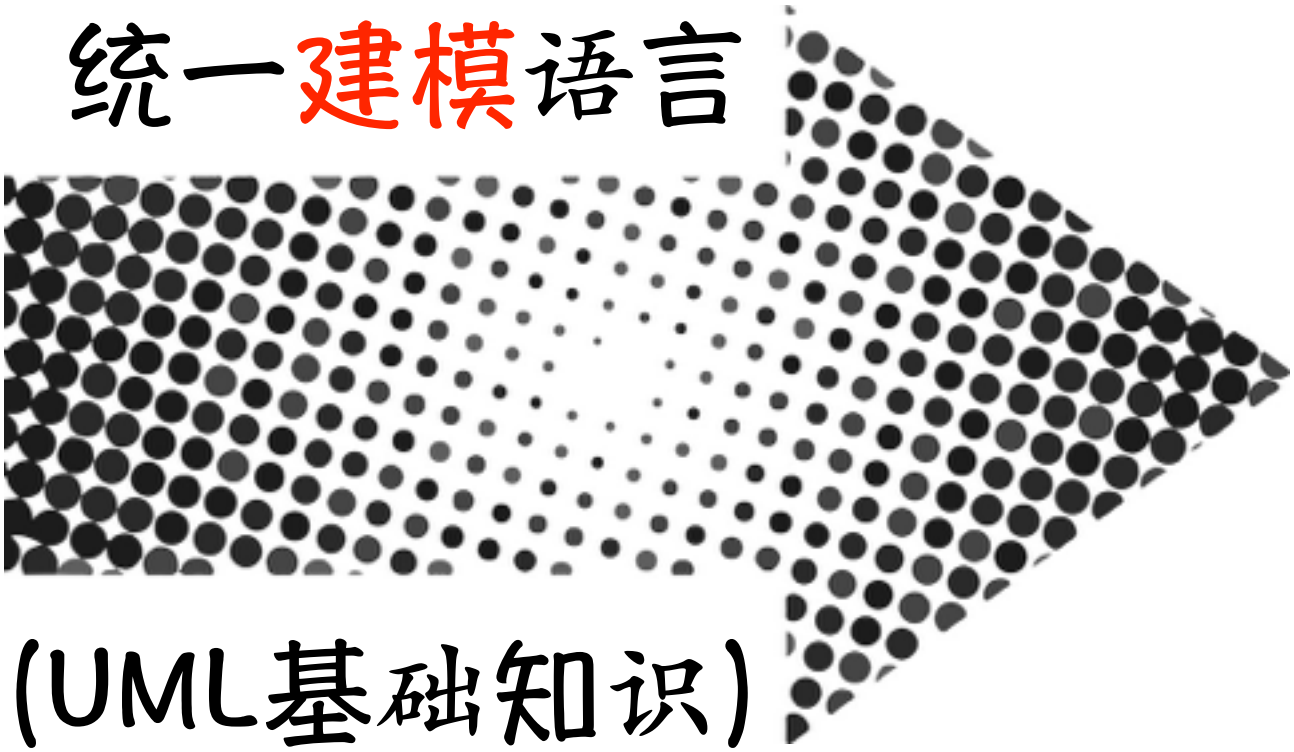
统一建模语言



(UML基础知识)

1.软件建模概述

1.1 什么是建模

任何事情都要先想清楚了才能做，软件开发更是如此！软件开发过程不可能一上来就开始盲目写代码，写代码之前必须搞清楚下面一些基本问题：

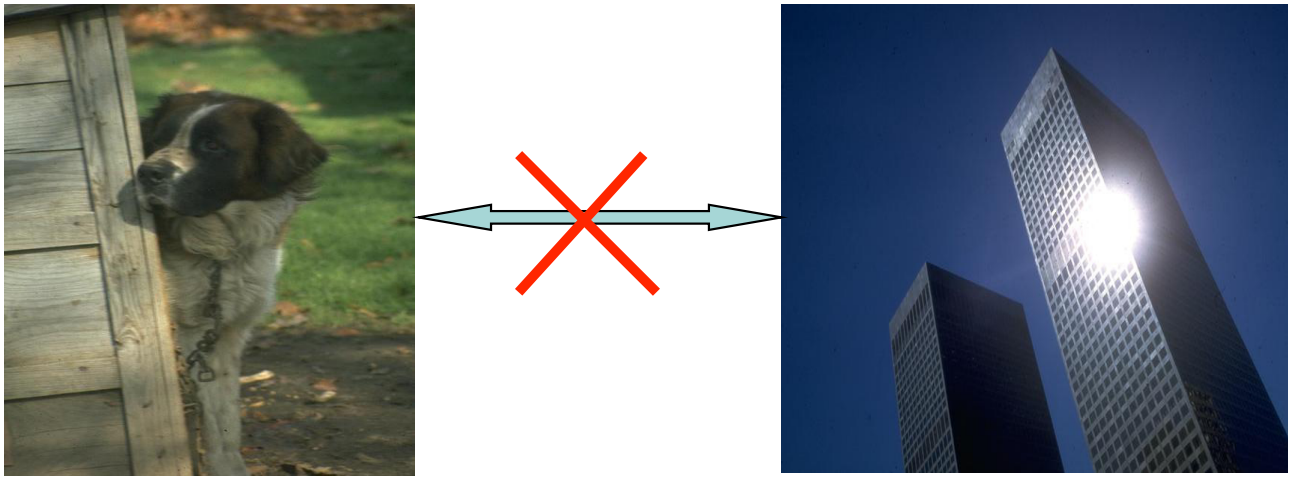
要做什么？
做成什么样？
怎么去做？

软件设计: 把软件开发想清楚的过程.
软件工程: 对软件开发全过程进行建模和管理.
模型: 对问题的书面上的无歧义文字或图形的描述.简言之, 模型是对现实的简化. 通过模型, 人们可以了解所研究事物的本质.

最杰出的模型: 地图

1.2 建模的原则

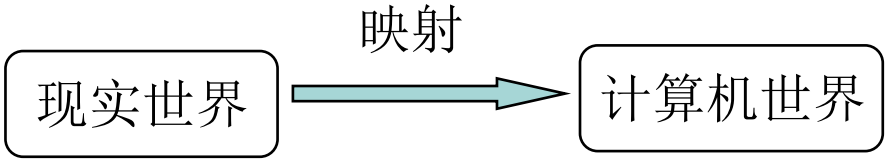
(1). 选择建立什么样的模型对如何发现和解决问题具有重要的影响。正确的模型有助于提高开发者的洞察力。



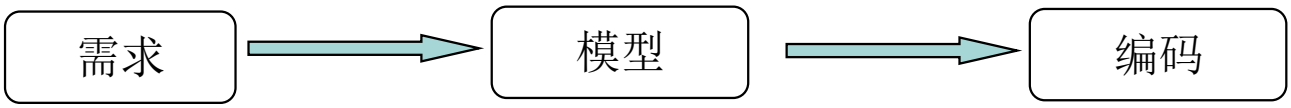
(2). 每个模型可以有多种表达方式. 使用者的身份和使用的原因是评判模型好坏的关键。
(**3**). 最好的模型总是能够切合实际. 模型是现实的简化，必须保证简化过程不会掩盖任何重要的细节。

1.3 软件建模的实现过程

软件建模的作用是把来源于现实世界的问题转化为计算机可以理解和实现的问题.



软件建模的实现过程是从需求入手，用模型表达分析设计过程，最终将模型映射成软件实现.



2.UML

（1）UML(United Modeling Language，统一建模语言)：是一种基于面向对象的可视化建模语言.
（2）**UML** 采用了一组形象化的图形(如类图)符号作为建模语言，使用这些符号可以形象地描述系统的各个方面
（3）UML 通过建立图形之间的各种关系(如类与类之间的关系)来描述模型.

2.1 **UML** 中一共有 **10** 种图

名称	重要性
类图	★★★★★
对象图	★★★☆☆
包图	★☆☆☆☆
组件图	★☆☆☆☆
部署图	★☆☆☆☆
用例图	★★★☆☆
时序图	★★★☆☆
协作图	★☆☆☆☆
状态图	★★☆☆☆
活动图	★★☆☆☆

2.2 UML 中的关系

UML 中的关系主要包括 4 种:

关联关系(association)
依赖关系(dependency)
泛化关系(generalization)
实现关系(realization)

3. 用例图

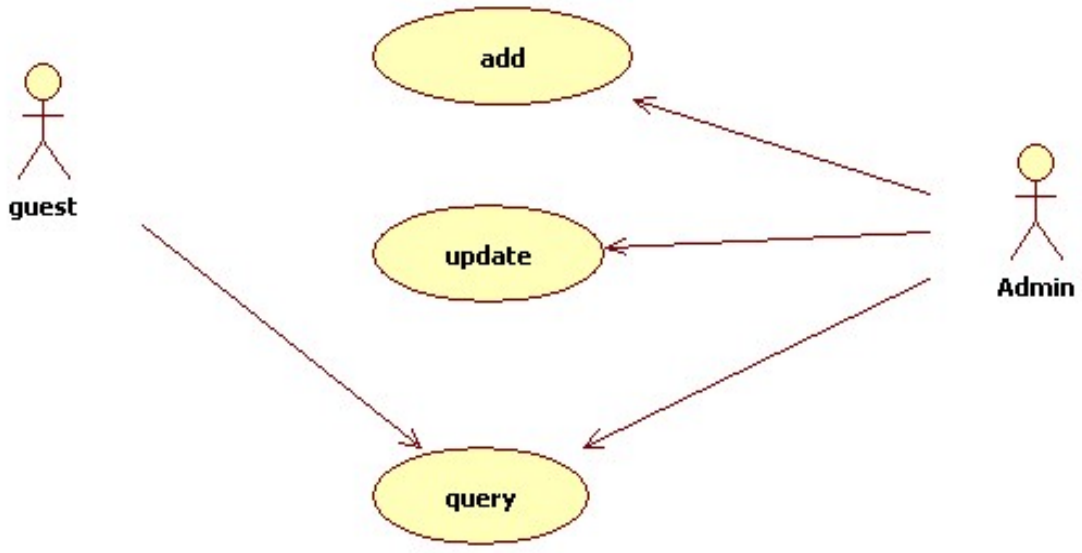
（1）用例图(Use Case Diagram): 也称为用户模型图, 是从软件需求分析到最终实现的第一步, 它是从客户的角度来描述系统功能.

