



统一建模语言

UML



UML的由来

- ✓ UML的全名: **unified modeling language**
统一建模语言
- ✓ UML在Rational Software公司的支持下于1994年开始成形。
- ✓ 是Grady Booch, James Rumbaugh, Ivar Jacobson三位从事面向对象方法研究的专家合作研究的成果。
- ✓ UML标准是**OMG**协会在1997年制定的。



为什么要建模？

- 模型帮助我们按照实际情况对系统进行可视化
- 模型可以让我们描述系统的结构和行为，并且利用它和同事沟通
- 模型提供了指导我们创建系统的模板，我们还可以利用它为使用系统的人提供帮助
- 模型对我们所做的决策进行文档化



UML概述

■ 什么是UML？

一种通用可视化建模语言。用来对软件密集型系统进行可视化、详述、构造和文档化。

- ✓ Unified: UML是一种标准语言，广泛运用于全世界
- ✓ Modelling: UML用途在建模
- ✓ Language: 一种建模语言



UML的层次

- UML在设计上分3个层次

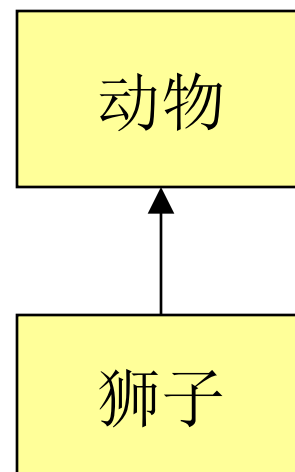
概念层

规格说明层

实现层

UML的层次

- **概念层**：一种图形表示方法，让相关人员可以一眼就看出所要表达的含义

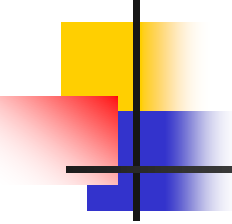




对象和类的概念

- ❖ **对象的引入：**能用于指定一台特定的电视机
- ❖ **类的引入：**把不同品牌、尺寸和型号的电视机捆绑在一个集合中，用于描述电视机的共同属性
- **应用类：**定义电视机模型
- **应用对象：**在类的基础上定义属于某一品牌和型号尺寸的一台特定的电视机

类是对象的抽象，对象是类的实例



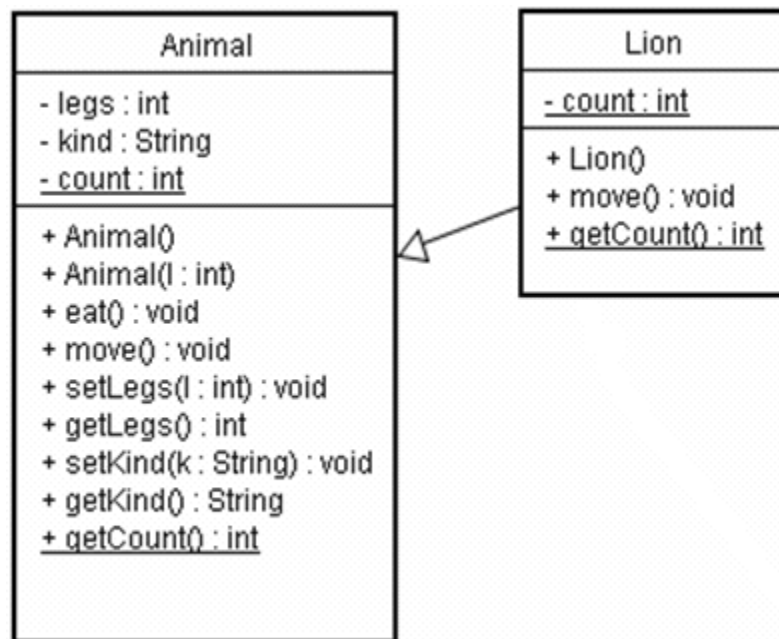
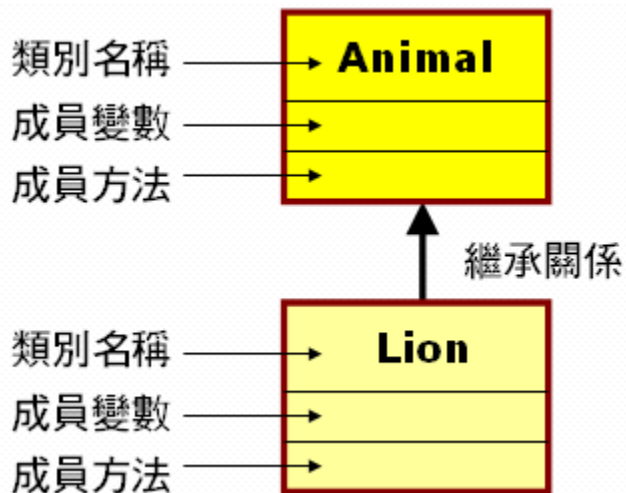
UML中的类

类名
属性： 类型=初始值
方法名(参数表)： 返回值类型

TV
brandName: string modelType: string size: integer
turnon(): boolean turnoff(): boolean

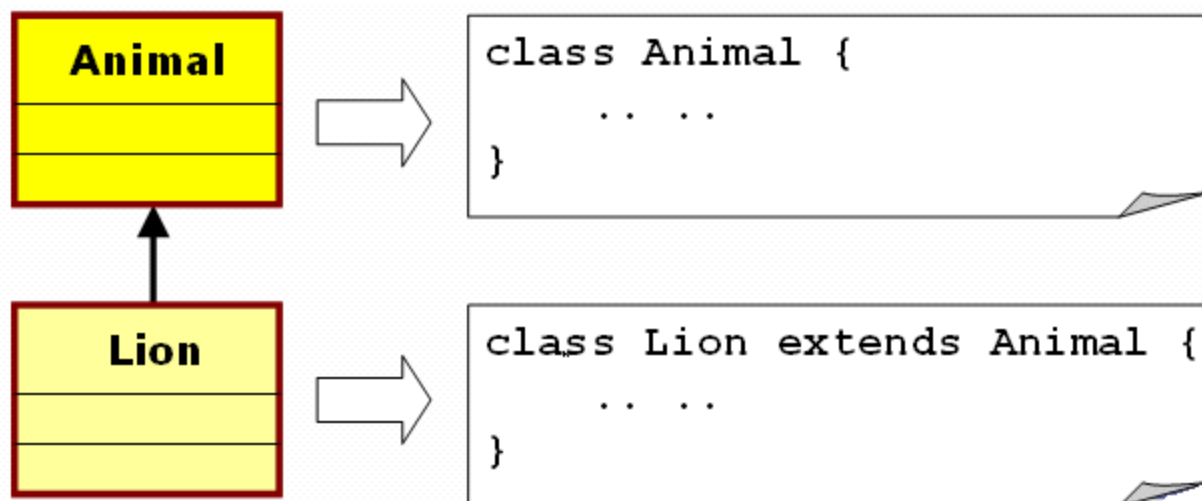
UML的层次

规格说明层： 一种图形，目的是将该图形转换为程序代码。



UML的层次


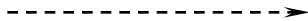


实现层： 将规格说明转换为程序代码。



UML基础知识-事物

事物类型	内容	表示法举例
结构事物	类, 接口, 用例, 组件, 结点等	 A UML class diagram notation example. It shows a rectangular box with a dashed border. Inside the box, the text is: "类的名称" (Class Name) at the top, followed by "+属性1 : String" (Attribute 1: String), "+属性2 : String" (Attribute 2: String), and "+方法1() : String" (Method 1(): String) at the bottom. The box is surrounded by small blue 'x' marks.
动作事物	交互, 状态等	 A UML state diagram notation example. It shows a rounded rectangular box with the text "Waiting" inside.
分组事物	包	 A UML package diagram notation example. It shows a rectangular box with a tab on the top-left corner. Inside the box, the text is "Business rules".
注释事物	解释部分	 A UML note diagram notation example. It shows a rectangular box with a folded top-right corner. Inside the box, the text is "Return copy of self".

