# 2020-05-11

回顾：

参与者之间的关系：泛化关系

参与者与用例之间的关系：关联关系——主要关系

用例之间的关系：泛化关系、包含关系、扩展关系

一、用例模型的构成

1、用例图

2、用例规约，用例图和每一个用例规约之后所得

表示形式：（1）纯文本（2）表格形式

（1）简要说明

（2）事件流：a基本流、b扩展流

（3）用例场景

（4）特殊需求

（5）前置条件

（6）后置条件

二、事件流

1、描述

（1）可描述的语言，便于用户阅读

（2）从主观出发

（3）以系统为主语

（4）不要涉及开发和界面细节

（5）分支和循坏

2、用例规约易出现的错误

（1）没有交互，从某个片面的角度出发

3、扩展流

描述：起点、条件、动作、恢复

4、用例场景

5、特殊需求

专属于该用例的特殊需求。

6、前置条件和后置条件

三、规约

1、补充规约

2、词汇表

4、用例模型的检查和调整

# 2020-05-12

智慧树第三章章测试本周内完成

测试：

“4+1”架构中用例视图将四个视图合为一个整体。

类的可见性可以通过UML的修饰来表示出来。

UML每个图形化符号都暗示了该元素的规约。

UML关系负责连接同类型或者不同类型的元素。

类

一、概念

类的模型

使用类图来可视化这些类和它们的关系。

为数据库模式建立模型。

类图主要是由类、类之间的关系构成的。

类图 = 类 + 关系 + 约束

二、表示

只有类型是类图中唯一不可缺少的部件。

三格：类名+属性+操作

2.1类的名字

2.2类的属性

可见性/属性名：属性的类型【多重性】=缺省值【特性描述和限制条件】

“/”表示当前属性是导出属性，只读，用户不能更改它的值。

1、属性名

2、属性类型

3、多重性

1：只有一个

0..1：0或1个

0..n：0-n个对象

4、属性字符

说明属性具有的其他性质

例如：unique

2.3类的操作

可见性 操作的名字（参数列表）：操作返回类型

1、可见性

2、操作名

3、参数列表

方向 参数名：类型【多重性】=缺省性【特性】

三、种类

3.1抽象类

抽象类是一种不能直接实例化的类型，不能用抽象类创建对象。

3.2接口

3.3模板类

不用说明属性、方法返回值和方法参数的实际类型。

3.4主动类

拥有一个控制线程并且能够控制线程的活动。

3.5嵌套类

将一个类定义放在另一个类中

四、关系

4.1关联关系

1、概念：

2、类型：

（1）关联的方向、导航

（2）关联名

（3）关联角色：角色、关联名