（2）用例图包含 3 个基本组件:

参与者(Actor): 与系统打交道的人或其他系统即使用该系统 的人或事物. 在 UML 中参与者用人形图标表示

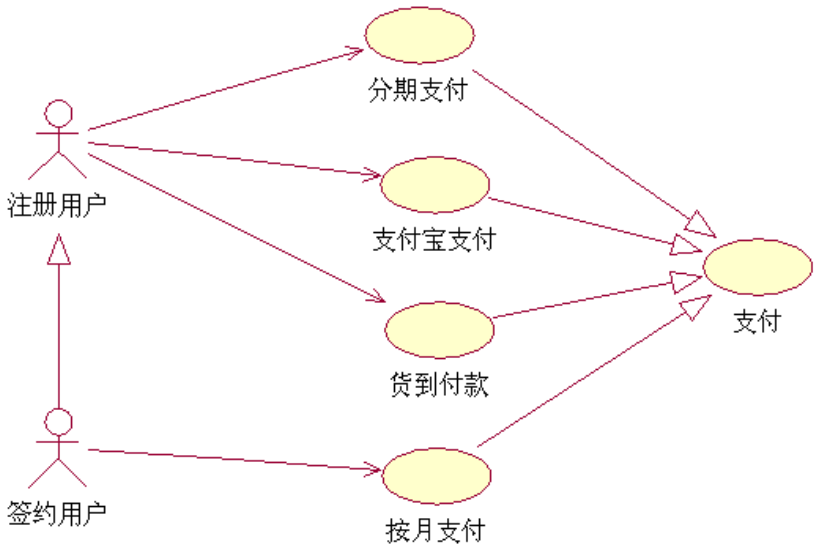
用例(Use Case): 代表系统的某项完整的功能. 在 UML 中使 用一个椭圆来表示

关系: 定义用例之间的关系 ------ 泛化关系, 扩展关系, 包含关系



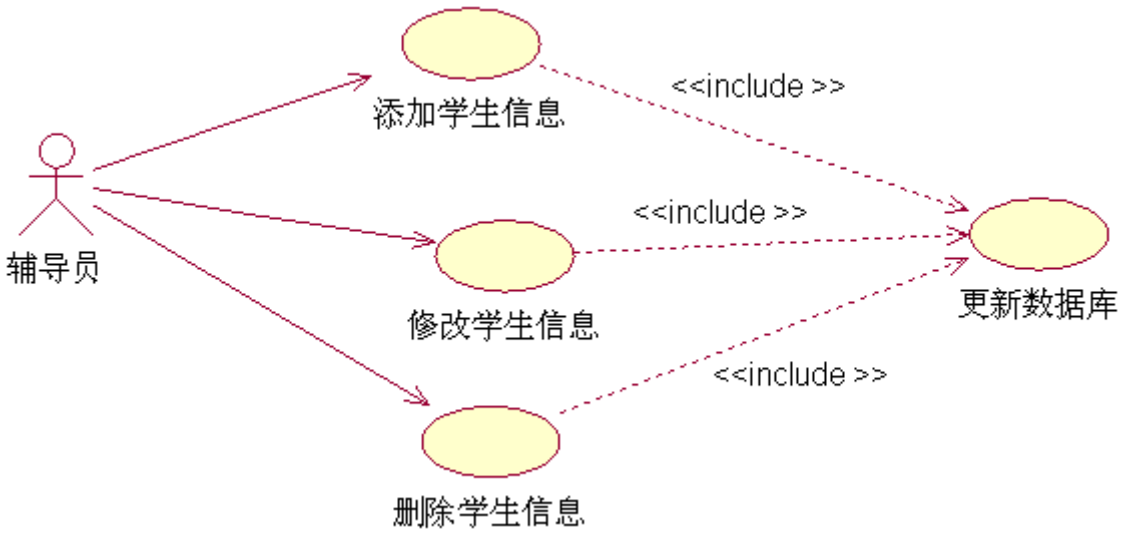
3.1 用例之间的关系——泛化关系

泛化关系: 表示同一业务目的(父用例)的不同技术实现(各个子用例). 在 UML 中, 用例泛化用一个三角箭头从子用例指向父用例. 以下是某购物网站为用户提供不同的支付方式



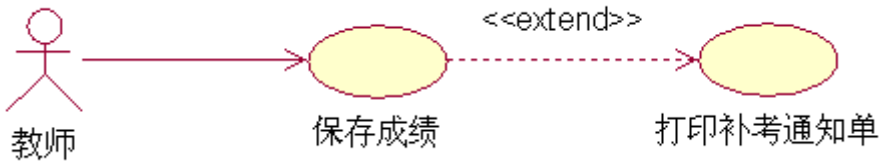
3.2 用例之间的关系——包含关系

一个用例可以包含其他用例具有的行为, 并把它包含的用例行为作为自身行为的一部分. 在 UML 中包含关系用虚线箭头加 “<<include>>”, 箭头指向被包含的用例



3.3 用例之间的关系——扩展关系

如果在完成某个功能的时候偶尔会执行另外一个功能，则用扩展关系表示. 在 UML 中扩展关系用虚线箭头加 “<<extend>>”，箭头指向被扩展的用例



3.4 用例图练习

下面是关于一个公司的人事管理系统的需求的简单描述，建立其相应的用例模型：该人事管理系统的用户是公司的人事管理干部. 该系统具有人事档案库，保存员工的人事信息，包括姓名，性别，出生年月，健康状况，文化程度，学位，职称，岗位，聘任时间，任期，工资，津贴，奖罚记录，业绩，论著和家庭情况等，系统提供的基本服务有人事信息的管理，包括人事规定的权调动与聘任，职称评定，奖罚等，并且可以按照限查询人事信息，生成与输出统计报表等. 该人事系统每月向公司的财务系统提供员工的工资，津贴等数据.

4 类图

类图是面向对象系统建模中最常用的图. 是定义其他图的基础.

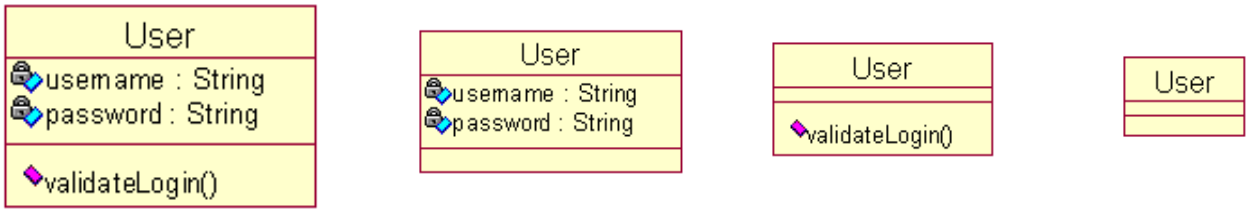
类图主要是用来显示系统中的类, 接口以及它们之间的关系.

类图包含的主要元素有类, 接口和关系. 其中关系有泛化关系,关联关系, 依赖关系和实现关系. 在类图中也可以包含注释和约束.

4.1 类的表示法

1. 类是类图的主要组件, 由 3 部分组成: 类名, 属性和方法. 在 UML 中, 类用矩形来表示, 顶端部分存放类的名称, 中间部分存放类的属性, 属性的类型及值, 底部部分存放类的方法, 方法的参数和返回

2.在 UML 中可以根据实际情况有选择的隐藏属性部分或方法部分或两者都隐藏

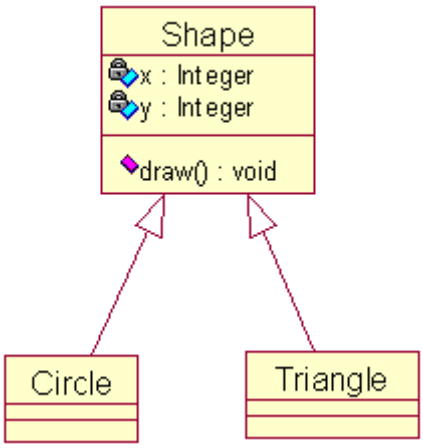


3.在 UML 中, 共有类型有 + 表示, 私有类型用 – 表示, 保护类型用 # 表示. UML 的工具开发商可以使用自己定义的符号表示不同的可见性

4.2 类之间的关系 ---- 泛化关系

1. 在 UML 中, 泛化关系用来表示类与类, 接口与接口之间的继承关系. 泛化关系有时也称为” is a kind of” 关系

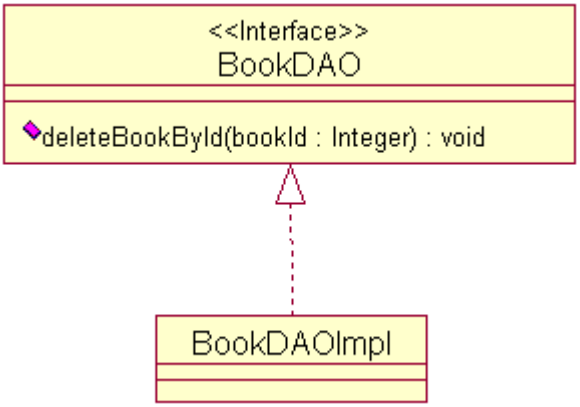
2. 在 UML 中泛化关系用一条实线空心箭头有子类指向父类



4.3 类之间的关系 ---- 实现关系

在 UML 中, 实现关系用来表示类与接口之间的实现关系.

