[**如何用Rational rose创建常用图**](http://www.cnblogs.com/perfei/p/7881752.html)**件**

UML中各种图形概要：

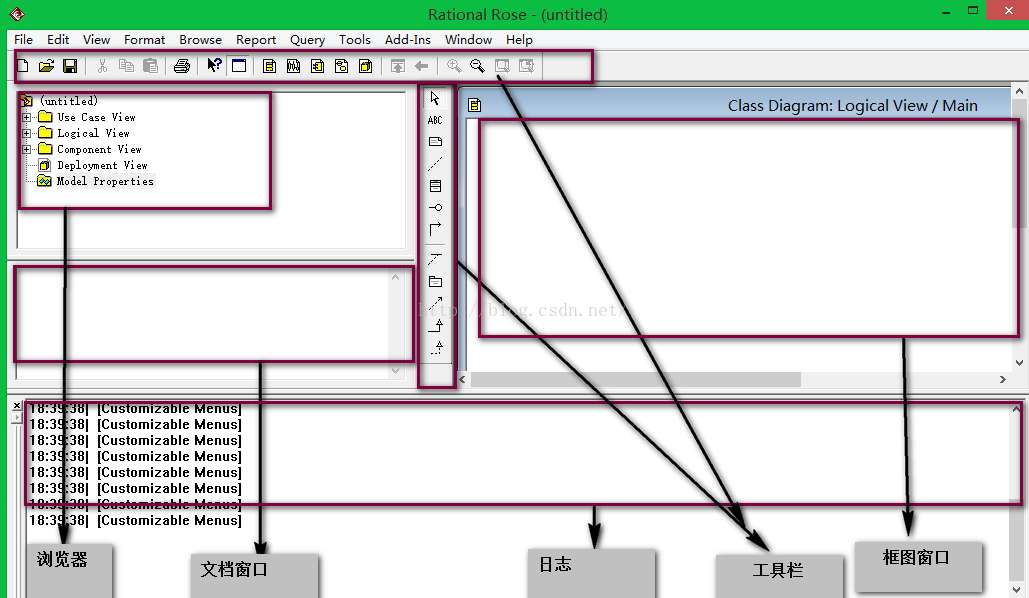
| 图名 | 对照 | 说明 |
| --- | --- | --- |
| 用例图 | use case diagram | 用例图表明系统做什么,与谁交互。用例是系统提供的功能，参与者是系统与谁交互,参与者可以是人、系统或其他实体。一个系统可以创建一个或多个用例图。 |
|  | 用例 | use case |
|  | 参考者 | actor |
|  | 关联关系 | unidirectional association |
|  | 泛化关系（继承） | generalization |
| 活动图 | activity diagram | 活动图显示了从活动到活动的流。活动图可以在分析系统业务时用来淙业务流，也可以在收集系统需求的时候显示一个用例中的事件流。活动图显示了系统中某个业务或者某个用例中，要经历哪些活动，这些活动按什么顺序发生。 |
|  | 泳道 | swimlane |
|  | 活动 | activity |
|  |  | state transition |
|  | 同步 | synchronization |
|  | 决策点 | decision |
| 类图 | class diagram | 类图显示系统之中类与类之间的交互 |
|  | 类 | class |
|  | 方法 | Operation |
|  | 属性 | Attribute |
| 序列图 | sequence diagram | 序列图显示用例中的功能流程 |
| 协作图 | collaboration diagram | |
| 状态图 | statechart diagram | |
| 构件图 | component diagram | 构件图显示模型的物理视图，也显示系统中的软件构件及其相互关系，模型中的每个类映射代码构件 ，一旦创建构件，就加进构件图中，然后画出构件之间的相关性。构件间的相关性包括编译相关性和运行相关性。 |
| 实施图 | deployment diagram | 实施图是显示网络的物理布局，系统中涉及的处理器、设备、连接和过程。一个项目中有一个实施图。 |

 Rose模型（包括所有框图、对象和其他模型元素）都保存在一个扩展名为.mdl的文件中。

## 1.环境简介

### 1.1 Rational Rose可视化环境组成

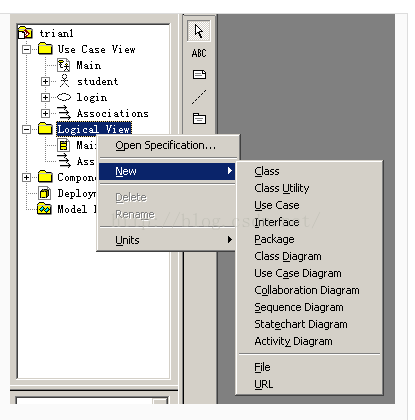
Rose界面的五大部分是浏览器、文档工具、工具栏、框图窗口和日志。



1、浏览器：用于在模型中迅速漫游。  
2、文档工具：用于查看或更新模型元素的文档。  
3、工具栏：用于迅速访问常用命令。  
4、框图窗口：用于显示和编辑一个或几个UML框图。  
5、日志：用于查看错误信息和报告各个命令的结果。

### 1.2浏览器和视图

浏览器是层次结构，用于在Rose模型中迅速漫游。在浏览器中显示了模型中增加的一切，如参与者、用例、类、组件等等。Rose浏览器见图1-2。  
浏览器中包含四个视图：Use Case视图、Logical视图、Component视图和Deployment视图。点击每个视图的右键，选择new就可以看到这个视图所包含的一些模型元素。



### 1.3框图窗口

我们可以浏览模型中的一个或几个UML框图。改变框图中的元素时，Rose自动更新浏览器。同样用浏览器改变元素时，Rose自动更新相应框图。这样，Rose就可以保证模型的一致性。

## 2.UML各类框图的建立

### 2. 1建立用例图use case diagram

从用例图中我们可以看到系统干什么，与谁交互。用例是系统提供的功能，参与者是系统与谁交互，参与者可以是人、系统或其他实体。一个系统可以创建一个或多个用例图。

用例图由参与者(Actor)、用例(Use Case)、系统边界、箭头组成，用画图的方法来完成。

1. 参与者

参与者不是特指人，是指系统以外的，在使用系统或与系统交互中所扮演的角色。

因此参与者可以是人，可以是事物，也可以是时间或其他系统等等。

还有一点要注意的是，参与者不是指人或事物本身，而是表示人或事物当时所扮演的角色。

比如小明是图书馆的管理员，他参与图书馆管理系统的交互，这时他既可以作为管理员这个角色参与管理，也可以作为借书者向图书馆借书，在这里小明扮演了两个角色，是两个不同的参与者。

参与者在画图中用简笔人物画来表示，人物下面附上参与者的名称。

参与者的画法：

  
  
2、用例  
是对包括变量在内的一组动作序列的描述，系统执行这些动作，并产生传递特定参与者的价值的可观察结果。这是UML对用例的正式定义。

我们可以简单的理解为：用例是参与者想要系统做的事情。



3、系统边界  
系统边界是用来表示正在建模系统的边界。边界内表示系统的组成部分，边界外表示系统外部。系统边界在画图中方框来表示，同时附上系统的名称，参与者画在边界的外面，用例画在边界里面。因为系统边界的作用有时候不是很明显，所以我个人理解，在画图时可省略。

