

软件工程课程实践辅导

基本概念和工具

一：面向对象的基本概念

什么是面向对象

面向对象 = 对象 (object)
+ 分类 (classification)
+ 继承 (inheritance)
+ 通过消息的通信 (communication
with messages)

对象 - 组成部分

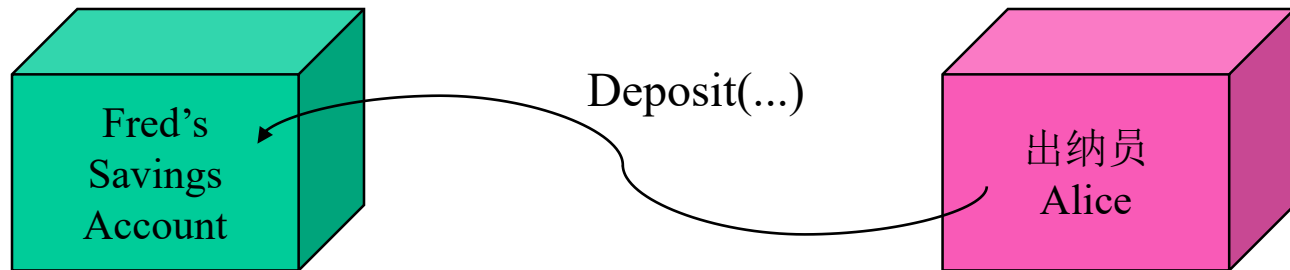
- 状态（state）— 对象的静态特性（属性）
- 行为（behavior）— 对象的动态特性
- OID: 对象具有惟一标识
 - 即使包含同样的数据，两个对象也是不同的。

```
Point point1 = new Point(0, 0);
```

```
Point point2 = new Point(0, 0);
```

对象 - 通讯

- 对象通过消息传递进行通讯
- 对象自己负责自己的行为



类—属性

**a named property of a class that describes a range of values
held by objects in a class**

- **name** (在类范围内惟一)
- **type**
- **visibility**: public (+), protected (#), private (-)
- **initial value** (optional)

类—操作

the implementation of a method for a class

- 由发送给对象的消息激发
- Visibility
- side effects
- polymorphic operation —
 - 一个操作具有不同的实现
- classoperation

File print

ASCII File ASCII File print method

Binary File Binary File print method

- dynamic binding — 运行时选择

Room	
-	roomDescription: String
-	roomid: int
-	roomName: String
-	roomPrice: double
-	roomState: int
+	getRoomDescription() : String
+	getRoomid() : int
+	getRoomName() : String
+	getRoomPrice() : double
+	getRoomState() : int
+	setRoomDescription(String) : void
+	setRoomid(int) : void
+	setRoomName(String) : void
+	setRoomPrice(double) : void
+	setRoomState(int) : void

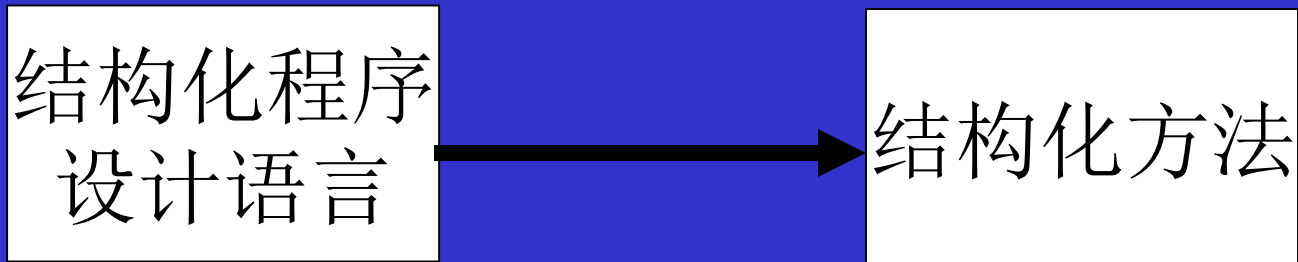
继承

- 继承：从一个祖先获得特性或特征
- 另一定义：在层次结构中共享属性和方法
- 继承的优点：
 - 允许一次性定义公共属性和服务（重用）
 - 同时，允许针对特殊情况特化和扩展那些属性和服务
 - 在需求分析的早期活动，OOA就使用继承来描述共性

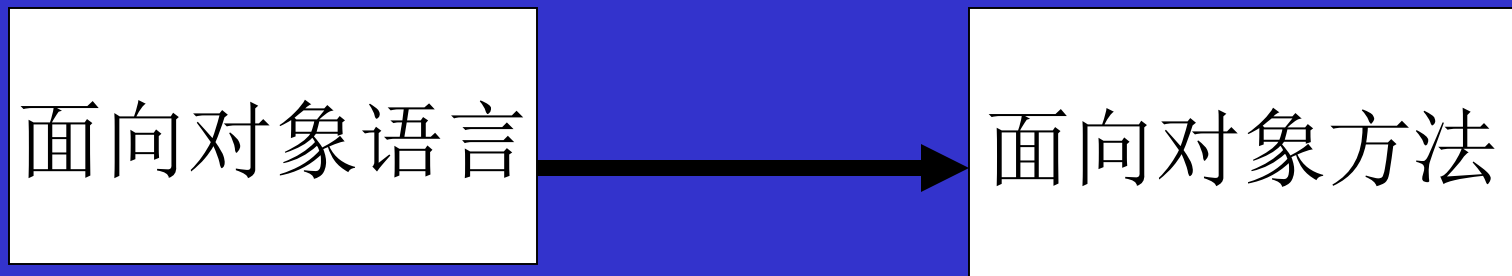
二：面向对象的开发方法

软件开发方法的演化

➤ 结构化开发方法



➤ 面向对象的软件开发方法



OO方法与结构化方法

- 结构化方法的两个鸿沟
 - DFD和ERD
 - 分析和设计
- 面向对象软件开发
 - 降低问题解决的复杂度
 - 无缝的开发过程
 - 可重用性
 - 易维护性

什么是UML?