（4）多重性：一个类的多少对象与另一个类一个对象相关

# 2020-05-15

供给接口：

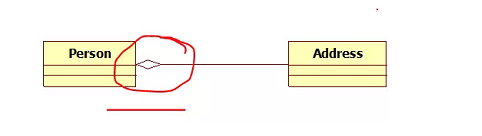
类是接口的具体实现，实现关系

需求接口：

类也可以通过接口使用另一个类，依赖关系

4.1关联关系

1、聚合关系中，空心菱形表示，整体不会拥有部件的生命周期，所以整体删除时，部件不会被删除。



多个整体可以共享一个部件。

2、组合关系中，整体拥有部件的声明周期，所以整体删除时，部件没有了存在的价值。

多个整体不可以同时共享一个部件。实心菱形表示。

在整体类的对象销毁之前，它要释放和销毁它的部分对象。

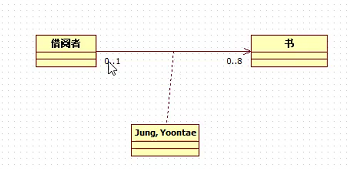
3、识别聚合：

物理上为整体的食物和它的部分，例如汽车和发动机，人体和器官

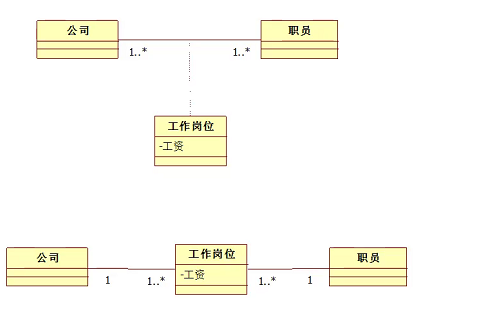
组织机构和它的下级组织或部门

4、关联类

关联类是兼有关联和类双重特征额建模，既可以看作是具有类的特征，又可以看作是具有关联特征的类。



关联类定义一组属于关联本身的特性，这些特性不属于关联两端任何一个类。



分析阶段（上）设计阶段（下）

5、自关联

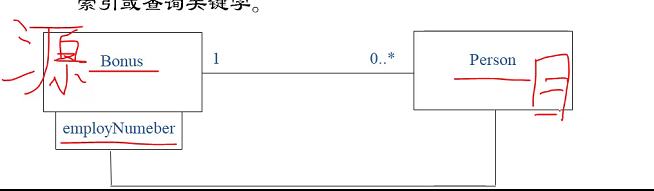
指一个类与其自身存在一种关联关系

自关联关系意味着该类的某个实例与该类的其他实例之间存在关联关系。

6、限定关联

限定符的表示法是在关联线靠近源类一端绘制一个小方框

是目标对象中的某个属性，也可以是表达式。



靠近源类，限定目标类

4.2泛化关系

从特殊到一般元素的分类

4.3实现关系

4.4依赖关系

关联、实现、泛化都是依赖关系。

宏观定义：一个事务发生变化，也会影响另一个事物

1、使用依赖

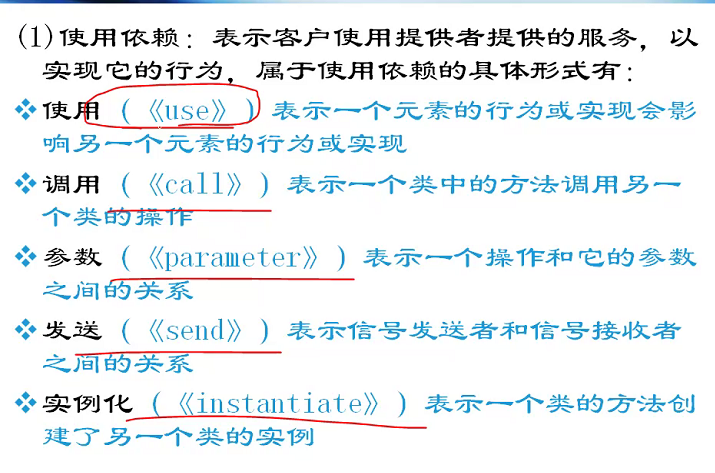
（1）使用

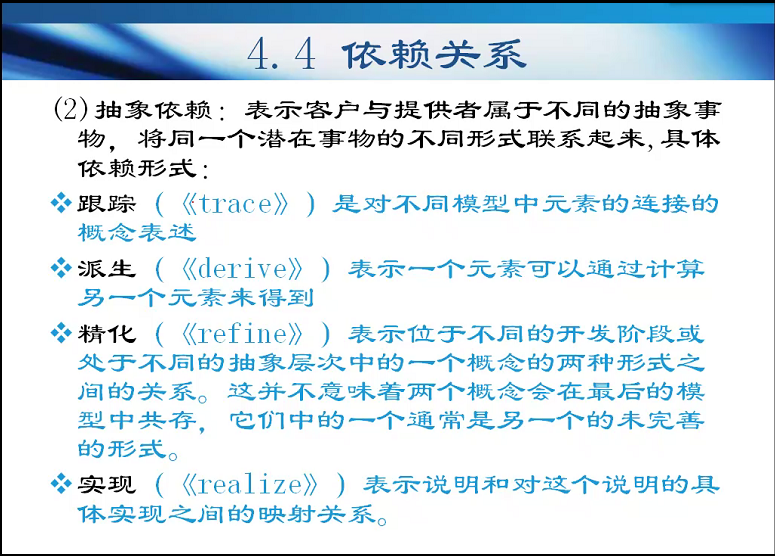
（2）调用

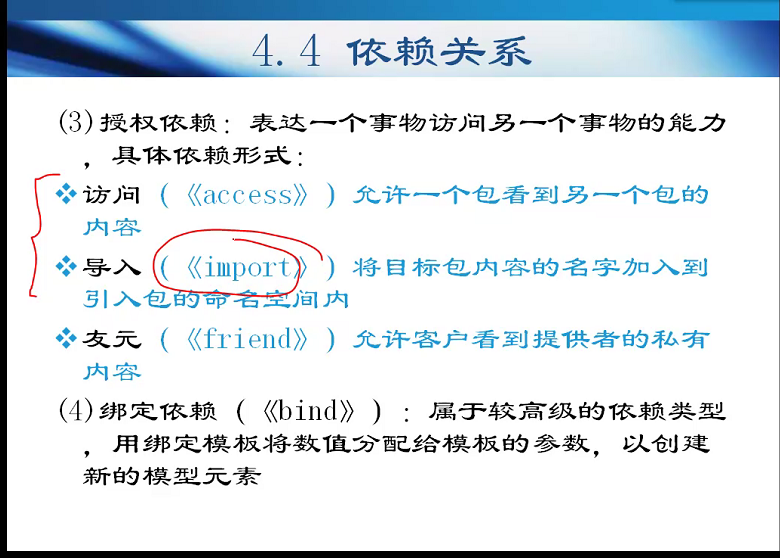
（3）参数

（4）发送

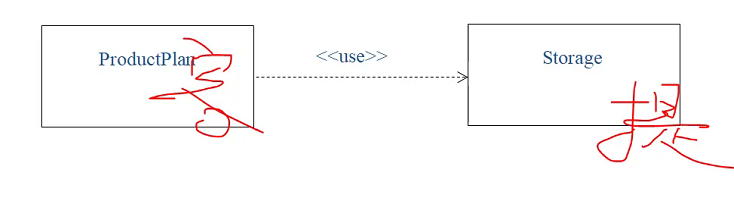
（5）实例化







表示方法：虚线+箭头



# 2020-05-18

回顾：用例规约中的前置条件和后置条件分别指的是用例执行前、后系统与参与者所处的状态。但两者不是必需的。

