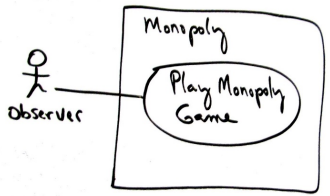
Object oriented Analysis &Design
面向对象分析与设计

Ch8~10,15 System Sequence Diagram
Ch11 Operation Contract

Presented by: Xiaohong Chen

2020/10/20

Review: Use case model



Scope: Monopoly application

Level: user goal

Primary Actor: Observer

Stakeholders and Interests:

- Observer: Wants to easily observe the output of the game simulation.

Main Success Scenario:

1. Observer requests new game initialization, enters number of players.
2. Observer starts play.
3. System displays game trace for next player move (see domain rules, and "game trace" in glossary for trace details).
4. Repeat step 3 until a winner or Observer cancels.

Extensions:

*a. At any time, System fails:

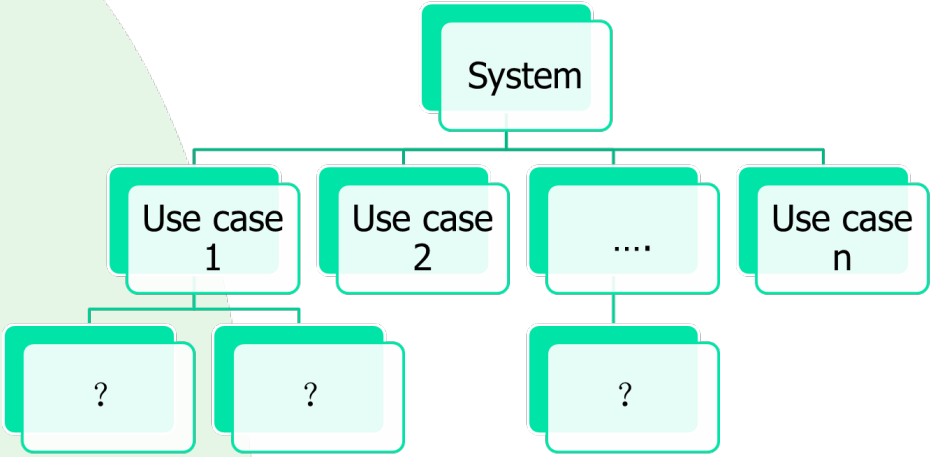
(To support recovery, System logs after each completed move)

1. Observer restarts System.
2. System detects prior failure, reconstructs state, and prompts to continue.
3. Observer chooses to continue (from last completed player turn).

Special Requirements:

- Provide both graphical and text trace modes.

Review-use case 需求模型



Outline

- Interaction Diagram （交互图）预备工作
- System sequence diagram（系统顺序图）
  - Identify system events.
  - Create system sequence diagrams for use case scenarios.
- Operation contract （操作契约）

要求掌握的内容

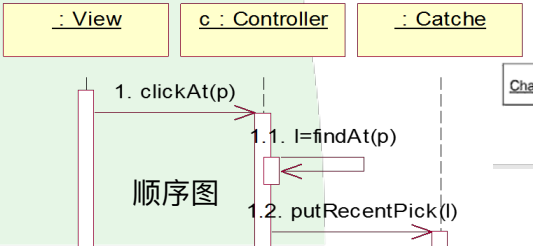
- 画好系统顺序图
  - 掌握黑盒子画法
- 写出正确的操作契约
  - 掌握操作契约的入口点
  - 掌握操作契约的格式

问题列表

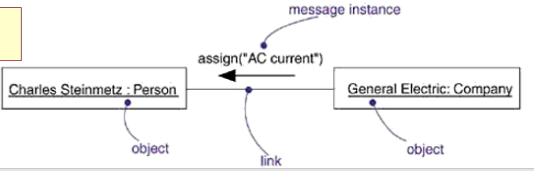
- 顺序图与系统顺序图的区别?
- 操作契约与领域模型之间的关系?
- 操作契约的输入?