- UML表示统一建模语言
(Unified Modeling Language)
 - 建模原则：准确、分层、分治、标准
- UML提供若干视图view（某个角度看系统）
 - 静态、设计、用例；状态机、活动、交互；部署；模型管理、剖面
 - 每个视图由若干幅图diagram描述（特定方面看）
 - 图由模型元素组成：类、用例、接口、包、节点、构件、注解、模型元素间互联的关系等
- UML主题域：
UML的视图分为四个主题域
 - 结构化structural、动态的dynamic、物理的physical、模型管理model management

主题域	视图——view	图——diagram
结构化	静态视图---static	类图---class
structure		内部结构---internal
	设计视图---design	协作图---collaboration
		构件图---component
	用例视图---use case	用例图---use case
动态的	状态机视图	状态图---state machine
dynamic	活动视图---activity	活动图---activity
	交互视图---interaction	时序图---sequence
		通信图---communication
物理的(phi)	部署视图---deployment	部署图---deployment
模型管理	模型管理视图	包图---package
model	剖面---profile	包图---package

UML基本元素

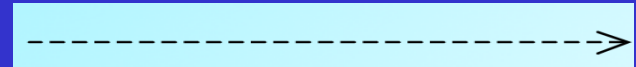
基本模型元素可分为四类：

- **结构模型元素** (structural things)
 - 类、接口、用例、构件、节点
- **行为模型元素** (behavioral things)
 - 交互、状态机
- **分组模型元素** (grouping things)
 - 包
- **注解元素** (annotational things)
 - 标注

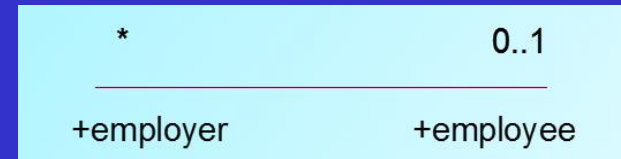
UML结构模型元素之间的联系

基本模型元素关系（relationship）：

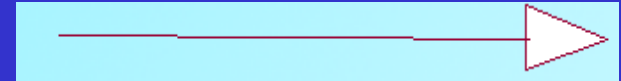
- 依赖关系(dependency)



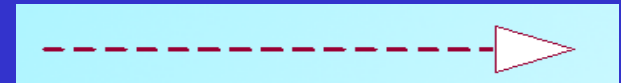
- 关联关系（association）



- 泛化关系(generalization)



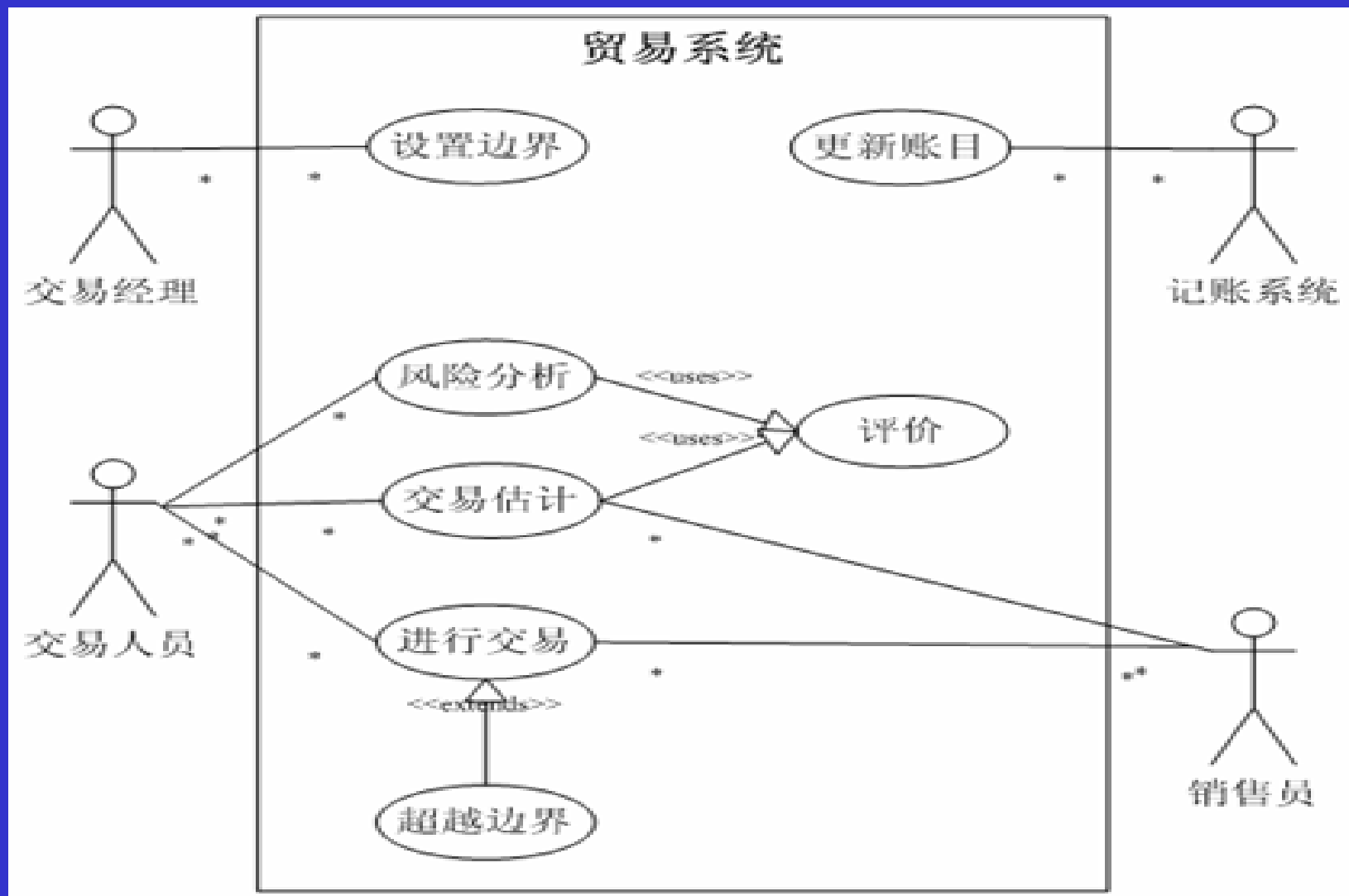
- 实现关系(realization)



