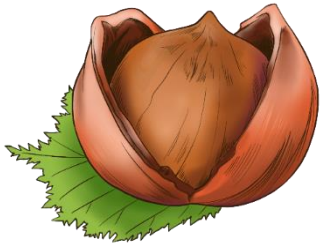
状态图的应用

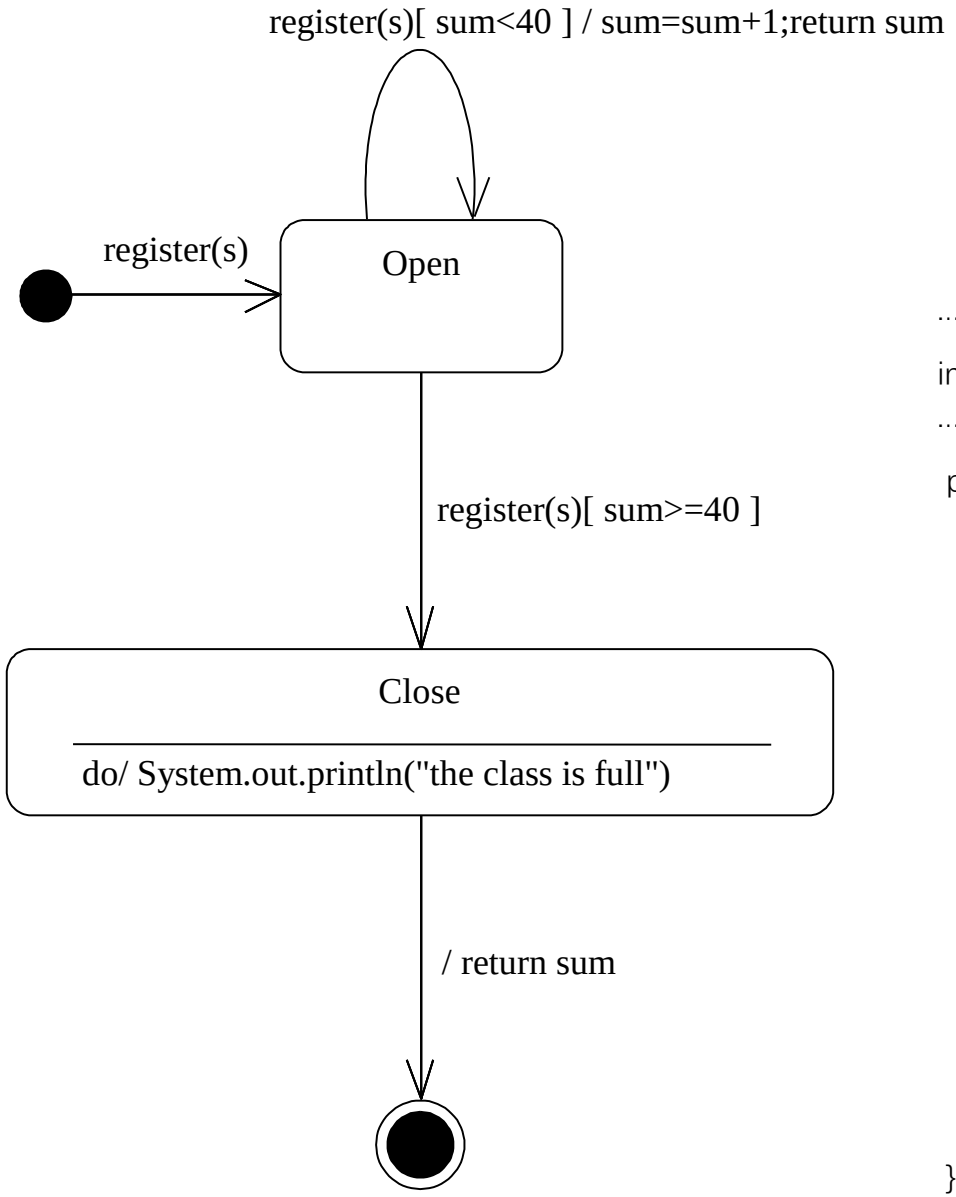
举个例子
Examples





举个例子
Examples






```
...
int sum = 0;
...
 public int register(Student s){
     switch(state){
          case Open:
               if (sum<40){
                                        state = Open;
                                        sum = sum + 1;
                                   }else
                    state = Close;
               break;
                  case Close:
                   System.out.println(“ the class is full”);
      }
      return sum;
}
```