《面向对象的系统分析与设计[UML]》实验 2 分析建模

[实验日期]　2022　年　5　月　9　日

[实验目的]

（1）理解面向对象系统分析和对象类建模(概念建模)的概念

（2）了解和掌握面向对象系统分析的方法和步骤

（3）了解和掌握寻找待开发系统中类(概念)的方法和技巧

（4）掌握使用 StarUML 5.0(免费开源)绘制概念模型的方法

[实验内容]

在“NextGen POS 系统”用例分析的基础上，选择第一个迭代周期打算开发的用例，建立相关的概念模型。

[实验原理和步骤]

1. 类的识别方法：

（1）词汇分析法（名词短语法）

当前迭代所开发的 UC 的 UCN（及其他）是分析的基础。

（2）概念和关联列表法（Larman）

从当前迭代所开发的 UC 的 UCN 中识别概念和关联；借助概念类分类列表+关联列表(见附录)。

（3）分析模式

重用和修改已有模型是首选、最佳和最有效的策略；来自以往项目文档、论文、书籍等；Peter Coad 的 12 种事务模式（Transaction Patterns）：以事务为中心；Martin Fowler 的《分析模式》2004：组织、财务、医疗；Len Silverston 的《数据模型资源手册-(卷 123)》：1,2 卷 2004，3 卷 2017（副标题：数据模型通用模式），数据库学派；David C.Hay 的《Data Model Patterns》；（其他面向对象、数据库应用专著）；不能机械套用，要根据实际情况做修改。

步骤：

（1）在“NextGen POS 系统”用例分析的基础上，采用上述方法识别类和类之间的关联（找名词和动词）。

（2）为体现是系统分析员的工作，可以采用“用例视图”（而非“逻辑视图”）下的类图描述概念模型，只不过每个类中只有类名和属性，没有方法。在概念建模阶段也没有必要确定属性的类型和访问属性。

（3）概念间的关联可以采用一般关联（无方向实线）（待设计阶段由设计师完善），当然，对于聚合和泛化，采用相应的连线（组合：实心菱形+实线；聚合：空心菱形+实线；泛化：空三角形+实线）

2．各种图之间相互启发和印证：

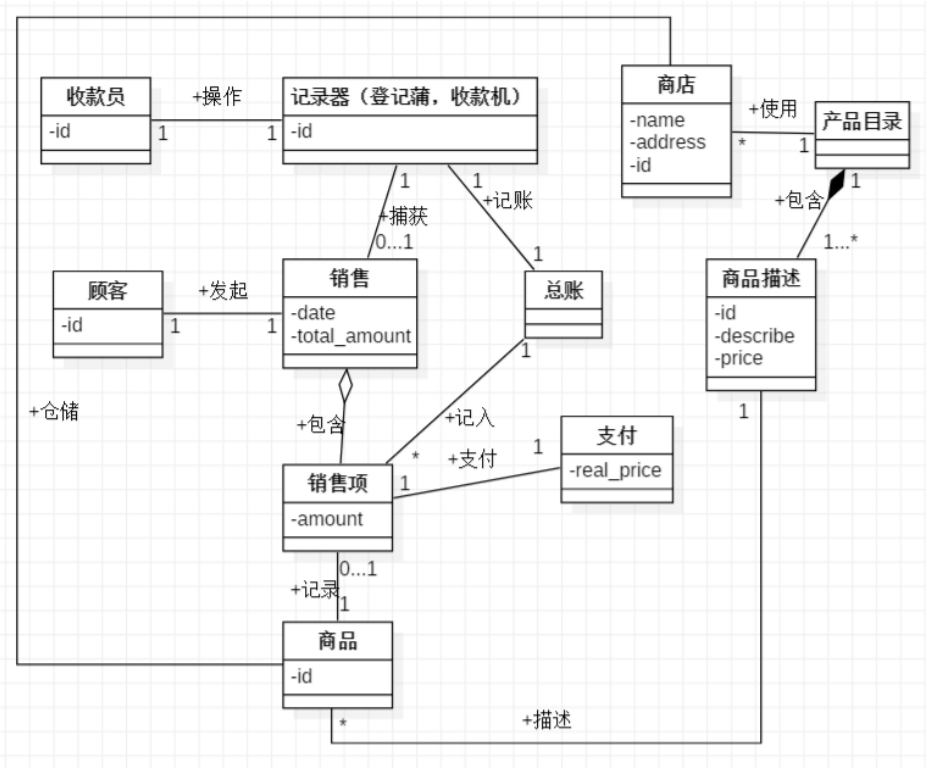
在分析阶段，分析师除了建立用例模型，还可以建立分析类图、高层顺序图等，主要目的是为了发现是否有遗漏的类。其中高层顺序图可以看成是分析类图与用例图的集成（横坐标是对象，纵坐标是用例的执行顺序），通过“走读”顺序图，研究当前这些对象是否能保证用例中的事件流顺利完成。

[实验结果]