4、箭头  
箭头用来表示参与者和系统通过相互发送信号或消息进行交互的关联关系。

箭头尾部用来表示启动交互的一方，箭头头部用来表示被启动的一方，其中用例总是要由参与者来启动。

USE CASE图的作用

主要的作用有三个:(1)获取需求;(2)指导测试;(3)还可在整个过程中的其它工作流起到指导作用。

use case图中的关系：

用例图中包含的元素除了系统边界、角色和用例，另外就是关系。

关系包括用例之间的关系，角色之间的关系，用例和角色之间的关系。

1、角色之间的关系

由于角色实质上也是类，所以它拥有与类相同的关系描述，即角色之间存在泛化关系，

泛化关系的含义是把某些角色的共同行为提取出来表示为通用的行为。

2、用例之间的关系:  
包含关系:基本用例图的行为包含了另一个用例的行为。

基本用例描述在多个用例中都有的公共行为。

包含关系本质上是比较特殊的依赖关系。它比一般的依赖关系多了一些语义。在包含关系中箭头的方向是从基本用例到包含用例。在UML1.1中用例之间是使用和扩展这两种关系，这两种关系都是泛化关系的版型。在UML1.3以后的版本中用例之间是包含和扩展这两种关系

3、泛化关系:

代表一般与特殊的关系。

它的意思和面向对象程序设计中的继承的概念是类似的。

不同的是继承使用在实施阶段，泛化使用在分析、设计阶段。在泛化关系中子用例继承了父用例的行为和含义，子用例也可以增加新的行为和含义或者覆盖父用例中的行为和含义。

4、扩展关系：

扩展关系的基本含义和泛化关系类似，但在扩展USE CASE图关系中，对于扩展用例有更多的规则限制，基本用例必须声明扩展点，而扩展用例只能在扩展点上增加新的行为和含义。与包含关系一样，扩展关系也是依赖关系的版型。在扩展关系中，箭头的方向是从扩展用例到基本用例，这与包含关系是不同的。

5、用例的泛化、包含、扩展关系的比较：

一般来说可以使用"is a"和"has a"来判断使用那种关系。

泛化和扩展关系表示用例之间是"is a"关系，包含关系表示用例之间是"has a"关系。

扩展与泛化相比多了扩展点，扩展用例只能在基本用例的扩展点上进行扩展。

在扩展关系中基本用例是独立存在。

在包含关系中在执行基本用例的时候一定会执行包含用例。如果需要重复处理两个或多个用例时可以考虑使用包含关系，实现一个基本用例对另一个的引用。当处理正常行为的变形是偶尔描述时可以考虑只用泛化关系。当描述正常行为的变形希望采用更多的控制方式时，可以USE CASE图在基本用例中设置扩展点，使用扩展关系。

扩展关系比较难理解，如果把扩展关系看作是带有更多规则限制的泛化关系，可以帮助理解。

通常先获得基本用例，针对这个用例中的每一个行为提问:该步骤会出什么差错?该步骤有不同的情况工作怎样以不同的方式进行等，把所有的变化情况都标识为扩展。通常基本用例很容易构造，

而扩展用例需要反复分析、验证。当我们发现已经存在的两个用例间具有某种相似性时，可以把相似的部分从两个用例中抽象出来单独作为一个用例，该用例被这两个用例同时使用，这个抽象出的用例和另外两个用例形成包含关系。

6、用例之间的关系举例  
      1、包含:业务中，总是存在着维护某某信息的功能，如果将它作为一个用例，那新建、编辑以及修改都要在用例详述中描述，过于复杂;如果分成新建用例、编辑用例和删除用例，则划分太细。这时包含关系可以用来理清关系。  
  
     2、扩展:系统中允许用户对查询的结果进行导出、打印。对于查询而言，能不能导出、打印查询都是一样的，导出、打印是不可见的。导出、打印和查询相对独立，而且为查询添加了新行为。  
  
  
    3、泛化:子用例将继承父用例的所有结构、行为和关系。子用例可以使用父用例的一段行为，也可以重载它。父用例通常是抽象的。  
  
7、关于用例图的总结

    用例图展示了用例之间以及同用例参与者之间是怎样相互联系的。

    用例图用于对系统、子系统或类的行为进行可视化，使用户能够理解如何使用这些元素，并使开发者能够实现这些元素。  
将每个系统中的用户分出工作状态的属性和工作内容，方便建模，防止功能重复和多余的类。  
用例图定义了系统的功能需求，它是从系统的外部看系统功能，并不描述系统内部对功能的具体实现。

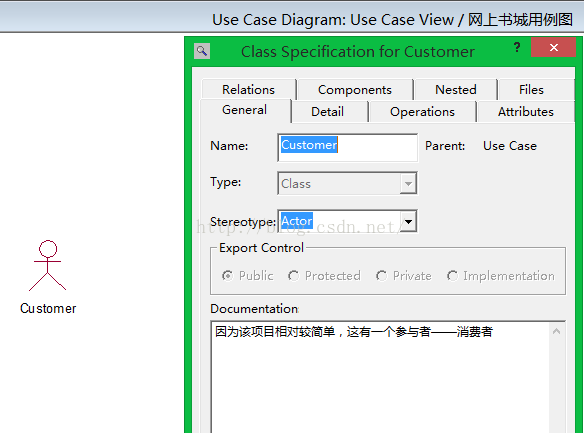
#### 2.1.1创建用例图

1、在浏览器内的Use Case视图中，双击Main，让新的用例图显示在框图窗口中。

2、新建一个包（右击Use Case视图，选择new→package，并命名），然后右击这个新建包的，选择new→use case diagram。  
3、对系统总的用例一般画在Use Case视图中的Main里，如果一个系统可以创建多个用例图，则可以用包的形式来组织。

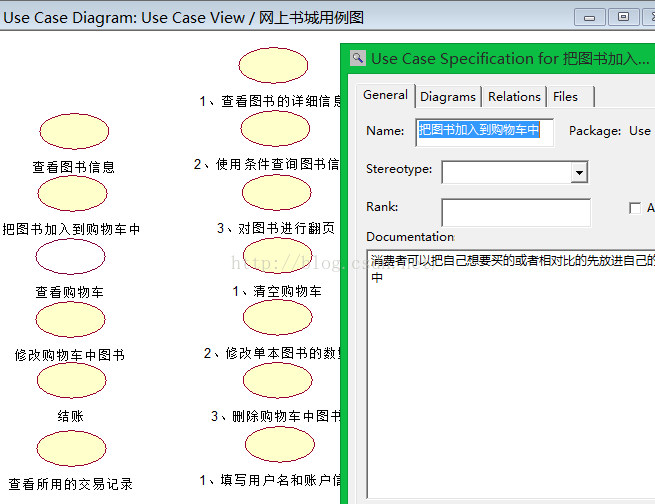
#### 2.1.2创建参与者

(1) 在工具栏中选择“Actor”，光标的形状变成加号。  
(2) 在用例图中要放置参与者符号的地方单击鼠标左键，键入新参与者的名称，如“客户”。  
若要简要的说明参与者，可以执行以下步骤：  
(1) 在用例图或浏览器中双击参与者符号，打开对话框，而且已将原型(stereotype)设置定义为“Actor”。  
(2) 打开“General”选项卡，在documentation字段中写入该参与者的简要说明。  
(3) 单击OK按钮，即可接受输入的简要说明并关闭对话框。