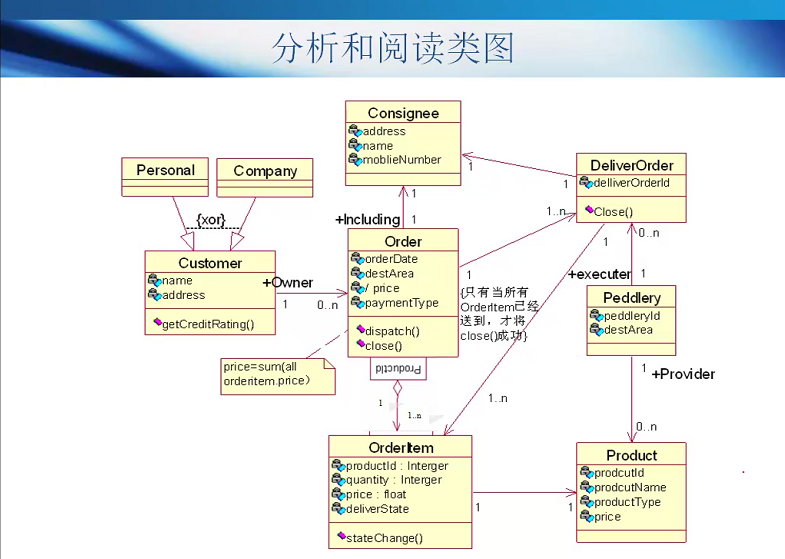
4.4依赖关系

A类是B类中的（某种方法）的局部变量；

A类是B类方法当中的一个参数；

A类是B类中方法的返回值；

分析和阅读类图



从最复杂的关系开始分析

# 2020-05-20

分析类：边界类、控制类、实体类

正向工程：模型转化为代码

逆向工程：代码转化为模型

对象图——类图的实例

1、概念：在某个时间点上，给系统所有参与对象拍下一张对象状态的快照。

对象图=对象+链

对象是类的实例，链是关系的实例。

与类图不同点：对象图显示类的多个实例。

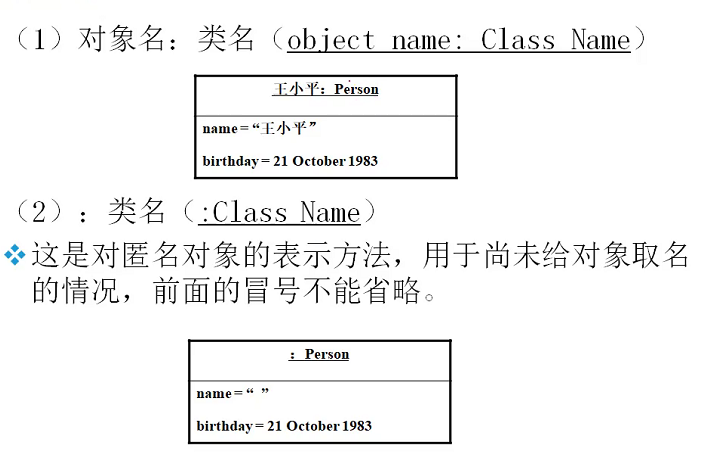
2、对象的表示

矩形框中放置对象的名字

（1）对象名：类名（object name:Class Name）

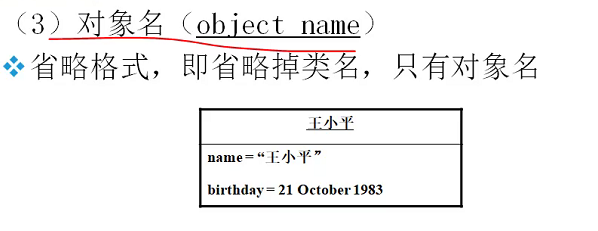
（2）：类名（:Class Name）

匿名对象的表示方法，冒号不可以省略



（3）对象名（object name）

省略模式，及省略掉类名，只有对象名



属性的表示：

第一栏放置对象名，第二栏放置对象的属性。

属性的表达方式：

属性名 = 值

链：单向链、双向链

# 2020-05-25

交互图：

顺序图：消息发送的时间关系

通信图（协作图）：对象之间的协作关系

定时图（时间图）、

交互概述图

一、顺序图的概念

时序图、序列图

强调的是消息交互的时间顺序

组成：对象、生命线、对象的激活、对象间交互的消息

纵向是时间轴，横向是对象的角色

一般来说，发起交互的对象或角色放到左边，较下级的对象或角色依次放在右边

二、顺序图的表示

2.1对象

将对象置于顺序图顶部意味着在交互开始的时候对象就已经存在了

从左到右：参与者对象、边界对象、实体对象

2.2生命线

表示对象在一段时间内的存在

2.3激活

UML1控制焦点

UML2执行规约

表示对象执行一个动作所经历的时间段

矩形的顶部表示动作的开始，底部表示动作的结束

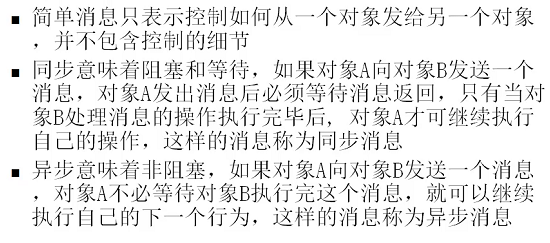
既可以自己直接执行，也可以通过下级执行

2.4消息

UML用带有实心箭头的实线表示消息，每条消息从发送对象指向接收对象

2.4.1命名

2.4.2简单消息、同步消息和异步消息



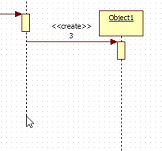
调用：对象A向对象B发送调用消息，可以理解为对象A调用对象B的一个操作。同步机制。