Interaction diagram 交互图

- 在UML中使用交互图建模系统中的交互行为：
  - **交互图**描述了一系列的对象之间传递的消息
- 交互图分为两种：
  - **顺序图 Sequence diagram**
  - **协作图 collaboration diagram**
- 它们在语义上是等价的
  - 顺序图：强调消息的时间顺序
  - 协作图：强调接收和发送消息的对象的结构组织

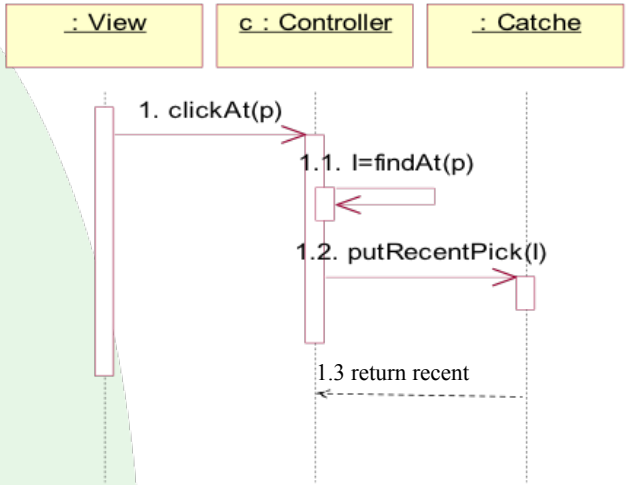


顺序图



协作图

顺序图


: View
c : Controller
: Catche
1. clickAt(p)
1.1. l=findAt(p)
1.2. putRecentPick(l)
1.3 return recent

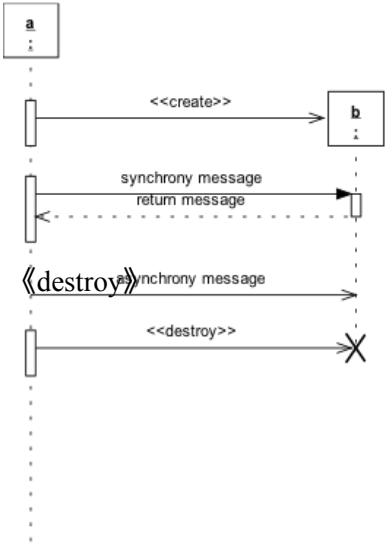
练习

- 把大象关进冰箱的过程

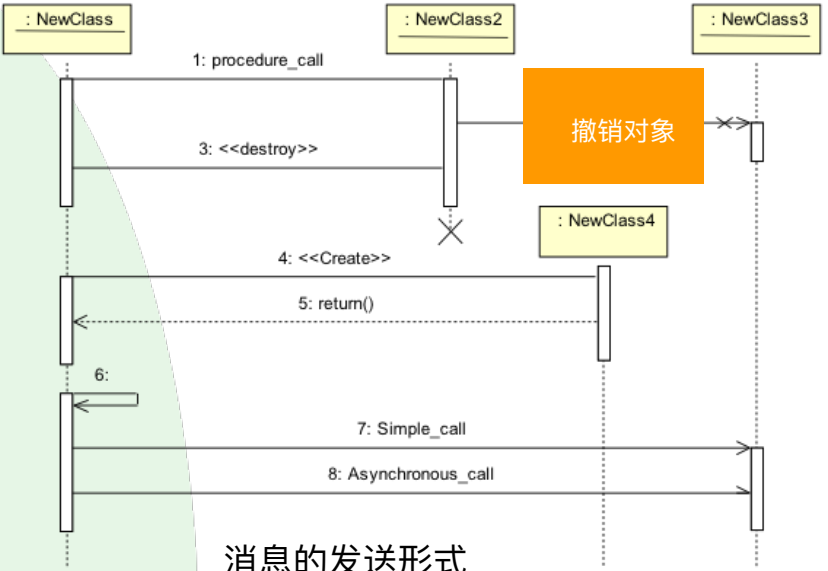
- 超市收银台买单付钱的过程?

不同的消息表示

- 在UML里，消息用箭头表示，从发送消息的对象指向接收消息的对象
  - 同步消息： 实心箭头
  - 异步消息： 枝状箭头
- 在消息的各种形式中
  - 创建和销毁消息
    - 用消息的 构造型来表示 《create》，《destroy》
    - 返回消息，用带虚线的箭头表示