Use Case图

- 是对一组动作序列的描述
- 包含系统与参与者的交互

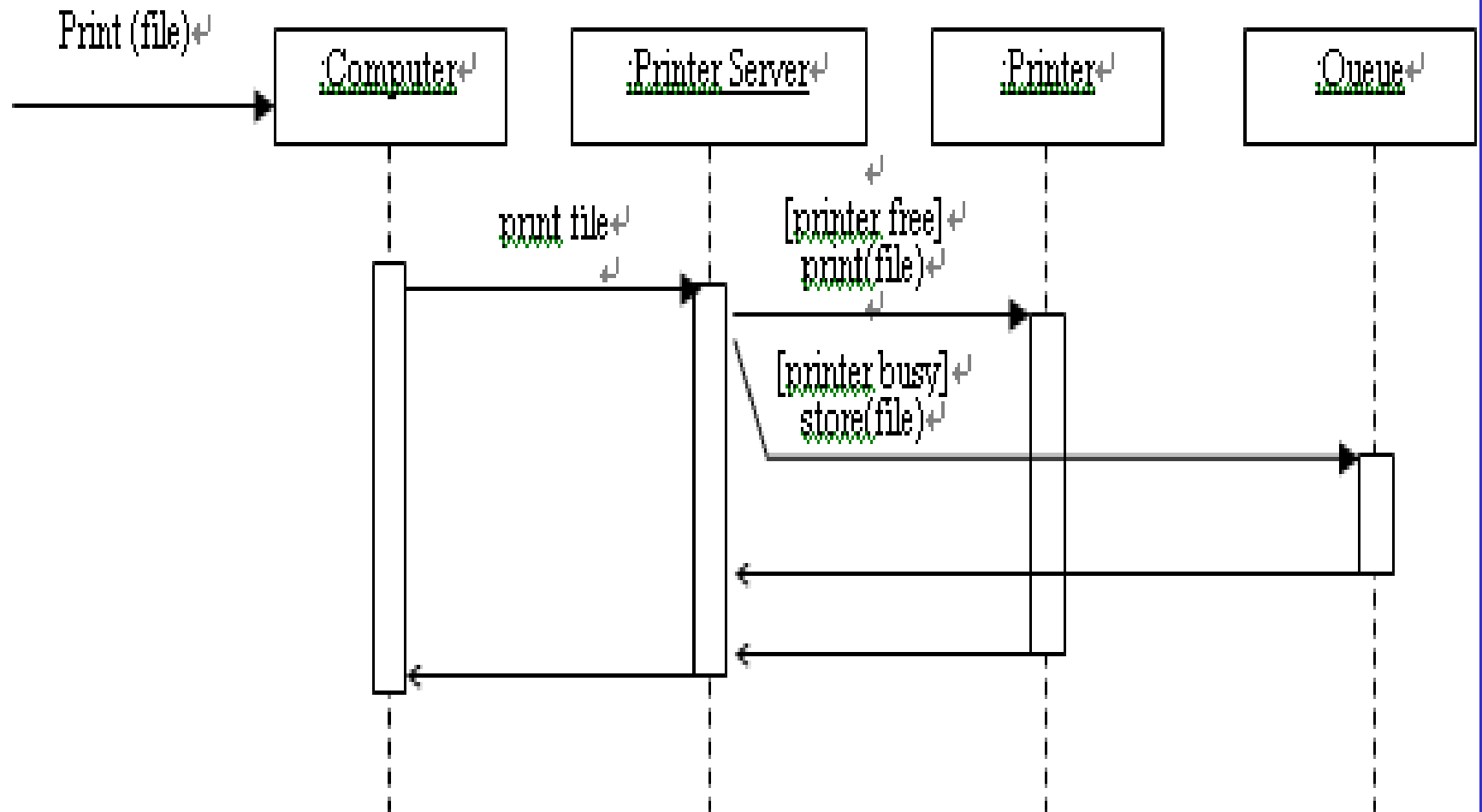
Use Case图



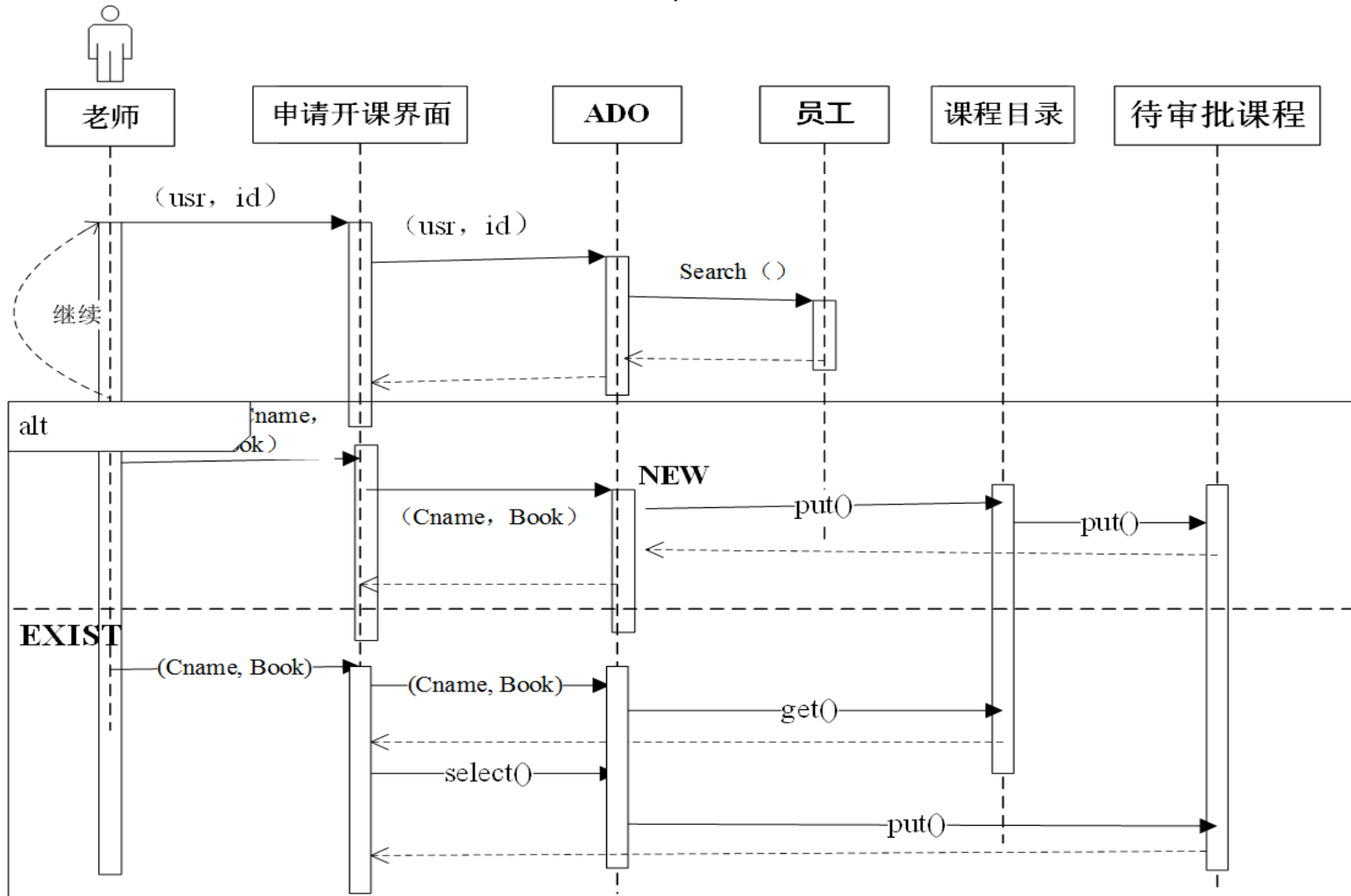
时序图

- 时序图用来描述对象之间的交互，即对象间消息的发送和接收的顺序
 - 时序图有两个坐标轴，垂直坐标表示时间（从上到下），水平坐标表示一组对象。