1.采用 StarUML 绘制的，与待开发用例相关的概念模型

【及涉及模型组织、绘制要点等重要步骤】

绘制要点：搞清楚名词和动词的寻找以及他们之间的数量关系



2.给出上述概念模型建立过程中，概念筛选与关联取舍的详细过程和理由。

【例如用的哪种方法、哪些重点部分的分析过程】

概念类的类别	示例
业务交易 准则：十分关键(涉及金钱)，所以作为起点	Sale，Payment
交易项目 准则：交易中通常会涉及项目，所以置为第二	SalesLineItem
与交易或交易项目相关的产品或服务 准则：(产品或服务)是交易的对象。所以置为第三	Item
交易记录于何处? 准则：重要	Register，Ledger
与交易相关的人或组织的角色；用例的参与者 准则：我们通常要知道交易所涉及的各方	Cashier，Customer，Store
交易的地点；服务的地点	Store
重要事件，通常包括我们需要记录的时间或地点	Sale，Payment
物理对象 准则：特别是在创建设备控制软件或进行仿真时非常有用	Item，Register
事物的描述 准则：对象需要“规格说明”来记录其描述。(如商品卖完也应能查到规格)	ProductDescription

使用的是第二种方法：概念和关联列表法

[实验总结]

① 对重点实验结果进行分析。

【例如：对概念模型中“主线”的理解：记录销售=>处理支付；两条子线索的理解：销售->销售项->商品->商品描述->产品目录->商店、收银员->收款机->销售->支付】

第一条主线索：销售 -> 销售项 -> 商品 -> 商品描述 -> 产品目录 -> 商店

销售最终来源于商店，在销售的动作里面包含了非常符合逻辑，每一步都很符合逻辑。

第二条主线索：收银员 -> 收款机 -> 销售 -> 支付

收银员的职责就是在支付现场完成支付动作，这个动作也非常符合逻辑。

② 实验中的问题和提高。对自己的分析或设计进行评价，指出合理和不足之处，提出改进的方案。

【例如是否找到新类。按“细粒度原则”，可以将 ItemID、Money 放大成类。】

实验我学会了如何进行简单的分析建，还有一些概念的理解和关联列表法的使用。

③ 收获与体会。

【例如筛选概念的要点；区分概念与属性的要点；关联取舍的要点；画图时如何防止关联重名。】

在分析和设计过程中，我学习到了很多东西：理解了面向对象系统分析和对象类建模(概念建模)的概念，熟悉了 StarUML 绘制概念模型的方法，也发现了自己知识的不足。我认为本次实验的难点在于概念之间的关系复杂，难以确定，需要我们不断地学习，才能清楚的区分这些。

附录 1：按方法（2）

概念类分类列表

概念类的类别	示例
业务交易 准则：十分关键(涉及金钱)，所以作为起点	Sale，Payment
交易项目 准则：交易中通常会涉及项目，所以置为第二	SalesLineItem
与交易或交易项目相关的产品或服务 准则；(产品或服务)是交易的对象。所以置为第三	Item
交易记录于何处? 准则：重要	Register，Ledger
与交易相关的人或组织的角色；用例的参与者 准则：我们通常要知道交易所涉及的各方	Cashier，Customer，Store
交易的地点；服务的地点	Store
重要事件，通常包括我们需要记录的时间或地点	Sale，Payment
物理对象 准则：特别是在创建设备控制软件或进行仿真时非常有用	Item，Register
事物的描述 准则：对象需要“规格说明”来记录其描述。(如商品卖完也应能查到规格)	ProductDescription
类别 准则：描述通常有类别	ProductCatalog
事物(物理或信息)的容器	Store，Bin
容器中的事物	Item
其他协作的系统	CreditAuthorizationSystem
金融、工作、合约、法律材料的记录	Receipt，Ledger
金融手段	Cash，Check，LineOfCredit
执行工作所需的进度表、手册、文档等	DailyPriceChangeList

关联列表

类别	示例
A 是与交易 B 相关的交易	CashPayment---Sale　　动词 pay
A 是交易 B 中的一个项目	SalesLineItem---Sale　　动词 contained-in

A 是交易(或项目)B 的产品或服务	Item---SalesLineItem(或 Sale)
A 是与交易 B 相关的角色	Customer---Payment
A 是 B 的物理或逻辑部分	Drawer---Register
A 被逻辑地或物理地包含在 B 中	Register---Store，Item---Shelf
A 是 B 的描述	ProductDescription---Item
A 存 B 中被感知/记日志/记录/生成报表/捕获	Sale---Register
A 是 B 的成员	Cashier---Store
A 是 B 的组织化子单元	Department---Store
A 使用/管理/拥有 B	Cashier---Register
A 与 B 相邻	SalesLineItem---SalesLineItem

按方法（1）

主成功场景（或基本流程）：

1. **顾客**携带所购**商品**或**服务**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**商品标识**。
4. 系统逐条记录出售的**商品**，并显示该商品**项目的描述**、**价格**和**累计额**。价格通过一组价格规则来计算。

收银员重复3－4步，直到结束。

5. 系统显示总额和所计算的**税金**。
6. 收银员告知顾客总额，并提请**付款**。
7. 顾客支付，系统处理支付。
8. 系统记录完整的**销售信息**，并将销售和支付信息发送到外部的**账务系统**（进行账务处理和**提成**）和**库存系统**（更新库存）。
9. 系统打印**票据**。
10. 顾客携带商品和票据（如果有）离开。