测试

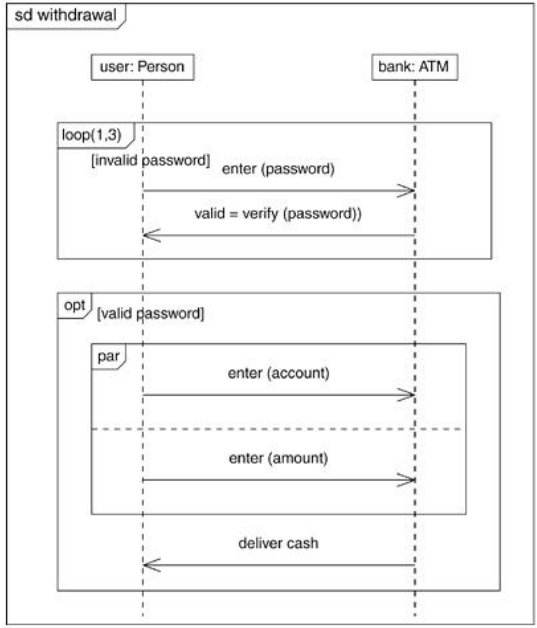

: NewClass
: NewClass2
: NewClass3
1: procedure_call
撤销对象
3: <<destroy>>
: NewClass4
4: <<Create>>
5: return()
调用自己的消息
6:
7: Simple_call
8: Asynchronous_call

消息的发送形式

顺序图中的结构化控制

- 顺序图中的结构化控制
  - 可选执行 （标签: opt)
  - 条件执行 （标签:alt)
  - 并行执行 （标签:par)
  - 循环（迭代）执行 （标签:loop)

Figure 19-3. Structured Control Operators



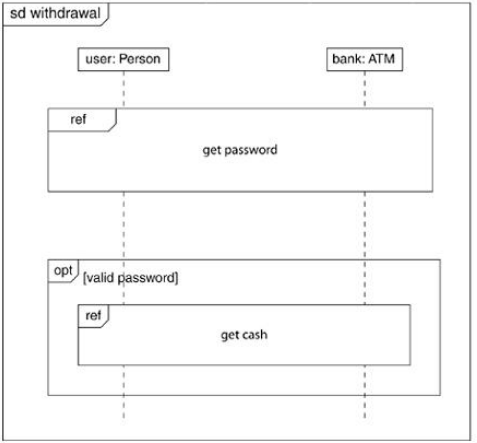
- 超市收银台买单付钱过程如何表示?

顺序图

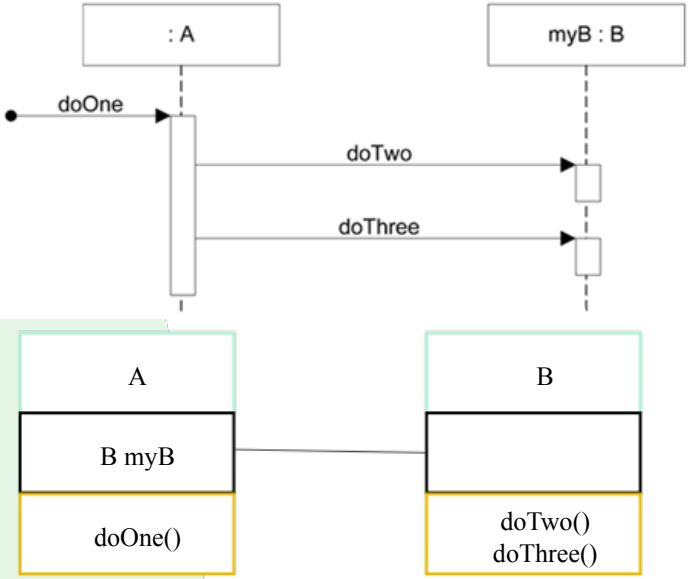
Ref (reference):