在 UML 中实现关系用一条虚线空心箭头由子类指向父类



4.4 类之间的关系 ---- 依赖关系

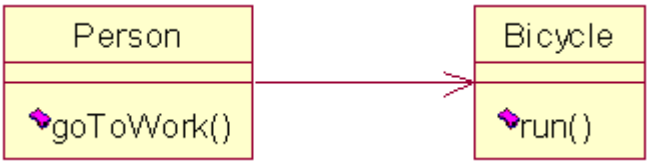
对于两个相对独立的系统，当一个系统负责构造另一个系统的实例，或者依赖另一个系统的服务时，这两个系统之间体现为依赖关系. 例如生产零件的机器和零件，机器负责构造零件对象; 充电电池和充电器，充电电池通过充电器来充电;自行车Bicycle和打气筒Pump，自行车通过打气筒来充气

在现时生活中，通常不会为某一辆自行车配备专门的打气筒，而是在需要充气的时候，从附近某个修车棚里借个打气筒打气。在程序代码中，表现为Bicycle类的expand()方法有个Pump类型的参数。以下程序代码表示某辆自行车先后到两个修车棚里充气：

```
//到第一个修车棚里充气
myBicycle.expand(pumpFromRepairShed1);
//若干天后，到第二个修车棚里充气
myBicycle.expand(pumpFromRepairShed2);
```

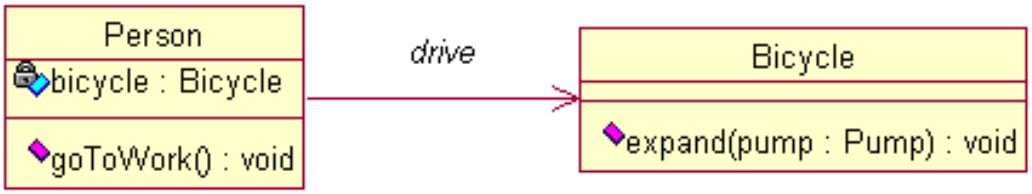
4.5 类之间的关系 ---- 关联关系

对于两个相对独立的系统，当一个系统的实例与另一个系统的一些特定实例存在固定的对应关系时，这两个系统之间为关联关系。例如客户和订单，每个订单对应特定的客户，每个客户对应一些特定的订单；公司和员工，每个公司对应一些特定的员工，每个员工对应一特定的公司；自行车和主人，每辆自行车属于特定的主人，每个主人有特定的自行车。而充电电池和充电器之间就不存在固定的对应关系，同样自行车和打气筒之间也不存在固定的对应关系。



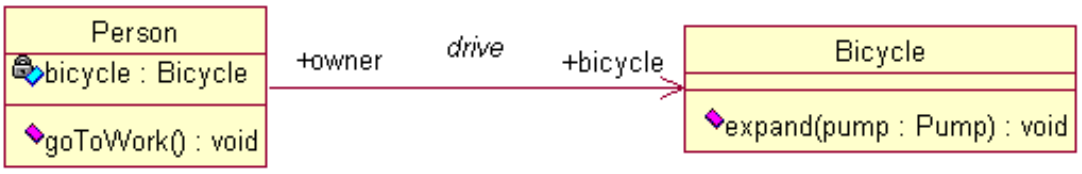
4.6 关联关系的名称

关联关系的名称: 关联关系可以有一个名称, 用于描述该关系的性质. 此关联名称应该是动词短语, 因为它表明源对象正在目标对象上执行动作.



4.7 关联关系的角色

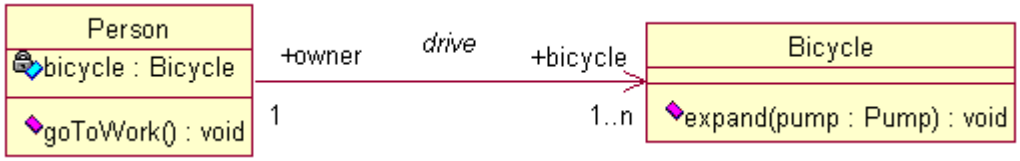
当一个类处于关联的某一端时, 该类就在这个关系中扮演一个特定的角色. 具体来说, 角色就是关联关系中一个类对另一个类所表现的职责. 角色名称是名词或名称短语.



4.8 关联关系的多重性

关联关系的多重性是指有多少对象可以参与该关联, 多重性可以用来表达一个取值范围, 特定值, 无限定的范围.

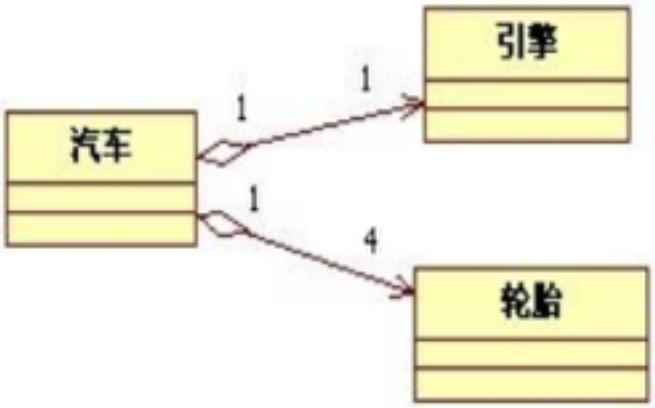
表示法	说明	表示法	说明
0	表示 0 个对象	1..n	表示 1 - n 个对象
0..1	表示 0 - 1 个对象	n	表示 n 个对象
0..n	表示 0 - n 个对象	*	表示许多个对象
1	表示 1 个对象		



关联关系是类与类之间的联结，它使一个类知道另一个类的属性和方法，关联可以是双向的，也可以是单向的。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头。在 Java和C++ 中，关联关系是通过使用成员变量来实现的。 如人与车。

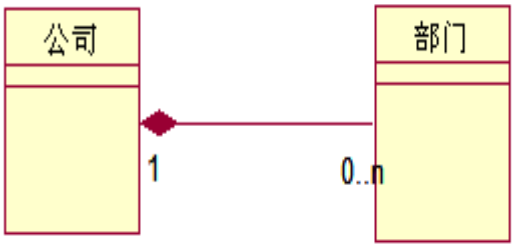
4.9 类之间的关系—关联—聚合

1. 聚合关系是关联关系的一种，是更强的关联关系。
2. 聚合是整体和部分之间的关系，例如汽车由引擎、轮胎以及其它零件组成。
3. 聚合关系也是通过成员变量来实现的。但是，关联关系所涉及的两个类处在同一个层次上，而聚合关系中，两个类处于不同的层次上，一个代表整体，一个代表部分。



4.10 类之间的关系—关联—组合

1. UML类图关系中合成关系是关联关系的一种，是比聚合关系还要强的关系。
2. 代表整体的对象负责代表部分对象的生命周期。

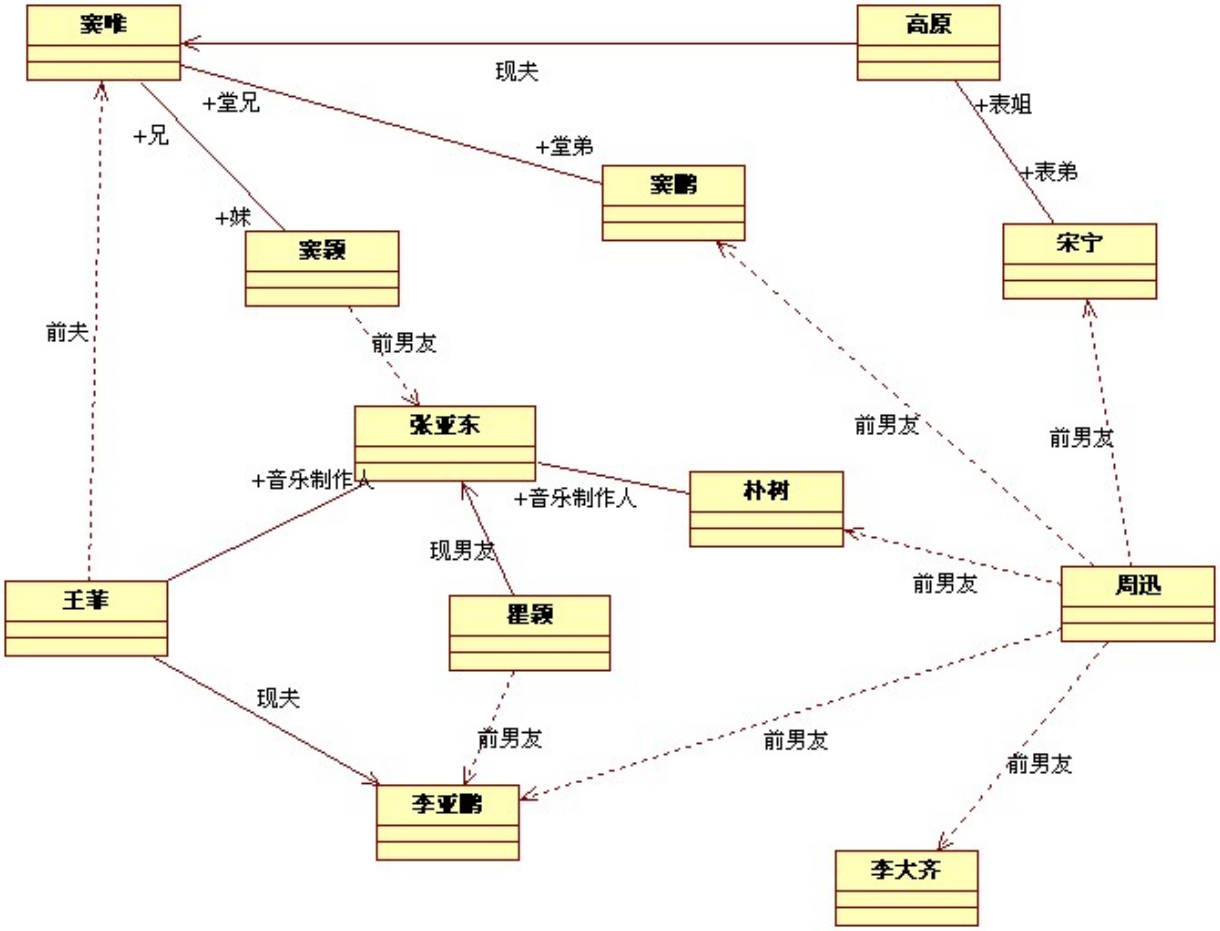


类图练习1

根据以下网友描述出艺人之间关系的”类图”

网易贵州安顺网友(222.86.*.*) 的原贴：

网友爆料简化版
网易北京海淀网友(221.221.*.*) 的原贴：
周迅前男友窦鹏是窦唯堂弟，窦唯是王菲前老公。周迅前男友宋宁是高原的表弟，高原是窦唯现任老婆，窦唯是王菲前老公。周迅前男友李亚鹏是王菲现任老公。周迅前男友朴树的音乐制作人是张亚东,张亚东是王菲的前老公窦唯的妹妹窦颖的前老公，也是王菲的音乐制作人。张亚东是李亚鹏前女友瞿颖现男友
解读
以下关系的阅读方法，在人称关系：现老公、现老婆、前老婆、现女友、前男友、哥哥、堂哥、表姐、音乐制作人的前面加上“的”，后面加
上“是”。就可以读通顺了。
周迅 前男友 窦鹏 堂哥 窦唯 前老婆 王菲
周迅 前男友 宋宁 表姐 高原 现老公 窦唯 前老婆 王菲
周迅 前男友 李亚鹏 现老婆 王菲
周迅 前男友 朴树 音乐制作人 张亚东 前老婆 窦颖 哥哥 窦唯 前老婆 王菲 音乐制作人 张亚东 现女友 瞿颖 前男友 李亚鹏 现老婆 王菲



