
第一部分：类图

类图-概述

- 类图是软件的蓝图，用于详细描述系统内各个对象的相关的类，以及这些类之间的静态关系
- 类组件：类名、属性、方法
- 类间关系：六种关系
- 关系要素：名称、多重性表达式、导航符号、角色名称
- 设计类是已经完成了规格说明并且达到能够被实现程度的类

类图-概述

- 在分析中，只要尽量捕获系统需要的行为，而完全不必考虑如何去实现这些行为
- 在设计中，则必须准确地说明类是如何履行它们的职责
 - 完整的属性集合，包括详细说明的名称、类型、可视性和一些默认值
 - 将分析类指定的职责转化成一個或多个操作的完整集合

类图-类组件

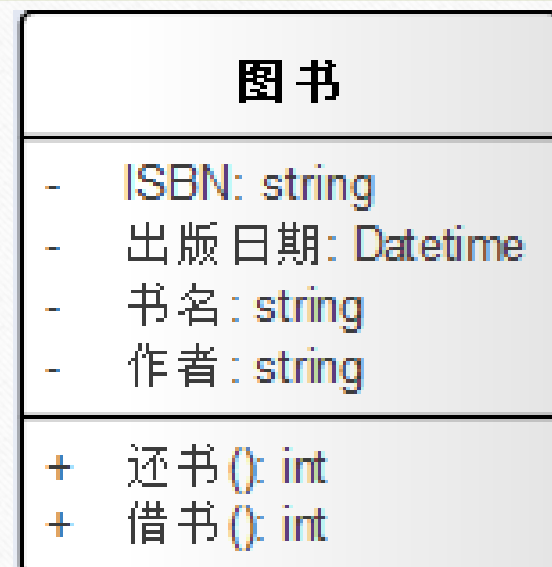
- **类名：** 类名

如果是抽象类需要用斜体字表示

- **类属性：** 可见性 名称：类型 [= 缺省值]

+(public) -(private) #(protected) ~ (package)

- **类方法：** 可见性 名称(参数列表) [: 返回类型]



某图书管理系统类图
中“图书”类示例

类图-类间关系-概述

- 依赖关系
- 关联关系
- 聚合关系
- 组合关系
- 泛化关系
- 实现关系(类与接口)



耦合度由低到高
实现和泛化相同

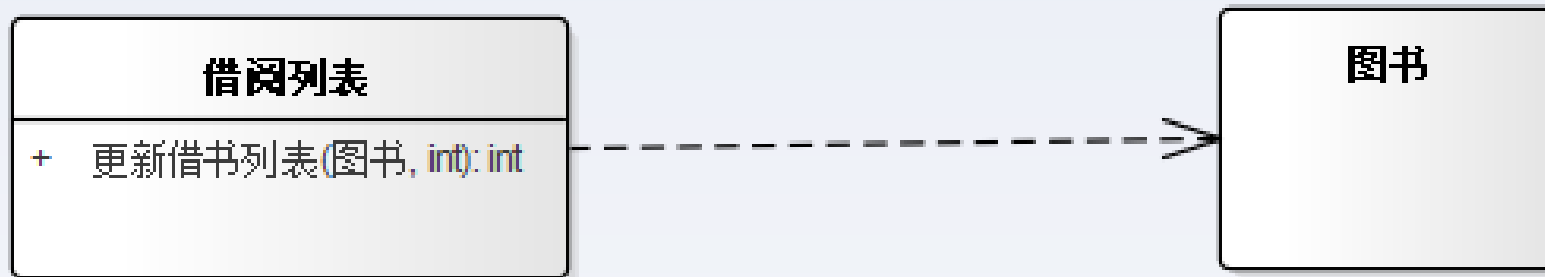
设计目标是最小化类间耦合!

类图-类间关系-依赖关系

依赖(Dependency)指出两个或多个模型语义上的关系，被依赖元素的改变会导致依赖元素的改变。

具体的依赖关系种类可参见教材P83页表5-2。

如何区分关联和依赖：某个类以成员变量的形式出现在另一个类中，二者是关联关系；



类图-类间关系-依赖关系

- 依赖关系是临时性的使用关系。
- 关联关系是“结构化”的关系，依赖关系是“非结构化”的，短暂的关系。
- 利用对象间的引用（即对象间互相了解的程度）来区分。
 - 属性引用：B对象作为A对象的某个属性(关联关系)
 - 参数引用：B对象作为A对象某个操作的参数 (依赖关系)
 - 局部声明引用：B对象作为A对象某个操作内部临时构造的对象 (依赖关系)
 - 全局引用：B对象是一个全局对象 (依赖关系)

类图-类间关系-关联关系

- 关联关系-单向关联(Association)

一个类知道另一个类的属性或方法

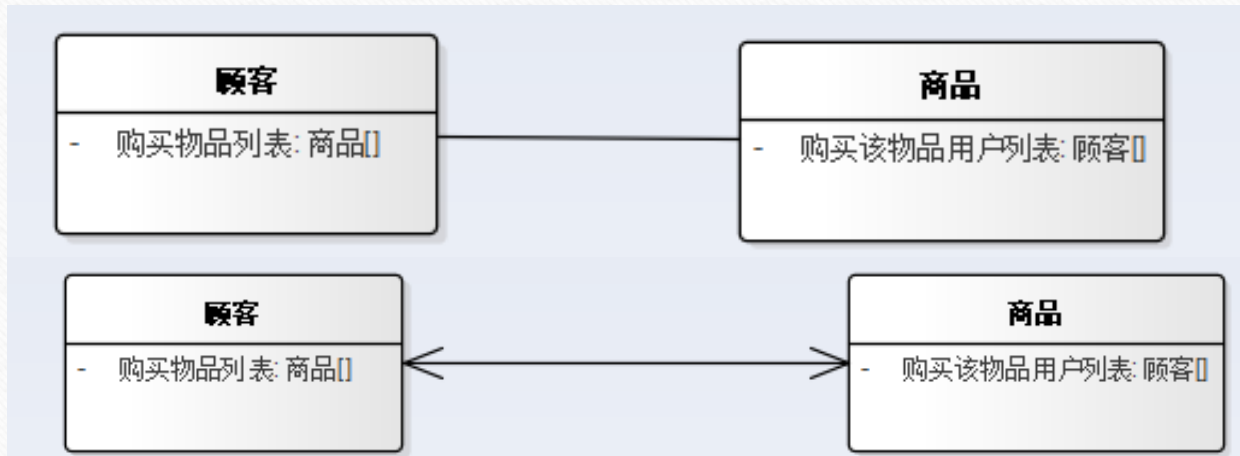
有箭头代表对方能访问到该类，可以理解为消息发送的方向



类图-类间关系-关联关系

- 关联关系-双向关联

两个类互相知道对方的属性或方法。尽量使用单向关联，降低耦合度。

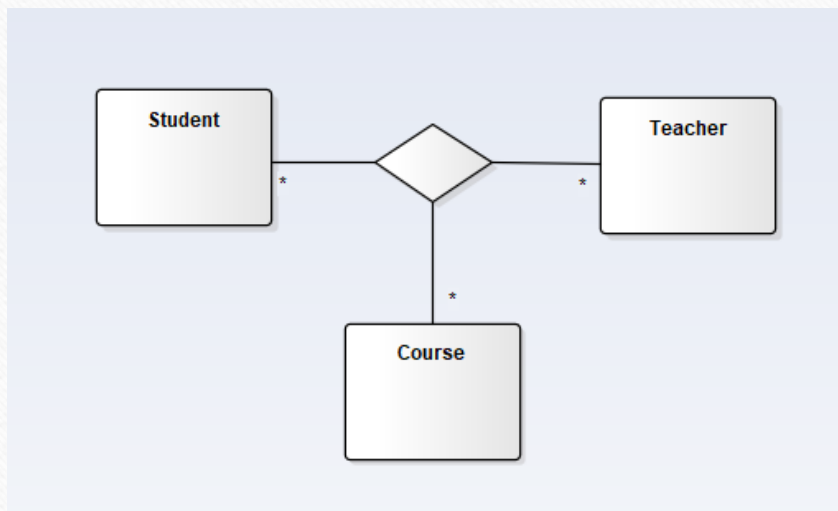


两种表示方式
均可无箭头版
更为常用

类图-类间关系-关联关系

- 关联关系-多元关联

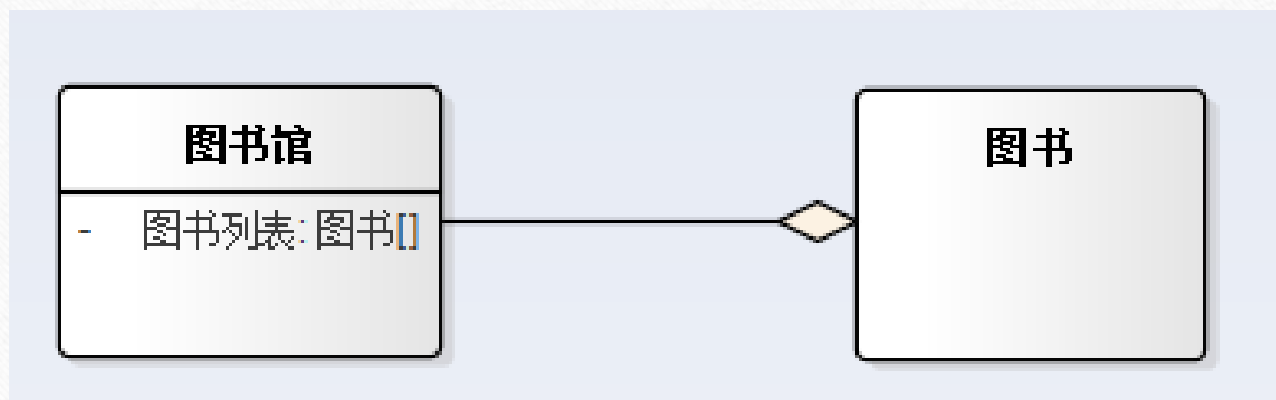
多个类之间互相关联/一个类与多个类关联



类图-类间关系-关联关系

- 关联关系-聚合(Aggregation)

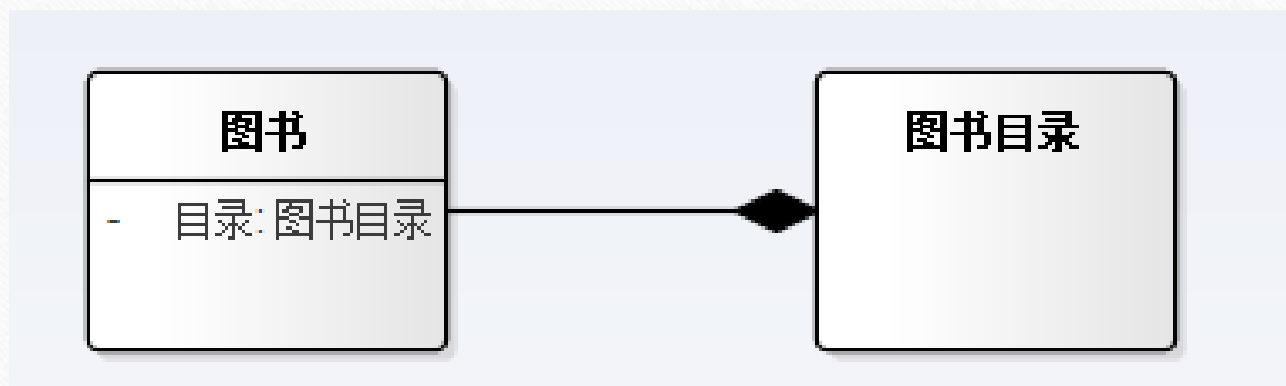
表示整体-部分的关联。整体包含部分，但是部分可以脱离整体单独存在。耦合度较弱。



类图-类间关系-关联关系

- 关联关系-组合

同样表示整体-部分的关联。整体包含部分，部分不能脱离整体单独存在。耦合度较强。



类图-类间关系-关联关系

- 关联关系-如何区分聚集和组合

鉴于类图是为后续的数据库设计打基础，可以从数据库的角度来理解：聚合关系相当于非级联关系（不会一起删除），组合关系相当于级联关系（会一起删除）。

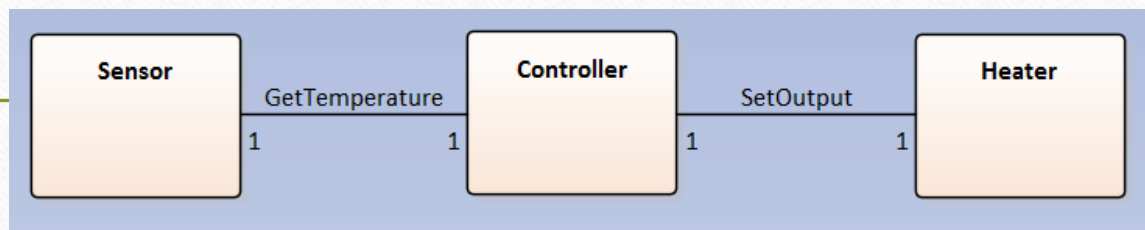
类图-类间关系-关联关系

- **关联关系-关联类**

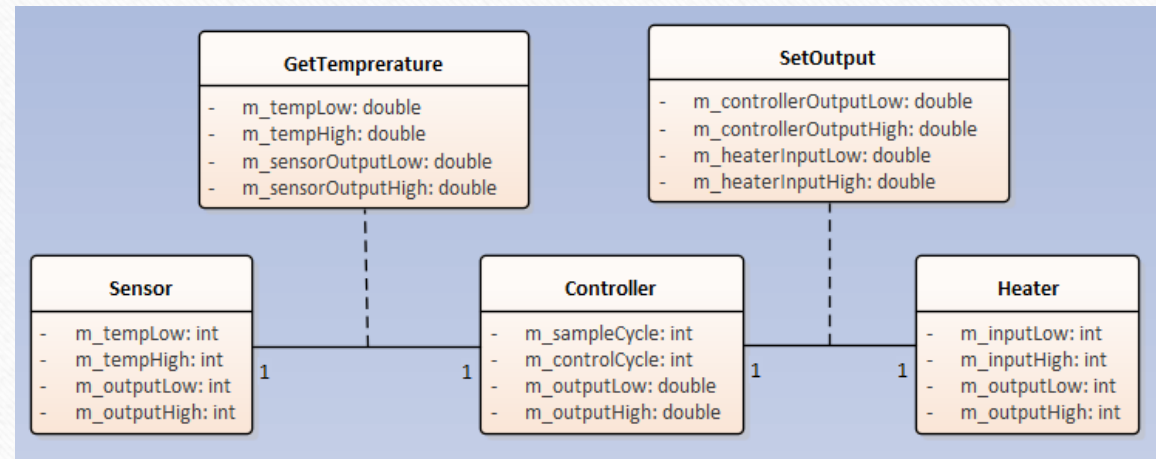
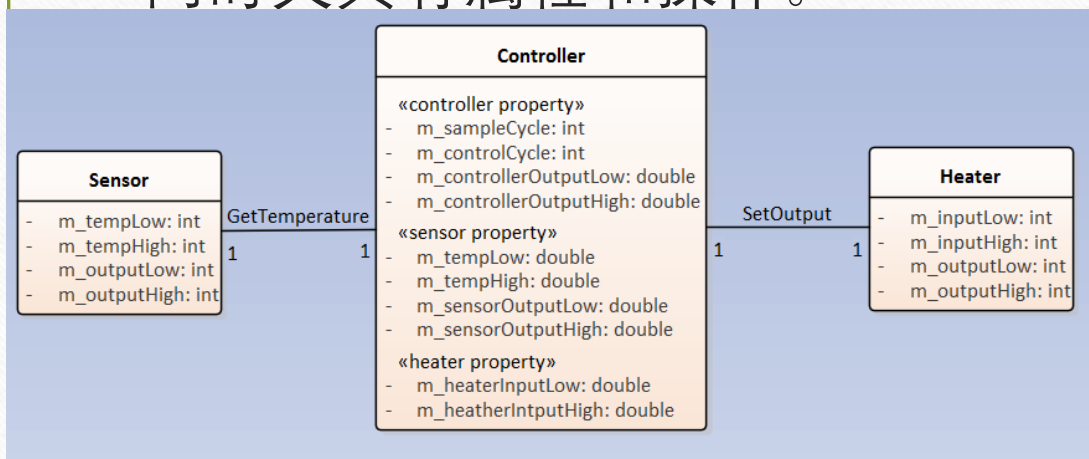
关联类既是类也是关联。它有着关联和类的特性。它将多个类连接起来同时又具有属性和操作。