表示引用其他交互

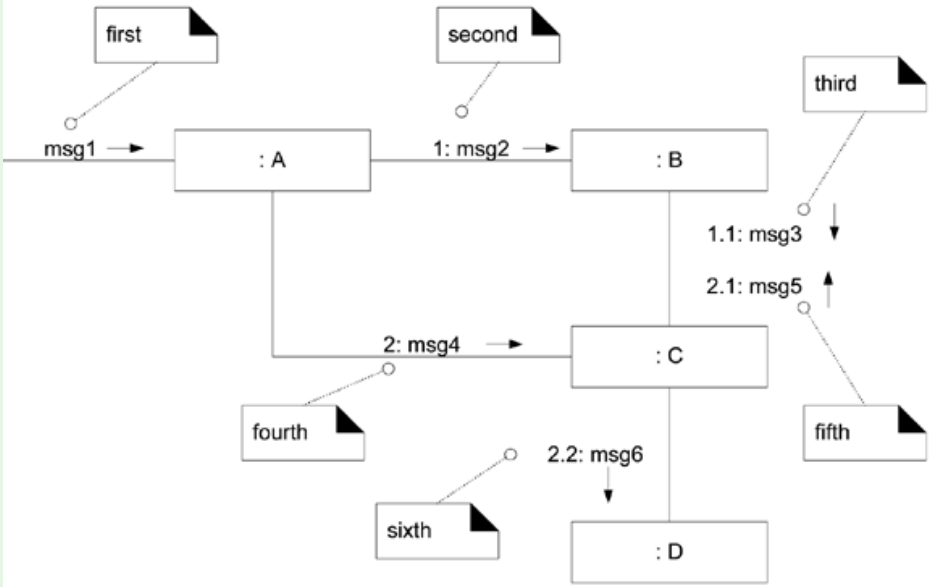
Figure 19-4. Nested activity diagram



SD到类图的映射



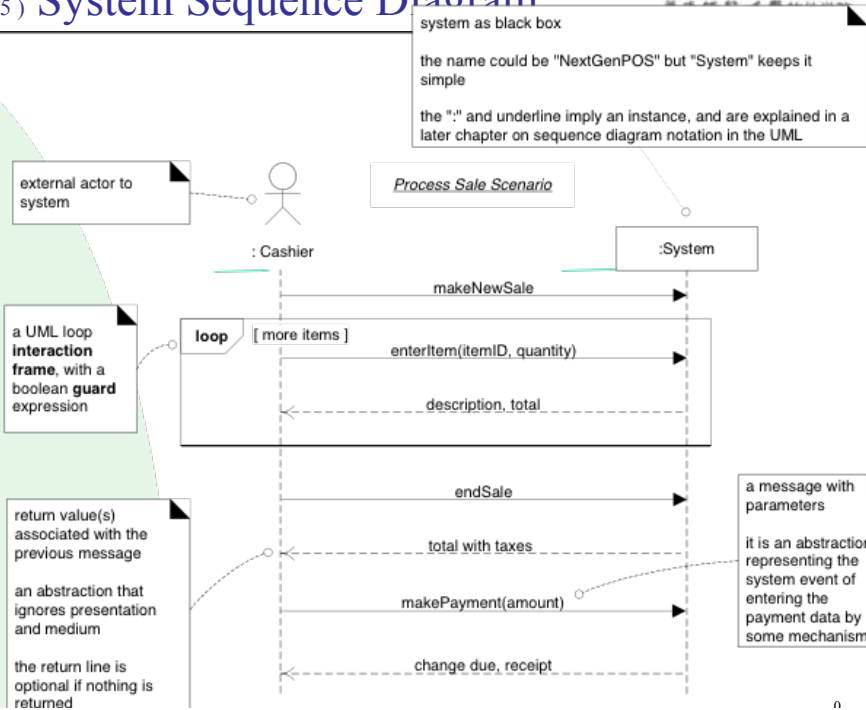
协作图


first
second
third
msg1
: A
1: msg2
: B
1.1: msg3
2.1: msg5
2: msg4
: C
fourth
fifth
2.2: msg6
sixth
: D

(Ch10,15) System Sequence Diagram

- SSD:
  - System Sequence Diagram
- What is SSD?
  - A SSD is a picture that shows, for one particular scenario of a use case, the events that external actors generate, inter-system events , and their order
  - All systems are treated as a black box;
  - the emphasis of the diagram is events that cross the system boundary from actors to systems
- System Event
  - external input events
    - actor generates events to a system
- system operation
  - to handle the system event , for example
  - when a cashier enters an item's ID, the cashier is requesting the POS system to record that item's sale (the enterItem event). That event initiates an operation upon the system.
  - Operations that the system offers in its public interface