UML基础知识-关系

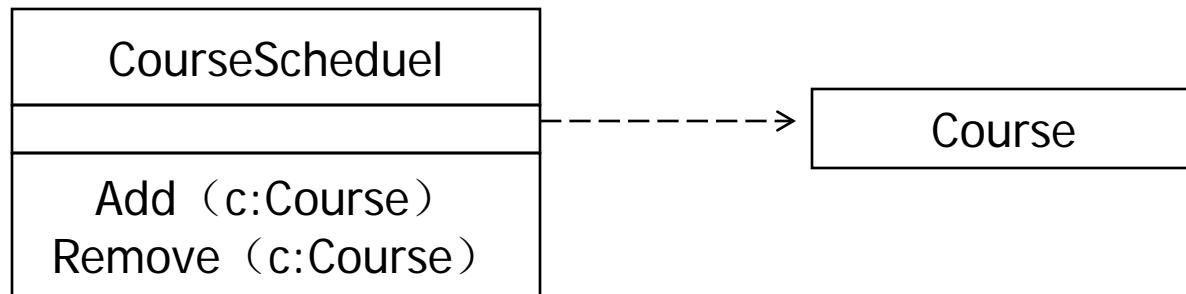
关系	功能	表示法
关联	实例之间连接的描述	
依赖	两个模型元素间的关系,对一个元素（提供者）的改变可能影响或提供信息给其他元素	
泛化	更概括的描述和更具体的种类间的关系，适用于继承	
实现	说明和实现间的关系	