类图-类间关系-关联关系

- 关联关系-关联类



关联类既是类也是关联。它有着关联和类的特性。它将多个类连接起来同时又具有属性和操作。



类图-类间关系-关联关系

- **关联关系-关联类**

- 面向对象的编程语言不支持关联类的实现
- 设计时需要将关联类直接定义为普通的类，从而将一个多对多的关系转变为两个一对多的关系

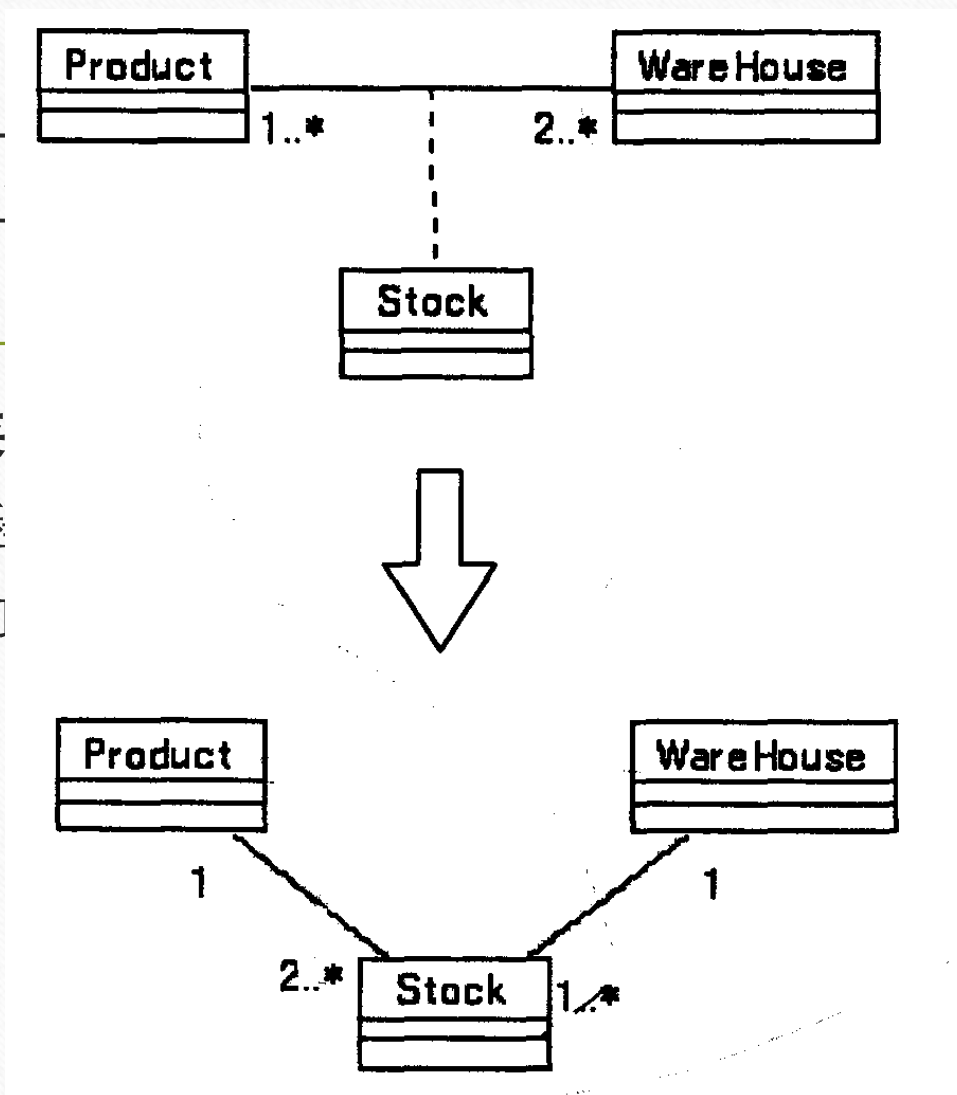
类

系

- 关联关系-关联类

- 面向对象的编程
- 设计时需要将关
多的关系

对多的关系转变为两个一对



类图-类间关系-泛化关系

泛化关系是类的一般描述和具体描述之间的关系，具体描述建立在一般描述的基础之上，并对其进行了扩展。

只有在两个设计类之间存在明确清晰的“is a”关系或为了复用代码才可以使用继承。

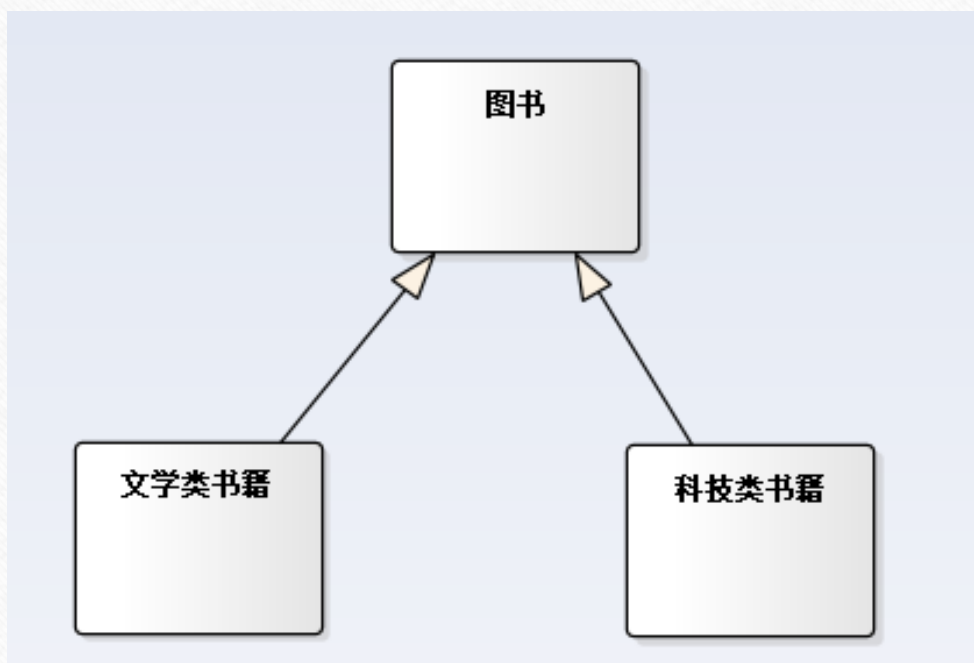
继承是耦合最强的关系，设计的主要原则是解耦。

对绝大多数语言来说，继承是一种静态关系。继承关系慎用！

注意LSP原则的应用

抽象类的使用

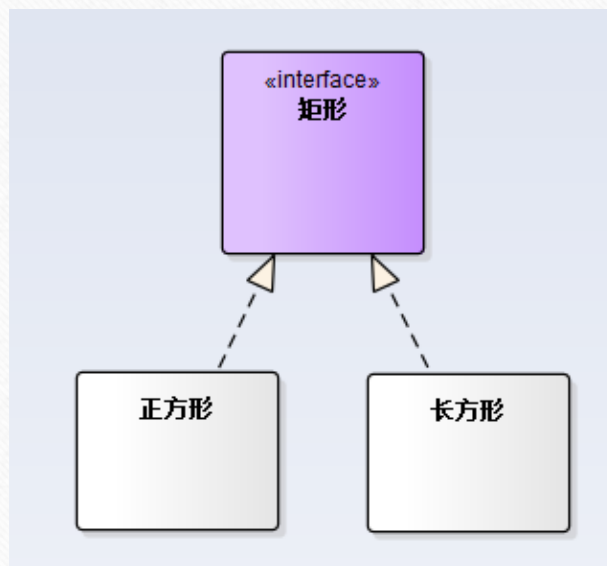
类图-类间关系-泛化关系



- ✓只要有可能，不要从具体类继承
- ✓行为集中的方向是向上的（抽象类）
- ✓数据集中的方向是向下的（具体类）

类图-类间关系-实现关系

实现关系将一种模型元素（如类）与另一种模型元素（如接口）连接起来。通常情况下，后者是行为的规约，前者要求必须支持至少支持后者的所有操作。如果前者是类，后者是接口，则该类是后者的实现。



类图-类间关系-多重性

- 关联的两端以"下限..上限"的格式标示出多重性,如果上下限数相同,标示出一个数目就可以了
- 1: 一个
- *: 零个或多个
- 1..*: 一个或多个
- 0..1: 零个或一个

第二部分：对象图

对象图-概述

- 对象图概述：对象图显示了某一时刻的一组对象及它们之间的关系。
- 对象图可以看做是类图的实例，用来表达各个对象在某一时刻的状态。
- 对象图中的建模元素主要有对象和链，对象是类的实例，链是类之间的关联关系的实例。

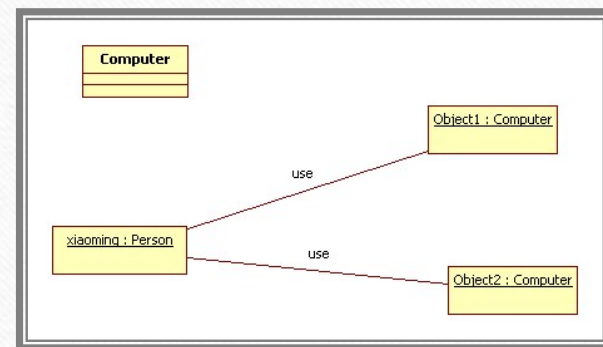
对象图-对象

- 对象是类的实例，是一个封装了状态和行为的具有良好边界和标识符的离散实体。对象通过其类型、名称和状态区别于其他对象而存在。
- 对象名：在矩形框的顶端显示。
- 类型：具体的类目
- 状态：由对象的所有属性以及运行时的当前值组成。
- 表示法：在对象名后跟一个冒号加上类型名，并且使用下划线与类进行区分。

- stu : Student 标准表示法↵
- : Student 匿名表示法↵
- stu 省略类名的表示法↵

对象图-链

- 链是关联关系的实例，是两个或多个对象之间的独立连接。因此，链在对象图中的作用就十分类似于关联关系在类图中的作用。
- 在UML中，链同样使用一根实线段来表示。
- 链主要用来导航。链一端的一个对象可以得到另一位置上的一个或一组对象，然后向其发送消息。链的每一端也可以显示一个角色名称，但不能显示多重性。



对象图-使用注意

- 对象图使用要点:
- 1) 注重于表达系统静态设计视图或静态交互视图的一个方面。
- 2) 表示由一个交互图描绘的动态场景的一个画面。
- 3) 只包含对理解该方面不可缺少的那些元素。
- 4) 提供与它的抽象层次相一致的细节，应该只显露出对理解是不可缺少的那些属性值和其他修饰。
- 5) 不包含方法，对象之间只能通过链来连接。

对象图-与类图的关系

类图	对象图
在类图中，每个类包含三部分：类名、类的属性和类的操作	在对象图中，每个对象包含二部分：对象名、对象属性
类的名称栏只包含类名	对象的名称栏包含对象名和类名
类的属性栏定义了所有属性的特征	对象的属性栏定义了属性的当前值
类中列出了操作	对象图中的对象不包含操作，因为对于属于同一个类的对象，其操作是相同的
类中使用了关联连接，关联中使用关联名、角色以及约束等特征定义	对象使用链进行连接，链中包含名称、角色
类是对象的抽象	对象是客观存在的抽象，对象是类的实例

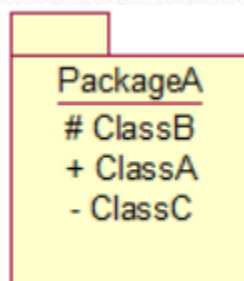
第三部分：包图

包图-定义

- 在面向对象软件开发的视角中，类显然是构建整个系统的基本构造块。但是对于庞大的应用系统而言，其包含的类将是成百上千，再加上其间“阡陌交纵”的关联关系、多重性等，必然是大大超出了人们可以处理的复杂度。这也就是引入了“包”这种分组事物构造块。
- 名称：每个包都必须有一个与其它包相区别的名称
- 拥有的元素：在包中可以拥有各种其它元素，包括类、接口、构件、节点、协作、用例，甚至是其它包或图