扩展（或替代流程）：

……

7a. 现金支付：

1. 收银员输入**收取的现金额**。
2. 系统显示**找零金额**，并弹出**现金抽屉**。

附录 2：概念建模的 StarUML 操作

打开上次实验的“NextGen POS 系统.UML”文件。

下面有多种组织方式：

方式一：直接在分析视图下绘制。

StarUML 缺省方法有“分析模型”，~~但符号较少，且采用 BCE 概念，需要更改构造型~~。

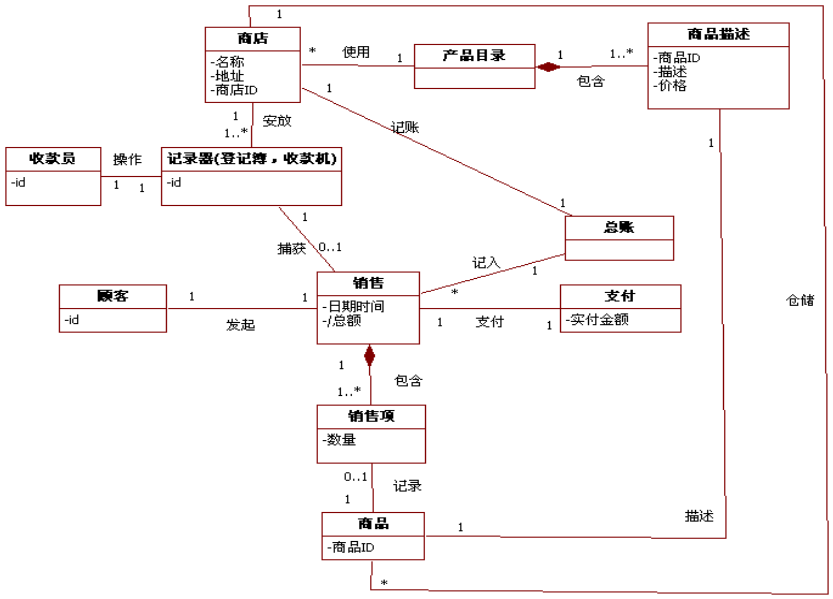
方式二：在设计视图下绘制。

为了与以后的设计相区别，可以先 ADD 一个 package“1.概念模型”，在此包下 ADD 一个类图，命名为“1. NextGen POS 系统概念模型”。

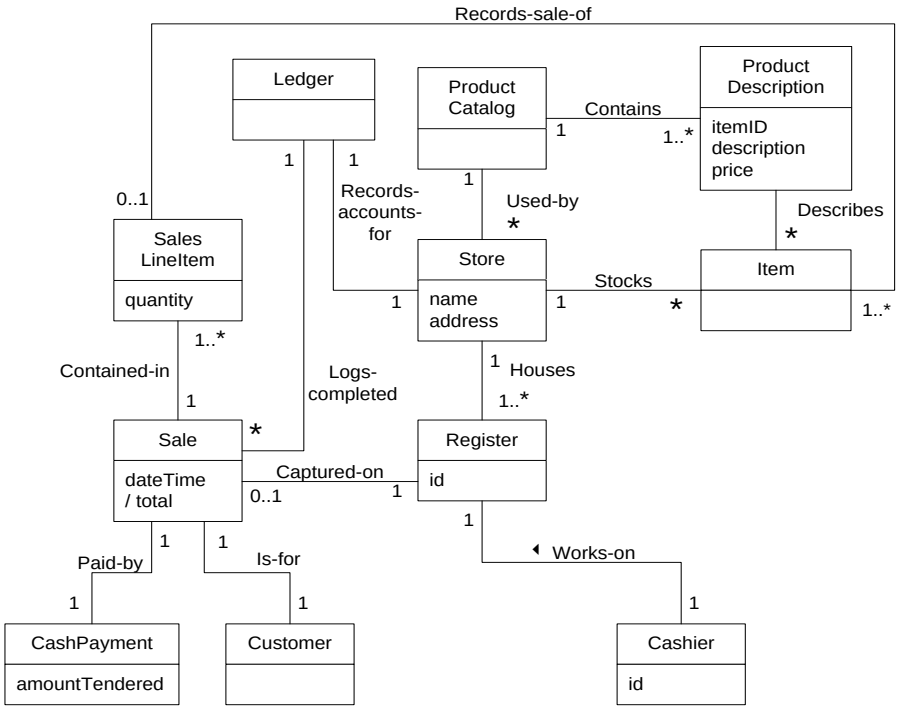
方式三：在用例模型下绘制。

在用例视图下 ADD 一个 package “1.概念模型”， 在此包下 ADD 一个类图，命名为“1. NextGen POS 系统概念模型”。

下图是按 **Peter Coad 的事务模式**的画法：



下图是 Larman 书本中的画法(还缺一些属性)：



附录 3：一些其他要点

（1） 关联画法

Stu 、Course 中间拉一根关联 SSC；若从学生拉向课程 ：

Rose,VS2010（新建项目-建模项目，菜单“体系结构”-“新建关系图”）均是宾语是 roleA；“聚合关系”似乎变成了“包含关系”，要从“班级”拉向“学生”，roleA 是学生，roleB 是班级。学生端可设置角色名 stu，多重性设为 1..n；班级端勾上 Aggregate 和 byValue，就变成实心菱形了。而 star uml 与它们相反，更符合中文习惯。

（2） 若对类图有不解之处，可“正向工程”生成代码看看，如自关联（c++就是两个角色指针）

ROSE：要在组件视图下新建组件，双击|实现哪些类，选语言|实现总切记 assigned

组件右键，ANSI c++才出现，产生代码

（除非你给出角色名或在对面类（班级）中加属性，否则无论设 1..n 还是对面是实心菱形,代码中均无反映）《可见这些元素也只是给人看的》

StarUML：检查 tools|add in manager;Model profiles 加入 c++ profiles；类，右键，generate code；支持未命名关联的属性可见性。