(Ch10,15) System Sequence Diagram

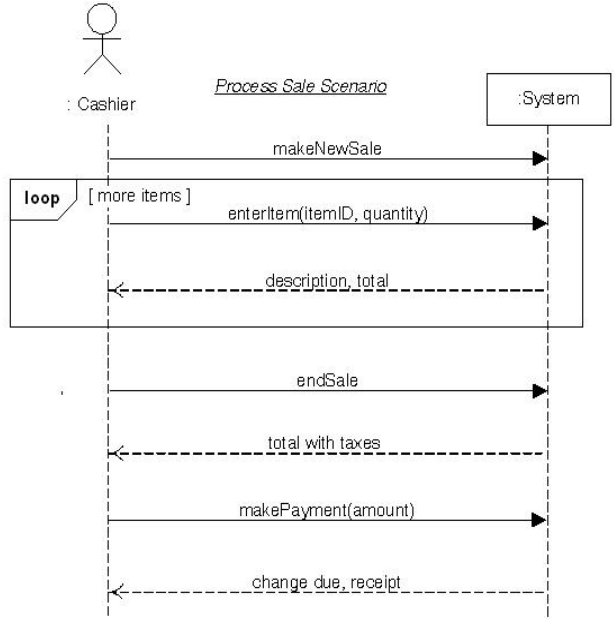


SD 案例 （POS系统）用例描述和用例图

用例：处理销售

1、顾客携带所购商品来到POS机付款处进行购买交易
2、收银员开始一次新的销售交易
3、收银员输入商品ID
4、系统逐条记录出售的商品条目，并显示该商品的描述、价格和累计额。价格通过一组价格规则来计算
5、收银员重复步骤3~4，直到结束
6、系统显示总额
7、收银员告知顾客总额并提请付款。
8、顾客支付，系统处理支付。

。。。。
（此处仅有主事件流）



(Ch10,15) System Sequence Diagram

- Cf: SSD & SD
  - SSD: to emphasize to treat systems as black boxes.
  - SD will be used to illustrate the design of interacting software objects to fulfill work
- SSD and Use case
  - it is generated from inspection of a use case

Monopoly Game-SSD

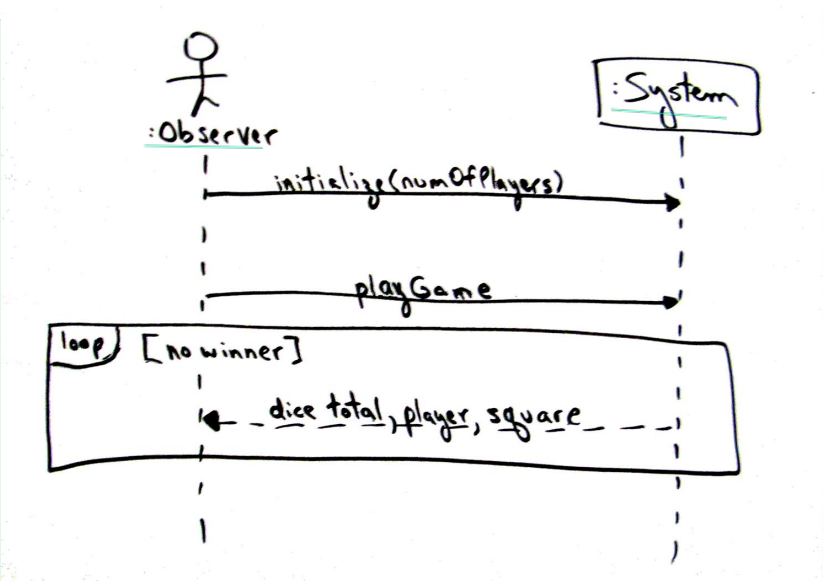


Fig 10-5

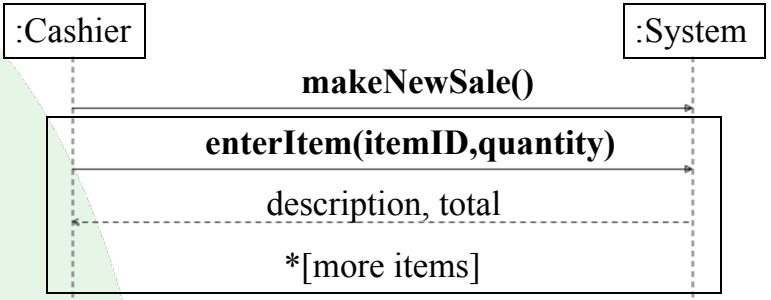
Next

- Operation Contracts:
- More on the Use Case Model

Operation Contracts: More on the Use Case Model

- **Use cases** - the way to describe requirements
  - For certain scenarios, **system sequence diagrams**
  - What is system operation?
- Use Case Model and Domain Model are developed in parallel
- **Operation Contracts:** a way to describe use cases in more detail
  - Part of the Use Case Model

System Operations



Operations that the system as a black box component offers in its public interface

设计类

System
makeNewSale() **enterItem(itemID,quantity)**

函数怎么写?