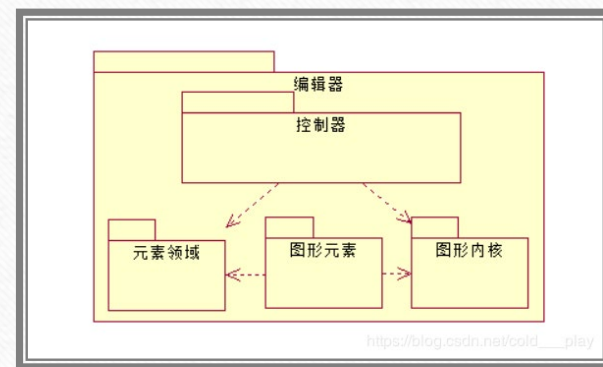
包图-可见性

- 包对自身所包含的内部元素的可见性也有定义，使用关键字private、protected或public来表示。private定义的私有元素对包外部元素完全不可见；protected定义的被保护的元素只对那些与包含这些元素的包有泛化关系的包可见；public定义的公共元素对所有引入的包以及它们的后代都可见。



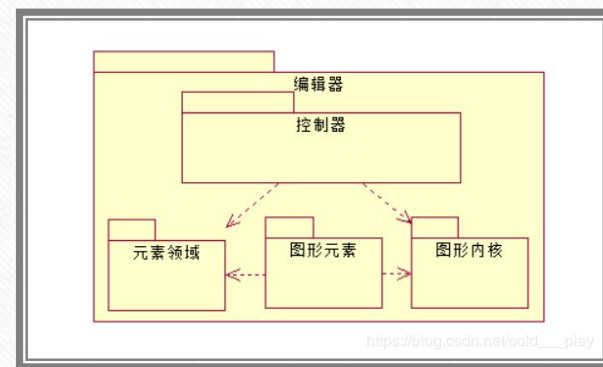
包图-嵌套

- 包可以拥有其他包作为包内的元素，子包又可以拥有自己的子包，这样可以构成一个系统的嵌套结构，比表达系统模型元素的静态结构关系。
- 包的嵌套可以清晰地表现系统模型元素之间的关系，但在建立模型时报的嵌套不宜过深，包嵌套的层数一般以二到三层为最佳。



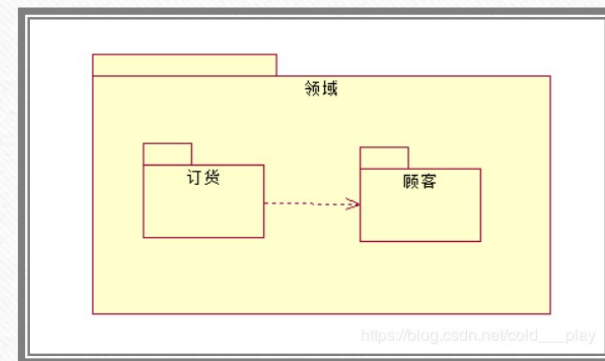
包图-嵌套

- 包可以拥有其他包作为包内的元素，子包又可以拥有自己的子包，这样可以构成一个系统的嵌套结构，比表达系统模型元素的静态结构关系。
- 包的嵌套可以清晰地表现系统模型元素之间的关系，但在建立模型时报的嵌套不宜过深，包嵌套的层数一般以二到三层为最佳。



包图-包间关系

- 包之间的关系总的来讲可以概括为**依赖关系**和**泛化关系**。两个包之间存在着依赖关系通常是值这两个包所包含的模型元素之间存在着一个和多个依赖。对于由对象类组成的包，如果两个包的任何对象类之间存在着一种依赖，则这两个包之间就存在着依赖。而包的嵌套实际为包的泛化关系。



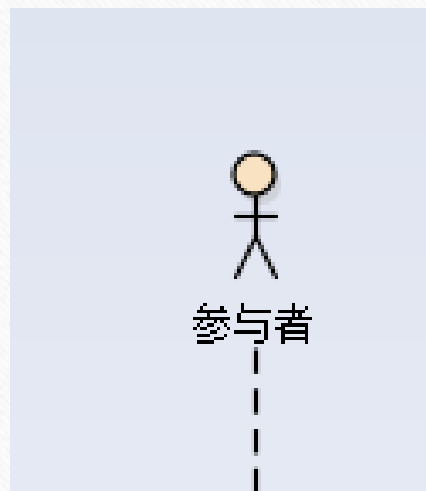
第四部分：顺序图

顺序图-概述

- 顺序图描述了一组对象的交互方式，它表示完成某项行为的对象和这些对象之间传递消息的时间顺序。
- 组成元素：参与者、对象、生命线、控制焦点、消息
- 用例图是系统外部对象（参与者）与系统这两大对象之间的互动，而类图是对系统中涉及到得所有对象，进行抽象描述。顺序图是参与者和系统进行交互，系统内部对象之间的具体互动实现。所以，顺序图关联了类图与用例图，可以通过用例图和类图进行整合。

顺序图-参与者

- 参与者是类中的用户类。参与者应该是脱离系统之外的一个单纯的角色。类中定义的属性和操作可以不考虑。
- 系统角色，可以是人、机器、其他系统、子系统。



顺序图-对象

- 系统内部参与用例的一群对象。
- 三类：边界类、控制类、实体类。依次按照从左至右的顺序排列。控制类只有一个，边界类和实体类可以有多个。
- 边界类：位于系统与外界的交界处，窗体、报表、以及表示通讯协议的类、直接与外部设备交互的类、直接与外部系统交互的类等都是边界类。
- 控制类：控制其他类工作的类。每个用例通常有一个控制类，控制用例中的事件顺序，控制类也可以在多个用例间共用。
- 实体类保存要放进持久存储体的信息。持久存储体就是数据库、文件等可以永久存储数据的介质。实体类可以通过事件流和交互图发现。通常每个实体类在数据库中有相应的表，实体类中的属性对应数据库表中的字段。

顺序图-对象

- 对象的三种命名方式
- 第一种方式包括**对象名和类名**，例如：直播课时:课时，在时序图中，用“对象：类”表示；
- 第二种方式**只显示类名**，即表示它是一个匿名对象，例如：:课程；在时序图中，用“：类”表示；
- 第三种方式**只显示对象名不显示类名**，例如：讲师；在时序图中，用“对象”表示。
- （鉴于EA中加不了下划线，所以手动放弃下划线的要求）



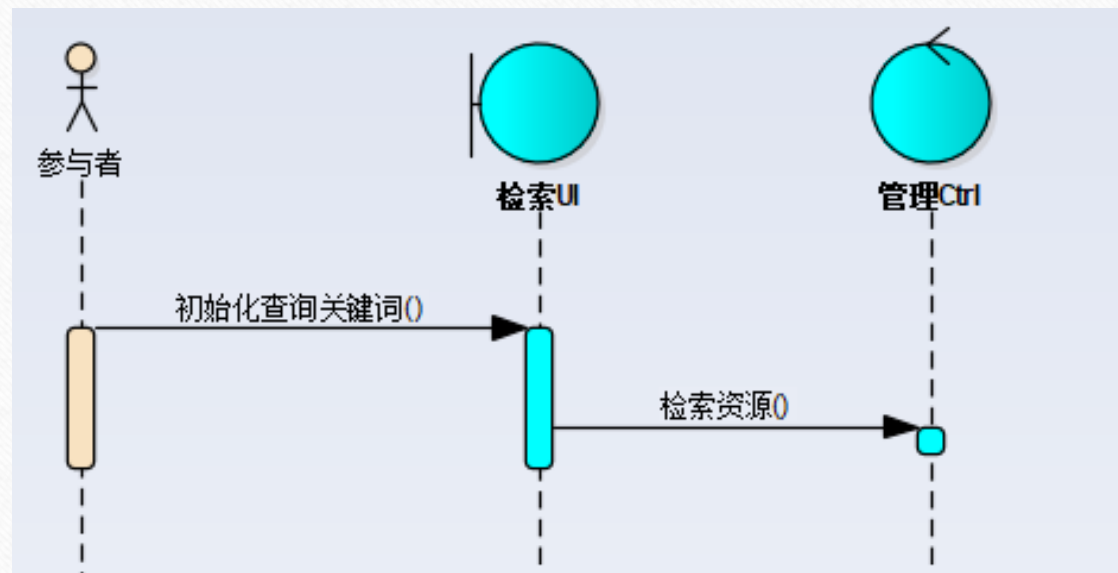
顺序图-生命线

- 从对象图标向下延伸的一条虚线，表示对象存在的时间。



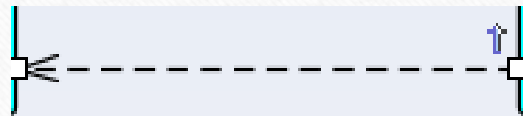
顺序图-控制焦点

- 表示时间段的符号，在这个时间段内对象将执行相应的操作。



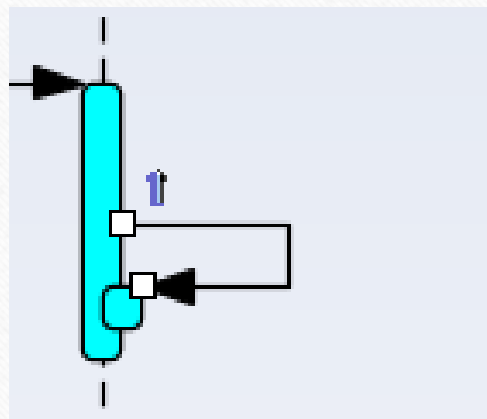
顺序图-消息

- 消息一般分为同步消息、异步消息和返回消息。
- 消息的发送者把控制传递给消息的接收者，然后停止活动，等待消息的接收者放弃或者返回控制。用来表示同步的意义。
- 消息发送者通过消息把信号传递给消息的接收者，然后继续自己的活动，不等待接受者返回消息或者控制。异步消息的接收者和发送者是并发工作的。
- 返回消息表示从过程调用返回。



顺序图-消息

- 自关联消息表示方法的自身调用以及一个对象内的一个方法调用另外一个方法。



顺序图-组合片段

- 组合片段用来解决交互执行的条件和方式，它允许在序列图中直接表示逻辑组件，用于通过指定条件或子进程的应用区域，为任何生命线的任何部分定义特殊条件和子进程。

顺序图-组合片段

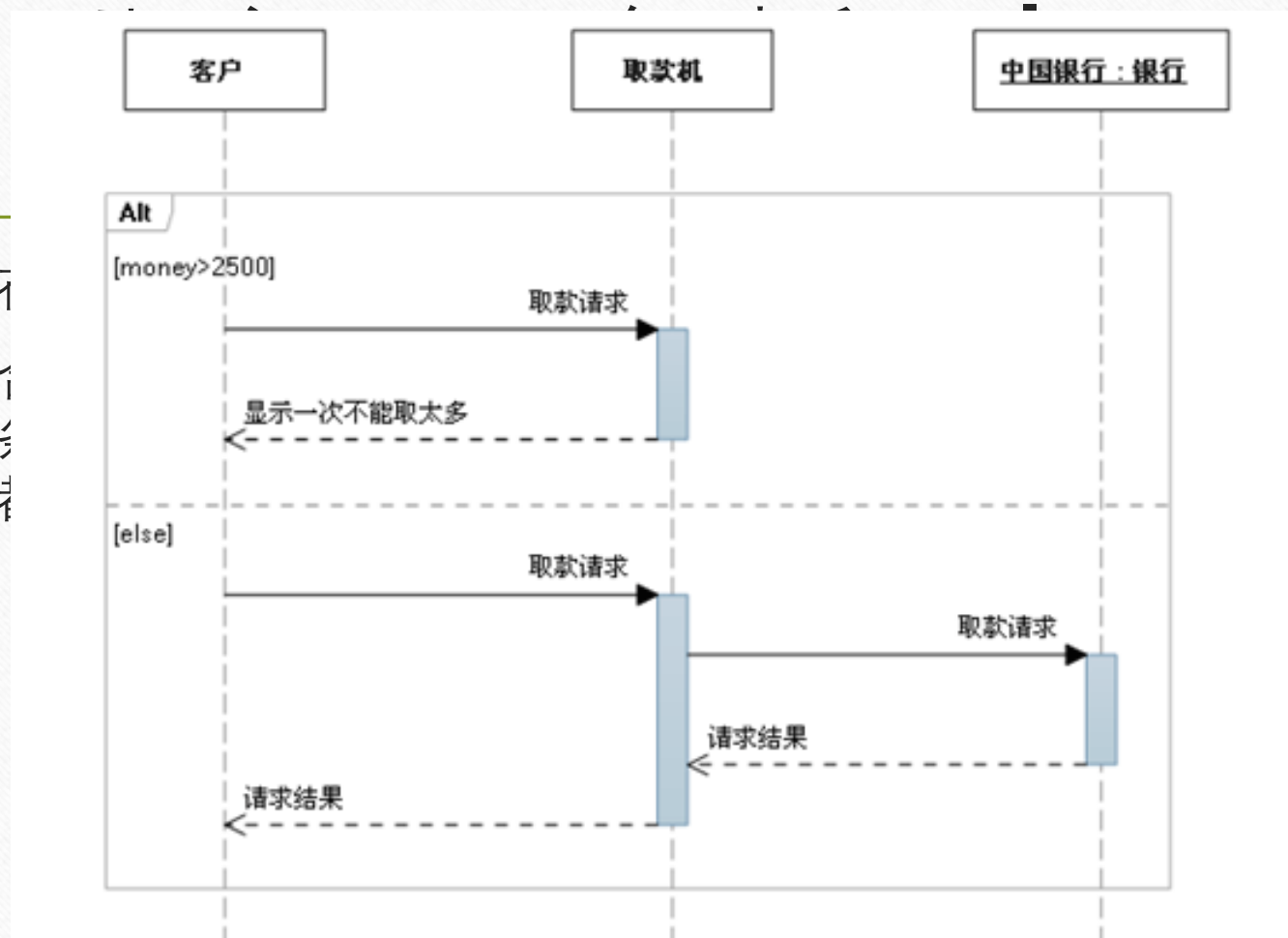
顺序图组合片段类型及属性

操作符	缩写	操作域	说明
Alternatives	alt	多个	备选组合片段，多个域表示多个条件。一次只能有一个操作域执行，类似switch-case语句。可以有一个else。若多个域条件都为真，则随机执行其中一个域
Option	opt	1个	选项组合片段，监护的alt，仅有if无else
Break	break	1个	如何执行此片段，则放弃序列的其他部分。可以使用临界来指定发生中断的条件
Parallel	par	多个	多个操作域的行为并行，操作域以任意顺序交替执行
Week Sequencing	seq	多个	有限制的并行。同一条生命线的不同操作域按顺序执行，不同生命线的操作域以任意顺序交替执行
Strict Sequencing	strict	多个	严格按顺序执行多个操作域的操作
Negative	neg	1个	不可能发生的消息系列，无操作
Critical Region	critical	多个	临界区，区域内操作不能与其他操作交织进行
Ignore	ignore	多个	消息可以在任何地方出现，但会被忽略，往外与其他片段组合在一起
Consider	consider	多个	域ignore相反，不可忽略的消息，往往和其他片段组合使用
Assertion	assertion	多个	断言，说明有效的序列
Reference	ref	1个	引用组合片段
Loop	loop	1个	循环组合片段，片段重复一定次数。可以在临界中指示片段重复的条件