UML基础知识-关系

■ 依赖 dependency ----->

一个事物（独立事物）发生变化会影响使用它的另一个事物（依赖事物），但反之则不然。箭头指向独立事物。

举例：“CourseScheduel” 类依赖于 “Course”类

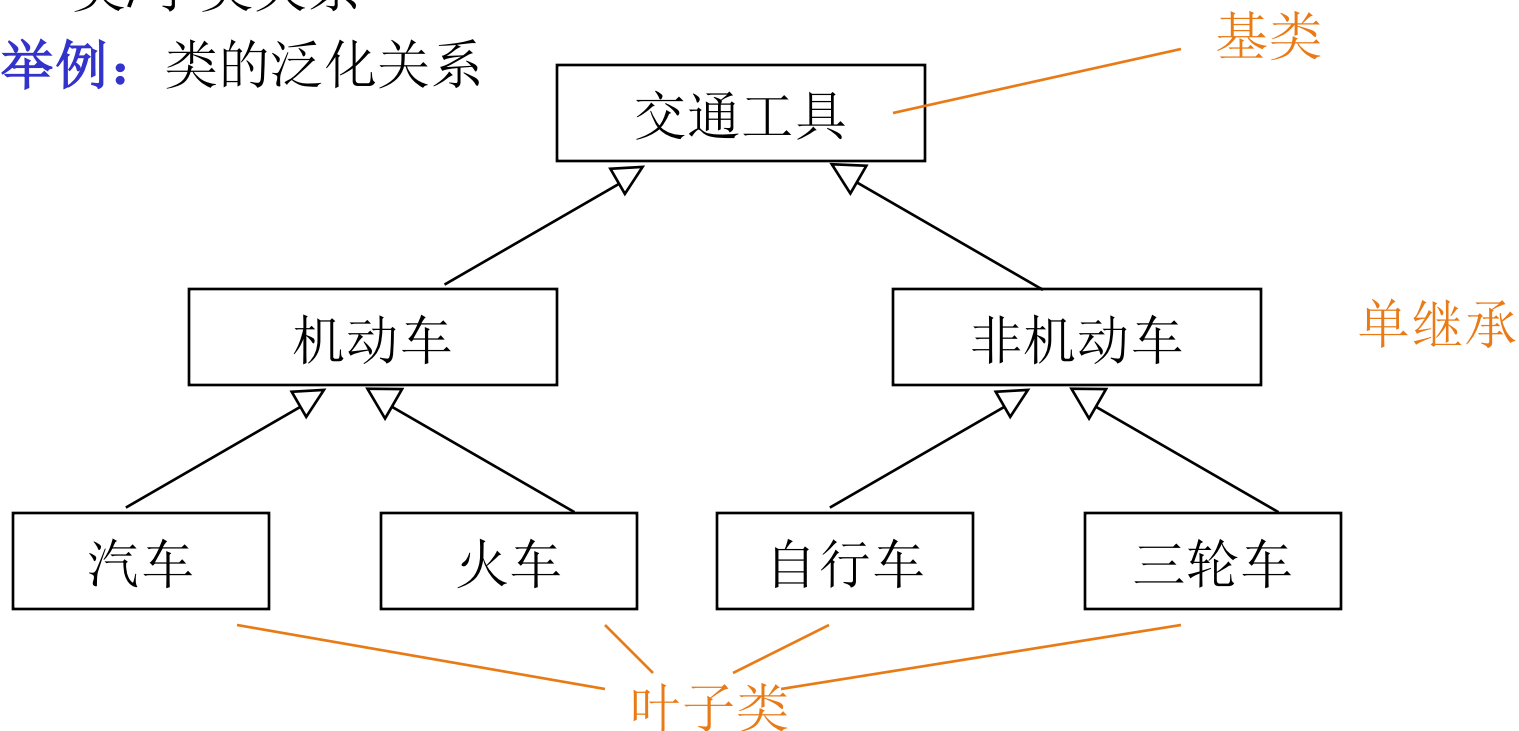


UML基础知识-关系

■ 泛化 generalization

泛化指把一般类连接到较为特殊的类，也称为超类/子类关系或父类/子类关系

举例：类的泛化关系



UML基础知识-关系

■ 关联 association

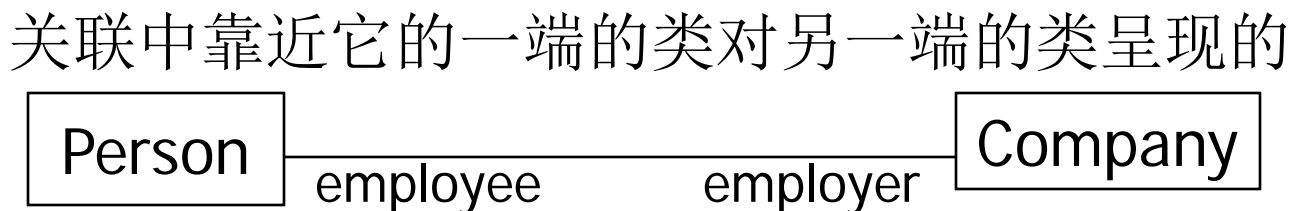
是一种结构关系，它指明一个事物的对象与另一个事物的对象间的联系

■ 关联可以有

■ 名称



■ 角色 职责

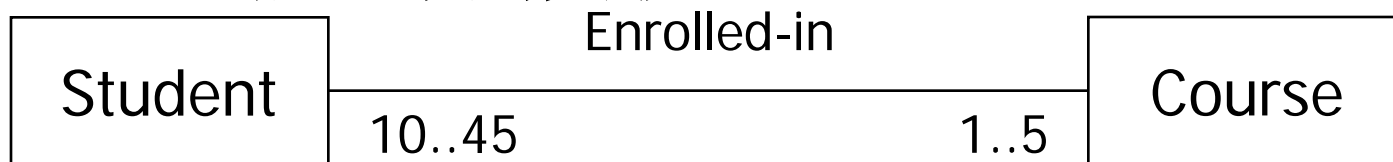


■ 多重性 在关联的另一端的每个对象要求在本端的类必须有多少个对象

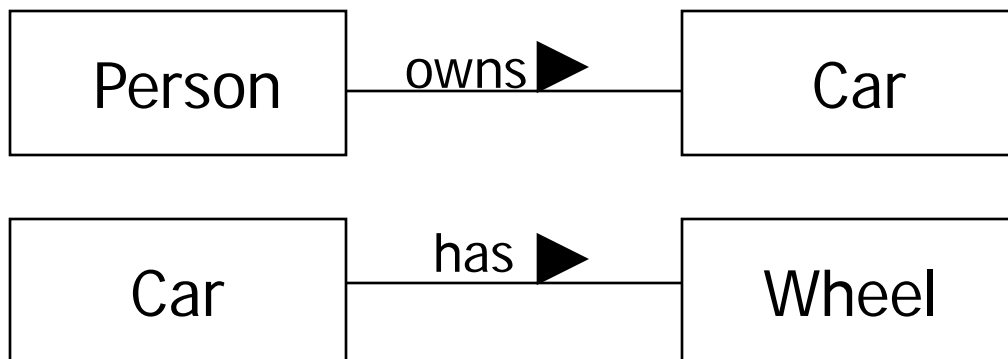


Question?

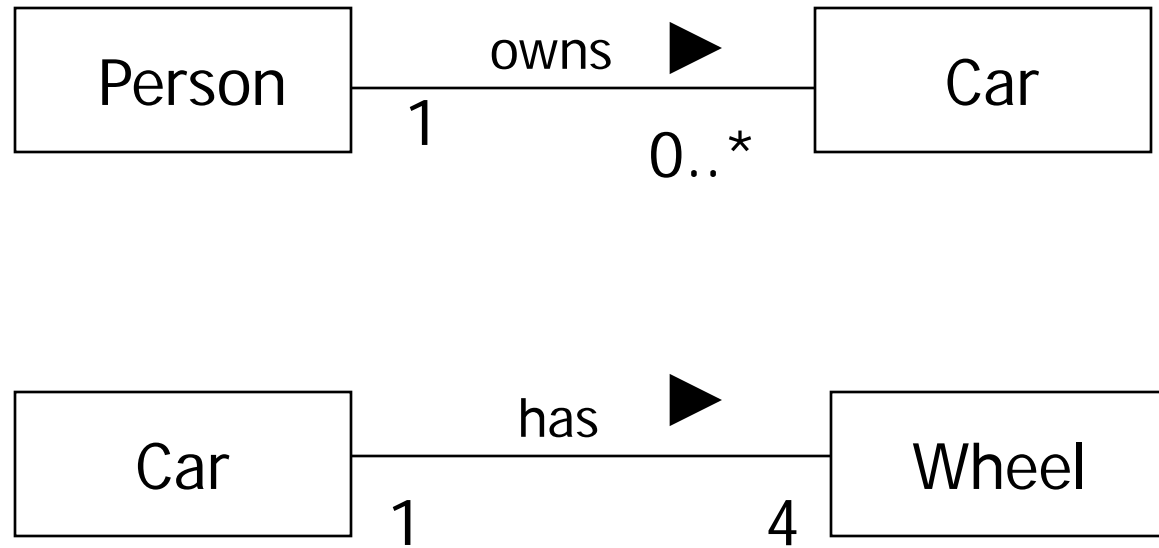
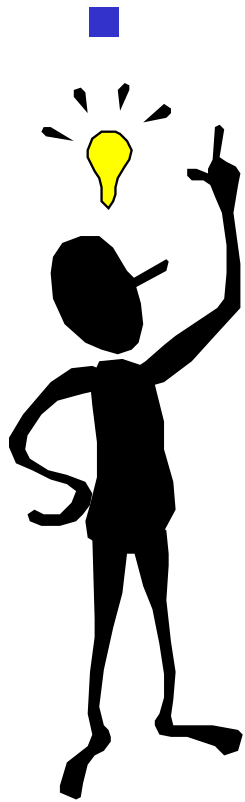
1、描述下列关联?



2、标注下列关联的多重性?

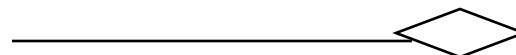


Answers



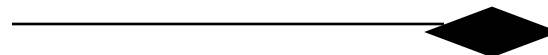
UML基础知识-关系

- 聚合 aggregation



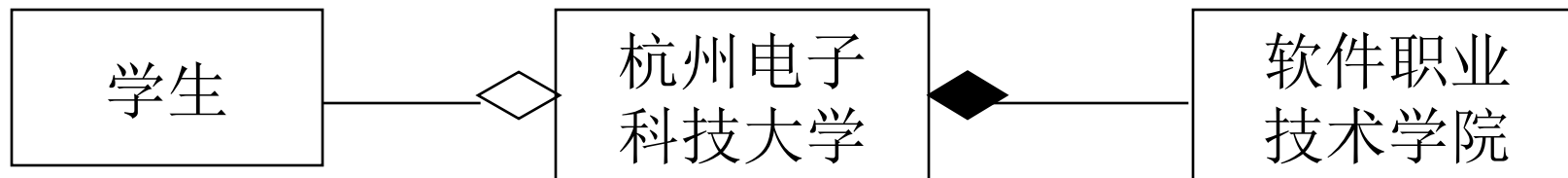
表示类之间的关系是“整体-部分”的关系。
“包含”、“组成”、“分成...部分”