（3） 类符号：三栏矩形，类名、属性、方法。

属性、方法的访问权限：公有、私有、保护（+、-、#） 【ROSE 换成了图形：空白、锁、钥匙】

属性、方法的数据类型是写在后面的，如- age:int

《面向对象的系统分析与设计(UML)》实验 3 设计建模（职责分配）

[实验日期]　2022　年　5　月　9　日

[实验目的]

（1）理解交互图（主要是顺序图、协作图）的基本概念。

（2）掌握用例逻辑时序的分析和设计（职责分配）方法。

（3）掌握使用 StarUML 绘制顺序图的方法。

[实验内容]

在用例模型和概念模型的基础上，对首选的用例（处理销售用例）进行事件分解，识别出系统事件，并为对应的系统操作并写出契约的后置条件；为每个系统操作画顺序图，为对象分配职责。

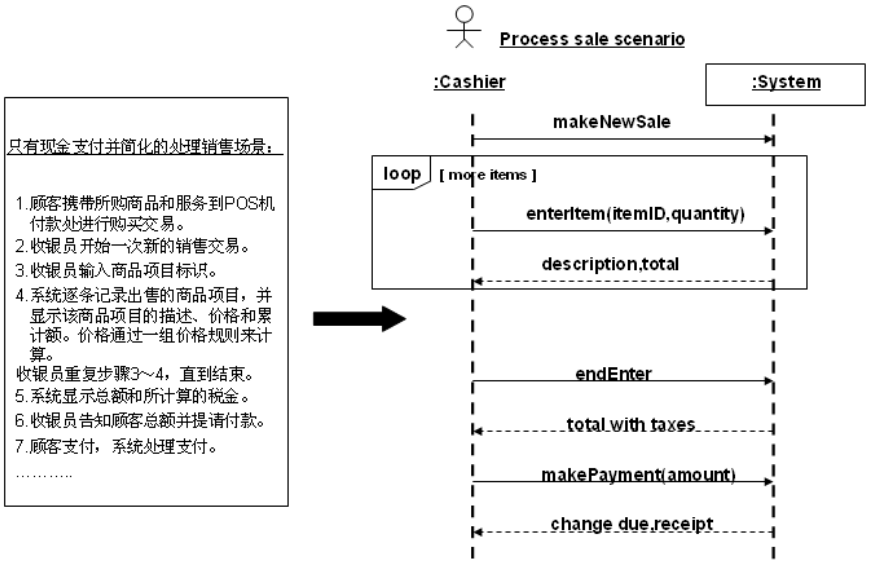
[实验原理和步骤]

原理：

(1) 在系统顺序图中，所有的系统都被当成黑盒子看待，顺序图的重点是参与者发起的跨越系统边界的事件。

(2) 系统事件是由某参与者发起的指向系统的输入事件。一个事件的发生能够触发一个响应操作的执行。

(3) 请仔细研究下图,考察它是如何从左边的"处理销售"用例的文字描述中分解出 4 个系统事件的。



（4）参照用例模型和概念模型，为每个系统操作估计后置条件。（创建实例、形成链接、修改属性）

（5）按照 GRAS(P)模式为对象分配职责。

步骤：

（1）分析首选用例的文字描述，按事件进行分解，识别出系统事件。

- √ makeNewSale
- √ enterItem
- √ endEnter
- √ makePayment

（2）为每个系统事件估计后置条件。

√ 契约CO1：makeNewsale

操作：makeNewSale()

交叉引用：用例：处理销售

前置条件：无

后置条件：·创建了Sale的实例s(创建实例)。
·s被关联到Register(形成关联)。
·s的属性被初始化(修改属性)。

√ 契约CO2：enterItem

操作：enterItem(itemID:ItemID,quantity:integer)

交叉引用：用例：处理销售

前置条件：正在进行的销售

后置条件：·创建了SalesLineItem的实例sli(创建实例)。
·sli被关联到当前Sale(形成关联)。
·sli.quantity赋值为quantity(修改属性)。
·基于itemID的匹配，sli被关联到ProductDescription(形成关联)。

√ 契约CO3：enterItem

操作：endEnter()

交叉引用：用例：处理销售

前置条件：正在进行的销售

后置条件：·Sale.isComplete被置为真(修改属性)

√ 契约CO4：makePayment

操作：makePayment(amount:Money)