发送：是指向对象发送一个信号，信号和调用不同，它是一种事件，用来表示各对象间进行通信的异步激发机制。

实体箭头表示同步，开放性箭头表示异步

2.4.3对象创建消息

Create



2.4.4销毁

2.4.5无触发对象和无接收对象消息

活动条开始端点用实心球表示

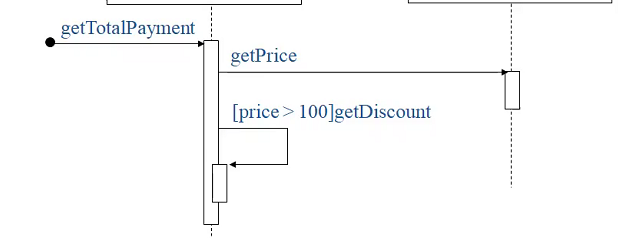
表示消息发送者未知，或者是一个随机的消息源

2.4.6自我调用消息

自己调用自己

2.4.7控制消息

条件：仅当条件为真的时候消息擦被发送



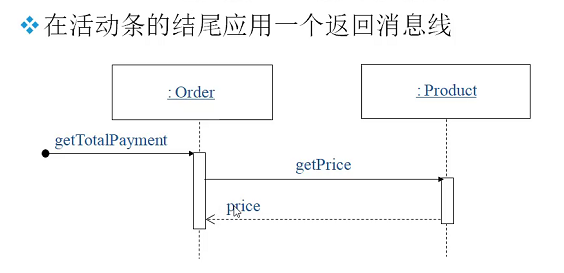
迭代：为了多次接受消息

条件前面加上\*

2.4.8返回值

表示：返回变量=消息（参数）

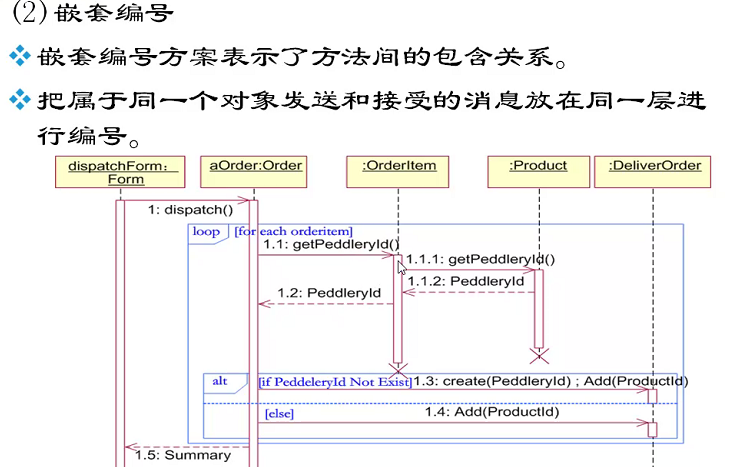
重要的返回消息即可



2.4.9 消息编号

（1）顺序编号

（2）嵌套编号



三、顺序图中的循环和分支

Break：跳出循环

Critical：临界区域，事务性操作

Par：并行，除非critical禁止

Ref：引用其他片段

# 2020-05-27

回顾：

构成：对象（生命线）+消息（激活）

对象：水平方向 排在顶端 代表的角色——匿名对象

从左到右：引发交互的参与者或对象、界面、控制、实体（、次要参与者）

消息：简单消息

调用（同步）、返回值、发送（异步）、创建、销毁

交互片段：操作符决定类型、多个区域，每个区域一个监护条件、一组消息传递

箭尾调用箭头

泛指 :类

确指 对象:类

顺序图分析阶段（汉字即可）

顺序图转为通信图 starUML Model

相同点：本质相同，语义相同

不同点：

1、侧重点不同

顺序图注重消息发生的时间顺序

通信图注重对象，对象关联关系，实现协作

2、画法不同

# 2020-05-29

一、通信图

1、通信图概念

特别适合少量对象之间的简单交互

消息编号必须加

2、通信图表示

通信图 = 交互的参与者+通信链+消息

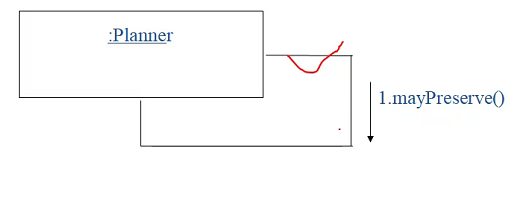
通信图的组成：对象、消息、链（连接器）