顺序图-组合片段-alt

- 抉择用来指明在两个或更多的消息序列之间的互斥的选择，相当于if..else..。
- 抉择在任何场合下只发生一个序列。可以在每个片段中设置一个临界来指示该片段可以运行的条件。**else** 的临界指示其他任何临界都不为 True 时应运行的片段。如果所有临界都为 False 并且没有 **else**，则不执行任何片段。

- 抉择用来指明在
- 抉择在任何场合
- 段可以运行的条
- 如果所有临界

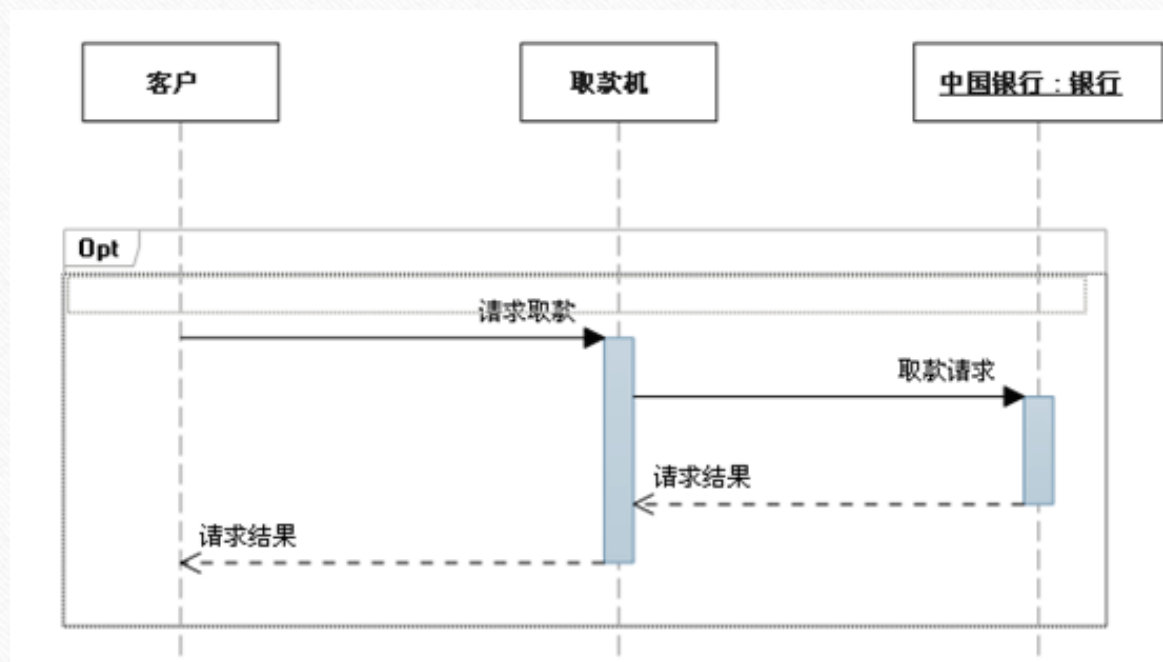


...

指示该片的
片段。

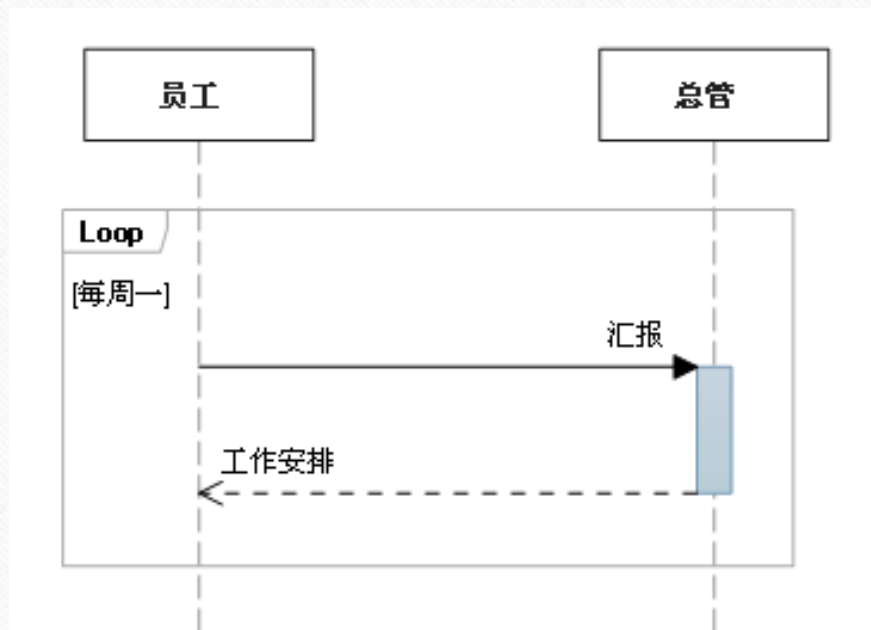
顺序图-组合片段-opt

- 包含一个可能发生或不发生的序列。



顺序图-组合片段-loop

- 片段重复一定次数。可以在临界中指示片段重复的条件。



第五部分：状态图

状态图-概述

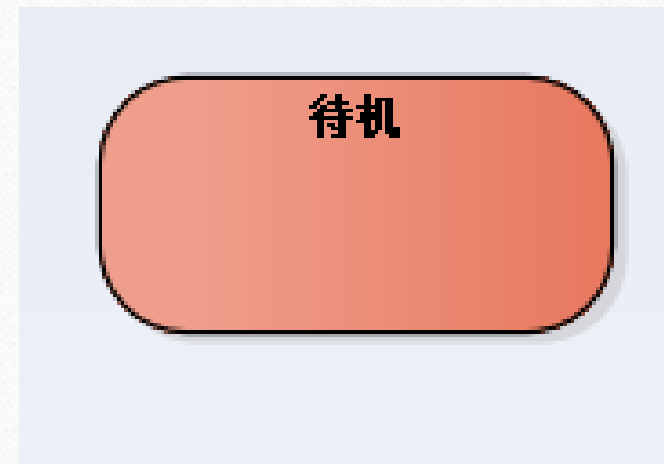
- 由状态机扩展而来，用来描述对象对外部对象响应的历史状态序列，即描述对象所有可能的状态，以及哪些事件将导致状态的改变。
 - 目标是关注在内部哪些事件导致状态改变
 - 在类设计期间，针对那些受状态影响的对象进行状态建模
- 组成元素：状态、事件、转换、活动、动作。
- 有一个初态和一或多个终态

状态图-状态

- 在对象的生命周期中满足某些条件，执行某些活动或等待某些事件时的一个条件或状态。
- 组成元素：名称、进入/退出动作、内部转换、子状态、延迟事件
- 举例：

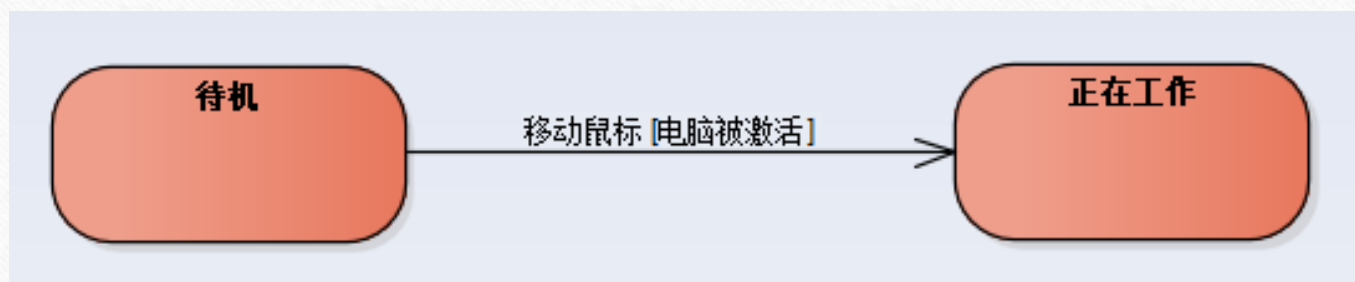
学生（对象）上机（状态）

电脑（对象）关着（状态）



状态图-转换

- 两个状态之间的一种关系，表示对象将在源状态中执行一定的动作，并在某个特定事件发生而且某个特定的警界条件满足时进入目标状态。
- 事件触发：是转移的诱因，可以是一个信号，事件、条件变化和时间表达式。
- 监护条件：当监护条件满足时，事件才会引发转移。
- 结果/动作：当转移发生时所执行的动作，该动作应当是原子操作。



状态图-转换-事件

- 对触发状态转换的事情的描述
- 状态和事件不能混淆：状态：Opened、Closed、Locked。
事件：Open、Close、Lock和Unlock。
- 不是所有事件都会引起状态转移。比如当门是处于Opened状态，不能进行Lock事件



状态图-转换-事件

- 事件触发:

1.信号事件: 实时系统运行中, 对象接受到一个系统外界的信号, 从而使对象的状态发生迁移的事件。

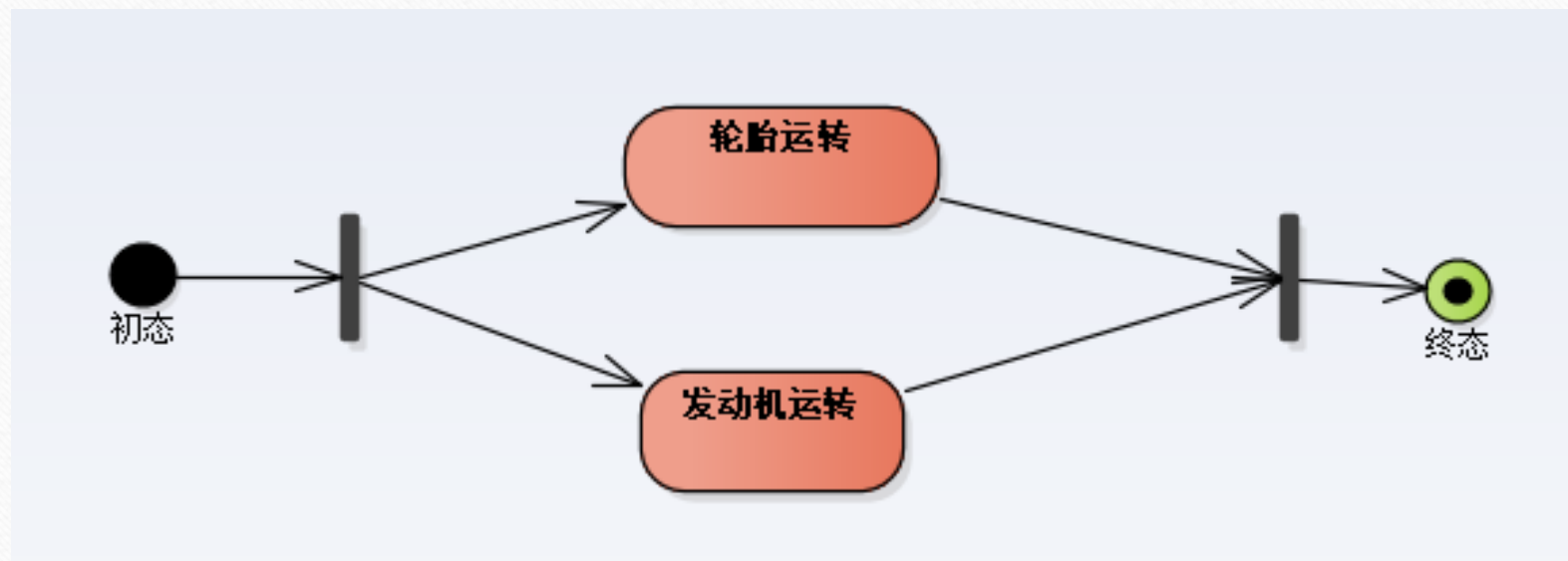
2.变化事件: 对象的内部或外部条件发生变化而引起对象状态发生变化的事件。

3.调用事件: 对象的状态在绝对时间上或某个时间段内自动发生迁移。

4.时间事件: 系统之外的其他系统通过接口或某种协议, 直接执行该系统内部的对象行为, 从而引发对象状态的迁移。

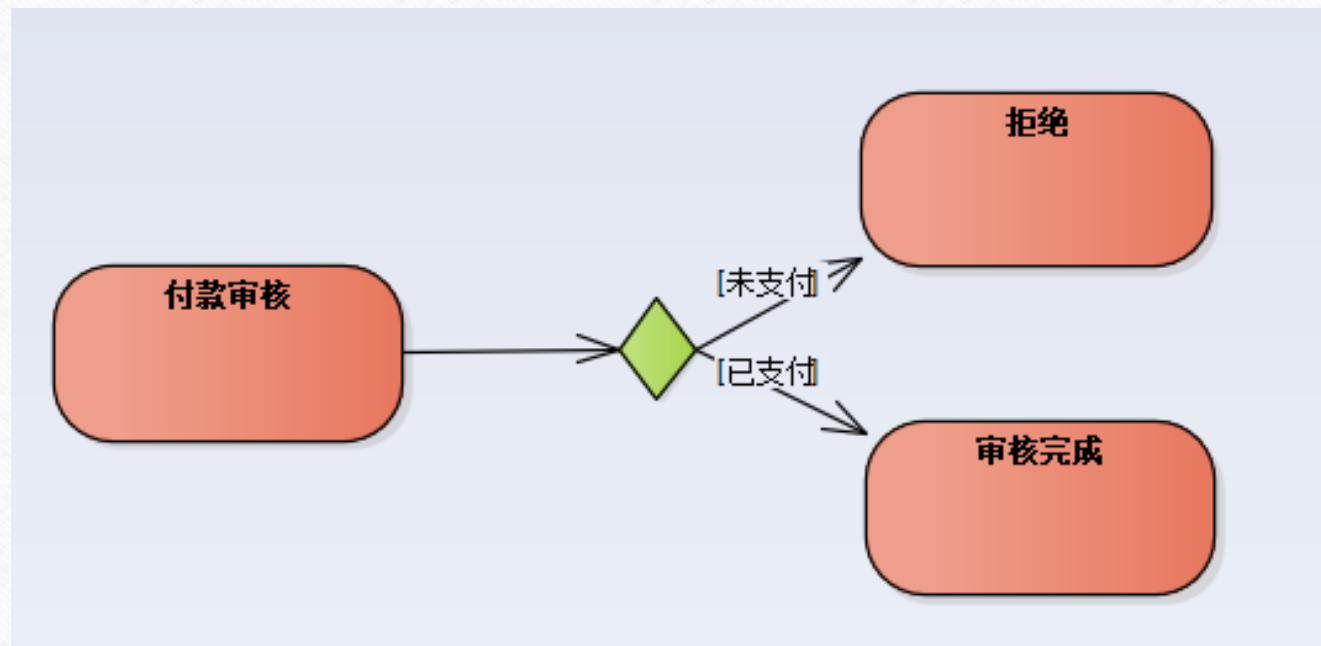
状态图-并发

- 分支同时进行，都会被执行



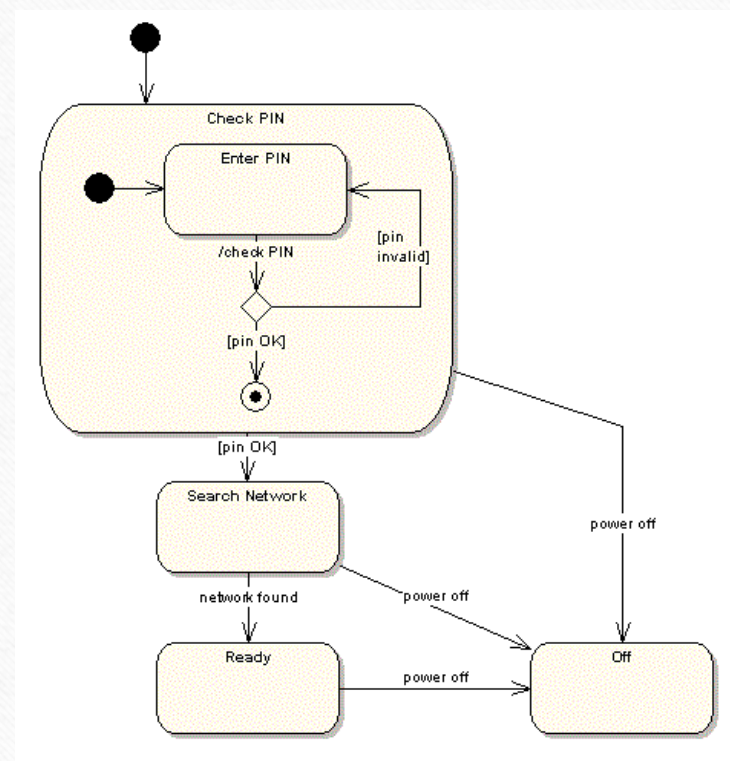
状态图-分支与合并

- 使用分支与合并表示条件选择



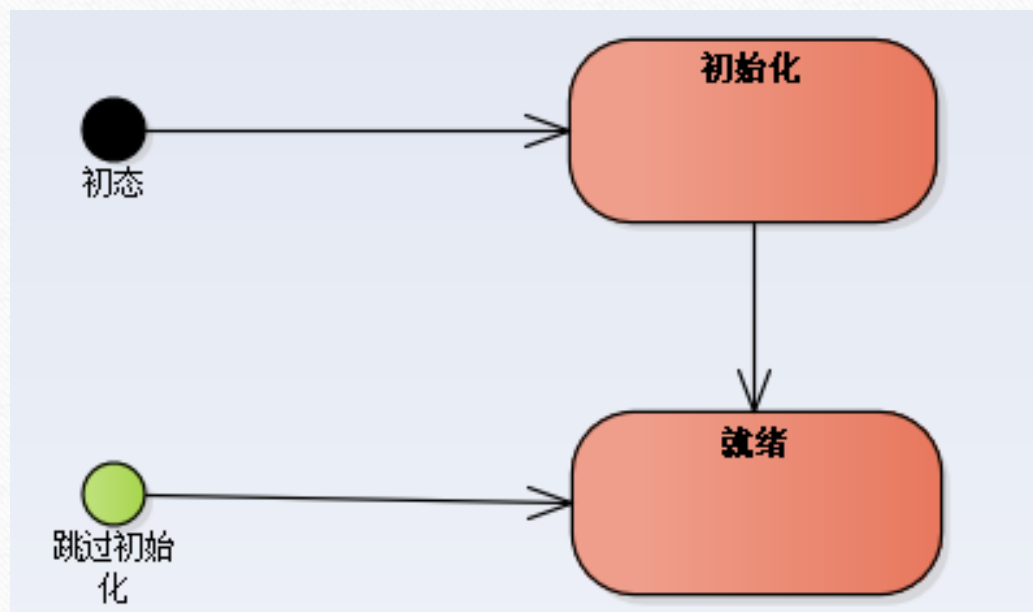
状态图-其它-组合状态

- 嵌套在另外一个状态中的状态称之为子状态, 一个含有子状态的状态被称作组合状态。



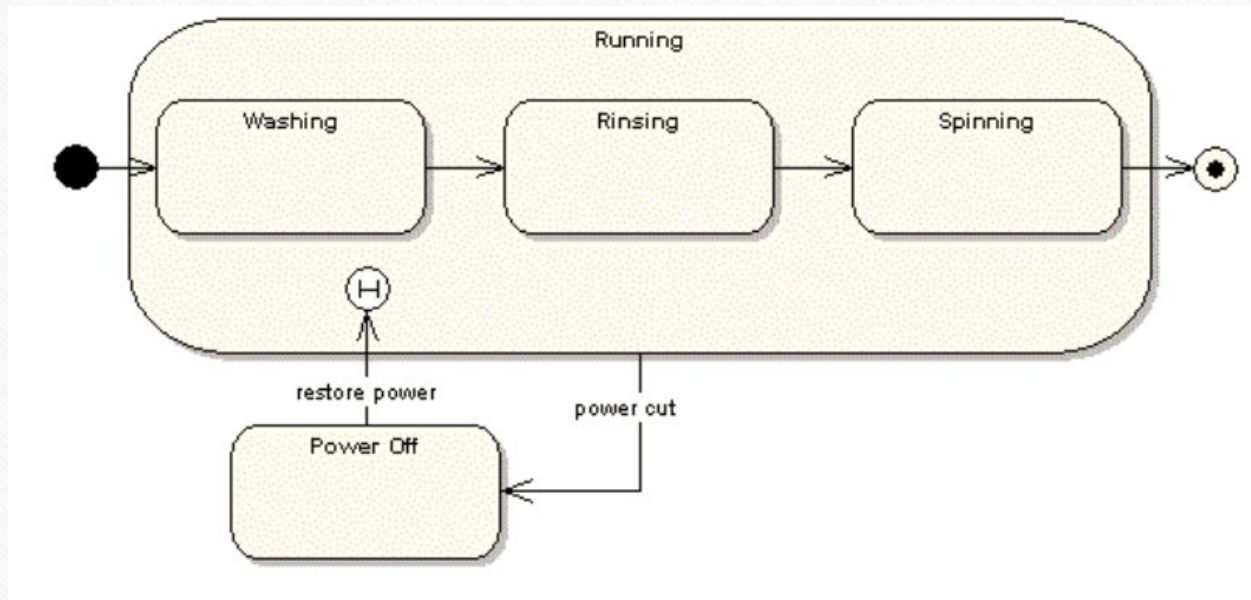
状态图-其它-进入节点

- 某些节点由于一些原因并不会执行初始化，而是直接通过一个节点进入状态，则此节点称之为进入节点。



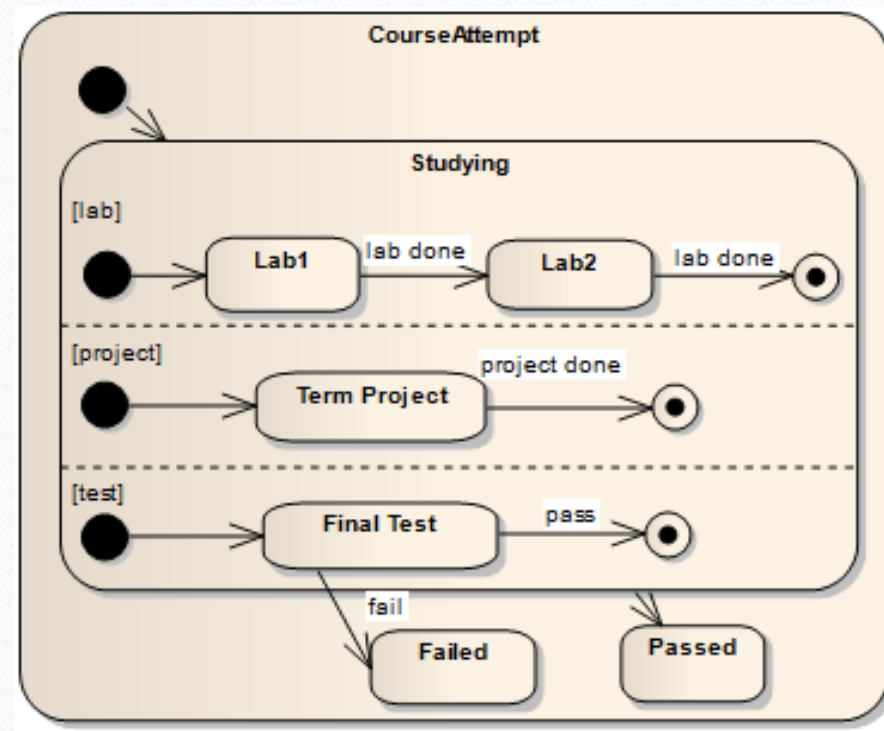
状态图-其它-历史状态

- 历史状态是一个伪状态,其目的是记住从组合状态中退出时所处的子状态, 当再次进入组合状态, 可直接进入这个子状态, 而不是再次从组合状态的初态开始。



状态图-其它-复合状态

- 含有一组子状态的状态



第六部分：活动图

活动图-概述

- 活动图是uml的动态模型的一种图形，一般用来描述相关用例图。准确的活动图定义：活动图描述满足用例要求所要进行的活动以及活动间的约束关系，有利于识别并行活动。目标是关注在内部哪些事件导致状态改变
- 强调从活动到活动的控制流
- 活动区和状态图：
- 状态图是描述某一对象的状态转化的，它主要是展示的是对象的状态。描述的是一个对象的事情。从状态图中我们可以看出，对象在接受了事件刺激后，会做出什么样的反应。
- 活动图是描述系统在执行某一用例时的具体步骤的，它主要表现的是系统的动作，描述的是整个系统的事情。

活动图-概述

- 活动图要素：
- 状态
- 转移
- 分支
- 泳道
- 对象流

活动图-状态

- 活动状态表示在工作流程中执行某个活动或步骤

- 起始状态



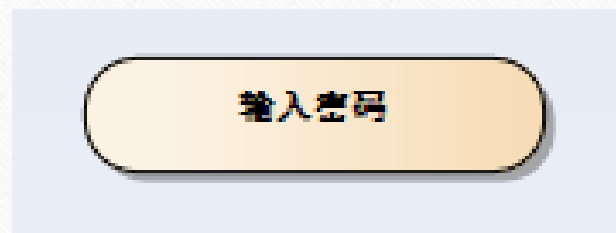
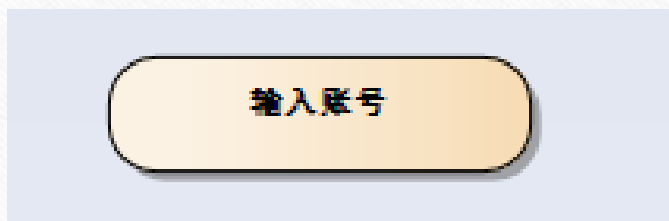
- 终止状态



(流程终止节点表示是子流程的结束)

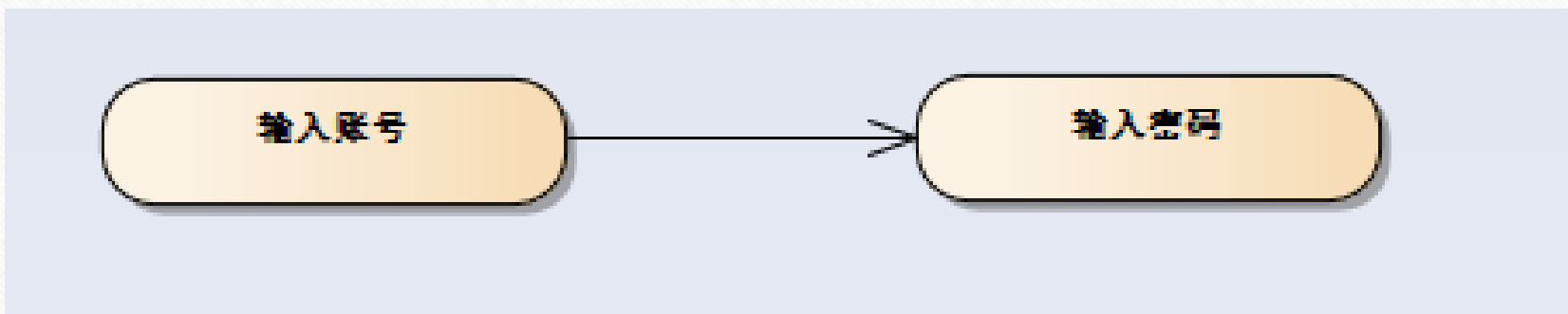


- 状态:



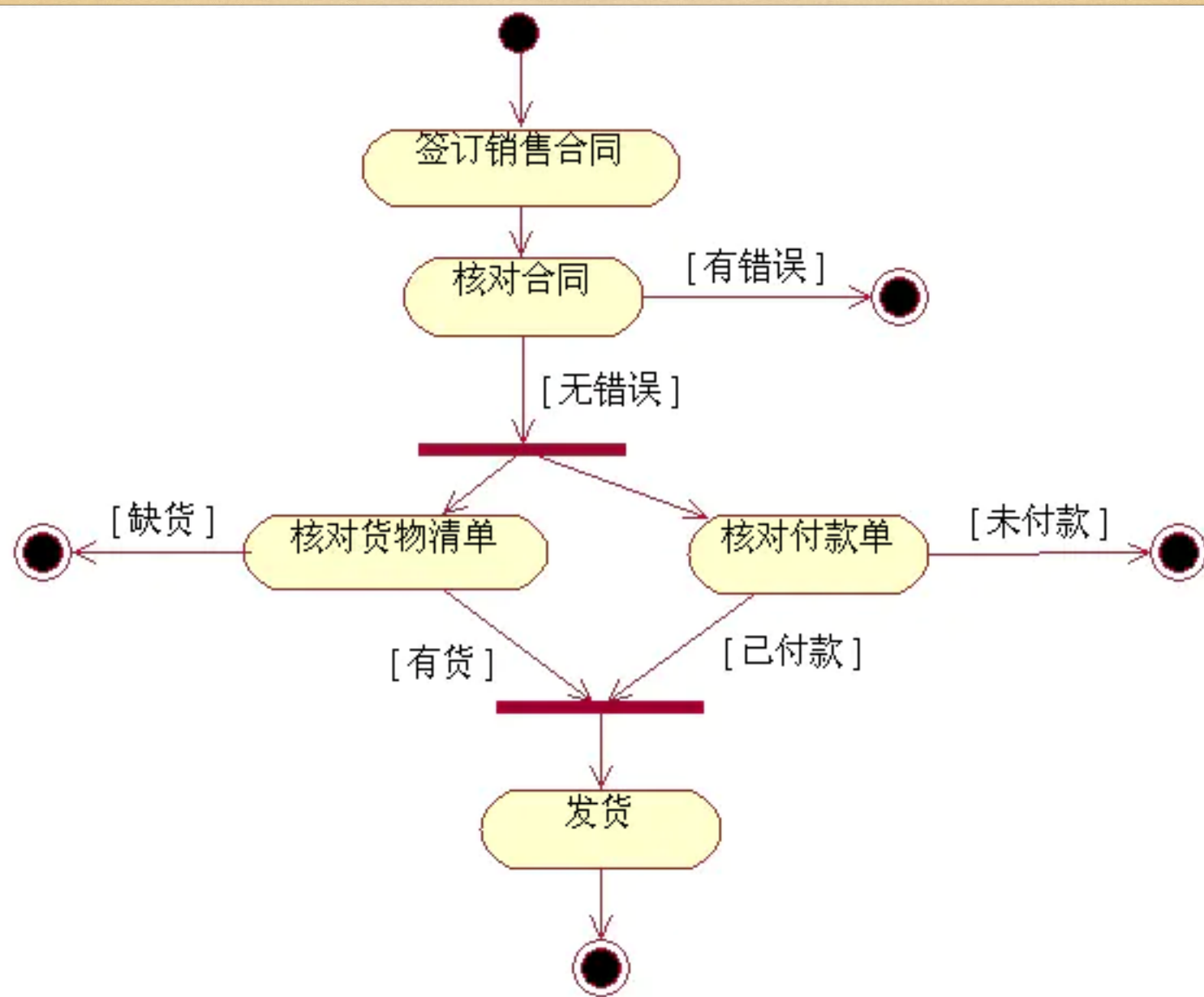
活动图-转移

- 转移表示各种活动状态的先后顺序。这种转移可称为完成转移。它不同于一般的转移，因为它不需要明显的触发器事件，而是通过完成活动(用活动状态表示)来触发。
- 通过完成活动(用活动状态表示)来触发。
- 例：



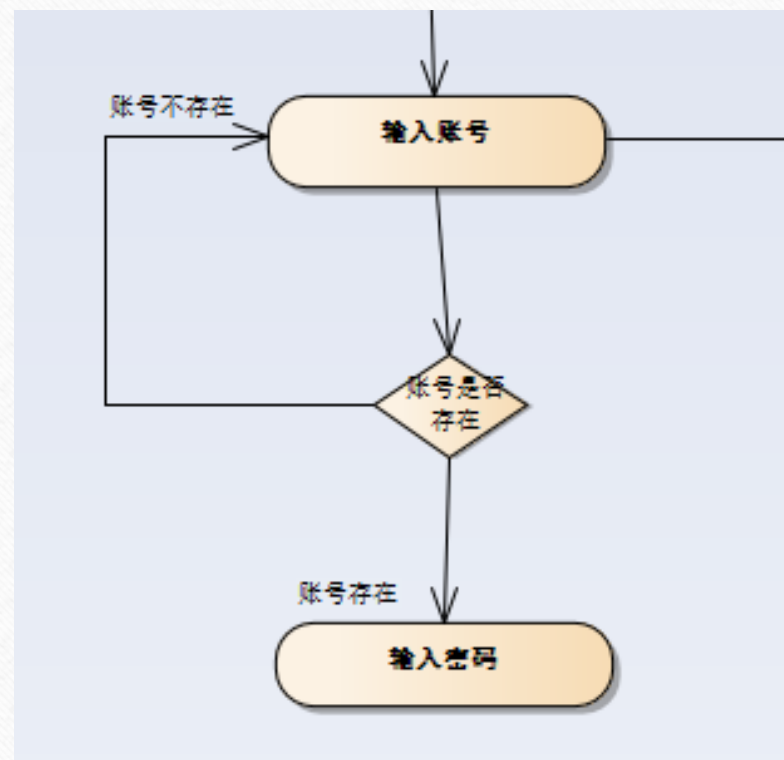
活动图-同步

- 同步示意条用于显示平行分支流
- 同步示意条使您能够显示业务用例的工作流程中的并行线程。



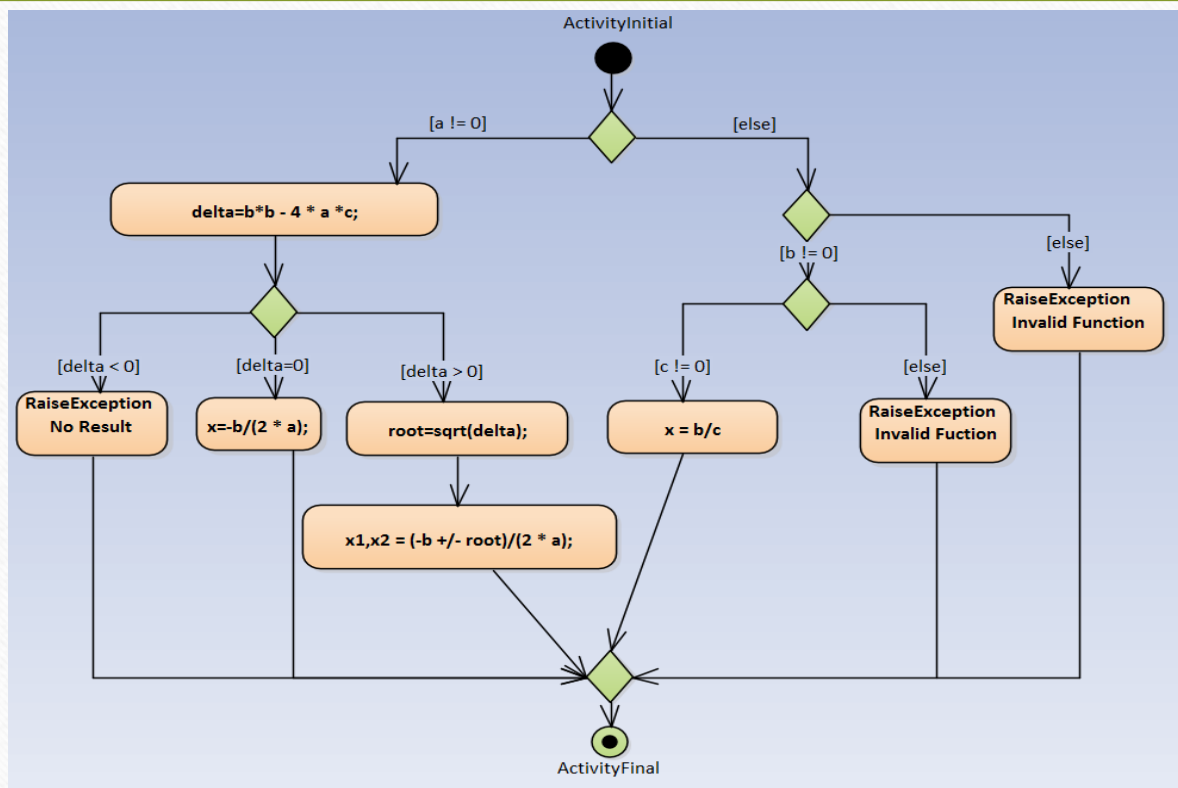
活动图-判断节点

- 选择分支
- 节点有一个输入，两个或多个输出
- 控制流带有监护条件
(else: 没有监护条件为真时控制流去向)



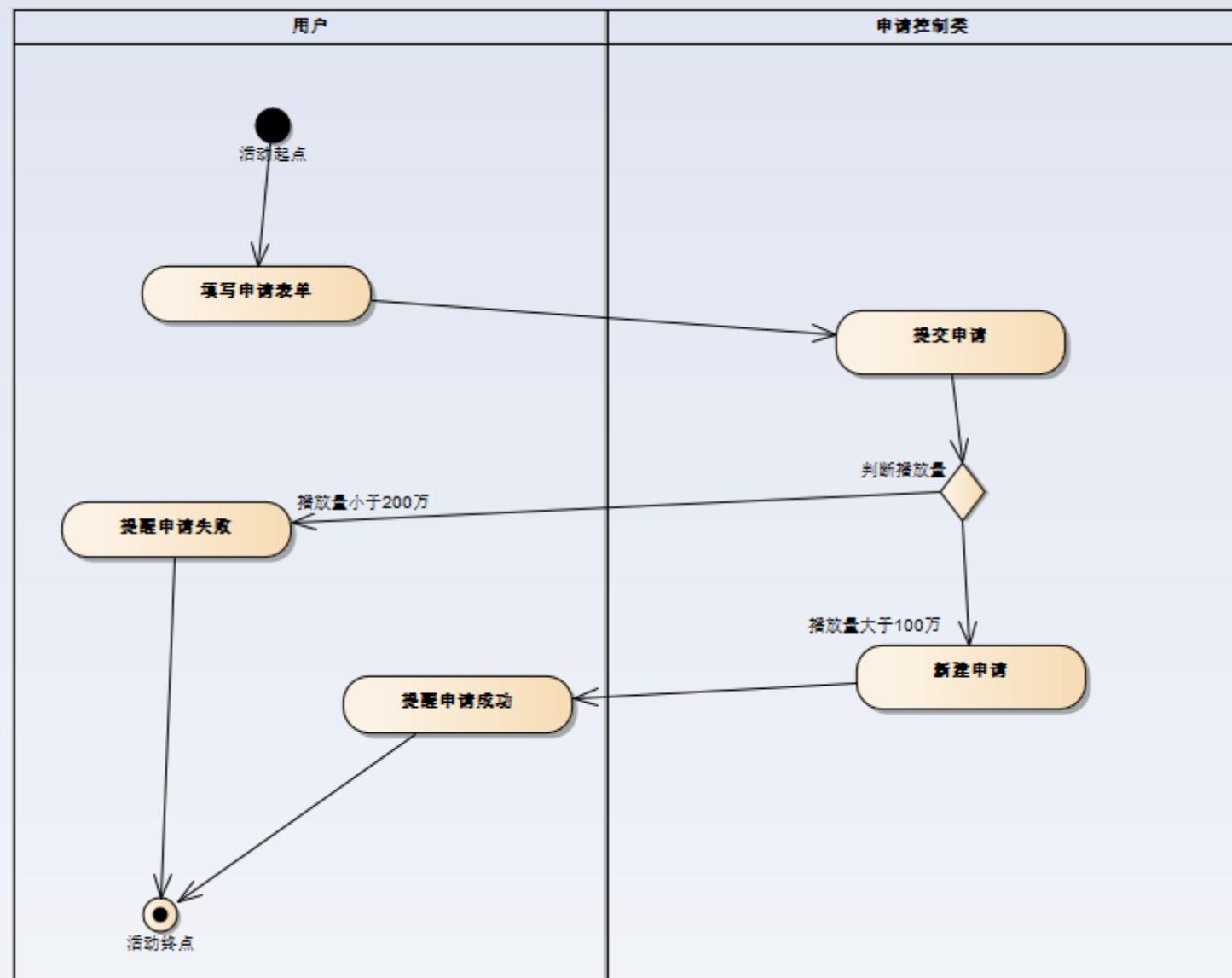
活动图-合并节点

- 控制流的合并



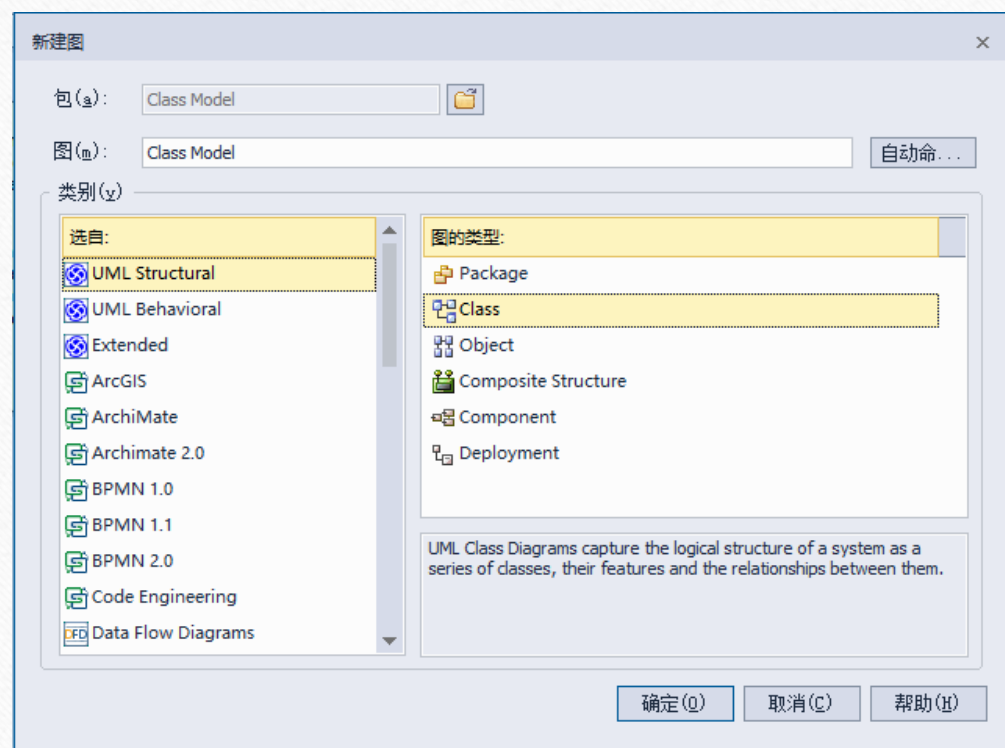
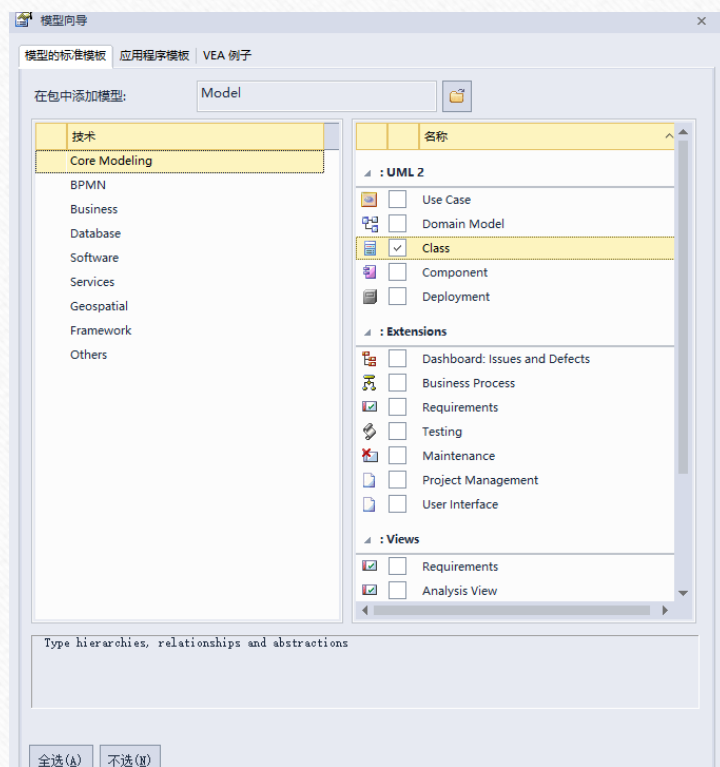
活动图-泳道