- 组合 composition



特殊的聚合

每个部分只能属于一个整体，且整体和部分具有一致的生命周期。

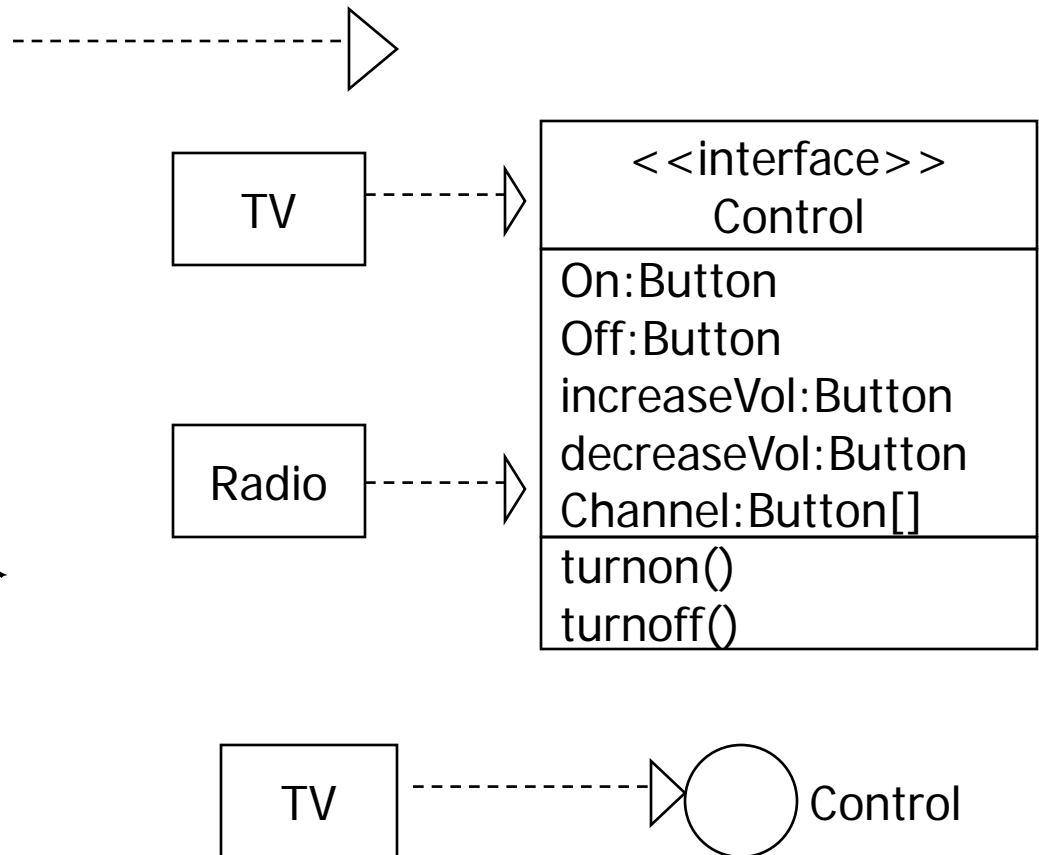


UML基础知识-关系

■ 实现

realization

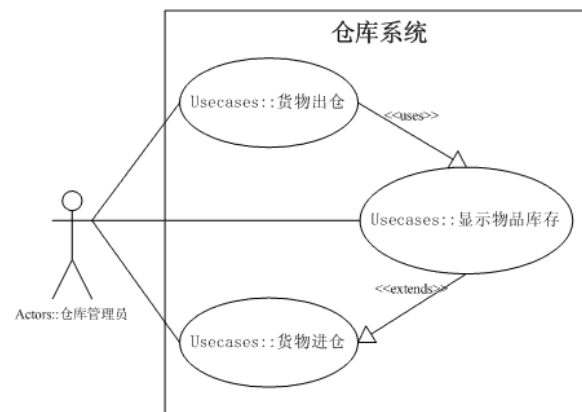
- ❖ 实现用于接口和实现它的类之间
- ❖ 什么是接口？
一个类提供给另一个类的一组操作



UML基础知识-图

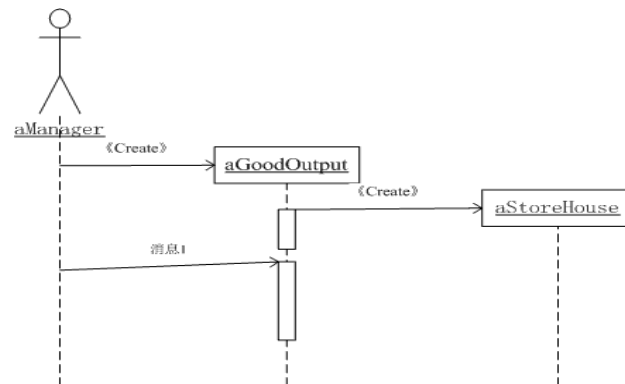
UML静态图

用例图(Use Case Diagram)
类图(Class Diagram)
对象图(Object Diagram)
构件图(Component Diagram)
实施图(Deployment Diagram)



UML动态图

状态图(State Diagram)
顺序图(Sequence Diagram)
协作图(Collaboration Diagram)
活动图(Activity Diagram)





UML项目实践

- 使用UML进行项目的分析和设计时，一般遵循的步骤是
 - 第一步，描述需求,产生用例图
 - 第二步，根据需求建立系统的静态模型，构造系统的结构,这个步骤产生：类图，对象图，组件图和部署图
 - 第三步，描述系统的行为，产生状态图，活动图，顺序图

UML基础知识-图

■ 用例图 use case diagram

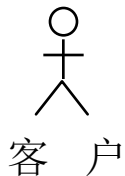
- ✓ 从系统的使用者的角度所理解的系统的总体功能。
- ✓ 建立于系统需求阶段，是开发者和用户对系统需求达成的共识。