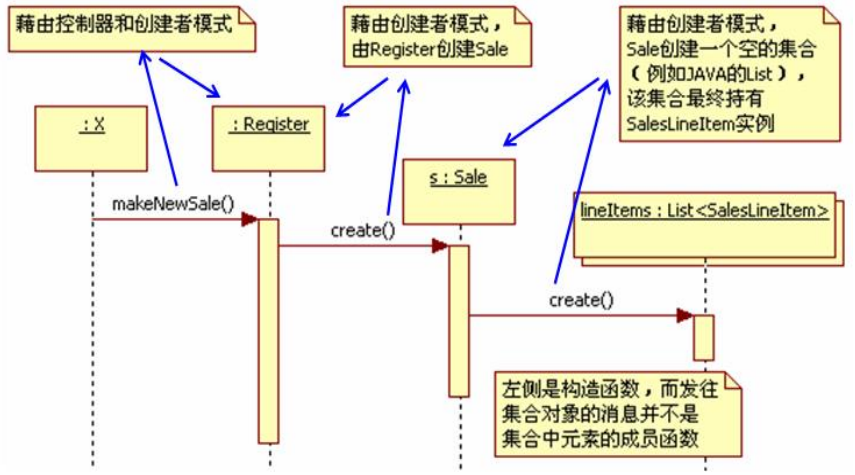
交叉引用：用例：处理销售

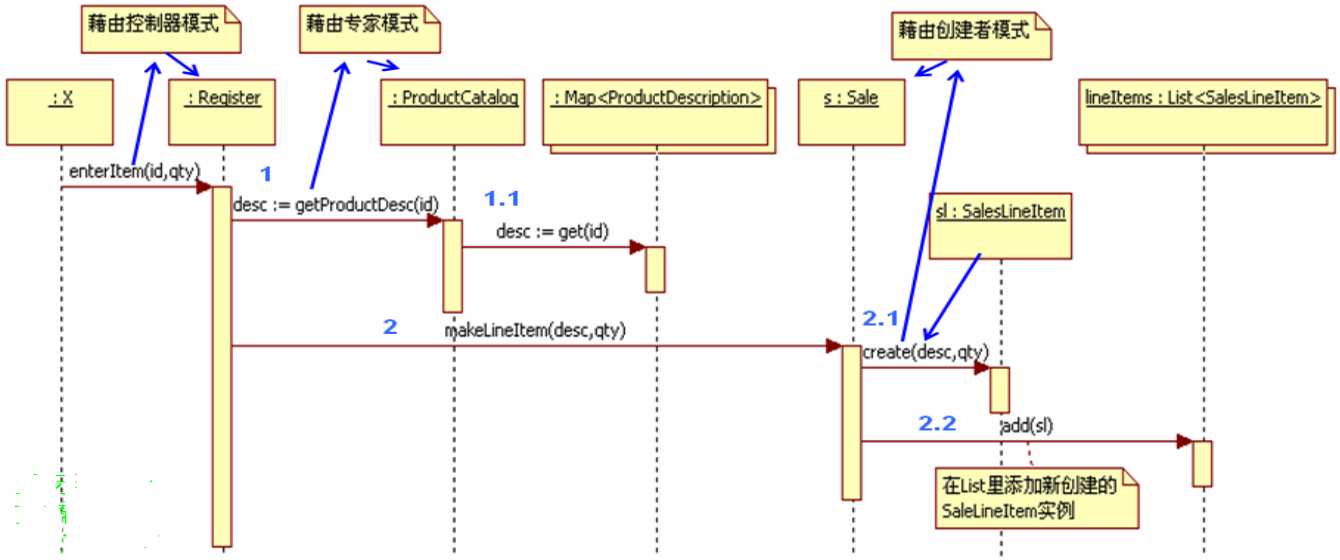
前置条件：有正在进行的销售

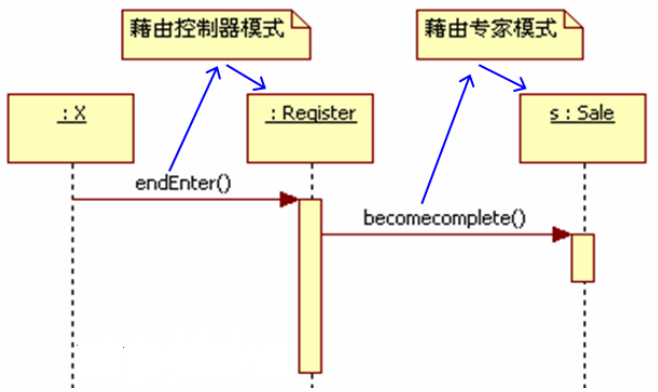
后置条件：·创建了Payment的实例p(创建实例)。
·p.amountTendered被赋值为amount(修改属性)。
·p被关联到当前的Safe(形成关联)。
·当前的Sale被关联到Store(形成关联)。(将其加入到已完成销售的历史日志中。)

（3）按 GRAS(P)模式进行设计。

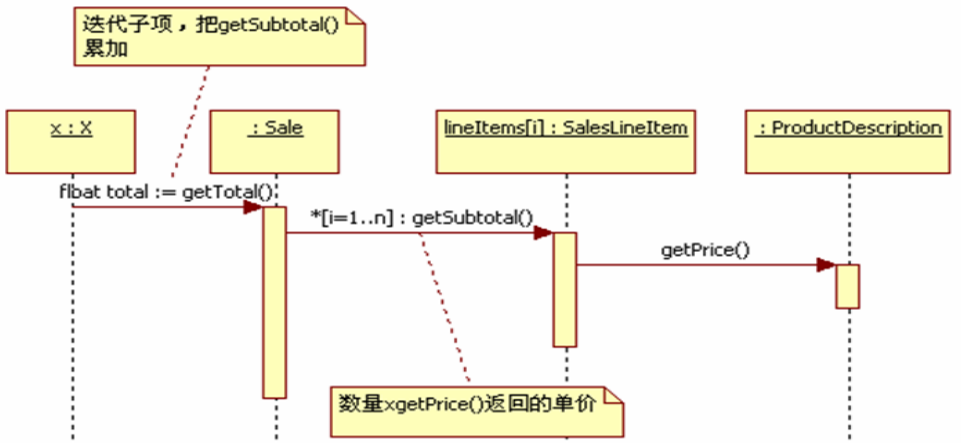
首先考虑控制者,领域控制者选 “Register”。为了避免使用 FORM,窗口等表示层对象,我们人造一个类“X”向控制者发送消息，然后按 GRASP 模式向后画。共有 6 张图。