类图练习2

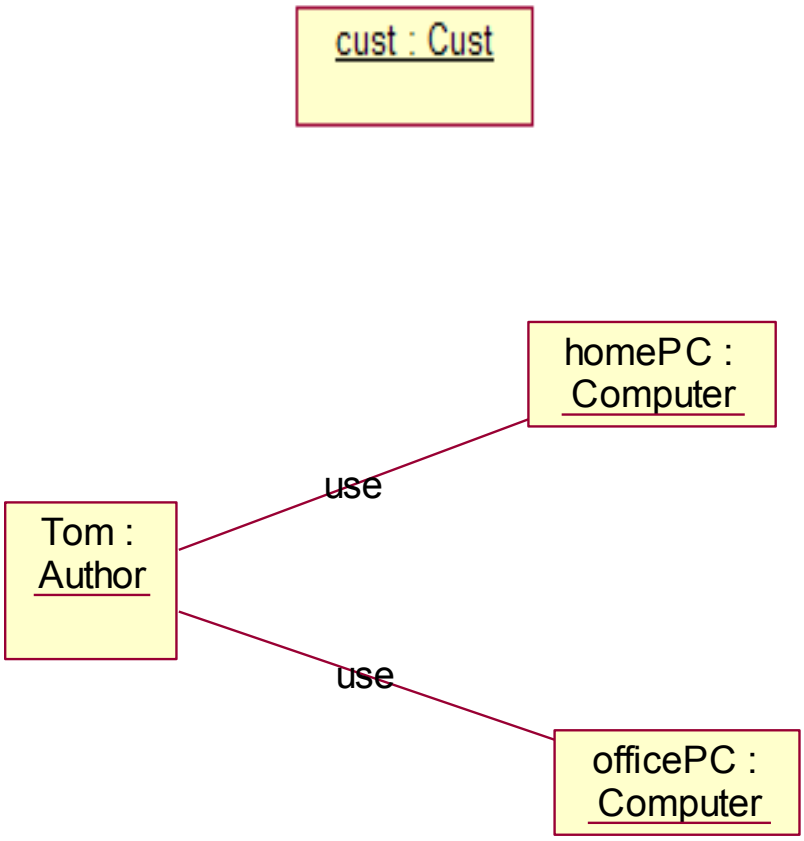
汽车和自行车都是交通工具(vehicle). 一辆自行车(bicycle)只归一个人(person)所有, 但一辆汽车(auto)可归一个人或两个人所有. 一个人可能没有自行车或汽车, 也可能有多辆自行车或汽车. 人分为男人(male)和女人(female). 每个人都有年龄(age)和名字(name). 每辆交通工具都有自己的颜色(color)和商标(brand). 每辆汽车都只有两个前灯(headlight)和一台发动机(motor)

5 对象图

1. 对象图是类图的一个实例, 用于显示系统执行时的一个可能的快照. 即在某一个时间上系统可能出现的样子. 对象图用带下划线的对象名称来表示对象.

2. 表现对象的特征

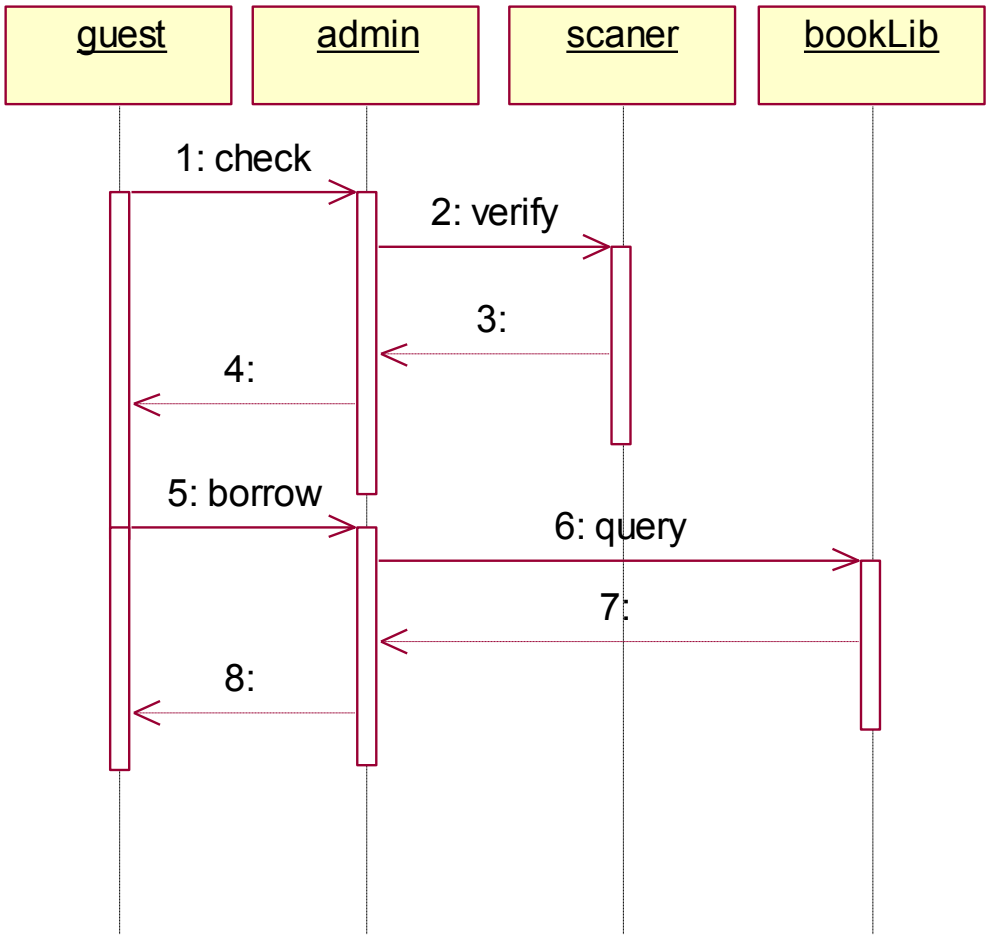
3. 对象图展现了多个对象的特征及对象之间的交互



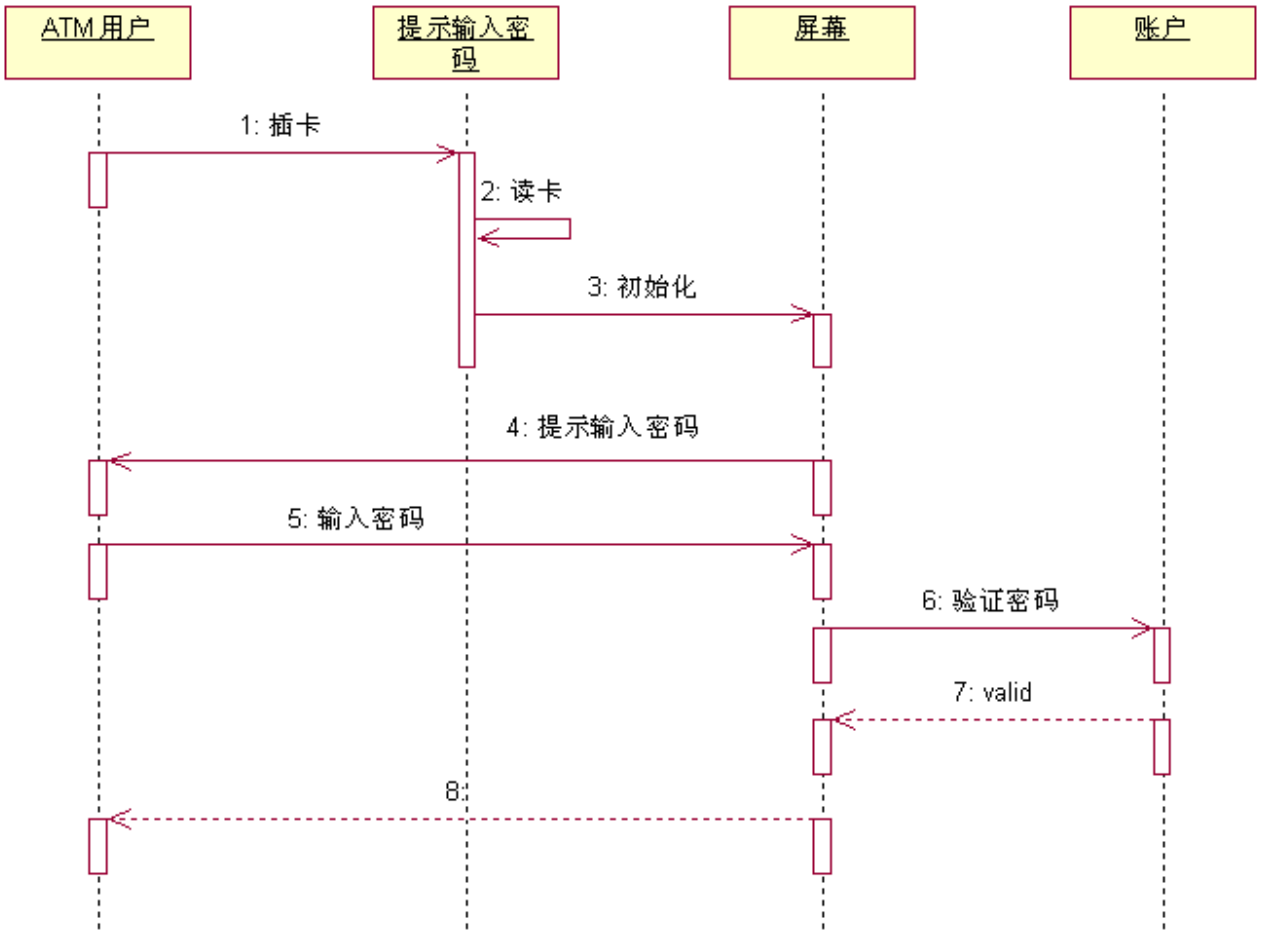
6 时序图

1. 时序图用于描述对象之间的传递消息的时间顺序, 即用例中的行为顺序.
2. 当执行一个用例时, 时序图中的每条消息对应了一个类操作或者引起转换的触发事件.
3. 在 UML 中, 时序图表示为一个二维的关系图, 其中, 纵轴是时间轴, 时间延竖线向下延伸. 横轴代表在协作中各个独立的对象. 当对象存在时, 生命线用一条虚线表示, 消息用从一个对象的生命线到另一个对象的生命线的箭头表示. 箭头以时间的顺序在图中上下排列.