Example of a Contract for **enterItem**

Operation: enterItem(itemID: ItemID, quantity: Integer)

Cross References: Use Cases: ProcessSale

Preconditions: There is a sale underway

Postconditions:

A SaleLineItem instance **sli** was created

sli was associated with the current Sale

sli.quantity became quantity

sli was associated with a ProductDescription, based on itemID match

Definition: What are the Sections of a Contract? (11.2)

- A description of each section in a contract is shown in the following schema.

Operation:	Name of operation, and parameters
Cross References:	Use cases this operation can occur within
Preconditions:	important assumptions about the state of the system or objects in the Domain Model before execution of the operation. These are non-trivial assumptions the reader should be told.
Postconditions:	This is the most important section. The state of objects in the Domain Model after completion of the operation. *Discussed in detail in a following section.*

Postconditions

- **Definition**
  - The **postconditions** describe changes in the state of objects in the domain model.
  - Domain model state changes include
    - instances created / deleted,
    - associations formed or broken,
    - and attributes changed.
- Why postcondition?
  - they aren't always necessary
    - If developers can comfortably understand what to do without them, then avoid writing contracts
  - postconditions support fine-grained detail and precision in declaring what the outcome of the operation must be
  - A contract is an excellent tool of requirements analysis or OOA
    - the design can be deferred, and we can focus on the analysis of what must happen, rather than how it is to be accomplished

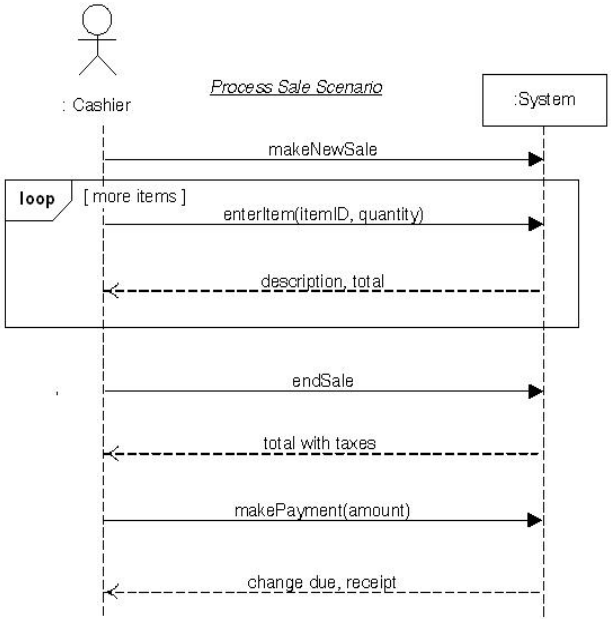
How to create and write Postcondition?

- Apply the following advice to create contracts
  - Identify system operations from the SSDs.
  - For system operations that are complex and perhaps subtle in their results, or which are not clear in the use case, build a contract.
  - To describe the postconditions, use the following categories:
    - instance creation and deletion
    - attribute modification
    - associations formed and broken
- What's the Most Common Mistake ?
  - Writing Contracts
    - (better) A SalesLineItem was created.
    - (worse) A SalesLineItem is created ; or Create a SalesLineItem
  - forgetting to include the forming of associations , ex,
    - The SalesLineItem was associated with the Sale (association formed).

Postcondition & Domain Model

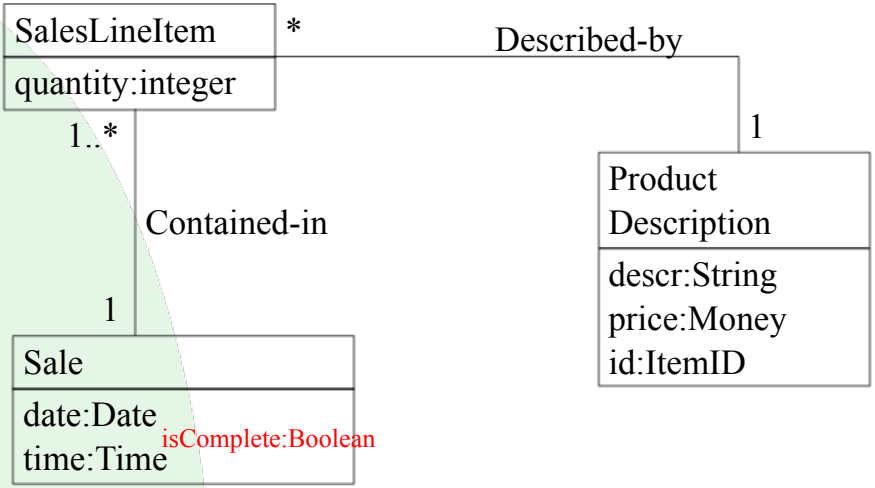
- postconditions are expressed in the context of the Domain Model objects
  - What instances can be created?
    - those from the Domain Model;
  - What associations can be formed?
    - those in the Domain Model;
  - and so on.
- It's common during the creation of the contracts to discover the need
  - to record new conceptual classes, attributes, or
  - associations in the domain model

For example



系统事件?