❖ 用例



描述一个系统做什么

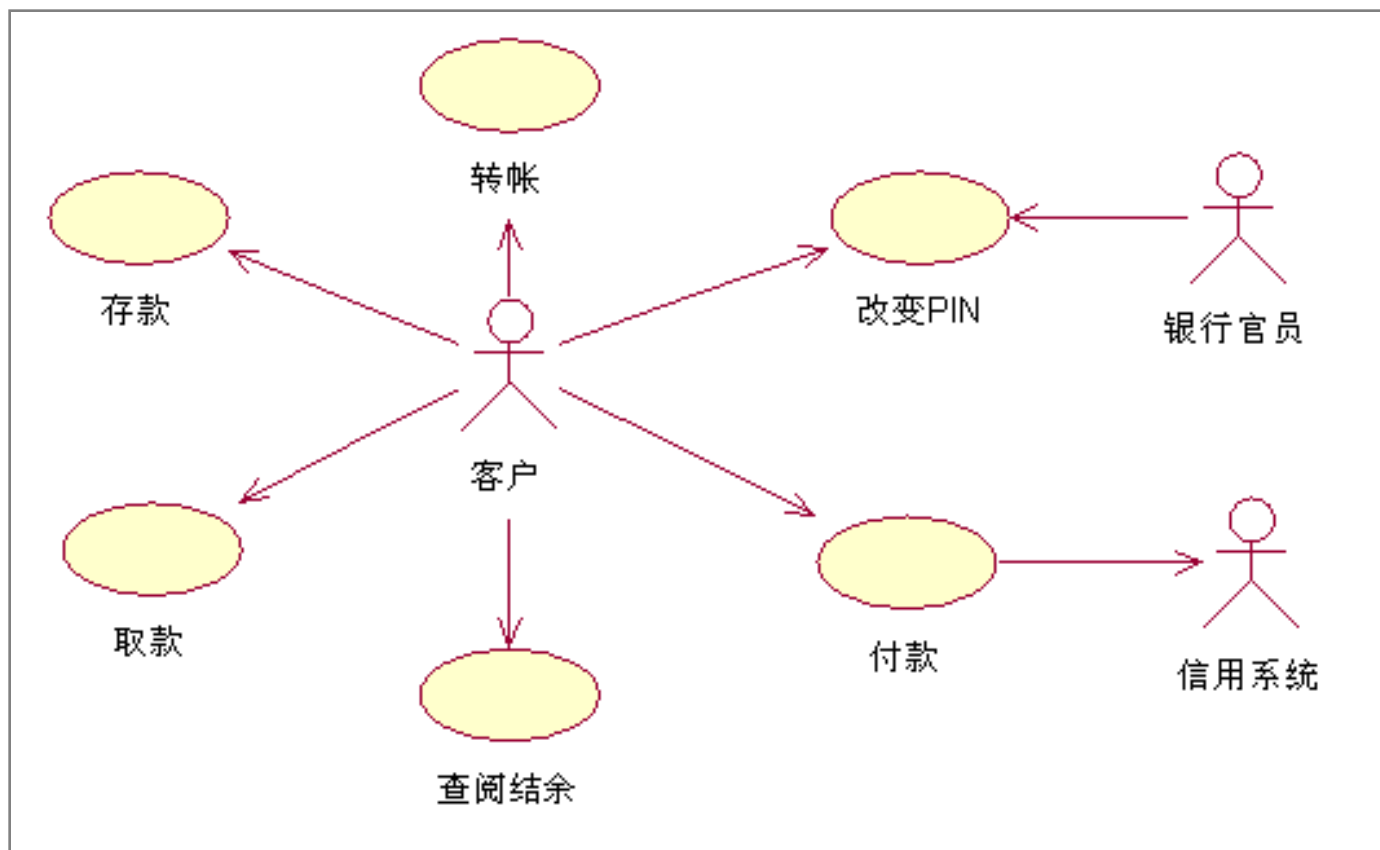
❖ 参与者



表示用例的使用者在与这些用例交互时所扮演的角色
可以是：人、硬件设备或一个系统

UML项目实践-举例

ATM（自动柜员机）系统的用例图





UML基础知识-图

- 类图 class diagram

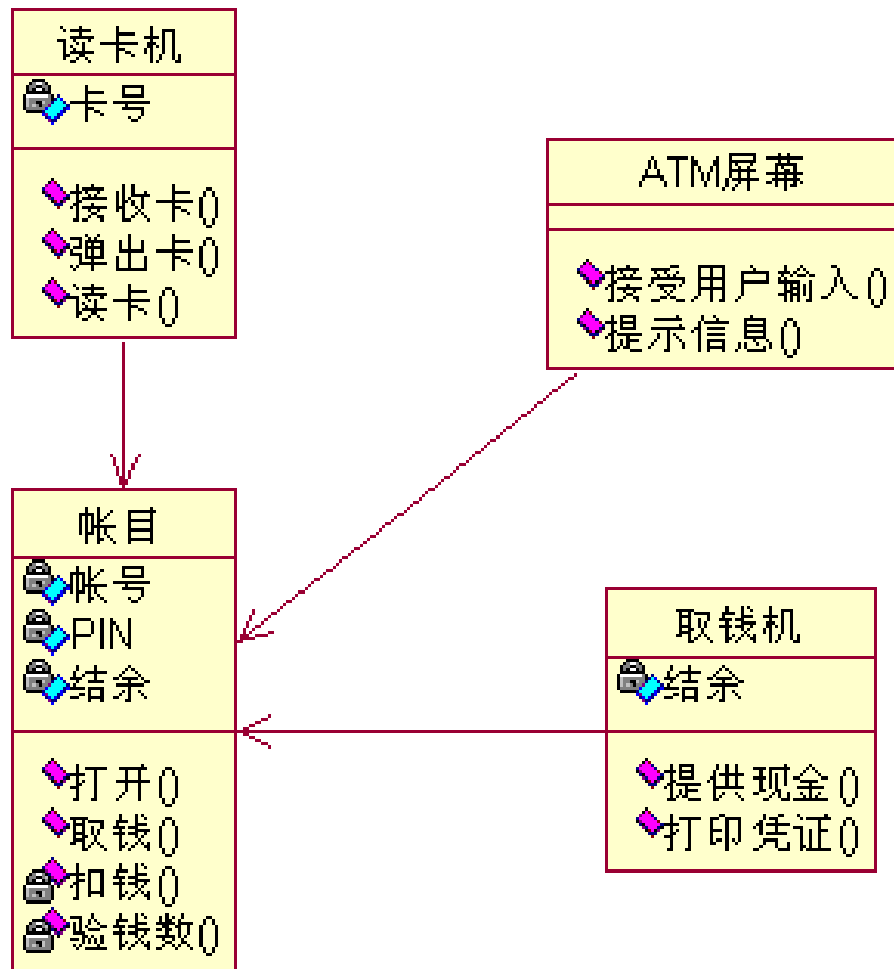
- ✓ 显示一组类、接口、协作以及它们之间关系的图

- 对象图 object diagram

- ✓ 显示某一时刻系统中一组对象以及它们之间关系

UML项目实践-举例

ATM系统中 取款用例 的类图







UML基础知识-图

■ 活动图 activity diagram

- ✓ 显示从活动到活动的流
- ✓ 与交互图不同：交互图观察传递消息的对象，而活动图观察对象之间传送的操作

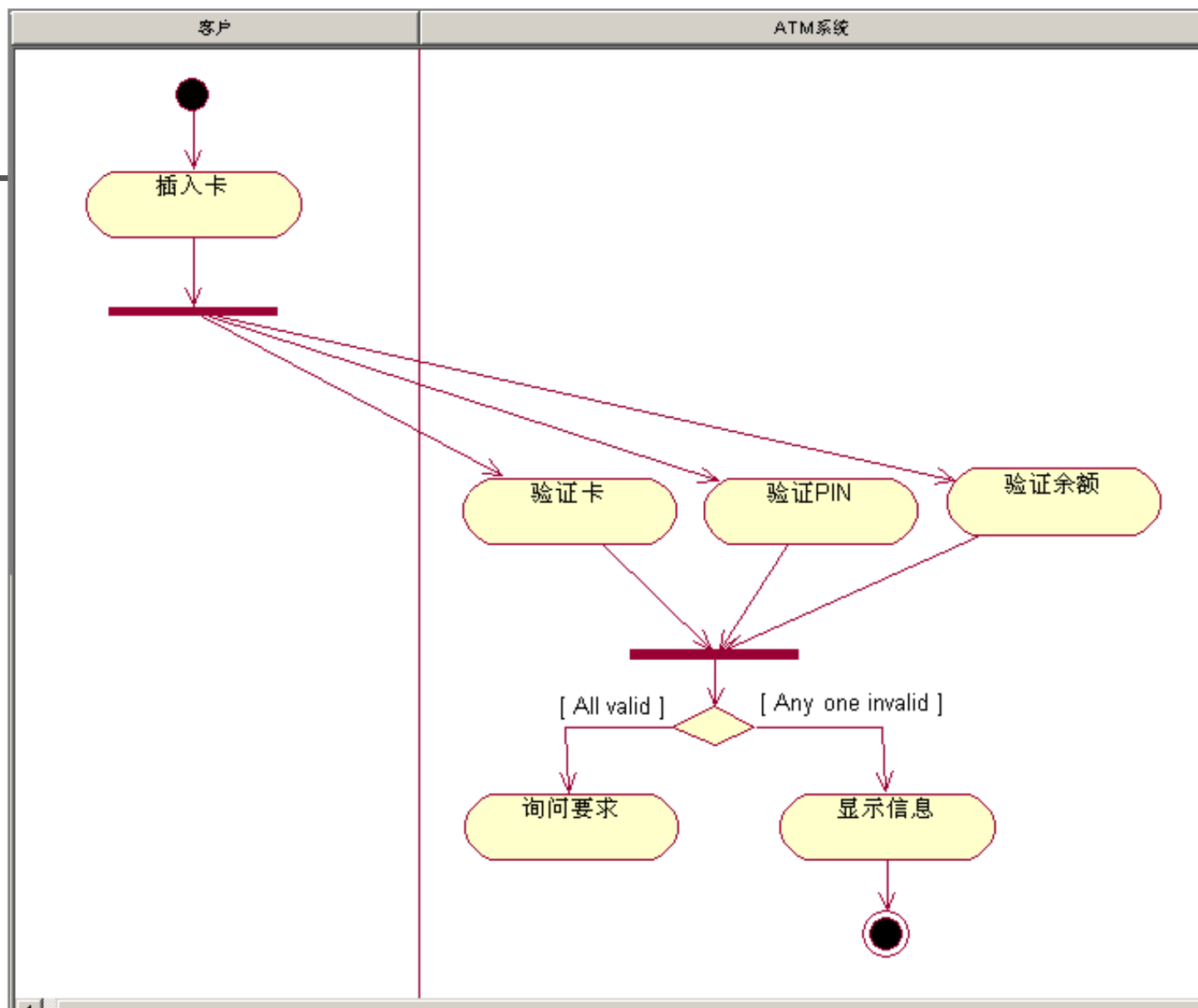