6.1 借书时序图



6.2 ATM 用户成功登陆的时序图



6.3 时序图中的基本概念

对象: 时序图中对象使用矩形表示, 并且对象名称下有下划线. 将对象置于时序图的顶部说明在交互开始时对象就已经存在了. 如果对象的位置不在顶部, 表示对象是在交互的过程中被创建的.

生命线: 生命线是一条垂直的虚线. 表示时序图中的对象在一段生命周期内的存在. 每个对象底部中心的位置都带有生命线.

消息: 两个对象之间的单路通信. 从发送方指向接收方. 在时序图中很少使用返回消息.

激活: 时序图可以描述对象的激活和钝化. 激活表示该对象被占用已完成某个任务. 钝化指对象处于空闲状态, 等待消息. 在 UML 中, 对象的激活时将对象的生命线拓宽为矩形来表示的. 矩形称为计划条或控制期. 对象就是在激活条的顶部被激活的. 对象在完成自己的工作后被钝化.

对象的创建和销毁: 在时序图中, 对象的默认位置是在图的顶部. 这说明对象在交互开始之前就已经存在了. 如果对象是在交互过程中创建的, 那么就应该将对象放到中间部分. 如果要撤销一个对象, 在其生命线终止点处放置 “ X” 符号.

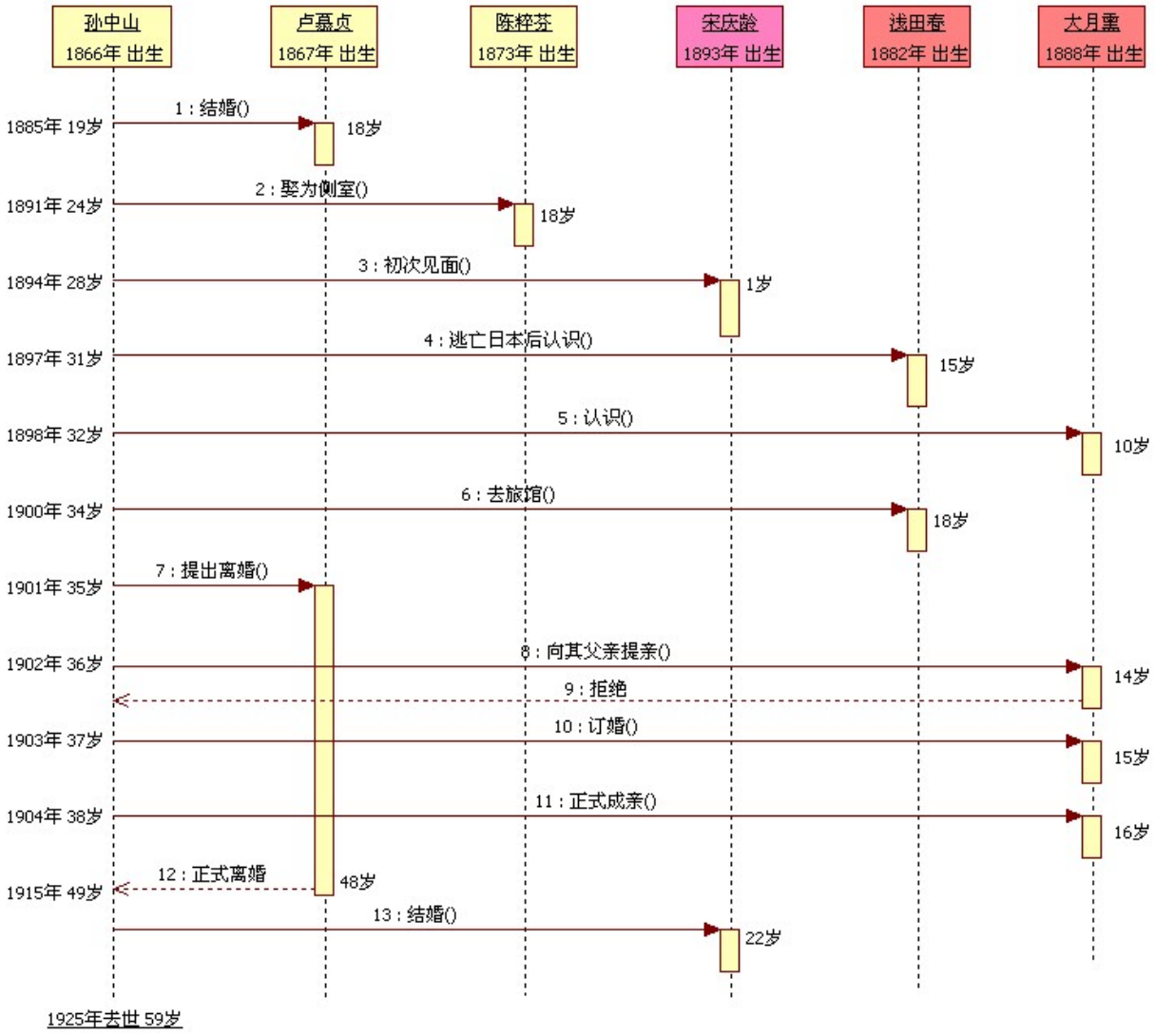
猜猜这个男人是谁？ ”

写道

有一个男人，他19岁娶了18岁的女友、
24岁時和只有18岁的秘书交往并結婚
28岁见到1岁的女嬰，开始光源氏计划、
在31岁到日本旅行认识一名15岁的女仆，
隔年认识10岁的萝莉，
在日本旅行期间就周旋于女仆和萝莉之间，
38岁和萝莉結婚、39岁回到中國，
49岁光源氏计划成功，把22岁的小妹妹带回家，
后來活到59岁死亡。
请问这人生的贏家是哪个历史人物？

不管是坛子里，还是QQ群里，这几天很流行这个问题[见天涯帖]，答案即刻揭晓：

孙中山先生
1885年 19岁 与卢慕贞(18岁)结婚，后育有三子
1891年 24岁 认识陈粹芬(18岁)，后成为侧室
1894年 28岁 初次见到宋庆龄(1岁岁女嬰)开始光源氏计划
1897年 31岁 流亡日本，认识浅田春(15岁女仆)
1898年 32岁 认识大月熏(10岁罗莉)
1900年 34岁 9月20日上午在神户市相生町加藤旅馆跟浅田春(18岁)...
1901年 35岁 向卢慕贞(34岁)提出离婚(当时似乎还没正式离婚
1902年 36岁 向大月熏(14岁)父亲提亲被拒绝
1903年 37岁 8月与大月熏(15岁)订婚
1904年 38岁 7月19日与大月熏(16岁)正式成亲
1915年 49岁 与卢慕贞(48岁)正式离婚 与宋庆龄(22岁)结婚

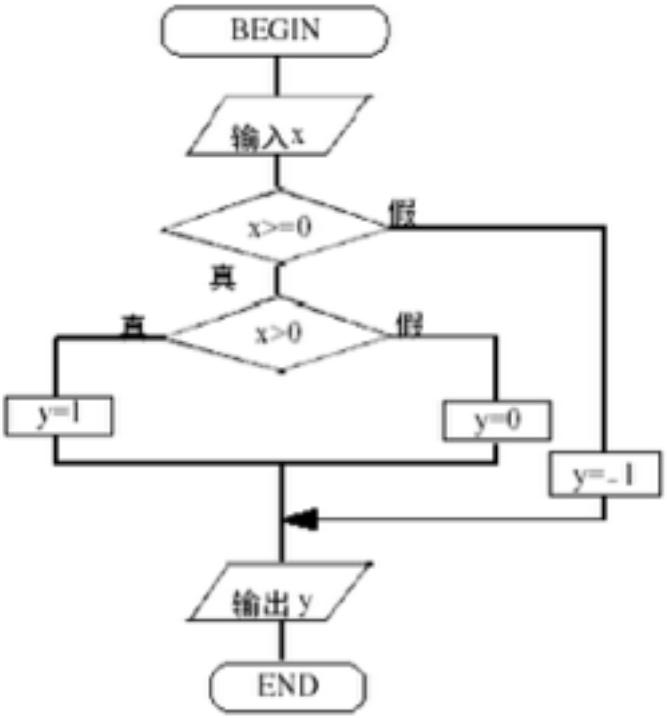


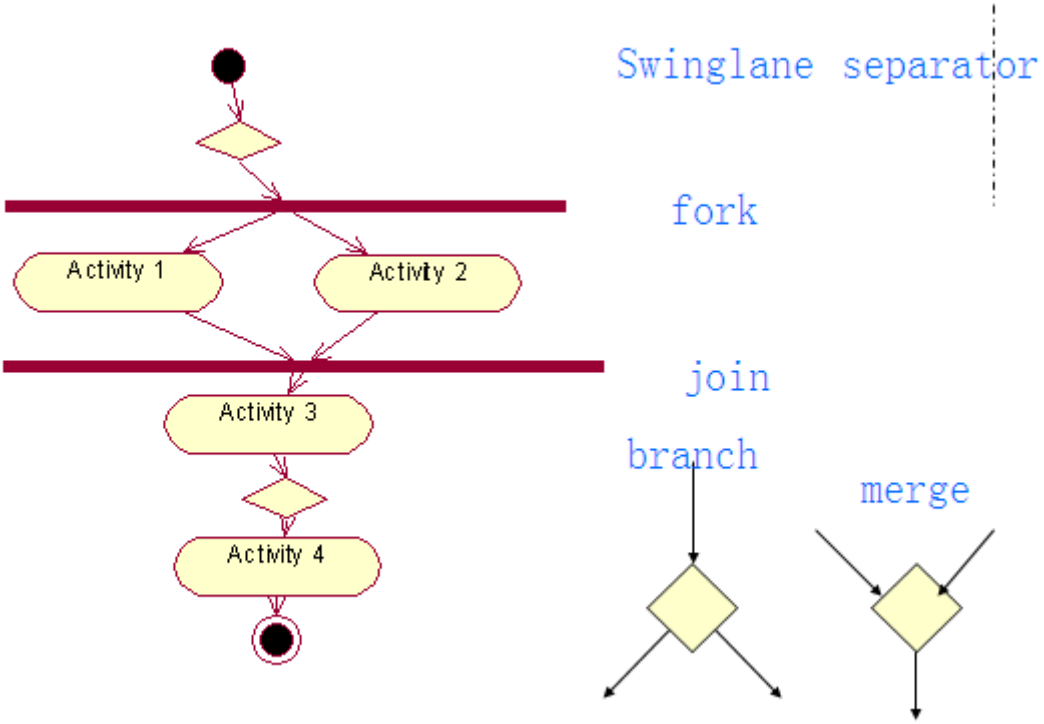
时序图练习

画出三层架构成功登陆的时序图

7 活动图

在 UML 中, 活动图本质上就是流程图. 它用于描述系统的活动, 判定点和分支等.