#### 2.1.3创建用例

(1) 在工具栏中选择“Use Case”，光标的形状变成加号。  
(2) 在用例图中要放置用例符号的地方单击鼠标左键，键入新用例的名称，如“存款”。  
若要简要的说明用例，可以执行以下步骤：  
(1) 在用例图或浏览器中双击用例符号，打开对话框，接着打开“General”选项卡。  
(2) 在documentation字段中写入该用例的简要说明。  
(3) 单击OK按钮，即可接受输入的简要说明并关闭对话框。



#### 2.1.4记录参与者和用例之间的关系（图2-1-4）

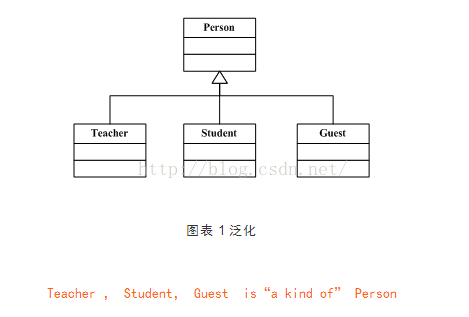
三个箭头的含义：  
Unidirection Association 单向关联

Dependency or instantiates 依赖和实例化

Generalization 泛化

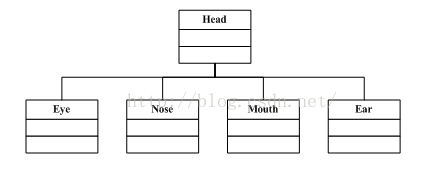
Association  关联   
Unidirectional Association  单向关联   
Association Class  关联类   
Dependency or instantiates依赖或实例化   
Generalization泛化  继承   
Realize实现  
Aggregation  聚合  
Unidirectional Aggregation  单向聚合   
Class 类   
Parameterized Class参数化类   
Class Utility类的实用程序   
Parameterized Class Utility参数化类实用程序

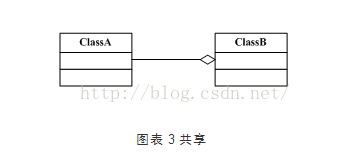
概念讲解  
  
1、泛化（Generalization）



在上图中，空心的三角表示继承关系（类继承），在UML的术语中，这种关系被称为泛化（Generalization）。Person(人)是基类，Teacher(教师)、Student(学生)、Guest(来宾)是子类。   
若在逻辑上B是A的“一种”，并且A的所有功能和属性对B而言都有意义，则允许B继承A的功能和属性。   
例如，教师是人，Teacher 是Person的“一种”（a kind of ）。那么类Teacher可以从类Person派生（继承）。   
如果A是基类，B是A的派生类，那么B将继承A的数据和函数。

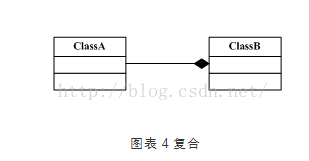
如果类A和类B毫不相关，不可以为了使B的功能更多些而让B继承A的功能和属性。   
若在逻辑上B是A的“一种”（a kind of ），则允许B继承A的功能和属性。

2、聚合（组合）   
  
  
若在逻辑上A是B的“一部分”（a part of），则不允许B从A派生，而是要用A和其它东西组合出B。   
例如，眼（Eye）、鼻（Nose）、口（Mouth）、耳（Ear）是头（Head）的一部分，所以类Head应该由类Eye、Nose、Mouth、Ear组合而成，不是派生（继承）而成。   
聚合的类型分为无、共享(聚合)、复合(组合)三类。   
  
   
1、聚合（aggregation）

  
上面图中，有一个菱形（空心）表示聚合（aggregation）（聚合类型为共享），

聚合的意义表示has-a关系。聚合是一种相对松散的关系，聚合类B不需要对被聚合的类A负责。

2、组合（composition）

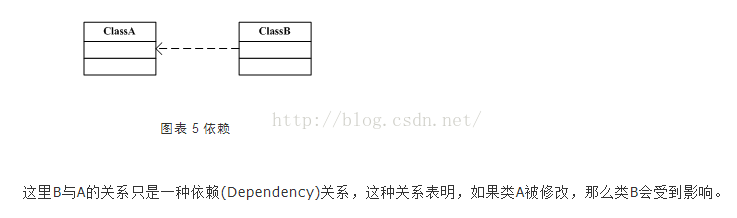


这幅图与上面的唯一区别是菱形为实心的，

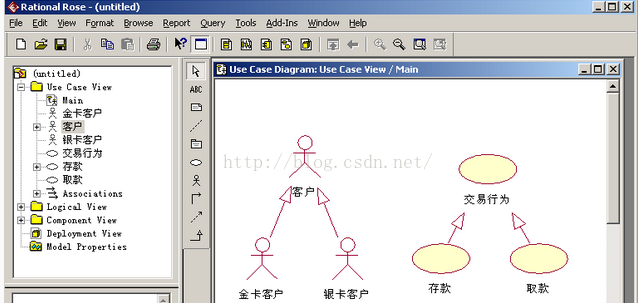
它代表了一种更为坚固的关系——组合（composition）（聚合类型为复合）。

组合表示的关系也是has-a，不过在这里，A的生命期受B控制。即A会随着B的创建而创建，随B的消亡而消亡。

3、依赖(Dependency)



2.1.5增加泛化关系  
(1) 从工具栏中选择泛化关系箭头。  
(2) 从子用例拖向父用例，也可从子参与者拖向父参与者。  
简要说明关系执行的步骤同上类似。

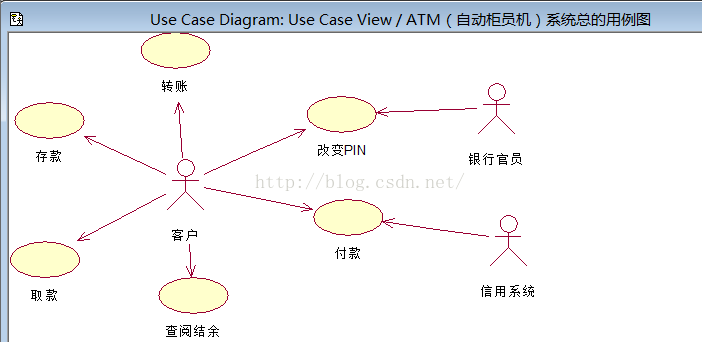


#### ★练习：画ATM（自动柜员机）系统总的用例图

理解：

对于银行的客户来说:可以通过ATM机启动几个用例：存款、取款、查阅结余、付款、转帐和改变PIN（密码）。

银行官员也可以启动改变PIN这个用例。参与者可能是一个系统，这里信用系统就是一个参与者，因为它是在ATM系统之外的。箭头从用例到参与者表示用例产生一些参与者要使用的信息。



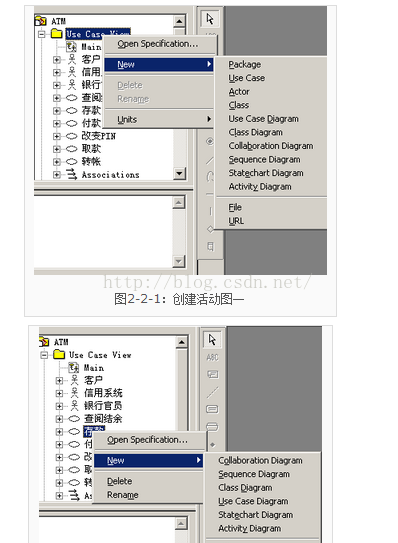
### 2.2建立活动图 activity diagram

活动图显示了从活动到活动的流。

活动图可以在分析系统业务时用来演示业务流，也可以在收集系统需求的时候显示一个用例中的事件流。活动图显示了系统中某个业务或者某个用例中，要经历哪些活动，这些活动按什么顺序发生。