■ 活动图包括

- ✓ 动作状态：不能被分解
- ✓ 活动状态：能被分解
- ✓ 转换： 
- ✓ 同步棒 用来说明并发分叉和汇合 
- ✓ 对象流

■ 泳道图

UML项目实践-举例

ATM系统
中
“客户插
入卡”
的活动图



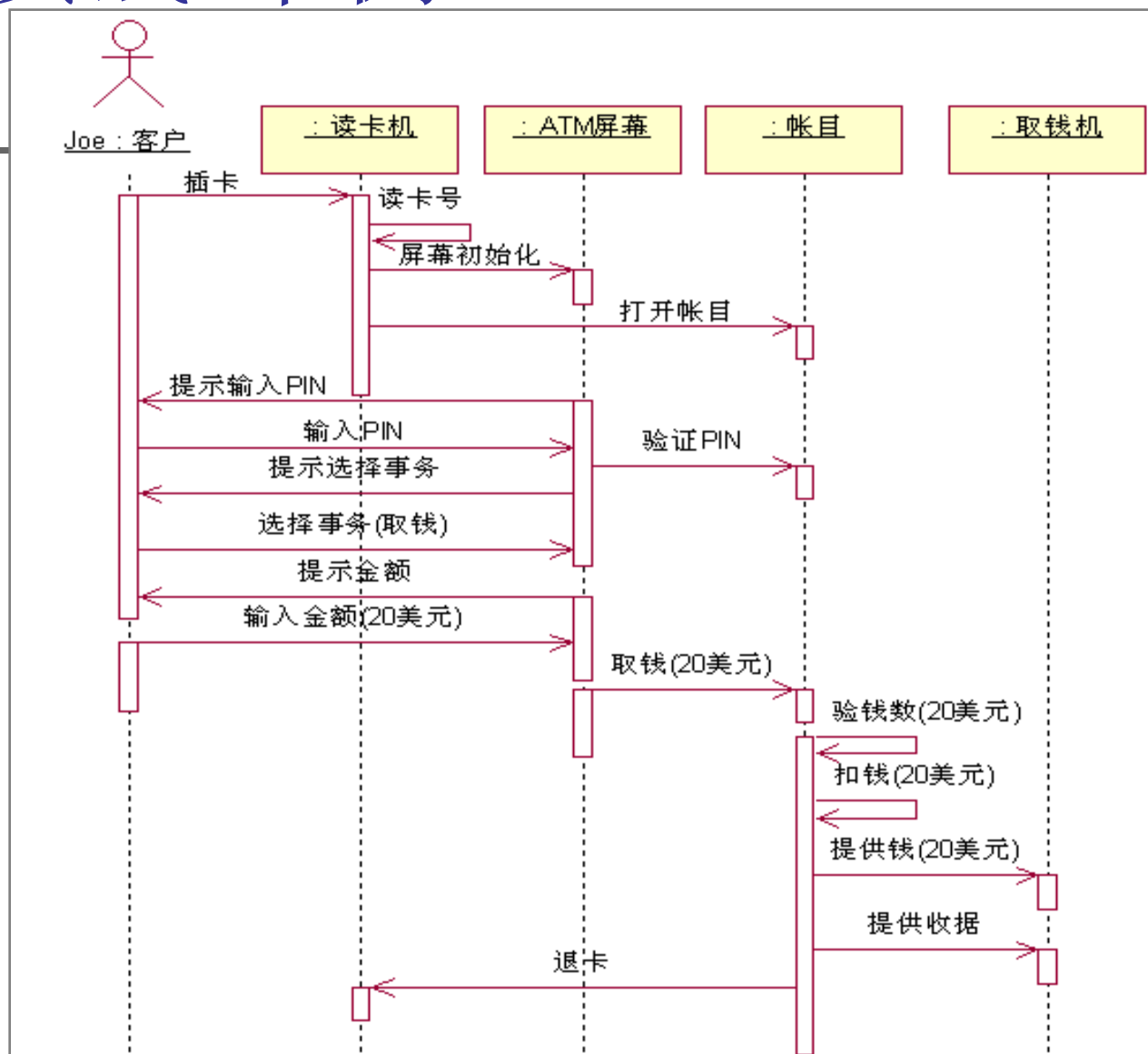


UML基础知识-图

- 顺序图和协作图均被称为**交互图 interaction diagram**
 - ✓ 由一组对象、对象间的关系、对象间发送的消息组成
 - ✓ 一种动态视图
 - ✓ 可以单独使用、也可以对用例中的特定控制流程建模
- **顺序图 sequence diagram**
 - ✓ 强调消息的时间顺序
 - ✓ 有对象生命线、有控制焦点
- **协作图 collaboration diagram**
 - ✓ 强调收发消息的对象的组织结构
 - ✓ 有路径、有顺序号
- **同构的：** 两种图之间可以相互转换，而没有任何信息损失