7.1 活动图中的基本概念

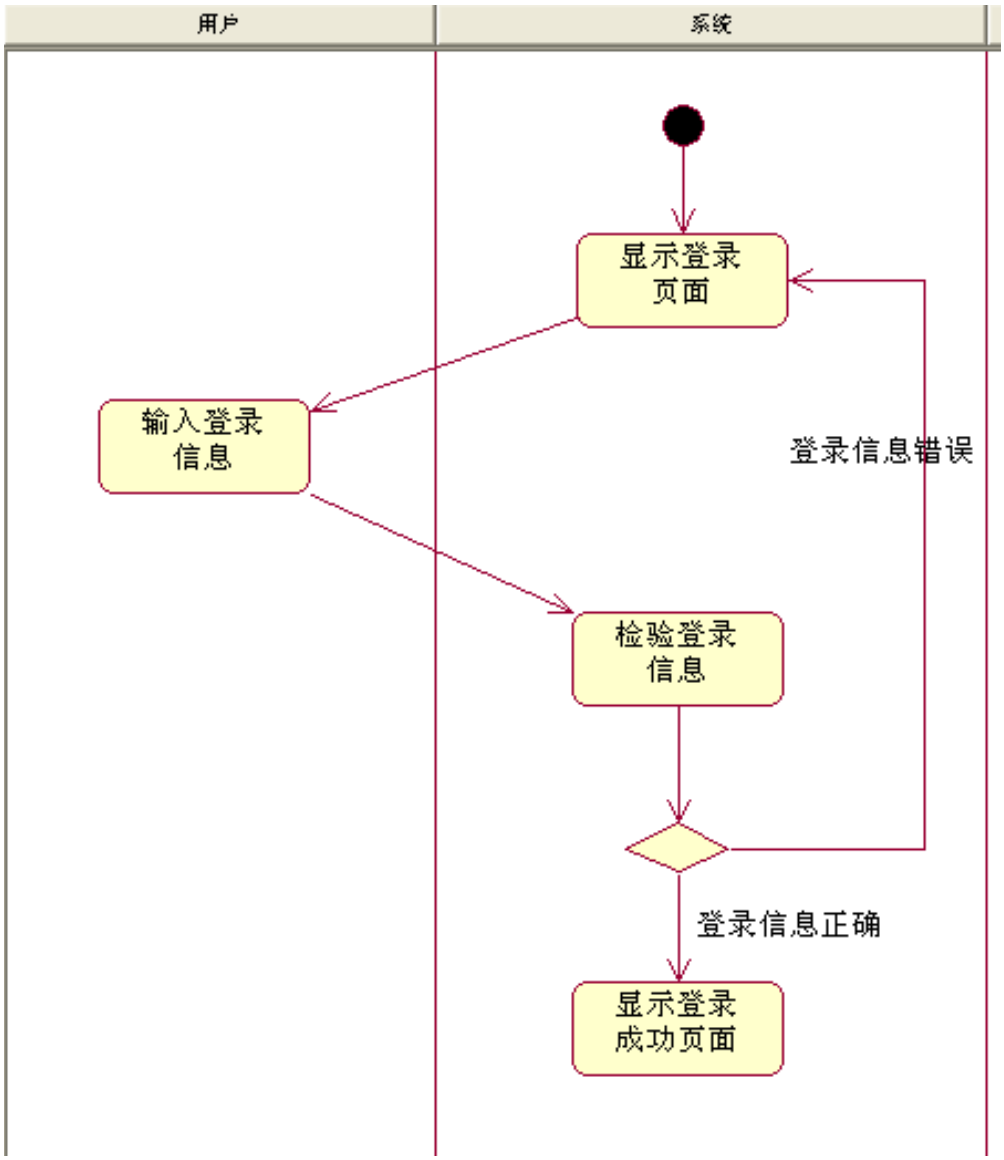
动作状态: 原子的, 不可中断的动作, 并在此动作完成之后向另一个动作转变. 在 UML 中动作状态用圆角矩形表示, 动作状态所表示的动作写在圆角矩形内部

分支与合并: 分支在软件系统中很常见. 一般用于表示对象类所具有的条件行为. 用一个布尔型表达式的真假来判定动作的流向. 条件行为用分支和合并表达.在活动图中, 分支用空心小菱形表示. 分支包括一个入转换和两个带条件的出转换, 出转换的条件应该是互斥的, 须保证只有一条出转换能够被触发. 合并包含两个带条件的入转换和一个出转换.

分叉与汇合: 分叉用来描述并发线程, 每个分叉可以有一个输入转换和两个或多个输出转换. 每个转换都可以是独立的控制流. 汇合代表两个或多个并发控制流同步发生, 当所有的控制流都达到汇合点后, 控制才能继续往下进行. 每个汇合可以有两个或多个输入转换和一个输出转换. 在 UML 中分叉和汇合用一条粗直线表示

泳道: 泳道将活动图中的活动划分为若干组, 并将每一组指定给负责这组活动的业务组织. 泳道区分负责活动的对象, 明确地表示哪些活动是由哪些对象进行的. 每个活动指定明确地属于一个泳道. 在活动图中, 泳道用垂直实线绘出, 垂直线分隔的区域即为泳道

7.2 用户登录活动图

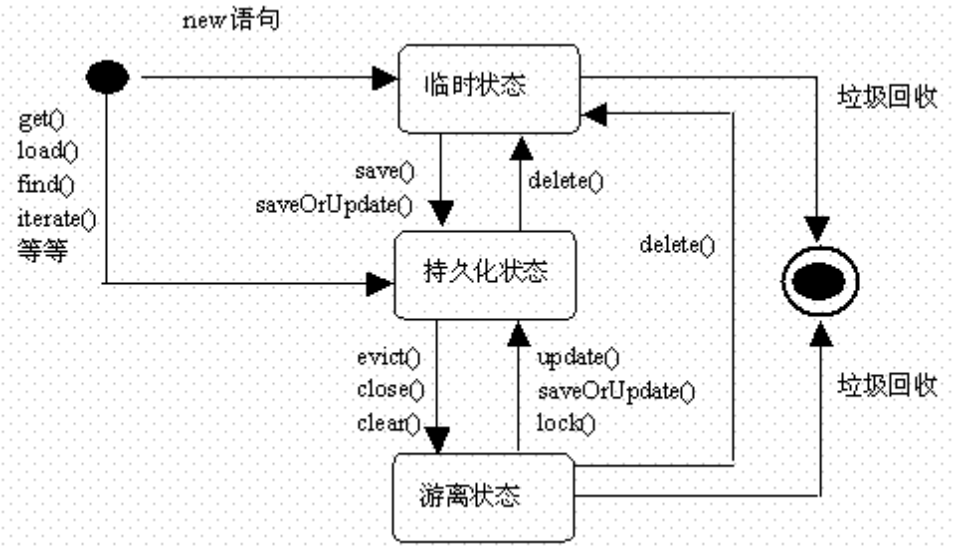


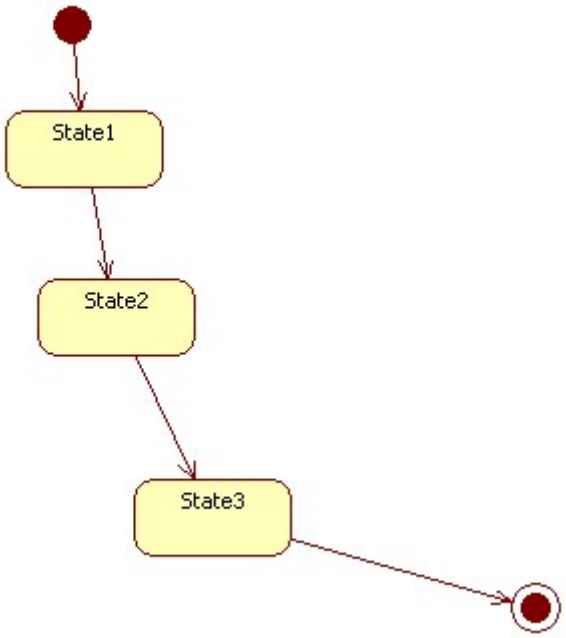
活动图练习

某公司销售人员接到订单后，将订单传给财务人员和仓库人员. 财务人员开具发票，并收款. 仓库人员准备货物，并查看是否货物加急，若是加急采用 EMS 方式发货，否则采用普通包裹方式发货. 完成之后由销售人员关闭该订单. 根据上面描述画出该公司销售过程的活动图.

8 状态图

状态图: 通过建立对象的生存周期模型来描述对象随时间变化的动态行为.





8.1 状态图中的基本概念

状态: 用圆角矩形表示. 状态名称表示状态的名字, 通常用字符串表示. 一个状态的名称在状态图所在的上下文中应该是唯一的.

转换: 用带箭头的直线表示. 一端连着源状态, 一端连着目标状态.