时序图



申请开课



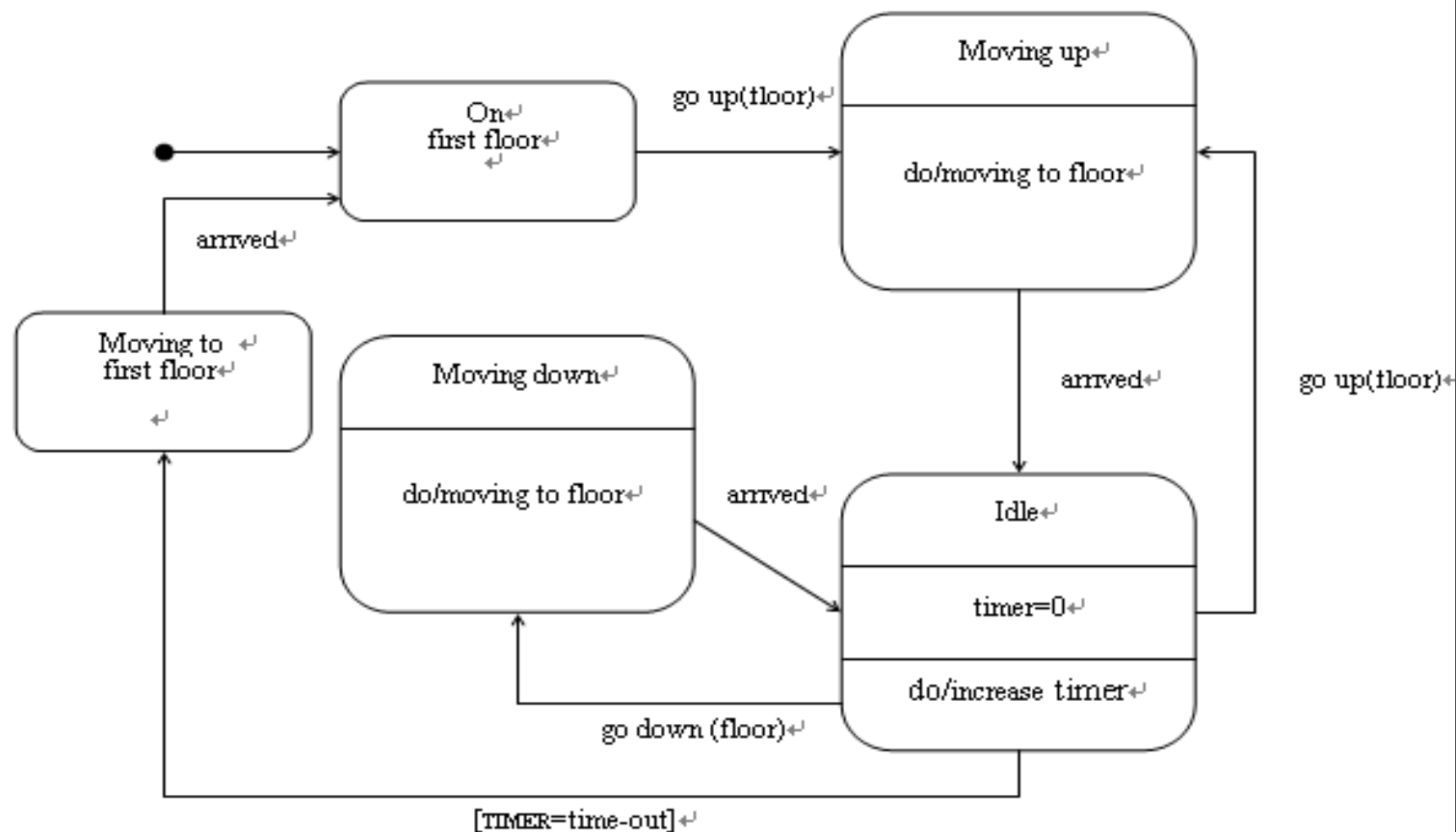
协作图

- 协作图用来描述对象之间的交互
 - 按时间顺序编号展示
- 协作图包括对象和对象间的连接

状态图

- 说明对象在它的生命期中响应事件所经历的状态序列以及它们对那些事件的响应
- 一个状态是指在对象的生命期中的一个条件或状况，在此期间对象将满足某些条件、执行某些活动或等待某些事件