#### 2.2.1创建活动图（图2-2-1）

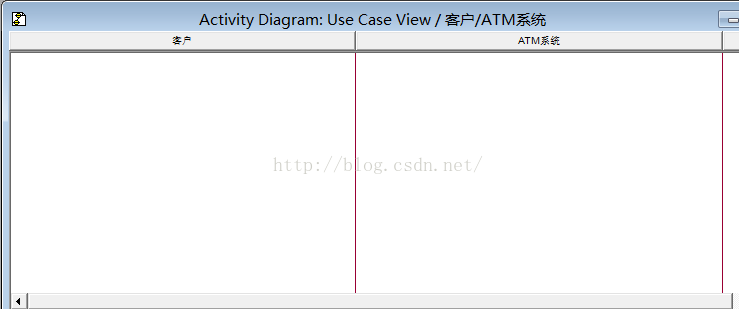
(1) 用于分析系统业务：在浏览器中右击Use Case视图，选择new→activity diagram。  
(2) 用于显示用例中的事件流：在浏览器中选中某个用例，然后右击这个用例，选择new→activity diagram。



#### 2.2.2增加泳道 (图2-2-2)

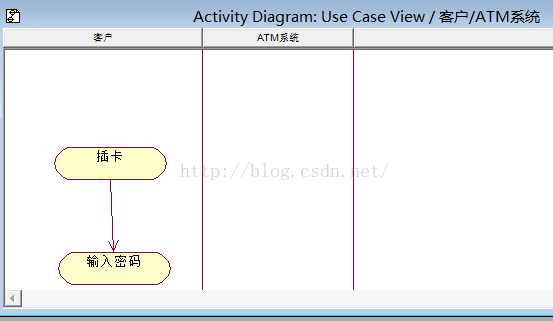
泳道是框图里的竖段，包含特定人员或组织要进行的所有活动。

可以把框图分为多个泳道，每个泳道对应每个人员或组织。  
在工具栏选择swimlane按钮，然后单击框图增加泳道，最后用人员或组织给泳道命名。



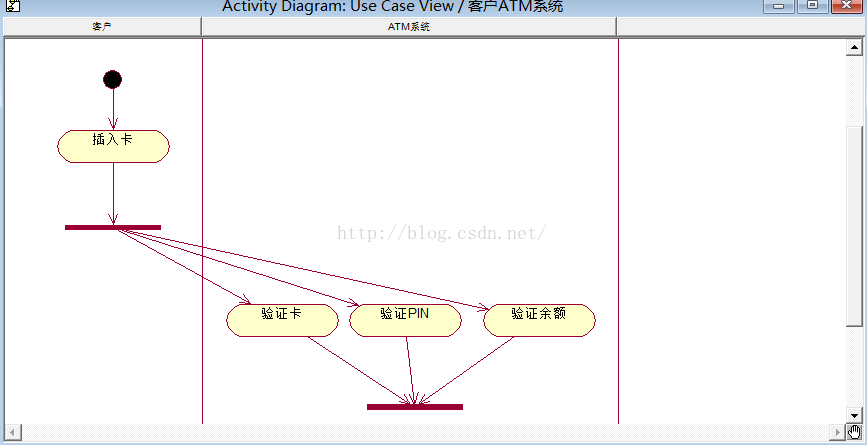
注意:先创建一个活动图，才会有swinlane

2.2.3增加活动并设置活动的顺序（图2-2-3）  
(1) 在工具栏中选择Activity 按钮，单击活动图增加活动，命名活动。  
(2) 在工具栏中选择Transition按钮，把箭头从一个活动拖向另一个活动。



#### 2.2.4增加同步（图2-2-4）

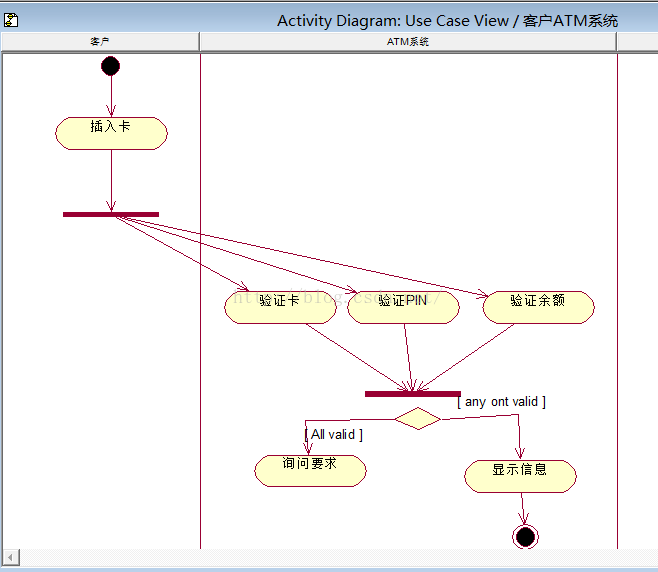
(1)选择synchronization工具栏按钮，单击框图来增加同步棒。  
(2)画出从活动到同步棒的交接箭头，表示在这个活动之后开始并行处理。  
(3) 画出从同步棒到可以并行发生的活动之间的交接箭头。  
(4) 创建另一同步棒，表示并行处理结束。  
(5) 画出从同步活动到最后同步棒之间的交接箭头，表示完成所有这些活动之后，停止并行处理。



2.2.5增加决策点(图2-2-5)  
  
  
决策点表示可以采取两个或多个不同的路径。从决策到活动的交接箭头要给出保证条件，控制在决策之后采取什么路径。保证条件应该是互斥的。  
  
  
(1) 选择decision工具栏按钮，单击框图增加决策点。  
(2) 拖动从决策到决策之后可能发生的活动之间的交接,双击交接，打开“detail”选项卡，在Guard Condition字段中写入保证条件。

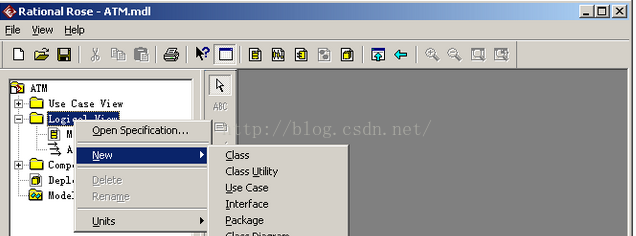
#### ★练习：画ATM系统中“客户插入卡”的活动图

理解：客户插入信用卡之后，可以看到ATM系统运行了三个并发的活动：验证卡、验证PIN(密码)和验证余额。这三个验证都结束之后，ATM系统根据这三个验证的结果来执行下一步的活动。如果卡正常、密码正确且通过余额验证，则ATM系统接下来询问客户有哪些要求也就是要执行什么操作。如果验证卡、验证PIN(密码)和验证余额这三个验证有任何一个通不过的话，ATM系统就把相应的出错信息在ATM屏幕上显示给客户。