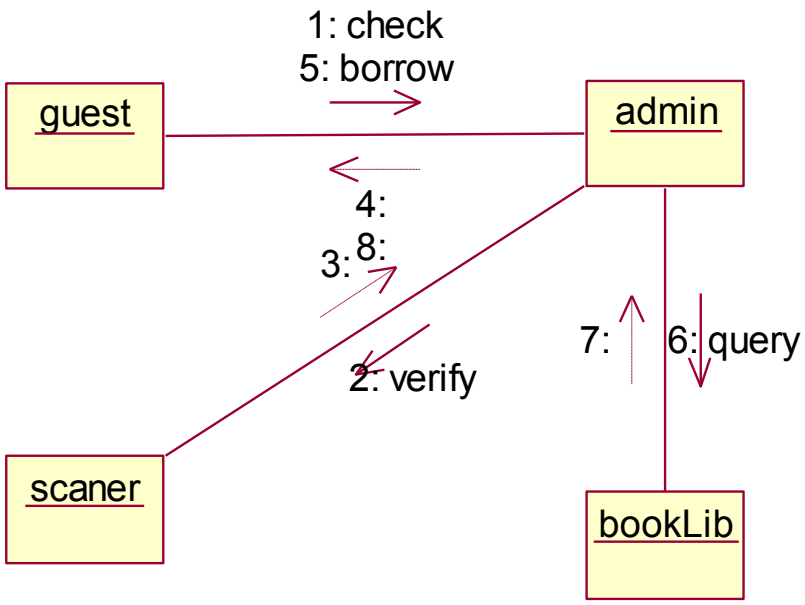
初始状态: 每个状态图都有一个初始状态. 此状态代表状态图的起始位置. 初始状态只能作为转换的源, 不能作为转换的目标, 并且在状态图中只能有一个. 初始状态用一个实心圆表示.

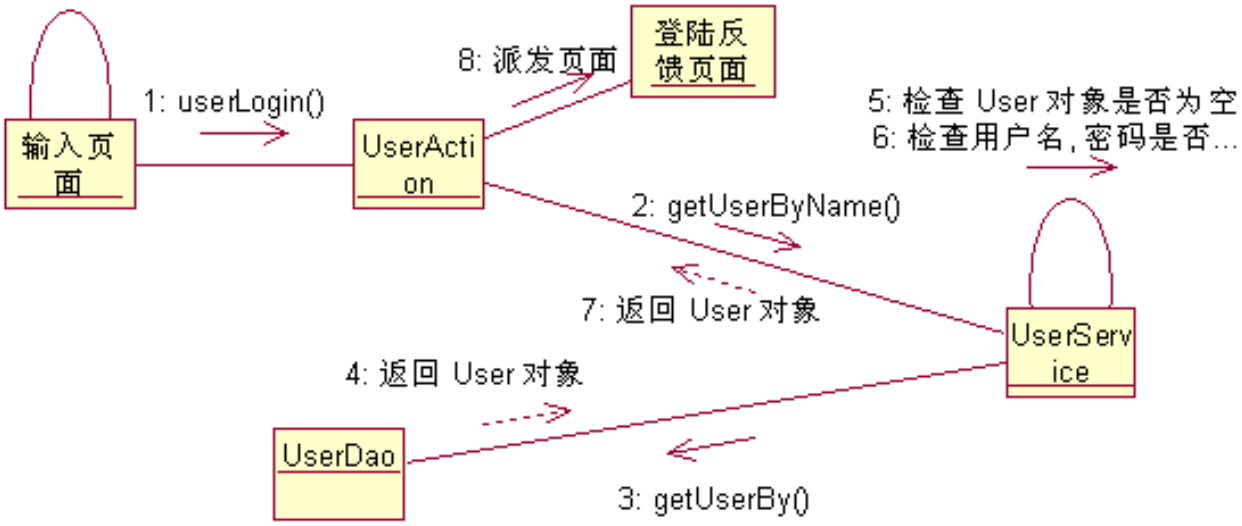
终止状态: 模型元素的最后状态, 是一个状态图的终止点. 终止状态在一个状态图中可以有多个.

9 协作图

1. 协作图(也叫合作图)是一种交互图.

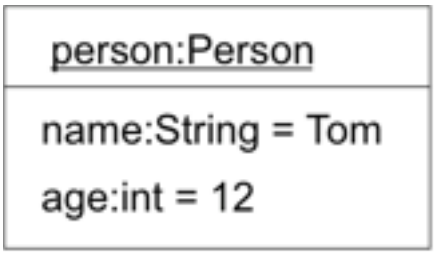
2. 时序图主要侧重于对象间消息传递在时间上的先后关系, 而协作图表达对象间的交互过程及对象间的关联关系。





10 对象图简介

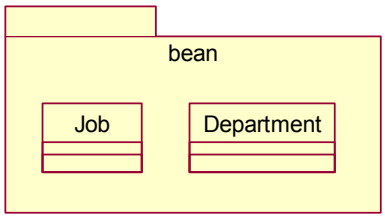
对象图是类图的一个实例, 用于显示系统执行时的一个可能的快照. 即在某一个时间上系统可能出现的样子. 对象图用带下划线的对象名称来表示对象.



11 包图简介

包图: 由包和包之间的关系组成. 包的图标就如同一个带标签的文件夹.

包提供了一种用于组织各种元素的分组机制. 在 UML 中, 包用来对元素进行分组, 并为这些元素提供命名空间.　包所拥有的或者引用的所有元素称为包的内容, 包没有实例.

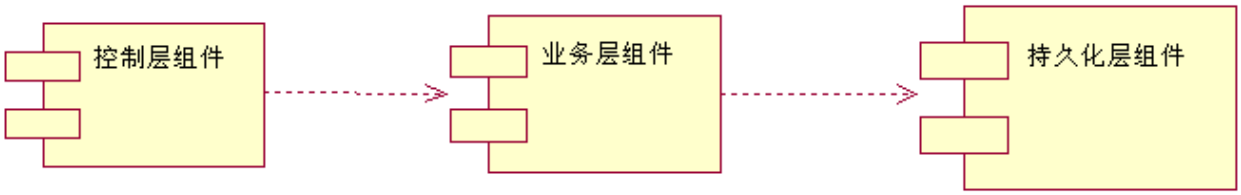


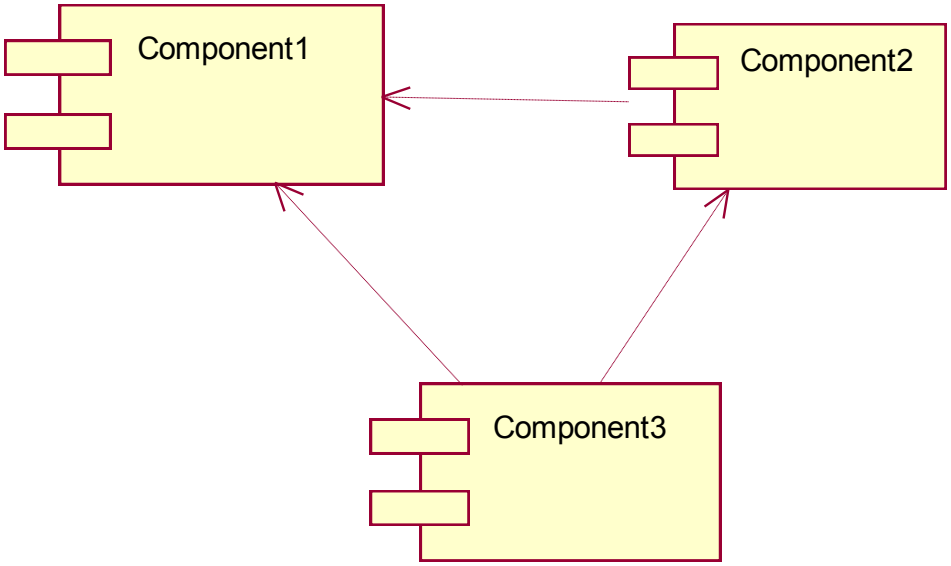
12 组件图简介

组件图用来建立系统中各组件之间的关系, 各组件通过功能组织在一起.