状态图

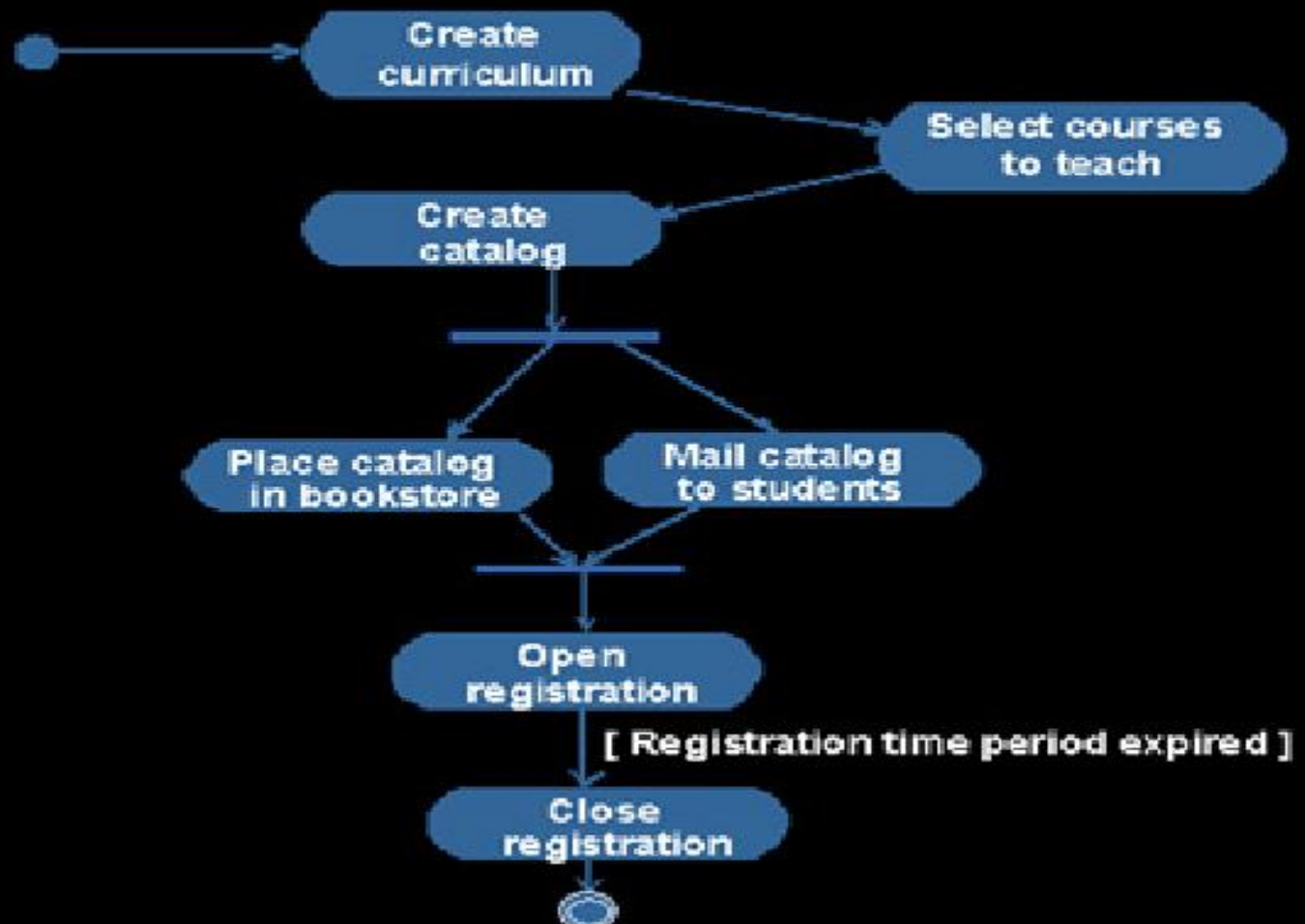


活动图

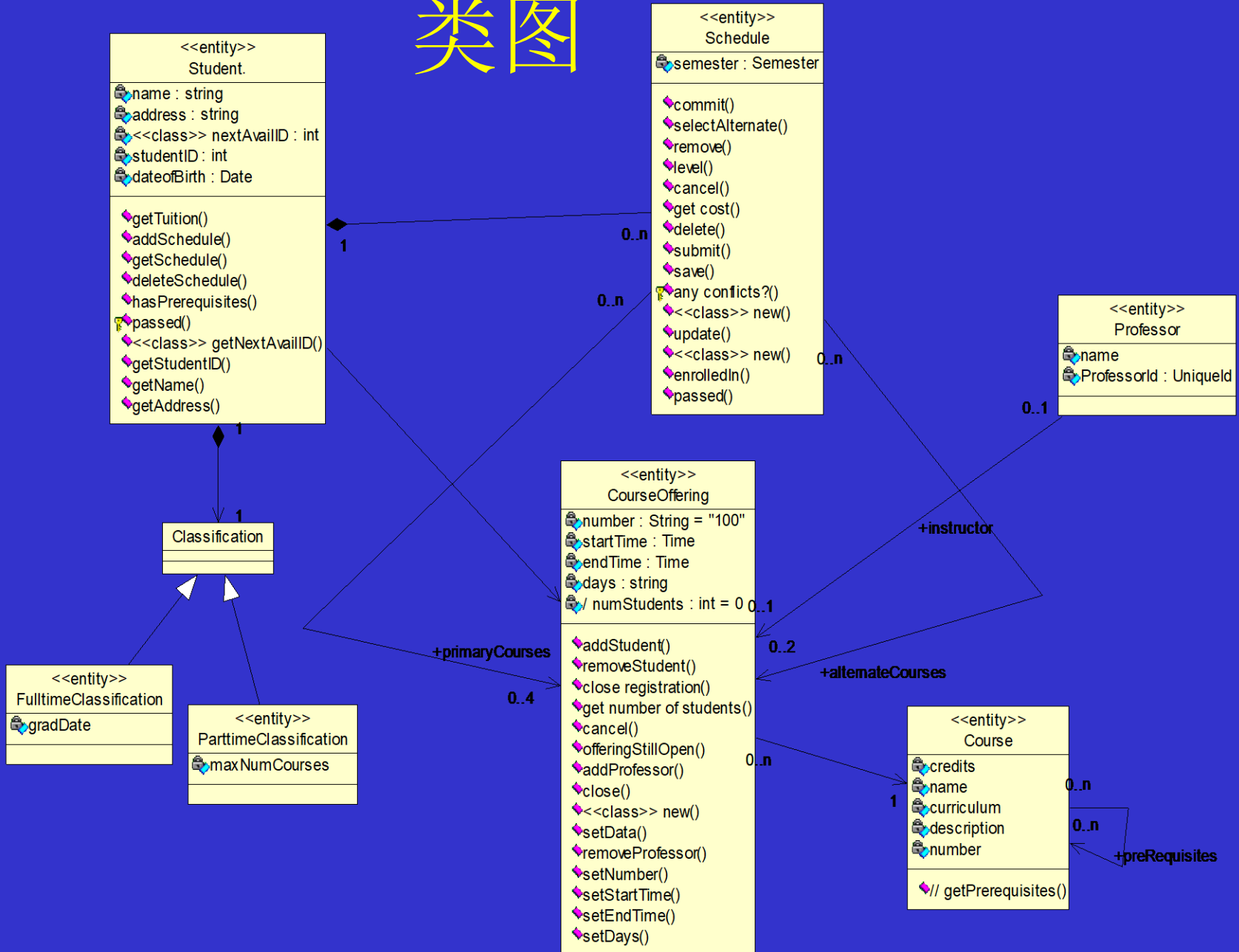
- 从本质上说，活动图是一个流程图，显示从一个活动到另一个活动的控制流
- 活动图是UML中用于对系统的动态方面建模的5种图的一种（其它的几种是时序图、协作图、状态图和用例图）