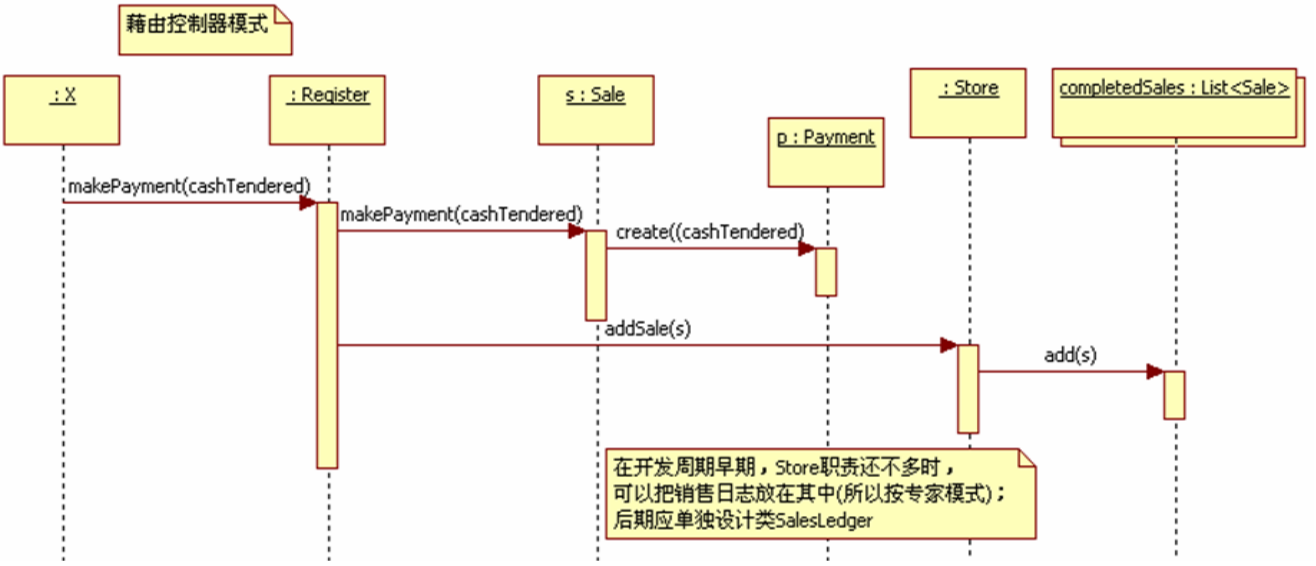




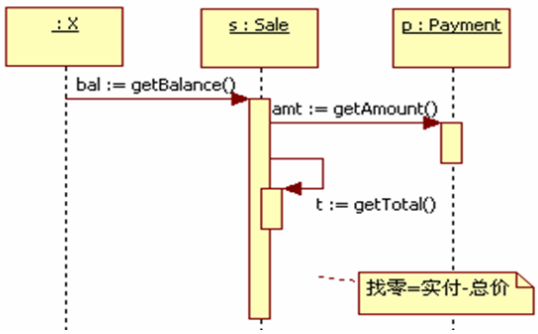


√ 不是系统操作级消息也可以单画一张图





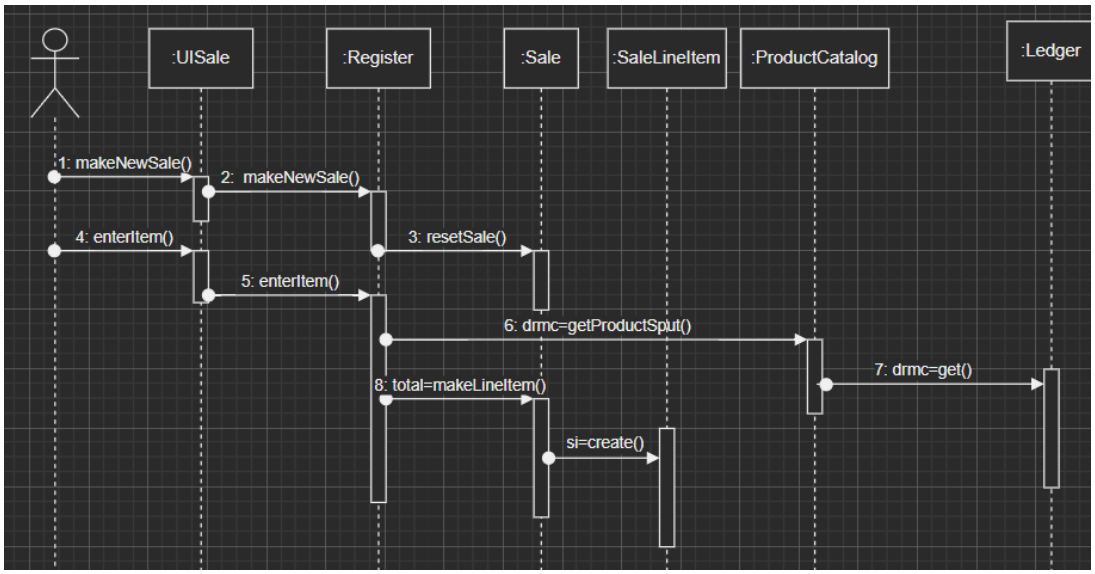
√ 不是系统操作级消息也可以单画一张图



[实验结果]