Javabean, ejb, jsp 都是组件。在UML中，组件使用在左侧有两个小矩形的大矩形来表示。

组件图可以用来设计系统的整体构架。



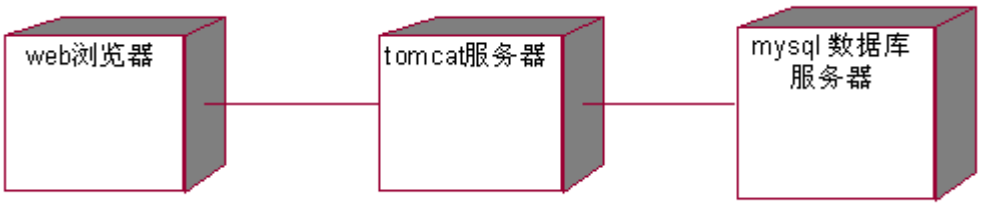


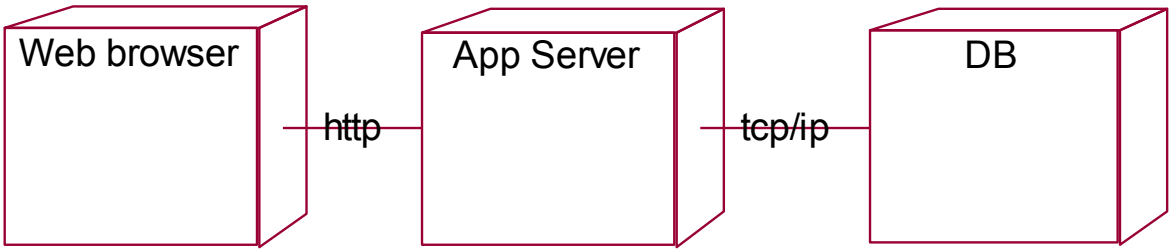
13 部署图简介

部署图用来帮助开发者了解软件中的各个组件驻留在什么硬件位置, 以及这些硬件之间的交互关系。

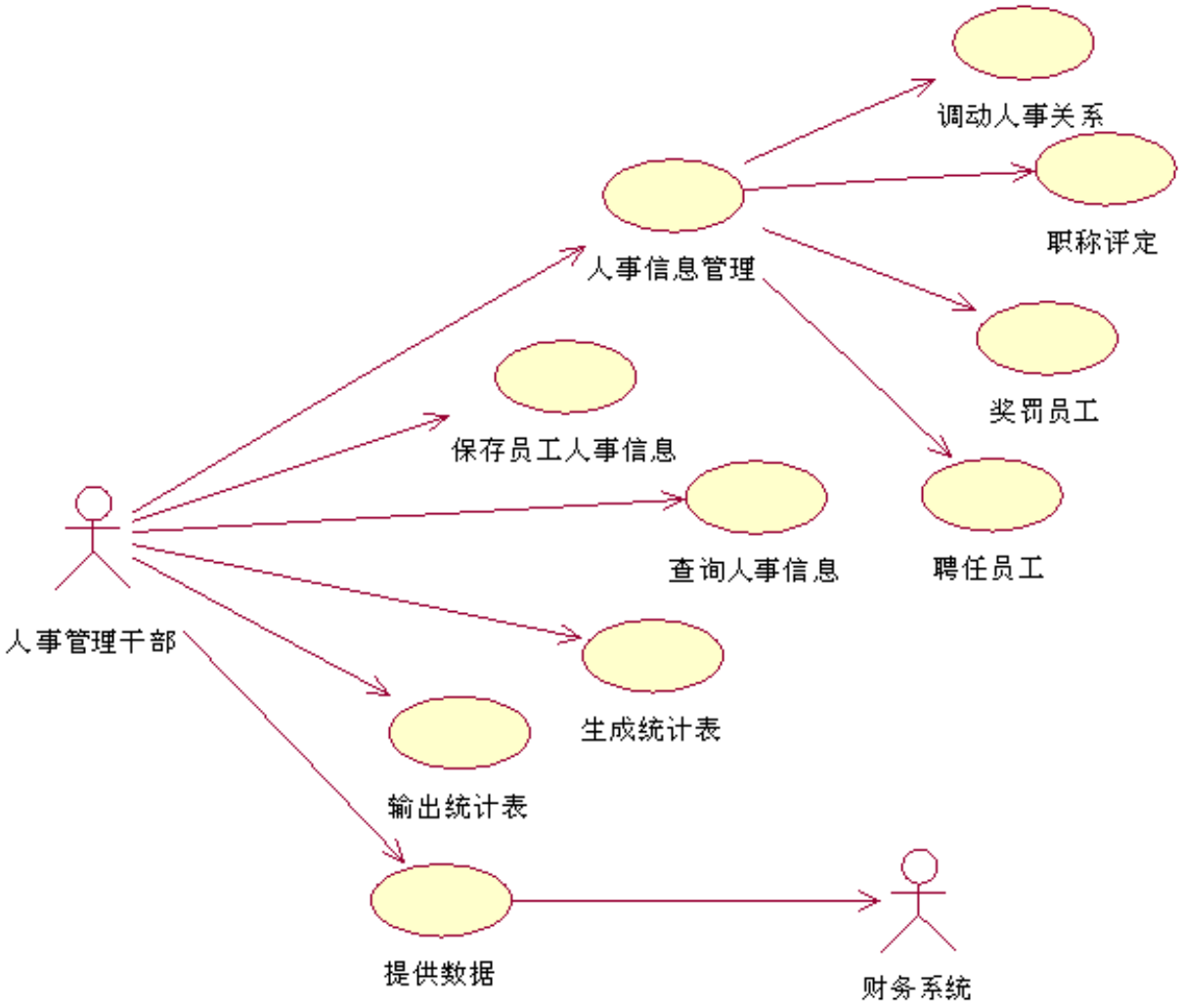
节点: 用来表示一种硬件, 可以是打印机, 计算机等.节点的标记符号是一个三维框，在框的左上方包含了节点的名称。

通信关联: 节点通过通信关联建立彼此的关系，采用从节点到节点绘制实线来表示关联。

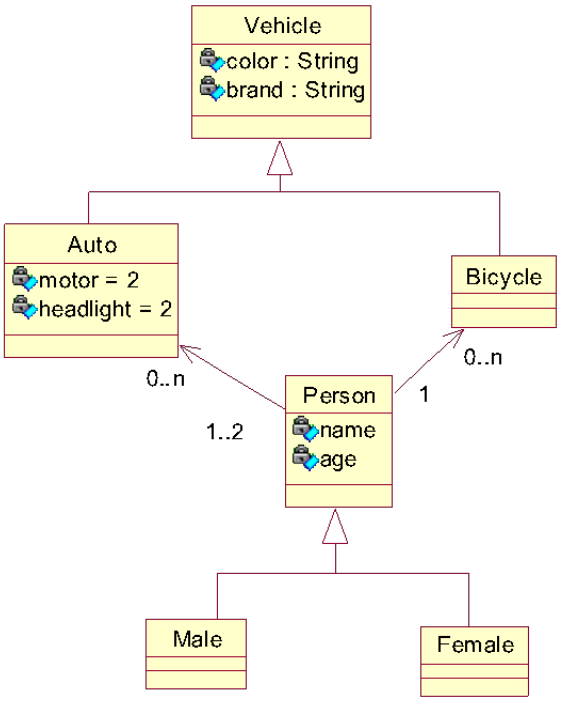




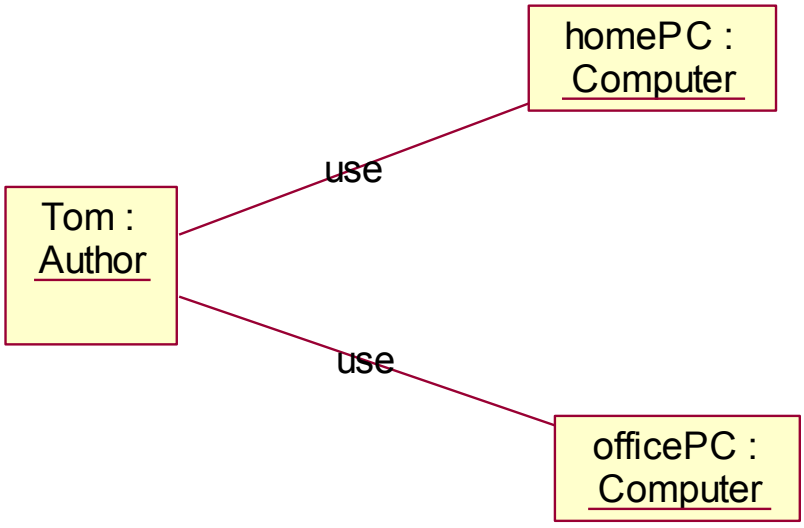
附录A
A.1 用例图


调动人事关系
职称评定
人事信息管理
奖罚员工
保存员工人事信息
查询人事信息
聘任员工
人事管理干部
生成统计表
输出统计表
提供数据
财务系统

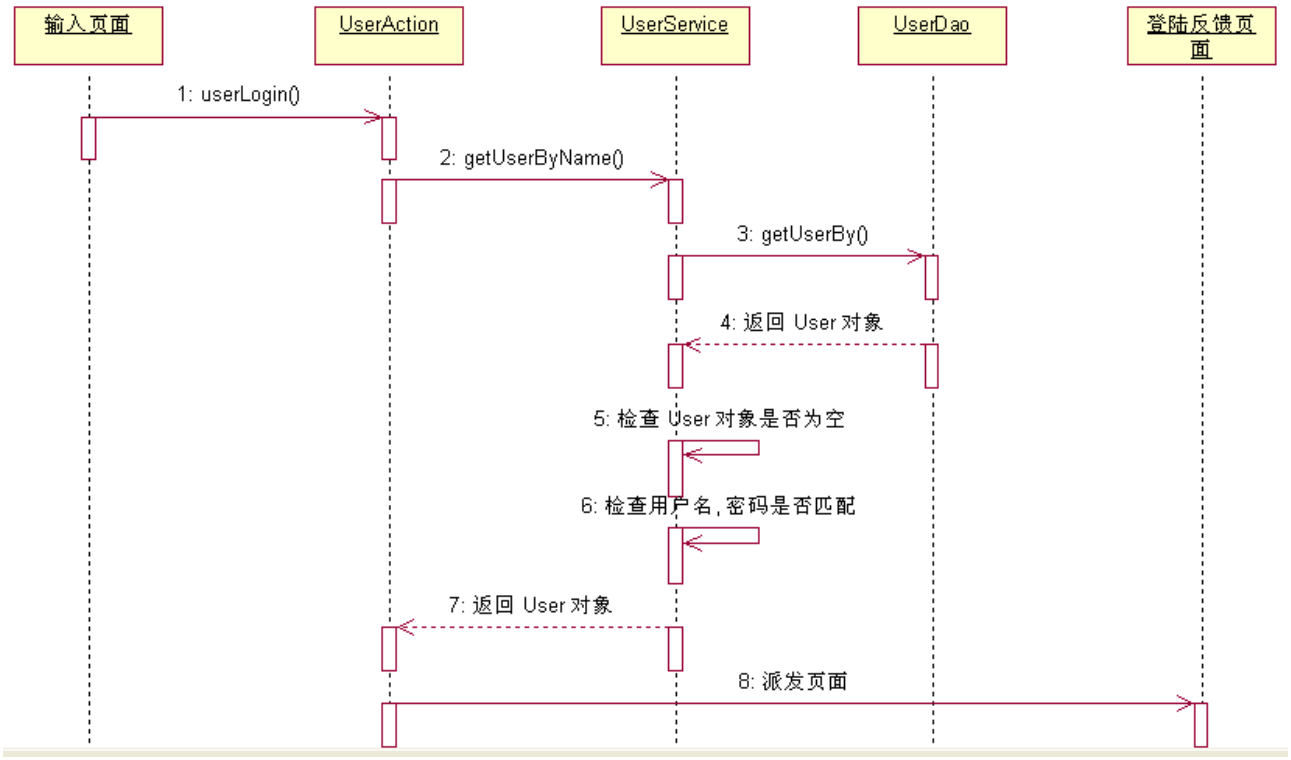
A.2 类图


Vehicle
color : String
brand : String
Auto
motor = 2
headlight = 2
Bicycle
0..n
0..n
1
1..2
Person
name
age
Male
Female

A.3 对象图


homePC :
Computer
use
Tom :
Author
use
officePC :
Computer

A.4 时序图


输入页面
UserAction
UserService
UserDao
登陆反馈页面
1: userLogin()
2: getUserByName()
3: getUserBy()
4: 返回 User 对象
5: 检查 User 对象是否为空
6: 检查用户名,密码是否匹配
7: 返回 User 对象
8: 派发页面

A.5 活动图