- 泳道将活动图中的活动划分为若干组，并把每一组指定给负责这组活动的业务组织，即对象。
- 在活动图中，泳道区分了负责活动的对象，它明确地表示了哪些活动是由哪些对象进行的。在包含泳道的活动图中，每个活动只能明确地属于一个泳道。节点有一个输入，两个或多个输出
- 动作流和对象流允许穿越分隔线。



第七部分：相关**EA**使用

相关EA使用-类图-创建



相关**EA**使用-类图-基本使用

特性

通用

模板

规则

需求

约束








场景

相关的

文件

连接

示例

B *I* U       构造型: ..状态: 别名: 关键词:

作者: miaor

复杂性: 容易

Language: Java

版本: 1.0

阶段: 1.0

包: Class Model

创建日 2019/4/16 23:43:11

修改日 2019/4/16 23:43:11

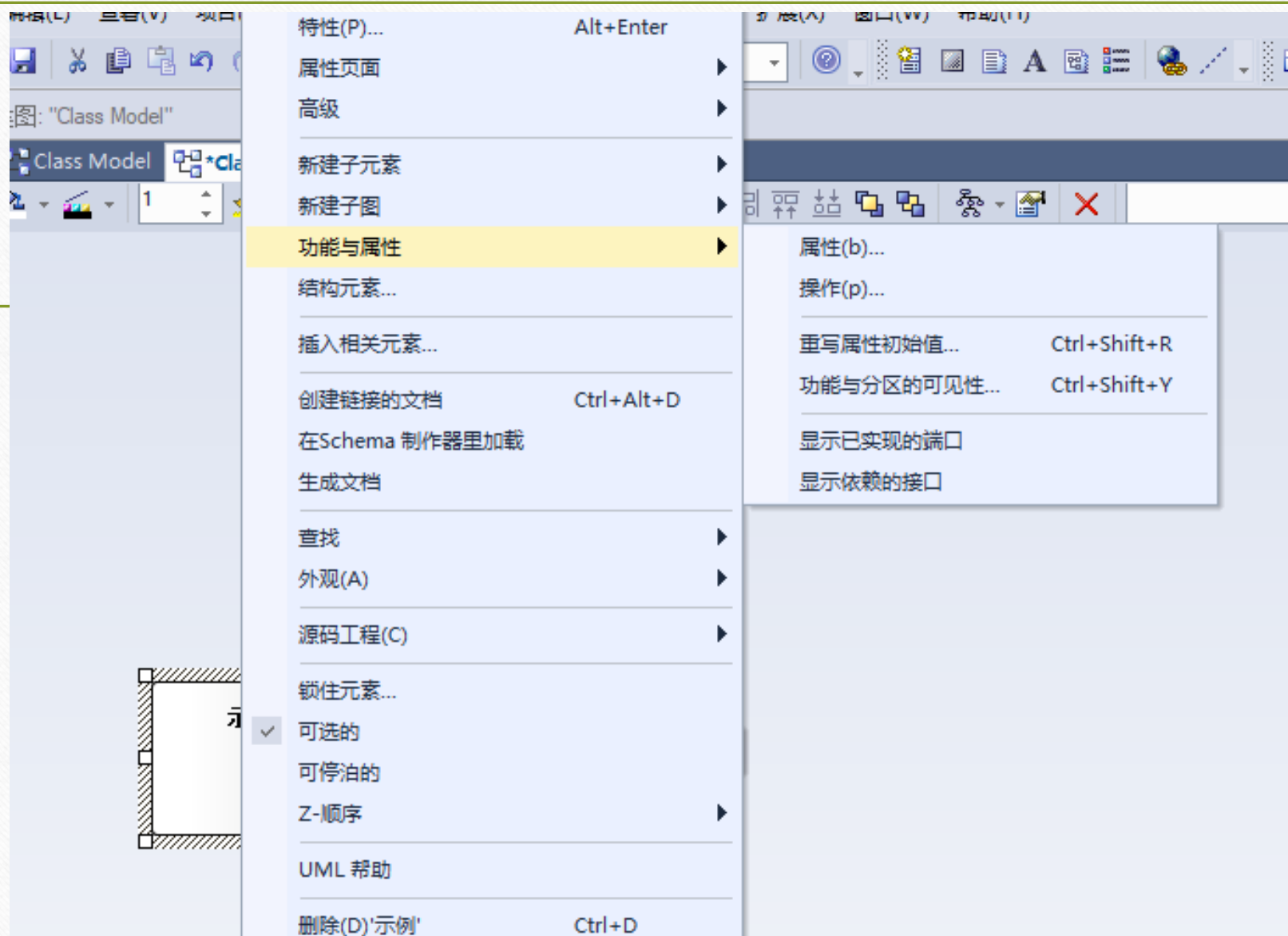
[主页](#) [详细](#) [高级](#) [标记](#)

确定

取消

应用(A)

帮助



属性
操作

选择类型

浏览

搜索

选择命名空间(m):

<any>

Model

Class Model

<none>

新元素

名称(m): newType

自动命...

类型(y): Class

工具

UML::Class

工具...

构造型

☐ 在当前的图中添加此元素

保存

取消(C)

帮助(H)

类型：分类器 比如 类，作用者，活动等

增加(N)

确定

取消

帮助

关闭

帮助

属性

操作

名称	参数	返回类型	作用域	构造型	别名
testIT	studentNumber	int	Public		
新操作...					

方法

开发

抽象的

False

静态的

False

修改者

高级

参数(m)

注释(N)

行为(B)

重新定义(R)

前置(e)

后置(t)

标记值(V)