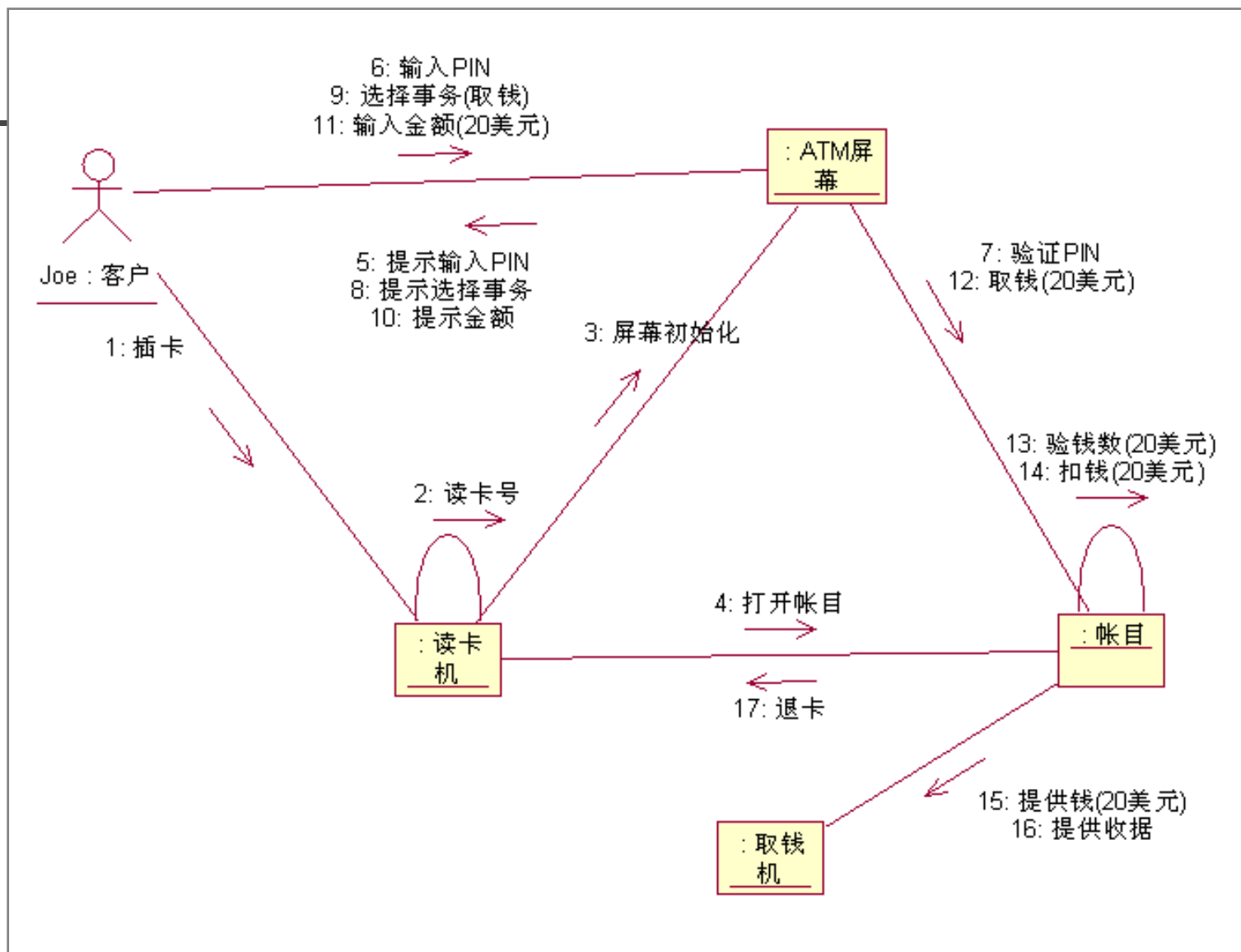
UML项目实践-举例

某客户 Joe取 20美 元的 顺序 图



UML项目实践-举例

某客户 Joe取 20美 元的 协作 图



UML基础知识-图

■ 状态图 statechart diagram

- ✓ 对一个对象按事件排序的行为建模
- ✓ 与交互图不同：交互图对共同工作的对象群体的行为建模，而状态图对单个对象的行为建模

■ 状态图包括状态、转换、事件、动作

- ✓ 初始态



每张状态图有1个初始态

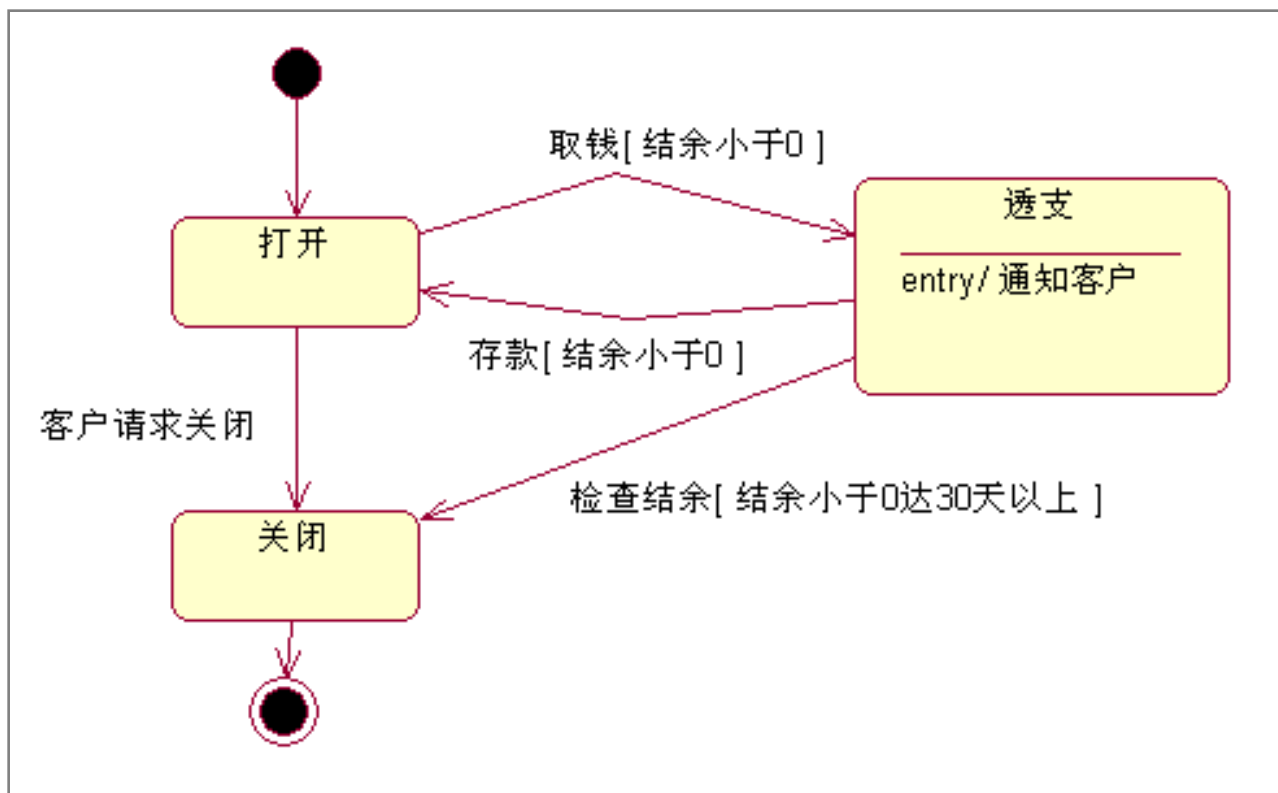
- ✓ 终止态



每张状态图有多个终止态

UML项目实践-举例

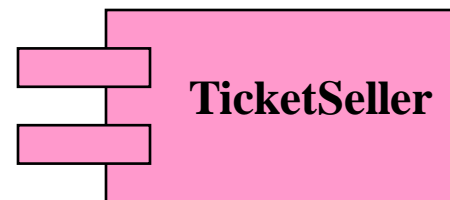
账目类的状态图



UML基础知识-图

■ 构件图 component diagram

- ✓ 构件：系统中遵从一组接口且提供其实现的物理的、可替换的部分
- ✓ 构件图显示系统中的构件以及它们之间的依赖、泛化和关联关系
- ✓ 构件图可以用来对源代码。可执行的发布体、物理数据库建模