活动图

- ◆ Activity diagrams show flow of control



类图



三：面向对象的分析

分析面临的挑战

- 问题空间：
 - 问题空间越来越复杂
- 人与人之间的通讯
 - 分析人员之间的通讯
 - 与用户之间的通讯
 - 与管理人员、复审人员等之间的通讯
- 不断的变化
 - 需求一直处于不断的变化之中

OOA所采用的机制

面向对象的方法＝

对象（属性及其服务的封装）

＋ 分类

＋ 继承

＋ 多态

＋ 通过消息的通讯

面向对象分析的一般步骤（部分）

- 确定类
- 确定属性
- 确定操作
- 建立数据字典
- 确定关联
- 使用继承优化类结构
- 完善对象模型

确定类

- 可以从问题陈述着手，通常陈述中的名词或名词短语是可能的对象，它们以不同的形式展示出来，如：
 - 事物，它们和问题信息域的一部分；
 - 发生的事件或事情，它们出现在系统运行的环境中；
 - 角色（如：管理者、工程师、销售员），他们由与系统交互的人扮演；
 - 组织单位（如，部门、小组、小队）；
 - 场所（如，制造场所、装载码头）；
 -

确定类 (cont.)






- 通过上述分析，我们可以得到一些潜在的对象，但并非所有的潜在对象都会成为系统最终的对象。我们可以用以下选择特征来确定最终的对象：
 - 去掉冗余类，如用户和顾客；
 - 去掉模糊类；
 - 需要服务：潜在对象必须拥有一组可标识的操作，它们可以以某种方式修改对象属性的值；
 - 多个属性：在需求分析阶段，关注点应该是“较大的”信息（仅具有单个属性的对象在设计时可能可用，但在分析阶段，最好把它表示为另一对象的属性）；
 - 去掉相关于实现细节的类；

确定属性

- 属性是对象的性质，常用修饰性的名词词组来表示。
- 使用下面的方法去掉不正确的属性：
 - 若某个属性有独立存在的必要，且有自己的结构，则应该是类
 - 标识符
 -

准备数据字典

- 类字典：

CourseOffering	
	number : String = "100"
	startTime : Time
	endTime : Time
	days : Enum
	/ numStudents : Int

- 类名 - CourseOffering
- 类描述 - 某个课程的特定的设置，包括日期，时间，学时。
- 所包含的属性有：课程设置编号（ number ），开始时间，结束时间，学生数

确定关联

- 类之间的相互依赖即为关联，常用描述性动词或动词词组来表示
- 使用下面的方法去掉不必要的关联：
 - 若某个类已经被删除，则与之相关的关联也一并删除
 - 去掉冗余的关联
 - 去掉包含实现细节的关联，如并发性

确定操作

- 操作标识对象对外提供的服务，常用描述性动词或动词词组来表示
- 还通常使用：
 - 事件踪迹图或时序图，在对象交互中发现操作
 - 通过建模对象行为（状态图），来发现操作

四：UML类图

类图

- 类图说明类、接口、合作以及它们之间的关系
- 用于建模系统的静态结构
- 提供模型的骨架，是其它模型的基础

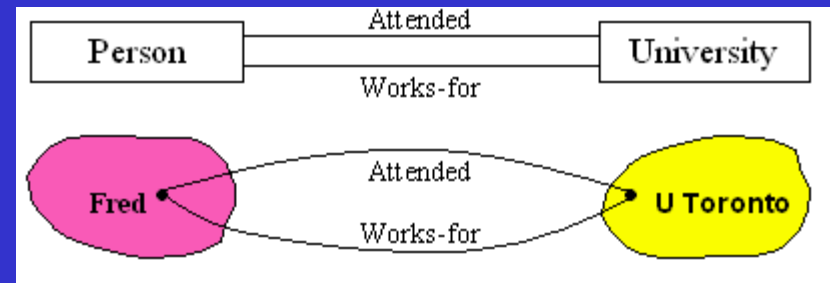
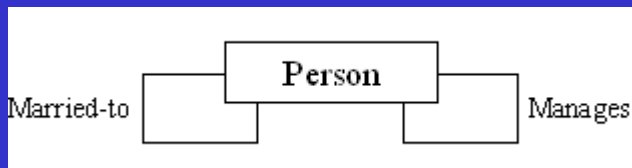
类

roomName ◊	房间名称 ◊
roomPrice ◊	房间价格 ◊
roomDescription ◊	房间描述 ◊
roomState ◊	房间状态 ◊

Room
<ul style="list-style-type: none">- roomDescription: String- roomid: int- roomName: String- roomPrice: double- roomState: int
<ul style="list-style-type: none">+ getRoomDescription() : String+ getRoomid() : int+ getRoomName() : String+ getRoomPrice() : double+ getRoomState() : int+ setRoomDescription(String) : void+ setRoomid(int) : void+ setRoomName(String) : void+ setRoomPrice(double) : void+ setRoomState(int) : void

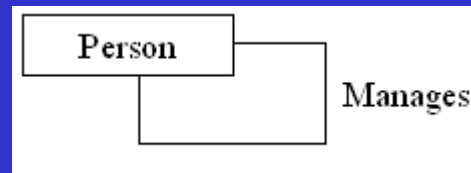
关联

- 两个类之间可以有多个关联
- 一个类的对象之间也可能存在关联

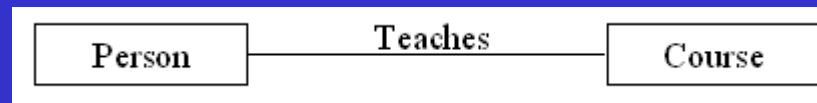


关联 - 度

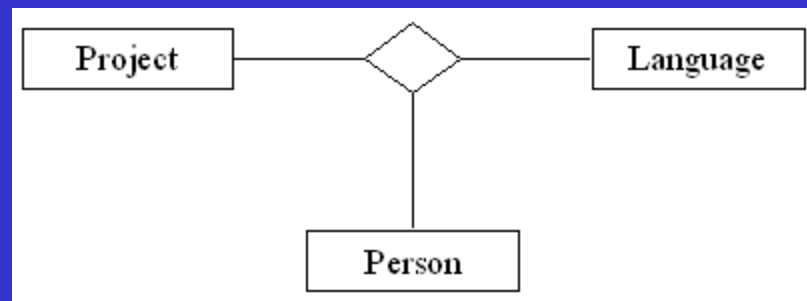
- Unary



- Binary

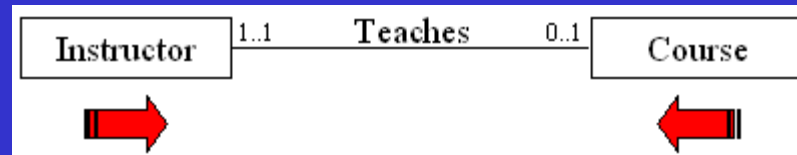


- Ternary



关联 - 多重性

- specifies restrictions on the number of objects in each class that may be related to objects in another class