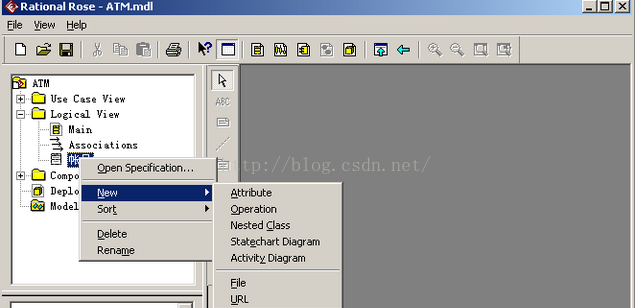


### 2.3建立类图class diagram

类图显示系统之中类和类之间的交互。  
2.3.1创建类(图2-3-1)  
在Rational Rose中可以通过几种途径来创建类。最简单的方法是利用模型的Logic视图中的类图标和绘图工具，在图中创建一个类。或者，在浏览器中选择一个包并使用快捷菜单的new→class。一旦创建了一个类，就可以通过双击打开它的对话框并在Documentation字段中添加文本来对这个类进行说明。



2.3.2创建方法(图2-3-2)  
(1) 选择浏览器中或类图上的类。  
(2) 使用快捷菜单的new→Operation  
(3) 输入方法的名字，可在Documentation字段中为该方法输入描述其目的的简要说明。

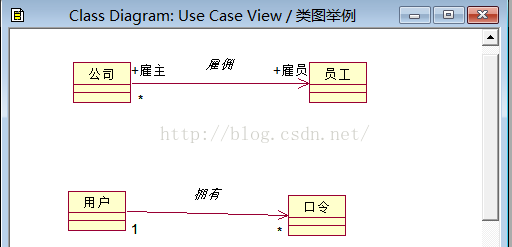


2.3.3创建属性(图2-3-2)  
  
  
(1) 选择浏览器中或类图上的类。  
(2) 使用快捷菜单的new→Attribute。  
(3) 输入属性的名字，可在Documentation字段中为该属性输入描述其目的的简要说明。  
2.3.4创建类图(图2-3-3)  
  
  
右击浏览器内的Logical视图，选择new→class diagram。把浏览器内的类拉到类图中即可。

2.3.5创建类之间的关系  
类之间的关系：

继承、关联、聚合、依赖

(1) 类之间的关系在工具栏中显示。  
(2) 对于关联关系来说，双击关联关系，就可以在弹出的对话框中对关联的名称和角色进行编辑(图2-3-4)。  
(3) 编辑关联关系的多重性：右单击所要编辑的关联的一端，从弹出的菜单中选择Multiplicity，然后选择所要的基数



**★练习：画ATM系统中取款这个用例的类图**

理解：类图显示了取款这个用例中各个类之间的关系，由四个类完成：读卡机、账目、ATM屏幕和取钱机。

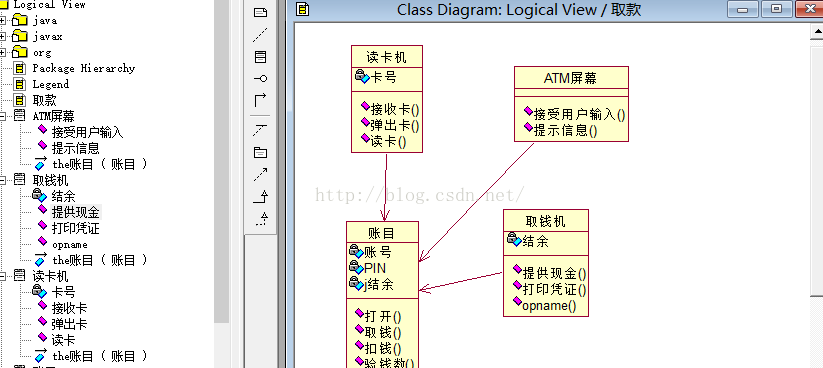
类图中每个类都是用方框表示的，分成三个部分。第一部分是类名；第二部分是类包含的属性，属性是类和相关的一些信息，如账目类包含了三个属性：账号、PIN(密码)和结余；最后一部分包含类的方法，方法是类提供的一些功能，例如帐目类包含了四个方法：打开、取钱、扣钱和验钱数。

类之间的连线表示了类之间的通信关系。例如，账目类连接了ATM屏幕，因为两者之间要直接相互通信；取钱机和读卡机不相连，因为两者之间不进行通信。

有些属性和方法的左边有一个小锁的图标，表示这个属性和方法是private的(UML中用’－’表示)，该属性和方法只在本类中可访问。

没有小锁的，表示public(UML中用’+’表示)，即该属性和方法在所有类中可访问。

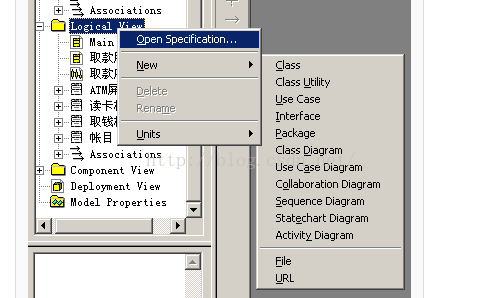
若是一个钥匙图标，表示protected(UML中用’＃’表示)，即属性和方法在该类及其子类中可访问。



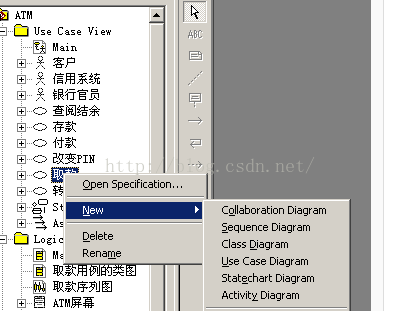
### 2.4建立交互图interaction diagram

#### 2.4.1序列图sequence diagram

序列图显示用例中的功能流程。  
创建序列图(图2-4-1)  
1、在浏览器内的Logic 视图中单击鼠标右键，选择new→sequence diagram就新建了一张序列图。



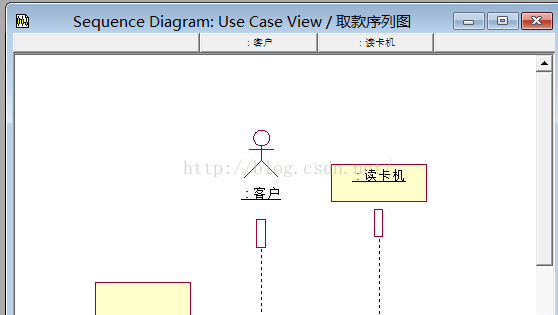
2、也可以在浏览器中use case视图中选择某个用例，然后右击这个用例，选择new→sequence diagram。



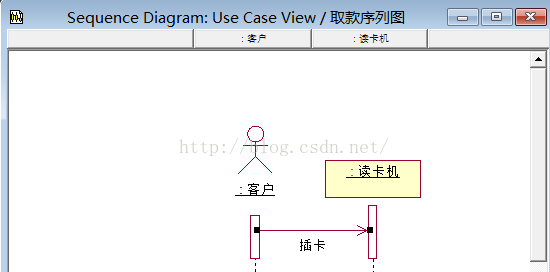
在序列图中放置参与者和对象(图2-4-2)