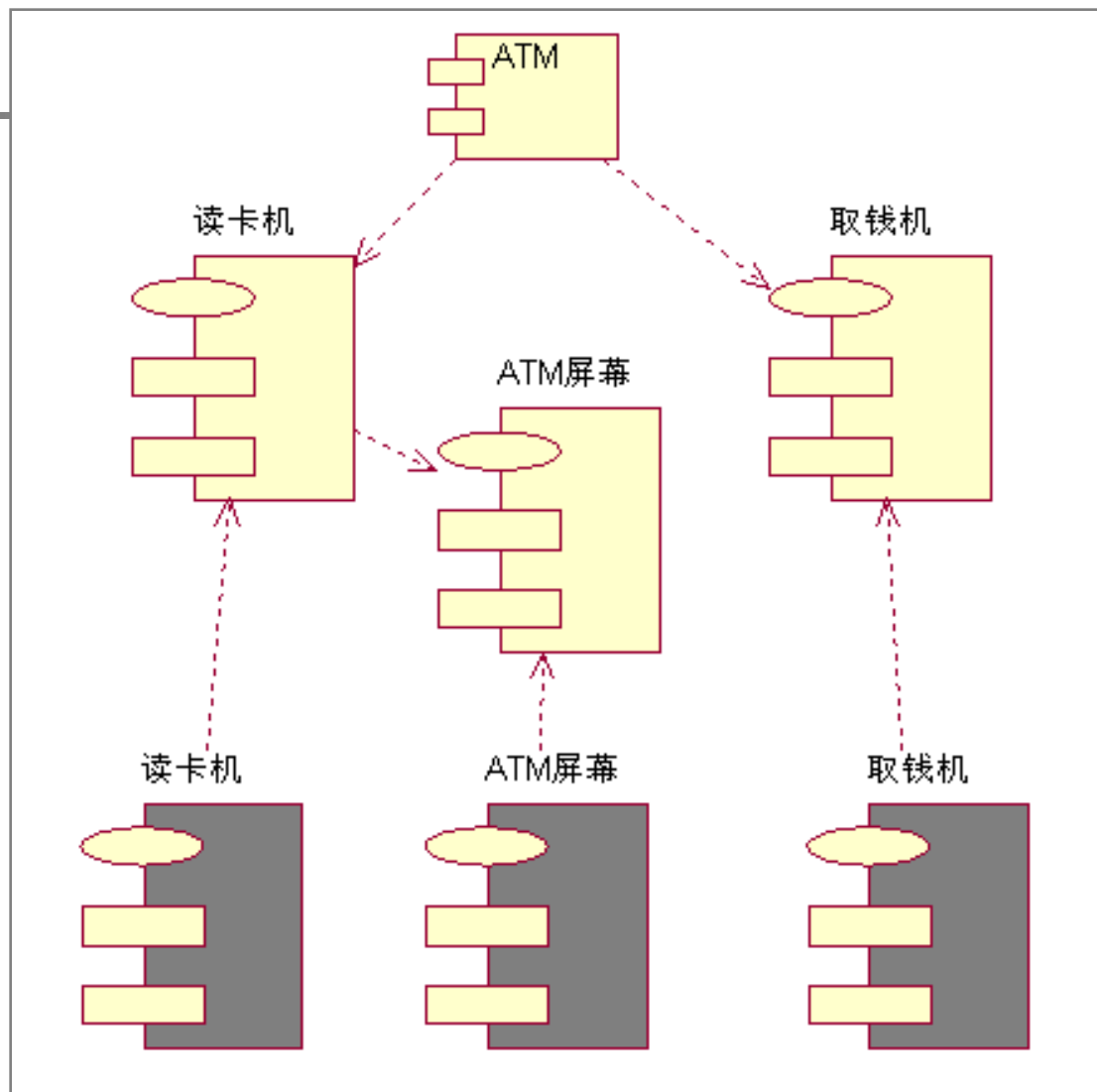


■ 实施图 deployment diagram

- ✓ 展现了系统运行时，系统内处理结点以及驻留在结点中的构件

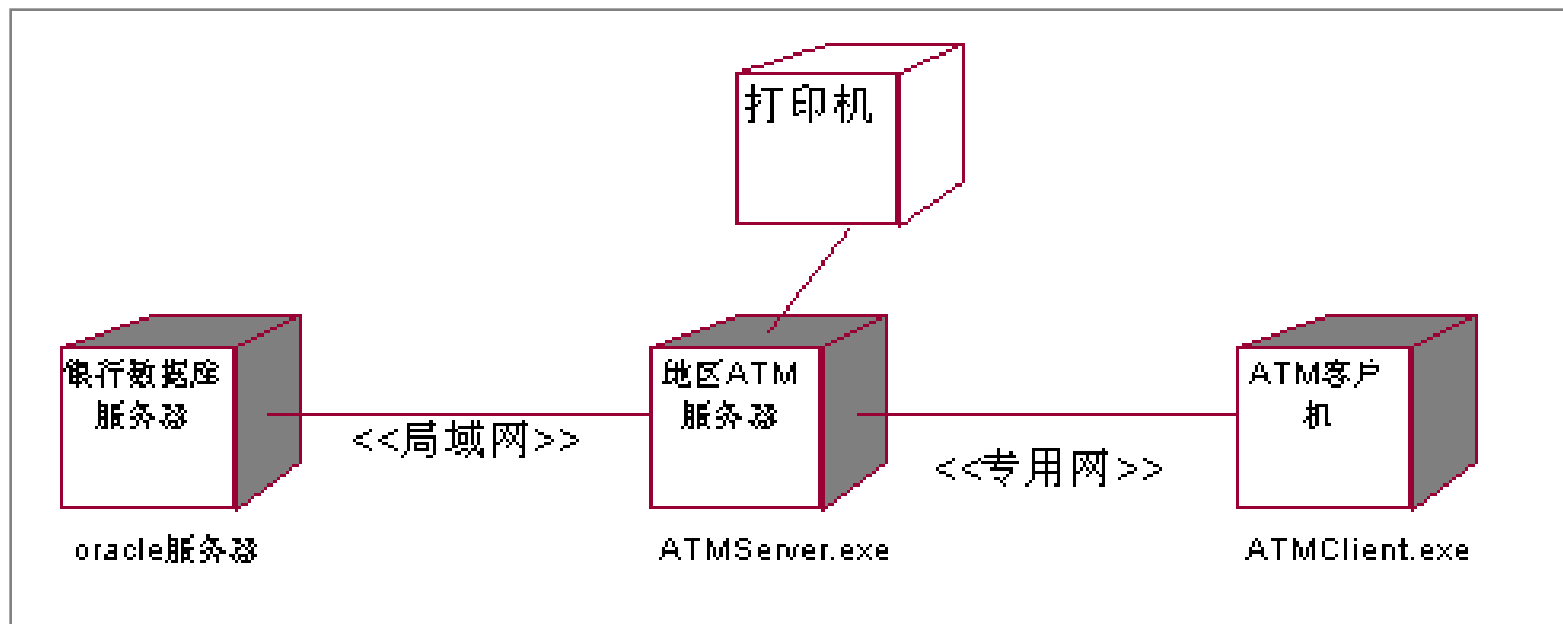
UML项目实践-举例

ATM系统 客户的 构件图



UML项目实践-举例

ATM系统的实施图





绘制UML图

- 工具软件
 - Rational Rose
 - Visio
 - Visual Modeler
 - Together
 - Visual UML
 - Enterprise Architect(EA)