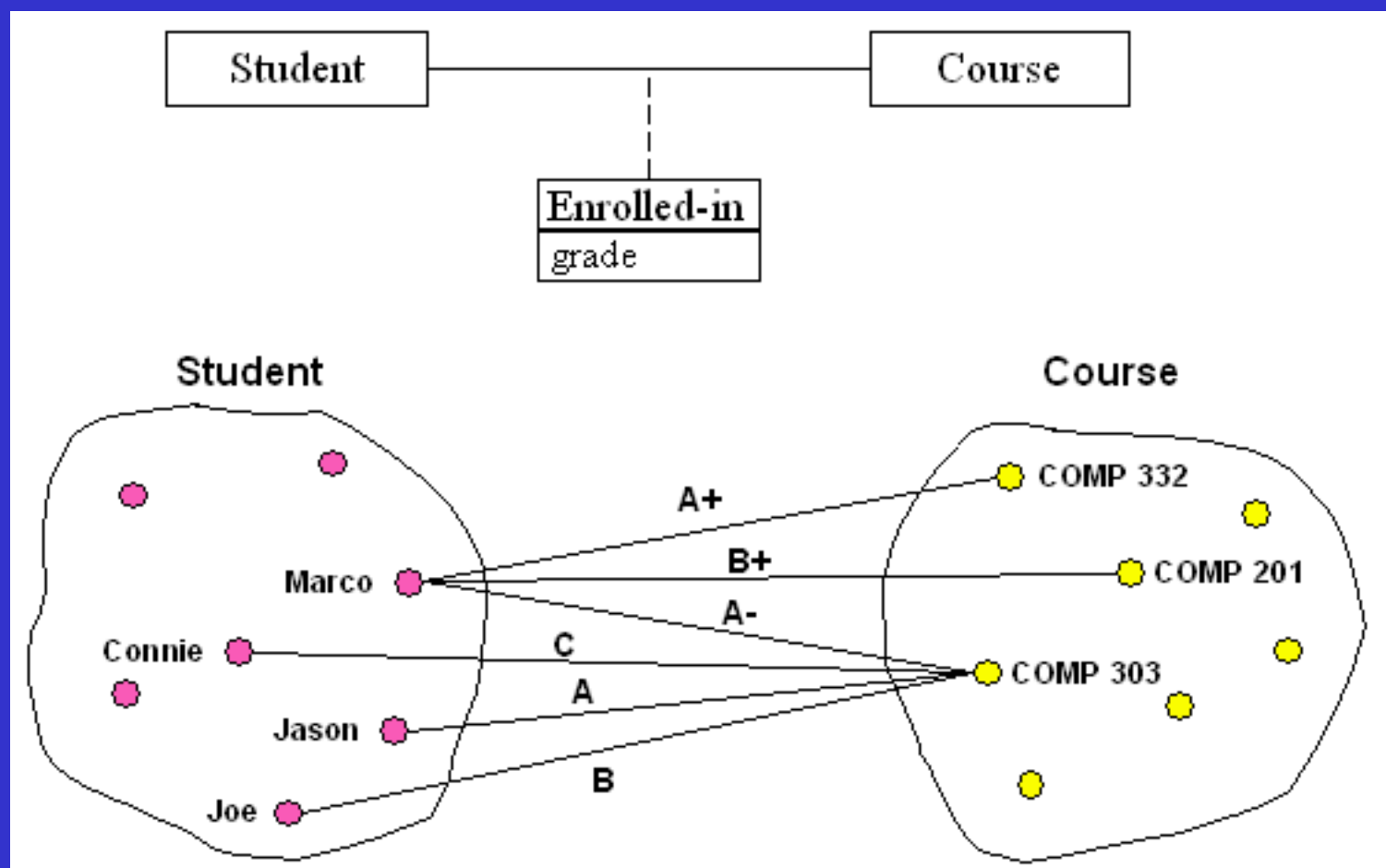
- “0..1”表示 “零或1”；
- “0..*”或 “*” 表示 “0或多”；
- “1..*”表示 “1或多”；
- “5..11”表示 “5至11”；
- “1, 3, 8”是枚举型，表示 “1或3或8”。
- 多重性的缺省值为1。