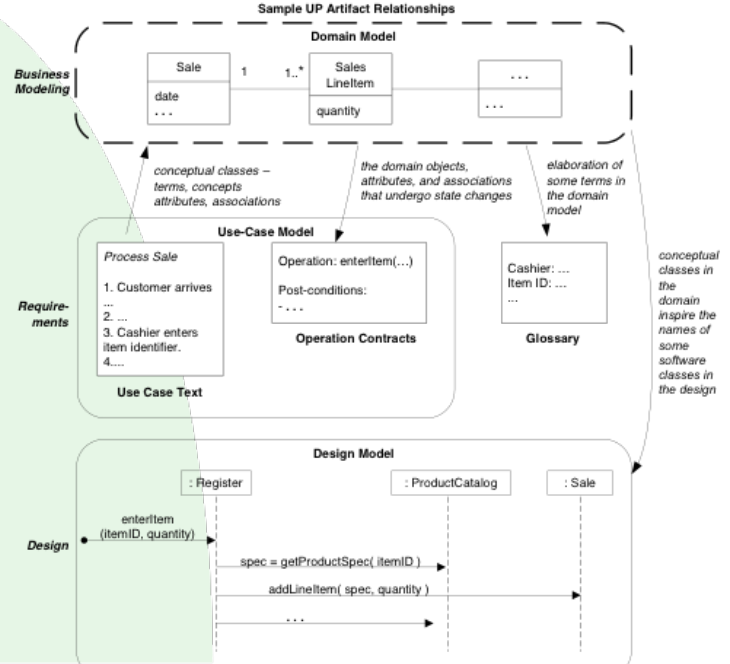
How to define the post condition of end sale?



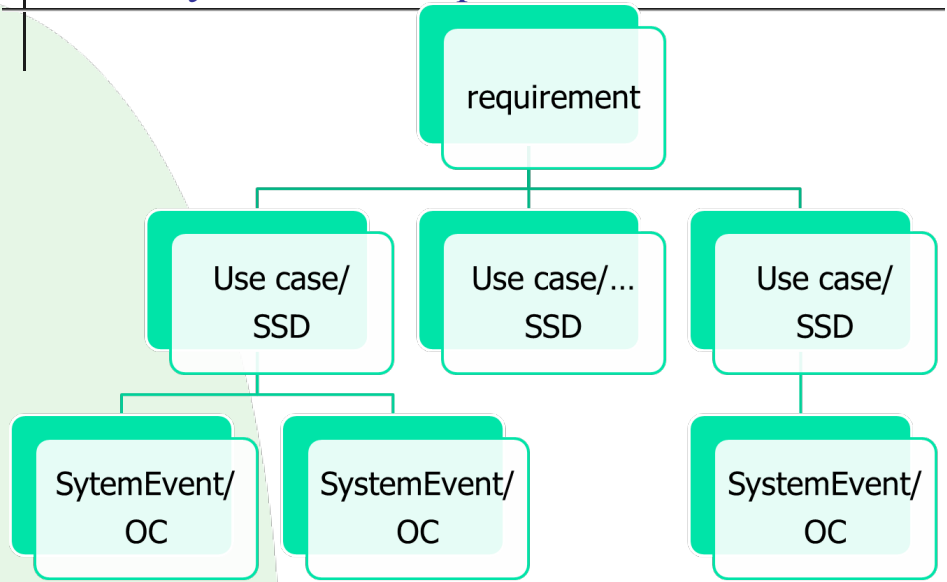
Summary

- System sequence diagram
- Operation contract

Review-where are we?



Summary: OOA Requirement?



HomeWork

- Read / review text book Ch9 ~14
- Preparation: Review of OOA
- 把上次的作业的两个用例描述转换为SSD，并给出两个系统事件的操作契约。
-
- End