在序列图中的主要元素之一就是对象，相似的对象可以被抽象为一个类。序列图中的每个对象代表了某个类的某一实例。

1. (1) 把用例图中的该用例涉及的所有参与者拖到sequence图中。
2. (2) 选择工具栏中的object按钮，单击框图增加对象。可以选择创建已有类的对象，也可以在浏览器中新建一个类，再创建新的类的对象。双击对象，在弹出的对话框中的“class”里确定该对象所属的类。
3. (3) 对象命名：对象可以命名也可没名字。双击对象，在弹出的对话框中的“name”里给对象取名

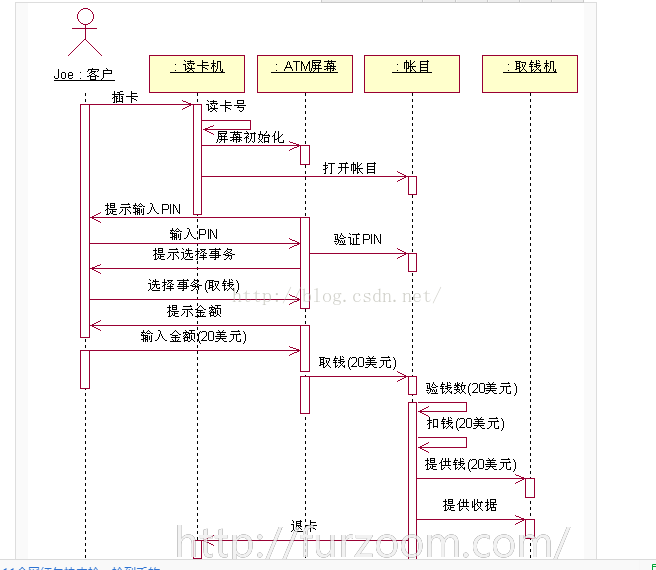


说明对象之间的消息(图2-4-3)  
(1) 选择message工具栏按钮。  
(2) 单击启动消息的参与者或对象，把消息拖到目标对象和参与者。  
(3) 命名消息。双击消息，在对话框中“General”里的“name” 中输入消息名称。



★练习：画某客户Joe取20美元的序列图  
理解：序列图显示了用例中的功能流程。我们对取款这个用例分析，

它有很多可能的程序，如想取钱而没钱，想取钱而PIN错等等，正常的情况是取到了钱，下面的序列图就对某客户Joe取20美元，分析它的序列图。  
    
  
序列图的顶部一般先放置的是取款这个用例涉及的参与者，然后放置系统完成取款用例所需的对象，每个箭头表示参与者和对象或对象之间为了完成特定功能而要传递的消息。  
  
  
取款这个用例从客户把卡插入读卡机开始，然后读卡机读卡号，初始化ATM屏幕，并打开Joe的账目对象。屏幕提示输入PIN，Joe输入PIN(1234)，然后屏幕验证PIN与账目对象，发出相符的信息。屏幕向Joe提供选项，Joe选择取钱，然后屏幕提示Joe输入金额，它选择20美元。然后屏幕从账目中取钱，启动一系列账目对象要完成的过程。首先，验证Joe账目中至少有20美元；然后，它从中扣掉20美元，再让取钱机提供20美元的现金。Joe的账目还让取钱机提供收据，最后它让读卡机退卡。

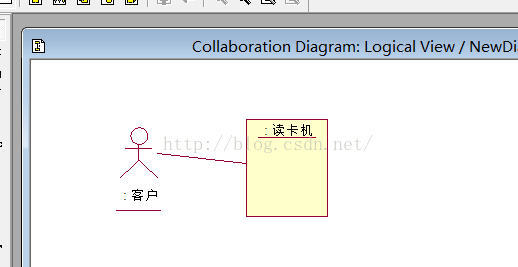


2.4.2协作图collaboration diagram

协作图的创建，以及在协作图中放置参与者和对象和序列图类似。只不过对象之间的链接有所不同。

增加对象链接(图2-4-4)

1. (1) 选择Object Link 工具栏按钮。
2. (2) 单击要链接的参与者或对象。
3. (3) 将对象链接拖动到要链接的参与者或对象。



加进消息(图2-4-5)

1. (1) 选择Link Message或Reverse Link Message工具栏按钮。
2. (2) 单击要放消息的对象链接。
3. (3) 双击消息，可以在弹出的对话框里为消息命名

1.Rose如何隐藏类的属性和操作？  
右击类，选Options->Suppress Attributes/Suppress Operations

2.Rose如何表示类的约束？  
在工具箱中选ABC图标（Text Box），输入用大括号括起来的自由文本。

3.Rose如何表示抽象类？  
打开类规格窗口。选择Detail页签，复选Abstract

4.Rose中接口的三种表示方法  
右击接口，点Options->Steretype Display->None/Label/Decoration/Icon

5.Rose如何表示关联类？  
关联类是关联和类的结合，在工具箱中选Association Class，连接类和关联

6.Rose如何表示模板类？  
打开类规格窗口，在General页签中的Type下拉框中选择ParameterizeClass

7.Rose如何表示嵌套类？  
打开类规格窗口，在Nested 页签中insert一个类，即新建了一个嵌套类的内部类

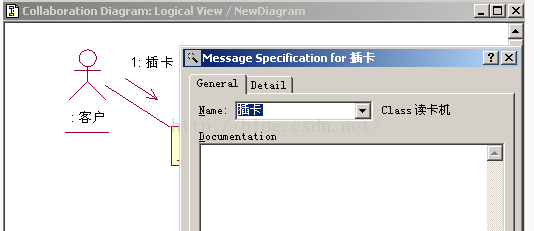
8.Rose中如何表示不同的依赖？  
在依赖的旁边加自由文本，在工具箱中选ABC图标（Text Box），输入用书名号括起来的自由文本。

9.Rose中如何表示类与接口的实现关系？  
在工具箱中选Realize，连接类与接口（从类到接口），改变接口的显示方式。  
10.Rose中如何表示双向关联？  
打开关联规格窗口，在Role A Detail或Role B Detail中取消对Navigable的选取。  
11.Rose中如何表示关联的角色名和多重值？  
右击关联的一端，点Role name增加角色名；右击关联的一端，点Multiplicity->选取值  
12.Rose中如何表示聚合和组合关系？  
打开关联规格窗口，在Role A Detail或Role B Detail中只选取Aggregate，即为聚合关系；在Role A Detail或Role B Detail中选取Aggregate和By Value，即为组合关系。  
Rose中的组合关系也可以没有箭头，打开关联规格窗口，在Role A Detail或Role B Detail中取消对Navigable的选取  
13.Rose中如何表示关联的限定符？  
右击关联，点New Key/Qualifier  
14.Rose中如何画对象图？  
Rose不支持对象图，但可以用协作图表达对象之间的关系  
15.Rose图如何输出到图片文件？  
    方法一 ：针对每一张绘制好的uml图形，使用ctrl+A  ctrl+C   最后打开word, 执行ctrl+V就可以了，一般这种情况下是可以将每张图都粘贴到文档中然后就可以发送给他人共享了。  
              如果中文乱码 ctrl+A全部选中，右键Format->Font把字体改为宋体在word里面就不会乱码  
              copy到windows的画图如果乱码的法，一般是设置仿宋gb2312    楷书gb2312 就不会乱码  
   
    方法二 ：针对每一张绘制好的uml图形，右击UML图，选择“Print Diagram”可以导出为Microsoft Document Imaging(MDI)文件。  
    方法三 ：上两个方法可以有效地将每张图片保存，但是一旦模型中有多个图的话就不是很方便，建议使用rational rose中的publish web功能，这个功能位于rational rose的菜单tools->web publisher，点击进去后修改导出的文件路径即可把弄好的uml模型都导出到web页面中，其中既有模型的图片还有各个图之间的关系，模型之间的联系。单个图可以在相应的目录中找到。