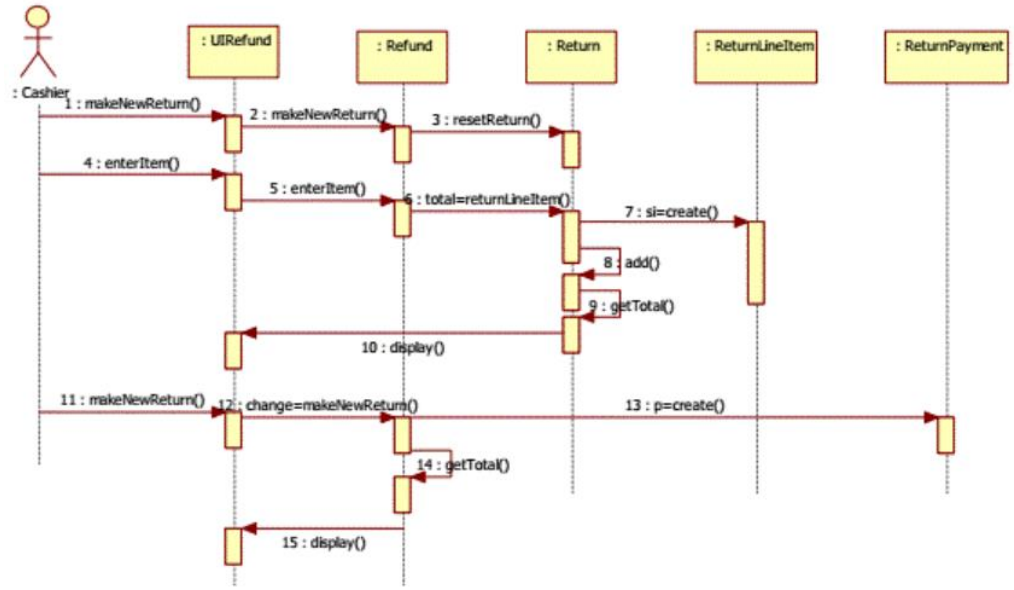
1.采用 StarUML 绘制的，与待开发用例（处理销售用例）相关的系统顺序图。

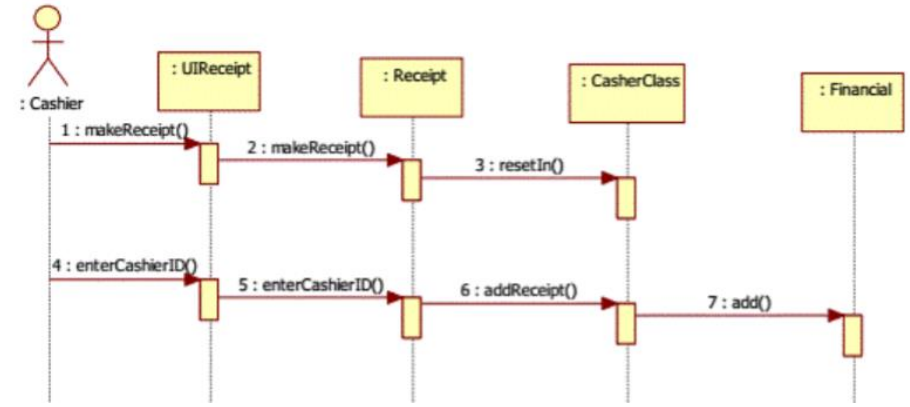
【及涉及模型组织、绘制要点等重要步骤】。

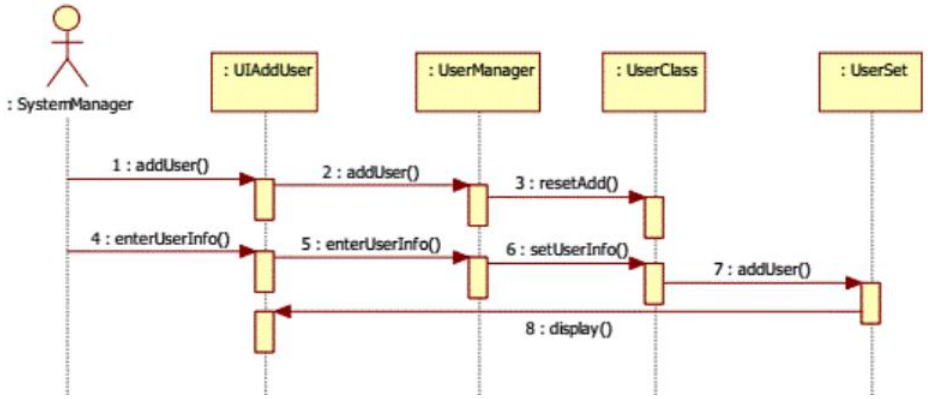


2.采用 StarUML 绘制的，每个系统事件对应的顺序图。

【以及该图中主要运用哪个模式的说明】。







[实验总结]