通信图内无法表示对象的创建和撤销

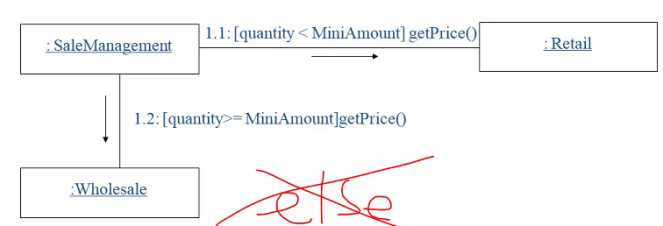
链表示对象之间的语义关系

3、消息的分类

（1）自我委派消息：消息从一个对象发送到它自身

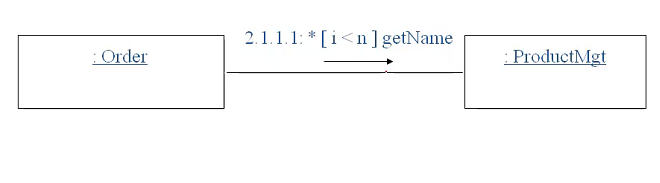


（2）控制消息：表示控制条件为真的时候才会被发送



（3）

（4）循环（右下角属性添加）\*表示

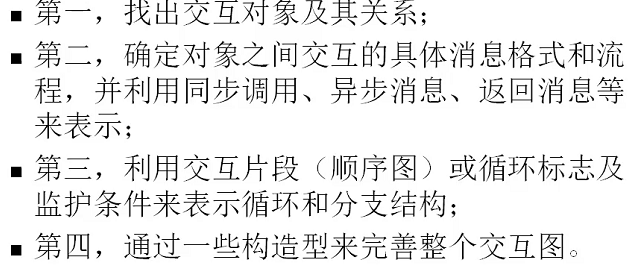


（5）并发（了解）：同时发送

二、交互图

2、绘制交互图

步骤：（分支循环右下角属性添加）



用例中事件转换为交互图中的消息

绘制策略：

（1）先确定交互发生的语境，即交互是发生在系统中、子系统中、操作中、类中还是用例或协作；

（2）识别对象在交互中扮演的角色设置和交互场景

（3）为每个对象设置生命线

三、定时图

常用实时控制图

1、定时图与顺序图的区别：

（1）坐标轴交换了位置，定时图的时间坐标是，从左到右表示时间的延续

（2）用生命线的“凹下凸起”来表示状态的迁移

2、包含：

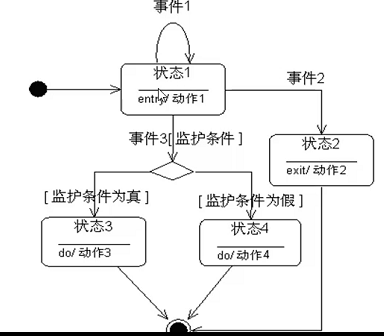
生命线、状态、状态变迁、消息、时间刻度

四、状态机图

对象既有行为又有状态，对象的行为由其状态决定，对象根据其状态的不同而产生的不同的行为。

特别适合为那些行为由其状态决定的对象建模，

1、状态机图的概念



初始状态、终止状态、状态、转换。

其中，转换将各种状态连接在一起，构成一个状态图。

2、状态图的表示

状态图 = 状态+转换

初始状态：实心圆

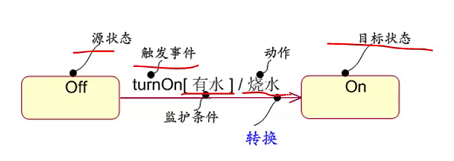
终止状态：空心圆套实心圆

状态：对象在某个时刻的的动作或者活动，圆角矩形，三层（嵌套状态图）/两层（名称、内部转换）

外部事件转换当时状态

内部转换未改变当前状态

转换的表示



# 2020-06-01

1. 状态图
2. 概念
3. 表示
4. 状态
5. 转换：

源状态

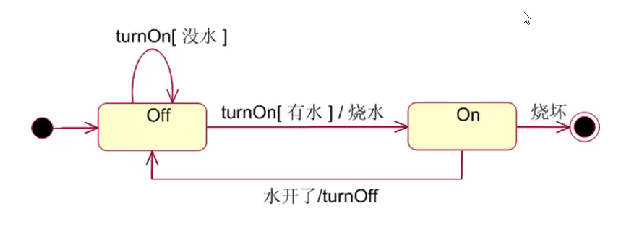
目标状态