16.如何修改Rose图形中的字体？  
改变缺省字体的方法是，点Tools->Options->General，修改Default font。  
改变图形中已有文本的字体的方法是，右击图形中的元素，然后点“Format->Font”；或者在图形中用ctrl+A全选，然后右击图形，点“Format->Font”；  
17.Rose中如何对消息进行多级编号？  
选择Tools-> Options； 单击Diagram标签；同时选中Sequence Numbering和Hierarchical Messages复选框。  
   
18.Rose协作图中消息（link message）的画法，Data Token的画法  
先画连接（object link），然后选择工具箱中的消息图标，点连接。选择工具箱中的Data Token图标，点消息。

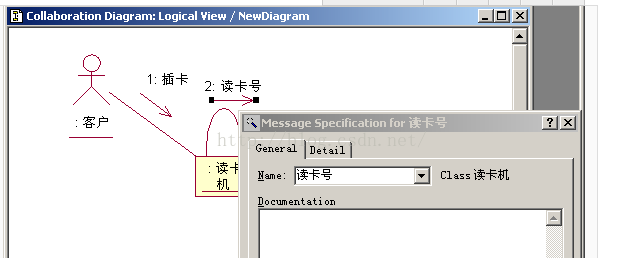
19. Rose协作图中多对象的画法  
            右击对象，点Open Specification，选择复选框“Multiple instance”  
20.Rose新建模型时，没有选择模板的对话框，如何设置才能出现选择模板的对话框？  
            Add-Ins->Add-In Manager，选择Framework Wizard  
   
 21.顺序图与协作图的转换  
      从顺序图转换为协作图：打开顺序图，选择Browse->Create collaboration diagram或按F5  
      从协作图转换为顺序图：打开协作图，选择Browse->Create sequence diagram或按F5

--------------------------------------------------------------------rose 2007小结-----------------------------------------------------



18.Rose协作图中消息（link message）的画法，Data Token的画法  
先画连接（object link），然后选择工具箱中的消息图标，点连接。选择工具箱中的Data Token图标，点消息。

自反链接(图2-4-6)  
  
  
(1) 选择Link to Self工具栏按钮。  
(2) 单击要链接的对象，会增加一个消息的箭头。  
(3) 双击消息，命名自反链接。



★练习：画某客户Joe取20美元的协作图  
理解：协作图显示的信息和序列图是相同的，只是协作图用不同的方式显示而已。

序列图显示的是对象和参与者随时间变化的交互，

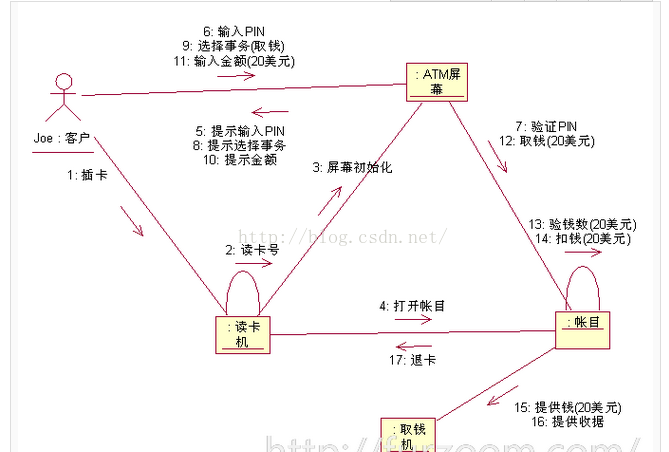
而协作图则不参照时间而显示对象与参与者的交互。

2.4.3序列图和协作图之间的转换  
  
  
在序列图中按F5键就可以创建相应的协作图；

同样，在协作图中按F5键就可以创建相应的序列图。

序列图和协作图是同构的，也就是说两张图之间的转换没有任何信息的损失。

例如，Joe取20美元的协作图中我们可以看到，读卡机和Joe的账目两个对象之间的交互：读卡机指示Joe的账目打开，Joe的账目让读卡机退卡。直接相互通信的对象之间有一条直线，例如ATM屏幕和读卡机直接相互通信，则其间画一条直线。没有画直线的对象之间不直接通信。



### 2.5建立状态图statechart diagram

状态图显示了对象的动作行为，

1、显示对象可能存在的各种状态，

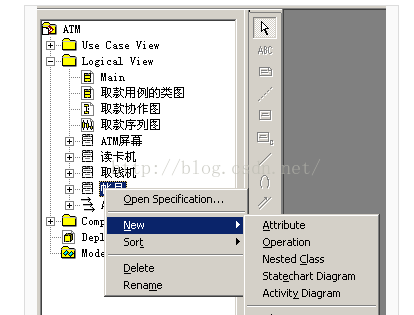
2、对象创建时的状态，

3、对象删除时的状态，

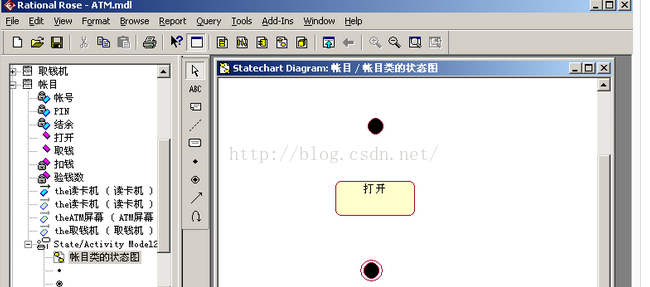
4、对象如何从一种状态转移到另一种状态，

5、对象在不同状态中干什么。

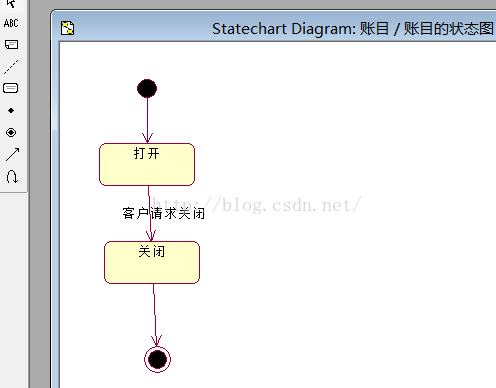
2.5.1创建状态图(图2-5-1)  
(1) 在浏览器中右击类。  
(2) 选择new→statechart diagram，对该类创建一个状态图，并命名该图。



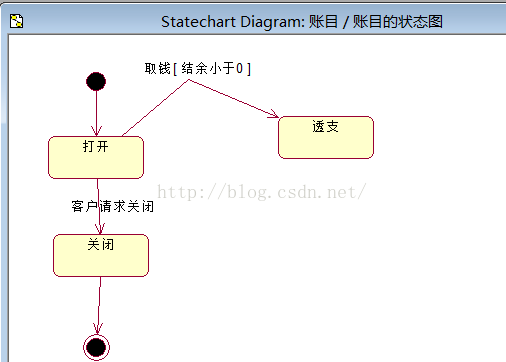
2.5.2在图中增加状态，初始和终止状态(图2-5-2)  
  