聚合与组装

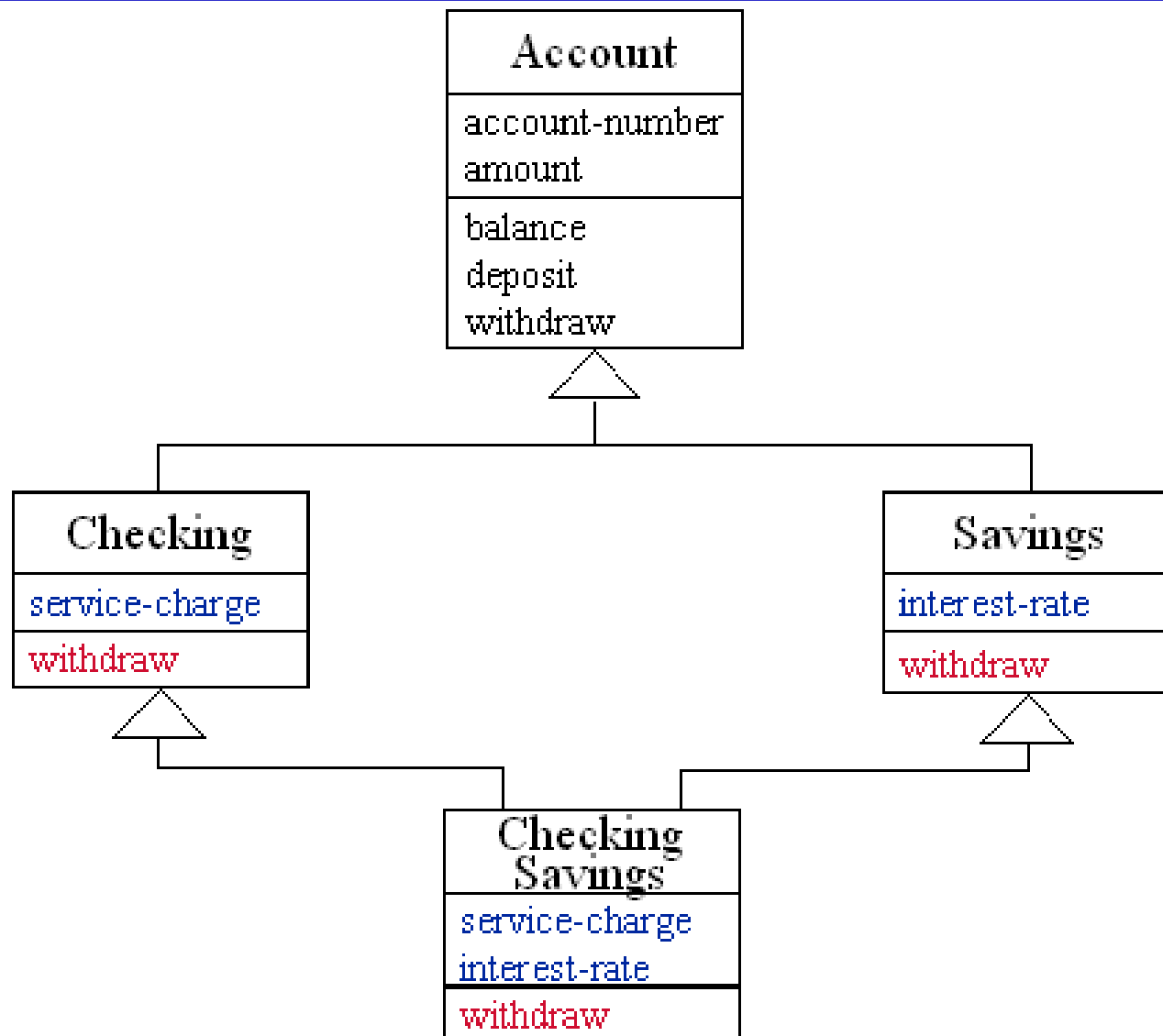
- 聚合 (aggregation) : 整体与部分关系
- 组装 (composition) : A Stronger form , 不可独立存在



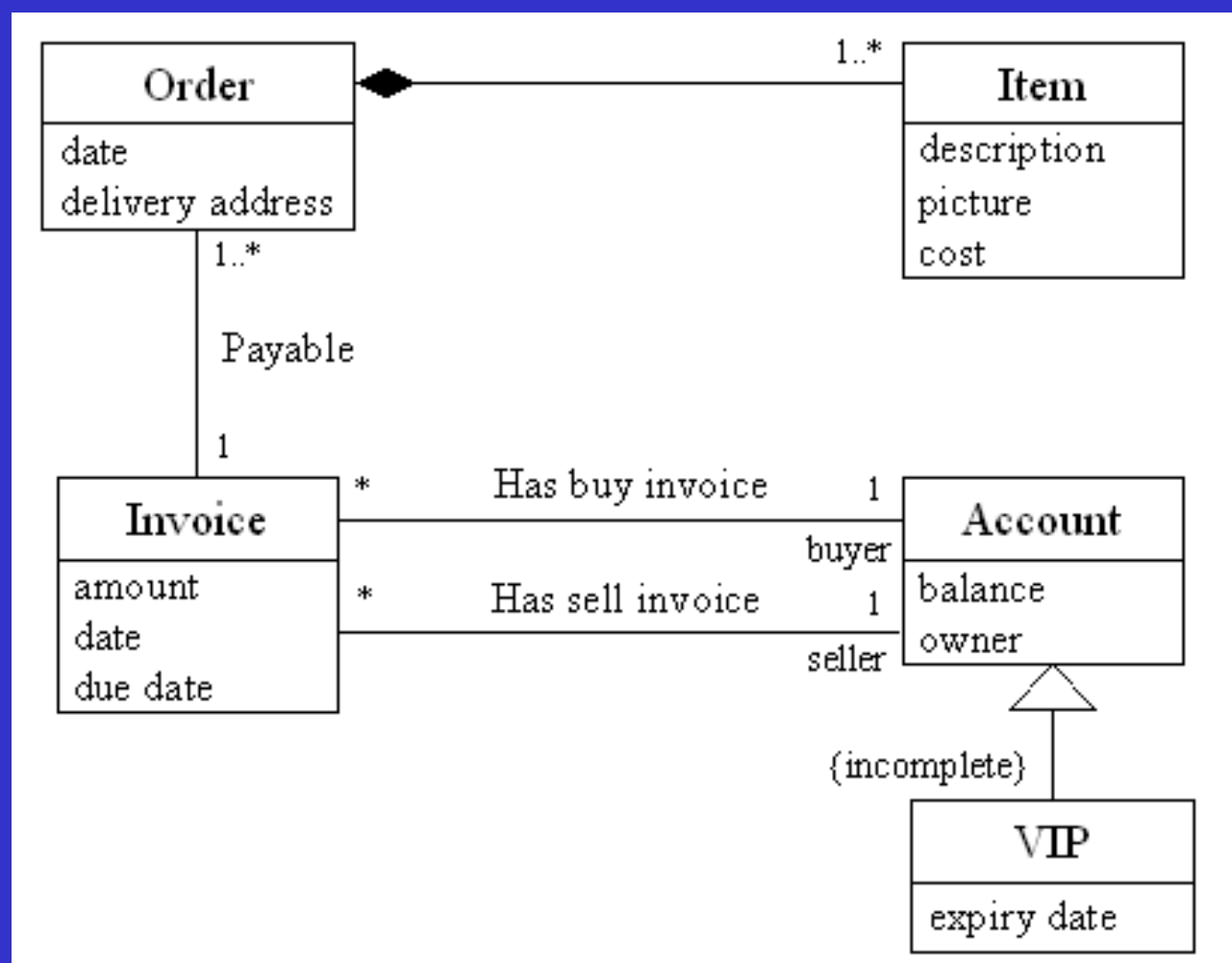
关联类



继承



UML类图一例子



包 (Package)

- 包：组织元素的常用机制
 - 将复杂问题分解的有效手段
 - 每一个包有一个与其它包不同的名字，构成一个名字空间