事件（触发器）：调用（改变对象状态）、改变（检测对象状态改变）、信号（对象之间发送信号实现通信）、时间（时间表达式）、可延迟（顺序）

事件不一定来执行

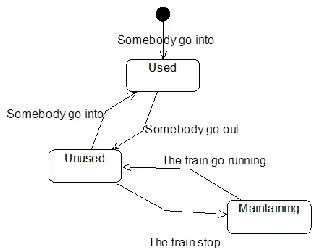
监护事件：触发事件发生之后，无事件监护无意义。

动作：当转换被激活后，如果定义了相应的动作，那么就执行这个动作。



1. 转换的分类

3.1外部转换：一种改变对象状态的转换，最常见的一种转换。



3.2内部转换

内部转换自始至终都离不开源状态，没有入口和出口动作。

入口动作：entry/要执行的动作，表示做了这个动作之后进入这个状态

出口动作：exit/要执行的动作，表示做了这个动作之后离开了状态

内部动作：do在这个状态下一直做得

自转换：离开了当前状态，之后返回该状态，执行入口动作和出口动作。

3.3自动转换：无外部事件作用下，完成了转换

3.4复合转换

由简单转换组成，通过分支判定，把多个简单转换组合在一起。

1. 状态的分类

状态机图中的状态分为简单状态和复合状态

4.1简单状态

不包含其他状态的状态。

4.2复合状态

多个子状态之间的关系，两种：并发关系、互斥关系

4.2.1顺序子状态：子状态是互斥关系

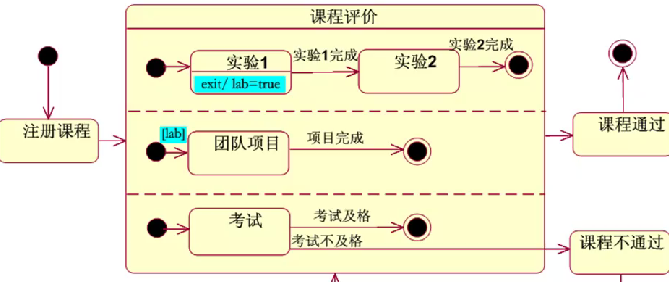
4.2.2并发子状态：子状态是并发关系

4.2.3复合状态的表示

4.3历史状态

4.4子状态间实现通信

借助监护条件、状态间的事件来描述，也可以采用异步模式



# 2020-06-03

回顾：

状态机图

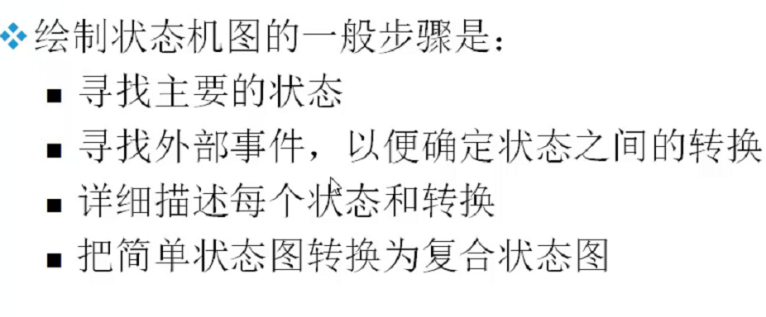
状态：状态名、

第二栏（入口/出口动作、内部活动、内部转换、延迟事件）

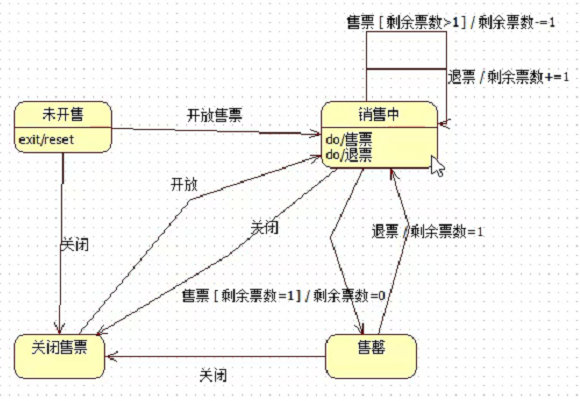
转换：源状态、目标状态、事件、监护条件和动作

一、状态机图

1、建立状态机图步骤：



示例：



1. 状态机图的应用

2.2反应性对象建模

状态改变动作行为

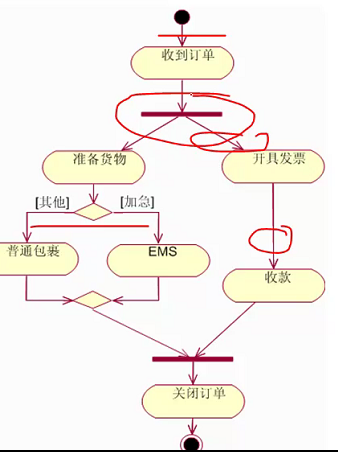
描述一个对象的多种状态

复合状态：顺序，并发