  
(1) 选择工具栏的state按钮，单击框图增加一个状态，双击状态命名。  
(2) 选择工具栏的start state 和end state ，单击框图增加初始状态和终止状态。初始状态是对象首次实例化时的状态，状态图中只有一个初始状态。终止状态表示对象在内存中被删除之前的状态，状态图中有0个、1个或多个终止状态。



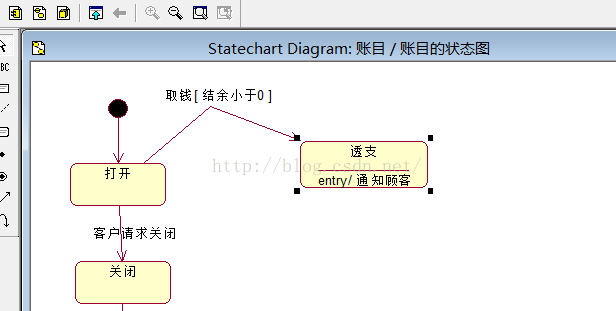
2.5.3状态之间增加交接  
(1) 选择state transition工具栏按钮。  
(2) 从一种状态拖到另一种状态。  
(3) 双击交接弹出对话框，可以在“General”中增加事件(Event) (图2-5-3)，在“Detail”中增加保证条件(Guard Condition) (图2-5-4)等交接的细节。事件用来在交接中从一个对象发送给另一个对象，保证条件放在中括号里，控制是否发生交接。



(3) 双击交接弹出对话框，可以在“General”中增加事件(Event) (图2-5-3)，在“Detail”中增加保证条件(Guard Condition) (图2-5-4)等交接的细节。



2.5.4在状态中增加活动(图2-5-5)  
  
(1)右击状态并选择open specification。  
(2)选择Action标签，右击空白处并选择Insert。  
(3)双击新活动(清单中有“Entry/”)打开活动规范，在“name”中输入活动细节。



★练习：账目类的状态图  
理解：

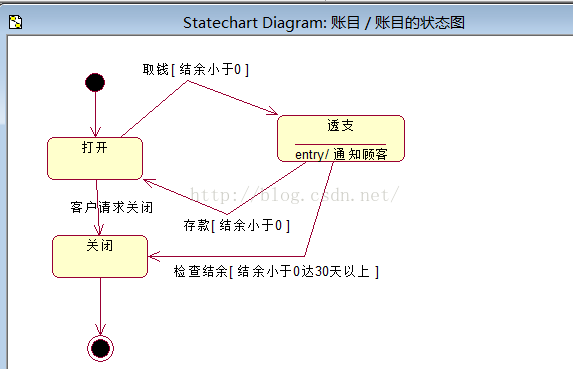
银行账目可能有几种不同的状态，可以打开、关闭或透支。

账目在不同状态下的功能是不同的，账目可以从一种状态变到另一种状态。

例如，账目打开而客户请求关闭账目时，账目转入关闭状态。

客户请求是事件，事件导致账目从一个状态过渡到另一个状态。

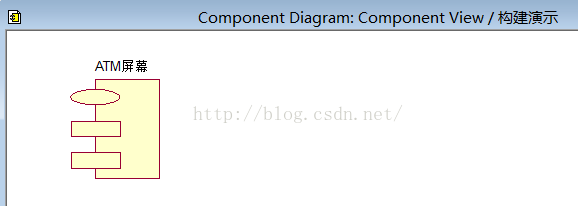
如果账目打开而客户要取钱，则账目可能转入透支状态。这发生在账目结余小于0时，框图中显示为[结余<0]。方括号中的条件称为保证条件，控制状态的过渡能不能发生。  
  
对象处在特定状态时可能发生某种事件。例如，账目透支时，要通知客户。



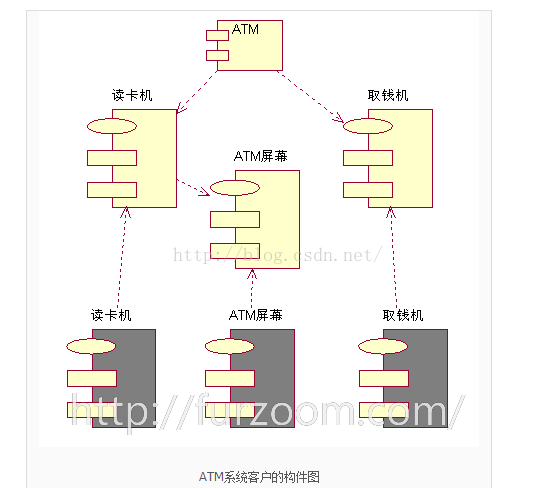
### 2.6建立构件图component diagram

构件图显示模型的物理视图，也显示系统中的软件构件及其相互关系。模型中的每个类映射到源代码构件。一旦创建构件，就加进构件图中，然后画出构件之间的相关性。构件间的相关性包括编译相关性和运行相关性。  
  
  
2.6.1创建构件图(图2-6-1)  
  
  
(1) 右单击浏览器中的Component 视图。  
(2) 选择New→Component Diagram，并命名新的框图。

2.6.2把构件加入框图(图2-6-2)  
  
  
(1) 选择Component工具栏按钮，单击框图增加构件，并命名构件。  
(2) 右单击构件，选择Open Specification，在“stereotype”中设置构件版型。



★练习：ATM系统客户的构件图  
理解： 例如我们用C++建立系统，每个类有自己的头文件和体文件，因此图中每个类映射自己的构件，例如ATM屏幕类映射两个ATM屏幕构件。这两个ATM屏幕构件表示ATM屏幕类的头和体。阴影构件称为包体，表示C++中ATM屏幕类的体文件(.cpp)，构件版型是package body。无阴影的构件称为包规范，这个包规范表示C++类的头文件(.H)，构件版型是package specification。构件ATM.exe是个任务规范，表示处理线程，是一个可执行程序。



### 2.7建立实施图deployment diagram

实施图显示网络的物理布局，系统中涉及的处理器、设备、连接和过程。

处理器是网络中处理功能所在的机器，包括服务器和工作站，不包括打印机扫描仪之类的设备。处理器用来运行进程（执行代码）。一个项目只有一个实施图。  
  
  
2.7.1创建实施图(图2-7-1)  
(1) 双击Deployment视图。  
(2) 选择Processor工具栏按钮，单击框图增加处理器，并命名处理器。  
(3) 在Deployment视图中右击处理器并选择New→Process，命名进程。  
(4) 在框图中右击处理器，对“Show Processes”打勾，可以在框图中显示该处理器的进程。

2.7.2把设备加入到框图中(图2-7-2)  
  
  
(1) 选择Device工具栏按钮。  
(2) 单击框图增加设备，并命名。

2.7.3把连接加进框图(图2-7-3)  
  
  
(1) 选择Connection工具栏按钮。  
(2) 单击要连接的一个处理器或设备，拖动到要连接的另一个处理器或设备。  
(3) 命名连接。

★练习：ATM系统的实施图  
理解：ATM系统的实施图显示了系统的主要布局。ATM客户机可执行文件在不同地点的多个ATM上运行。ATM客户机通过专用网与地区ATM服务器通信。ATM服务器可执行文件在地区ATM服务器上执行。地区ATM服务器又通过局域网与运行Oracle的银行数据库服务器通信。最后，打印机与地区ATM服务器连接。  
  
  
ATM系统采用了三层结构，分别针对数据库、地区ATM服务器和客户机。