参数

类型

参数

默认值

构造型

别名

方向

固定值

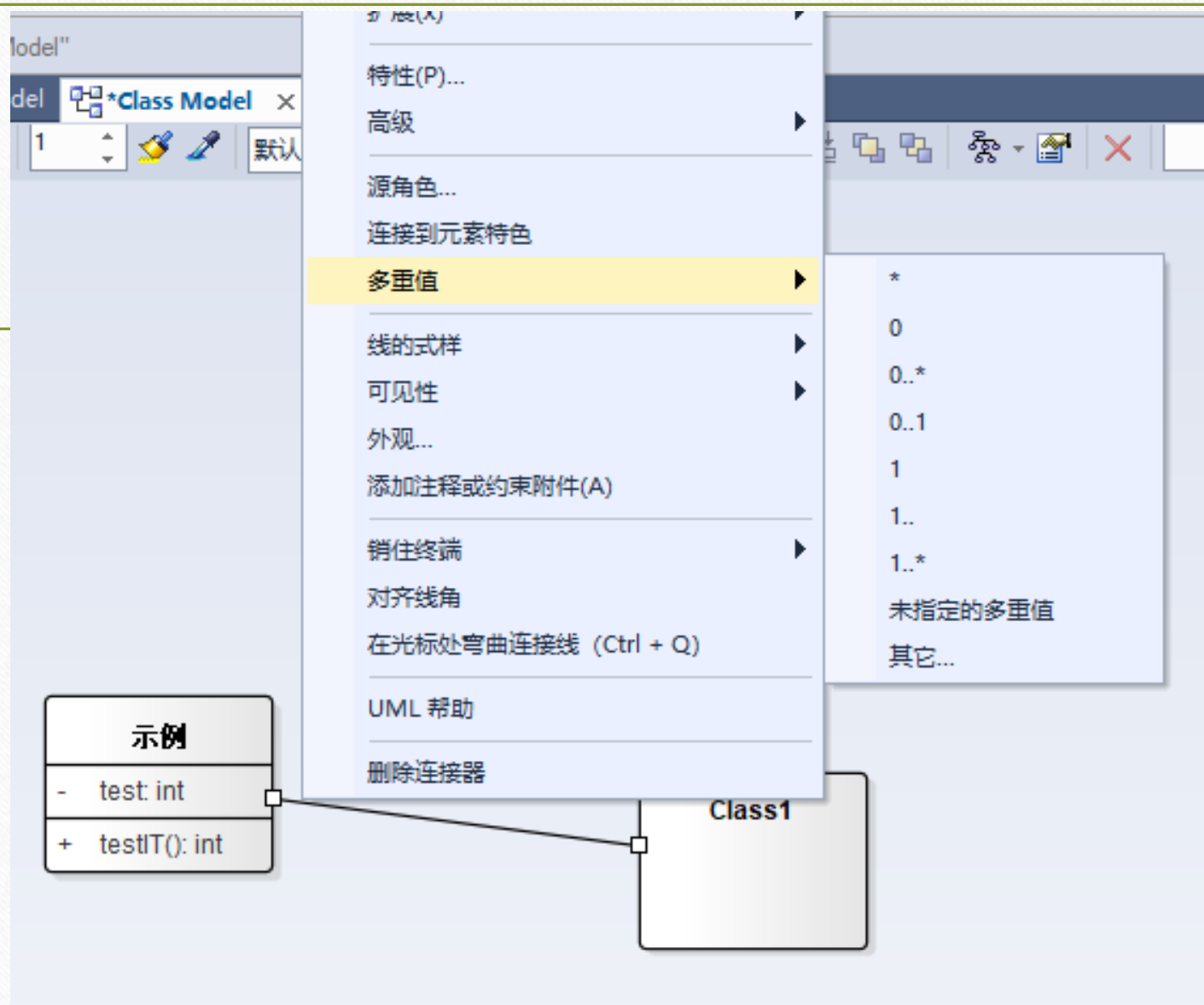
False

多重值

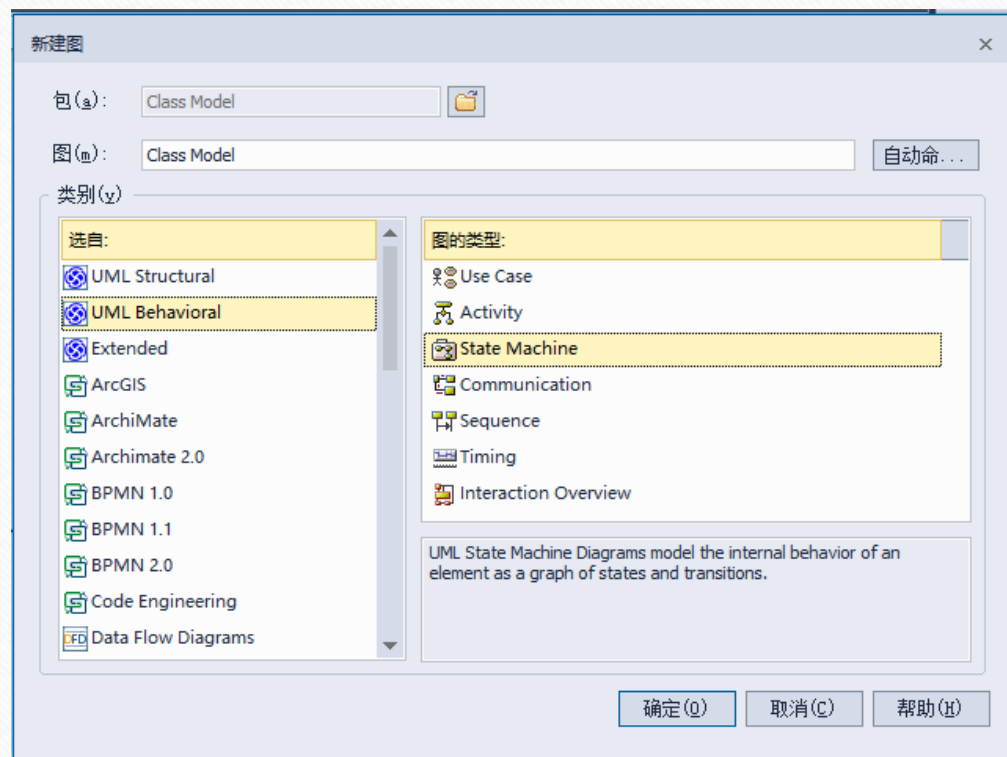
注释

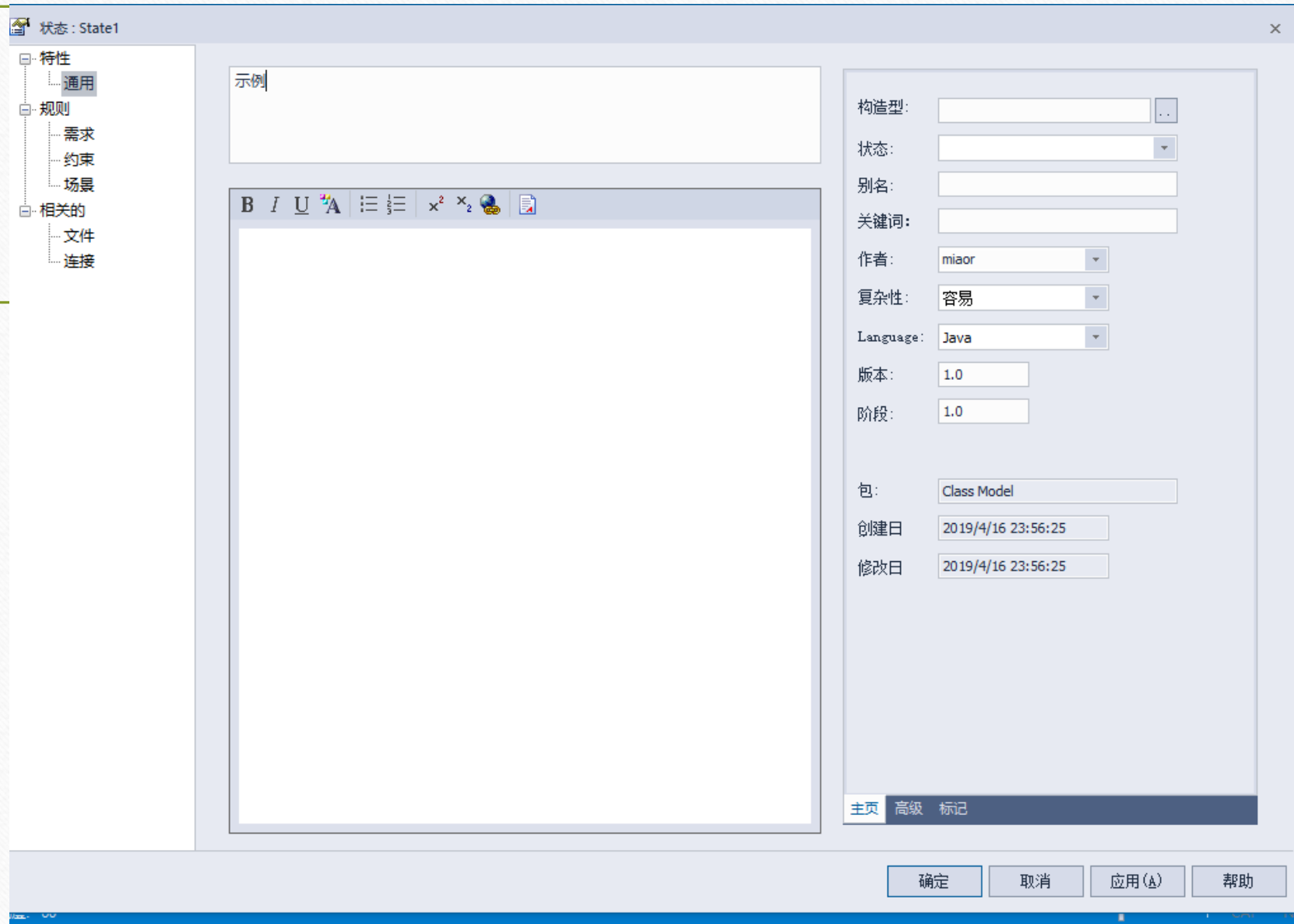
关闭

帮助



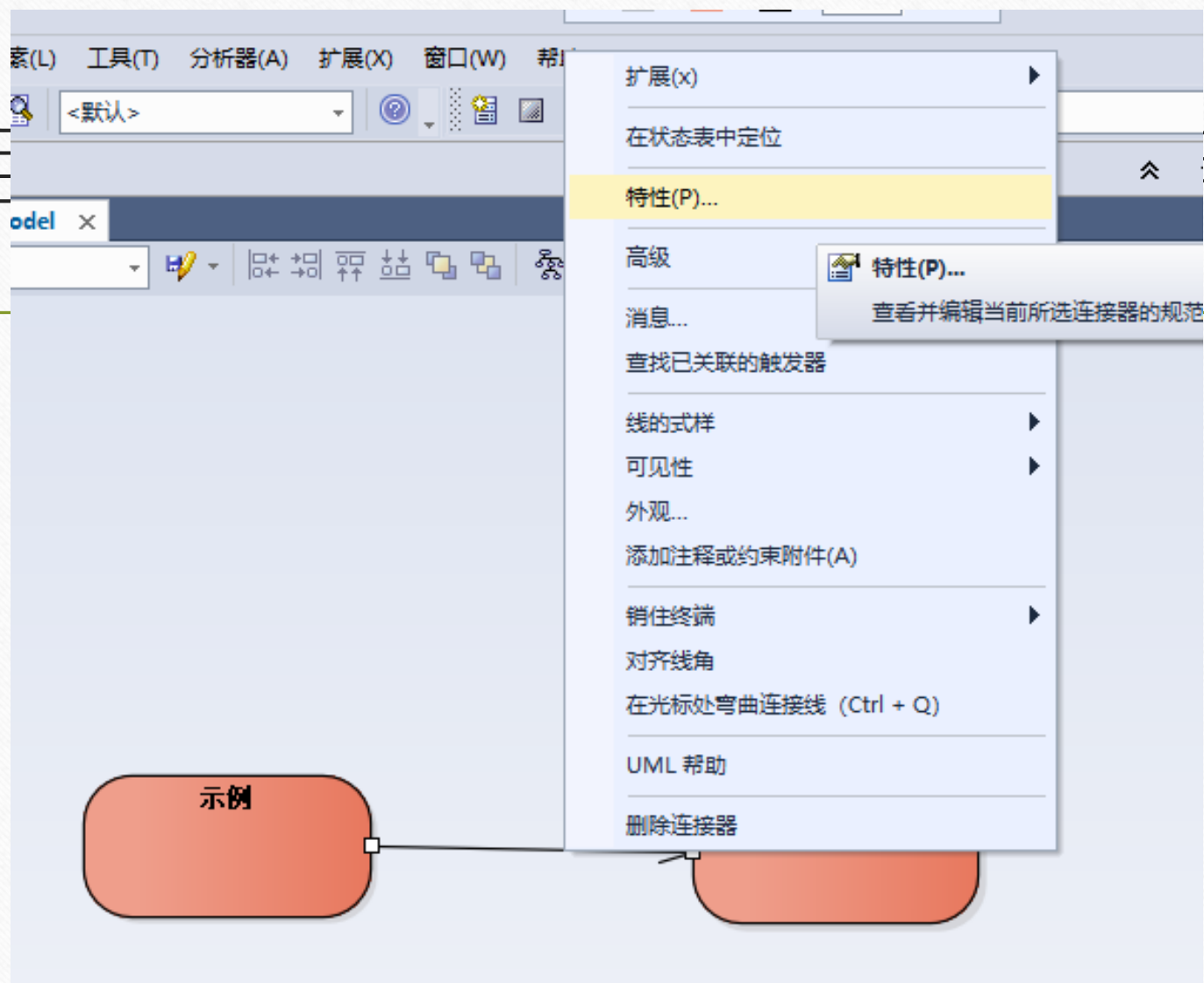
相关EA说明-状态图-基本使用





相

用



过度 特性

通用

约束

监护条件 (u):

监护条件

☐ 作用是一行为 (y) 元素。

作用 (f):

作用

...

触发器

名称 (n):

名称

...

类型 (y):

规范 (g):

...

新建 (N)

保存 (S)

删除 (D)

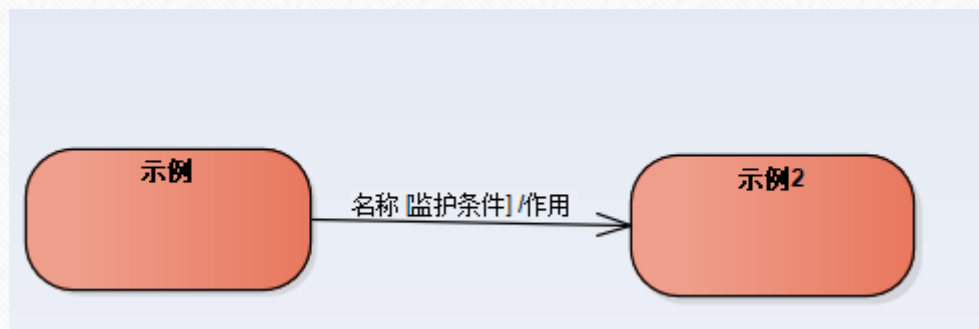
名称	类型	规范
----	----	----

确定

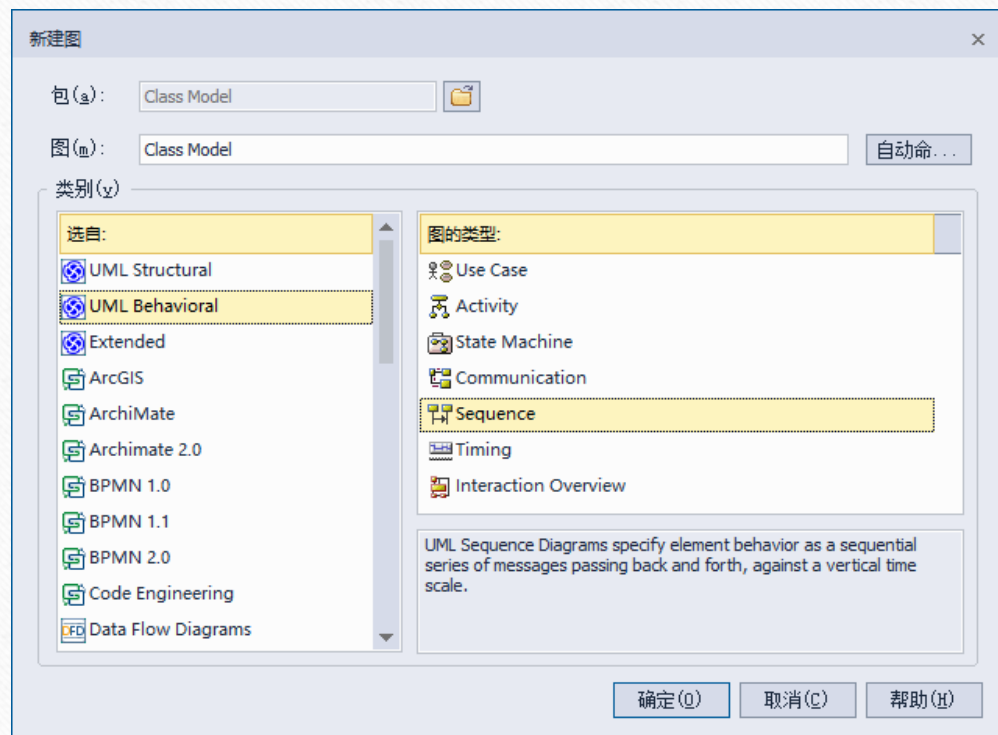
取消

帮助

相关**EA**说明-状态图-基本使用



相关EA说明-顺序图-基本使用





参与者



检索UI



管理Ctrl



书籍列表

- ^ Lifeline
- Boundary
- Control
- Entity
- Fragment
- Endpoint
- Diagram Gate
- State/Continuation

Interaction Relationships



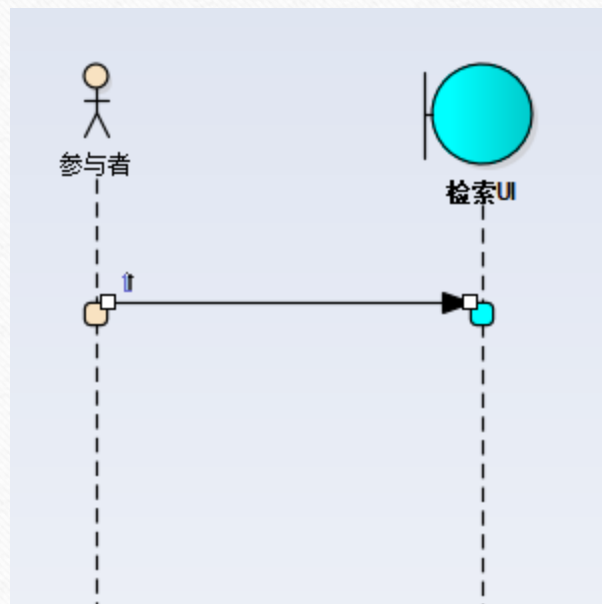
Message

Common

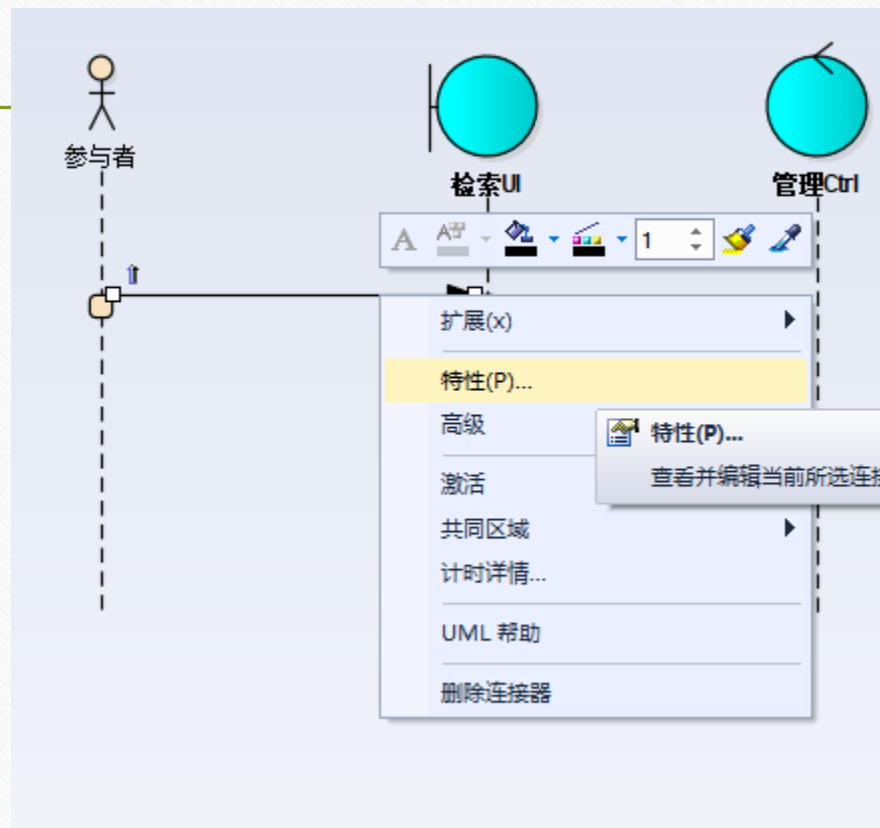


Artifacts

相关EA说明-顺序图-基本使用



相关EA说明-顺序图-基本使用



相关EA

本使用

消息特性

签名

消息(s): test 操作

参数

参数(u):

返回值(y): void ☒ 显示继承的方法(a)

分配给(z):

构造型(p): ...

别名(l):

顺序表达式(e)

条件(d):

约束(x):



☐ 重复(i)

控制流类型(f):

同步: 同步 生命周期: 是返回

类型: 同步 异步

备注(t):

B I U A $\frac{1}{3}$ $\frac{2}{3}$ \times^2 \times_2  

确定(O) 取消(C) 帮助(H)