历史状态：当离开一个复合状态，然后重新进入复合状态，并不希望从复合状态机的子状态重新开始

1. 活动图

在流程图的基础上添加大量软件工程的专业术语

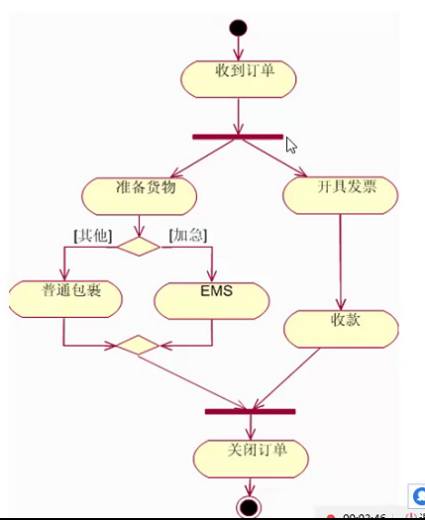


活动图的的表达方式：

活动图=活动+动作+活动边+活动节点

# 2020-06-08

1. 活动图
2. 活动图的概念



1. 表示

2.1活动和动作

活动是由一个或者多个动作组成的行为，一个活动是多个动作的集合。

2.2活动边

2.2.1控制流

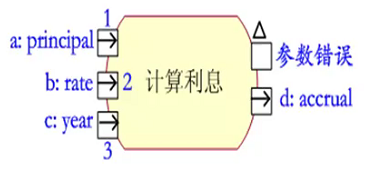
当活动边连接的是两个动作或活动节点时

2.2.2对象流

活动边连接动作与数值或活动与数值时

2.3活动节点

2.3.1参数节点



参数标识在活动节点的边界上，输入参数标识在左，输出参数和错误参数在右。参数名称和序号标识在参数节点旁边。

2.3.2对象节点

对象节点表示活动中移动的数据。

对象节点用矩形表示

输入输出可以用栓来表示

2.3.3控制节点

起始：活动开始

判断：通过布尔值的选择给出不同的输出流控制节点，不同条件称为警戒条件

汇合：多个输入，无论哪一个到达即可运行

分叉：一个动作在该点同时产生多个并发活动边

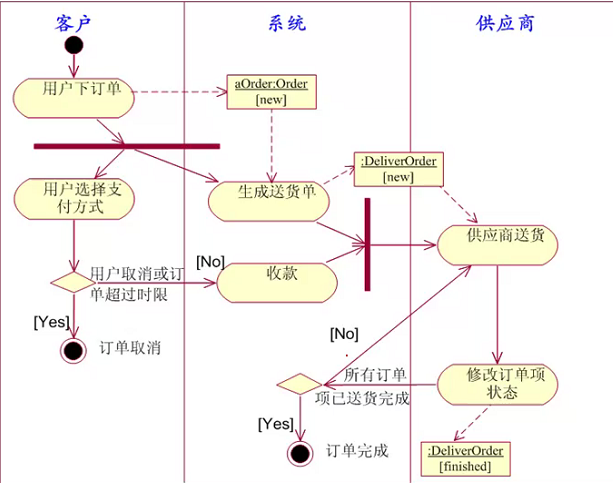
结合：多个并发活动边均到达后才能输出

终点：活动结束

2.4活动划分或泳道

1. 活动图分类

3.1简单活动图



每个活动节点，分支必须只属于一个泳道，而转换、分支和结合是可以跨泳道的。

3.3.标识对象流的活动图

可能存在一些对象进入一个活动节点，经过活动处理，修改了对象的状态；活动节点创建或删除了一些对象。

3.4标识信号的活动图



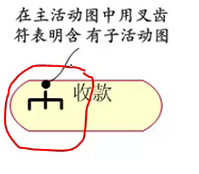
3.5标识参数的活动图

每个活动节点执行前，需要输入哪些参数

3.6标识扩展区的活动图

表示一个活动多次执行的情况。

3.7嵌套



1. 构建活动图

4.2对操作流程建模

1. 包图

一种有效的建模工具，基于包的系统在静止状态下的结构建模

1. 包图的概念

定义命名空间的方法

包图 = 包 + 关系

1. 包的表示

2.1包的表示

2.2包的名称

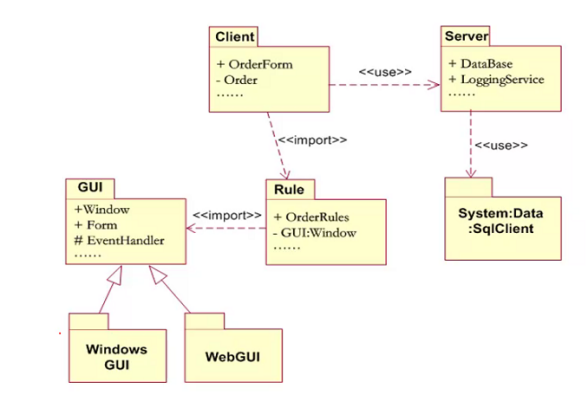
简单名和全名

2.3包的元素

2.4包的可见性

+：公共

-：私有



简单包图示例

虚线箭头依赖关系

实线箭头泛化关系

1. 包的关系

3.1《use》关系

默认的依赖关系

3.2《import》关系

提供者的包添加到客户包，客户包中的元素能够访问提供者包的所有公共元素。

使命名空间合并。

3.3《access》关系

只是想使用提供者包中的元素，而不是想把两个包合并，在客户包中必须使用路径名，才能访问提供者包中的所有公共元素。

# 2020-06-10

1. 阅读和创建包图

4.1

4.2创建包图

1. 确定包；
2. 确定包之间的关系
3. 标出包内元素的可见性

4.2.1寻找包

4.3包图的用途

4.3.1对组成元素建模

一、组件图

1. 组件的概念

组件是系统可替代的物理接口，定义了良好接口的物理实现单元，是系统中可以替代的部分，依赖于组件支持的接口

1. 配置组件：组成系统的基础组件
2. 工作产品组件：开发系统中的中间产物
3. 执行组件：运行时创建的组件
4. 组件
5. 组件的表示
6. 组件图的绘制
7. 组件图的应用

5.1对可执行结构建模

5.2对源代码进行建模

组件图应该在软件开发的设计阶段

二、部署图

1、概念：也叫配置图、实施图

软硬件之间的物理关系以及软件组件在处理节点上的分布情况。

组成元素：节点、连接。

描述一个运行时的硬件节点，以及在这些节点上运行的软件组件的静态视图。

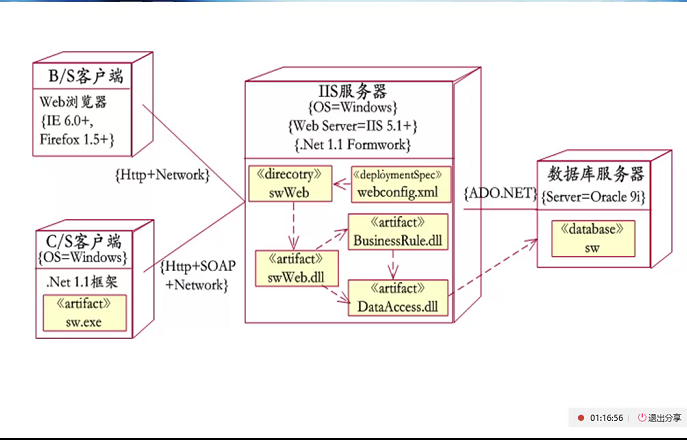
2、部署图的表示

2.1节点

计算资源的物理元素。拥有一些内存，并具有处理能力

2.1.1节点的表示

2.1.3节点上的组件



2.2连接

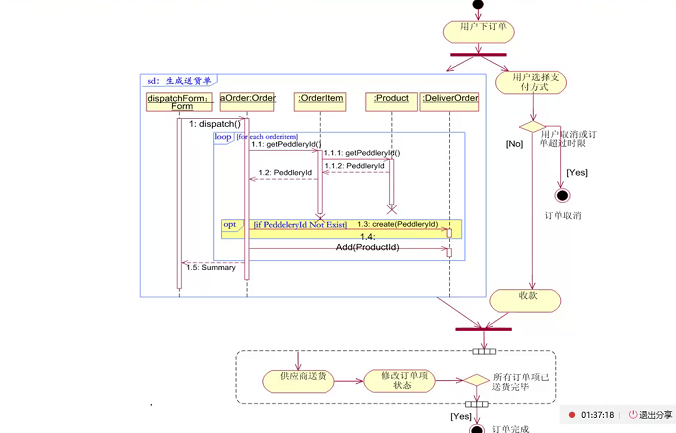
3、部署图应用

设计与实现

设计阶段主要用来描述硬件节点以及节点之间的连接

实现阶段生产出来了软件组件

1. 交互概述图
2. 概念



1. 绘制

对操作建模