① 对重点实验结果进行分析。

【例如：运用前 5 个 GRASP 模式的口诀】

顺序图绘制步骤：1、确定参与交互的执行者，2、确定与执行者直接交互的对象 3、确定与交互相关的全部对象（顺藤摸瓜）注意用描述性的文字叙述消息的内容。

② 实验中的问题和提高：对自己的分析或设计进行评价，指出合理和不足之处，提出改进的方案。

【例如：是否可以运用 UML2.0 的图框表达循环 enterItem; 是否可以运用 GoF 模式对局部进行优化。】

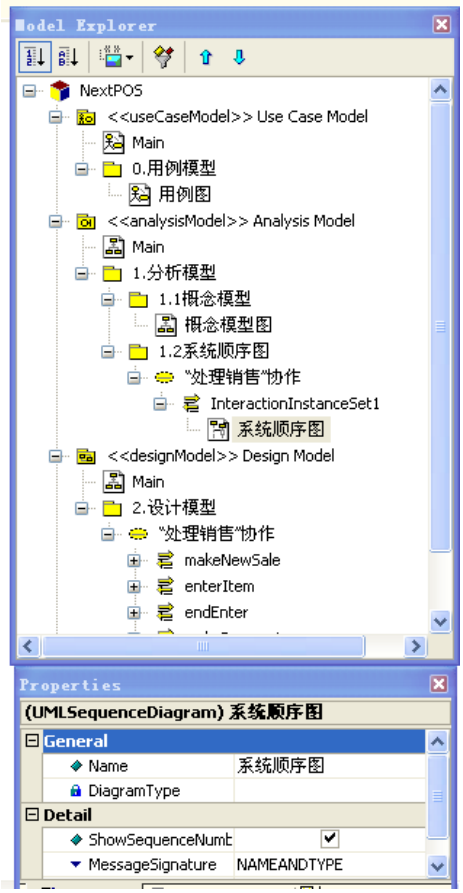
在交互阶段我画的不是很熟悉，可以运用 UML2.0 的图框表达循环 enterItem，不可以运用 GoF 模式对局部进行优化。

③ 收获与体会。

【例如：事件分解的要点；控制者选择的要点；绘制顺序图的要点。】

在本次实验中，我理解了交互图的基本概念，掌握了用例逻辑时序的分析和设计方法，掌握使用 StarUML 绘制顺序图的方法。

附录 1：在 StarUML 中如何组织模型



《面向对象的系统分析与设计(UML)》实验 4 设计建模（关联设计）

[实验日期]　　2022　　年　5　月　10　日

[实验目的]

（1）了解和掌握分析类向设计类转换的方法

（2）完善设计类图，掌握使用 StarUML 进行设计类图建模的过程

[实验内容]

根据“实验 3 设计建模（职责分配）”中的交互分析，从分析类向设计类转换，进一步完善设计类图。设计类间可见性、填上类中的方法、补上遗漏的属性可见性、补上关键的参数可见性。

[实验原理和步骤]

原理：

（1）根据具体的技术方案（架构、语言、数据库），对分析类进行适当的拆分或组合。

（2）根据顺序图中的“构造”消息确定类间导航。

（3）把顺序图中的“非构造”消息填到类中的方法栏。

（4）补上遗漏的属性可见性。

（5）补上关键的参数可见性（依赖）。

步骤：

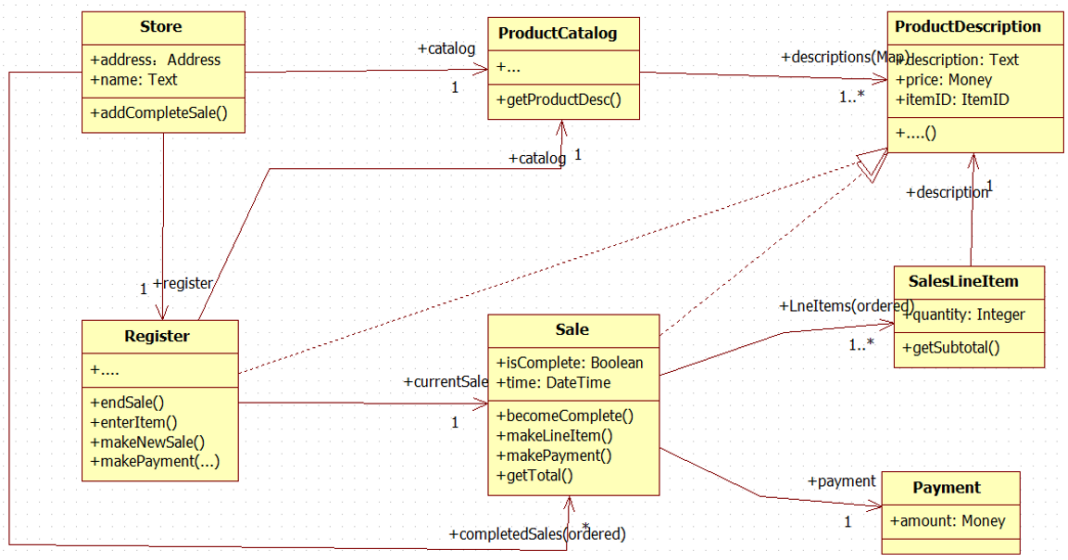
（1） 从分析类向设计类转换。可以在“2.设计模型”包里新建一个包“2.2 设计类图”（2.1 是交互模型），在此包下建一张类图“迭代 1 设计类图”，把分析类图中的元素复制过来。

（2） 进一步完善设计类图。设计类间可见性、填上类中的方法、补上遗漏的属性可见性、补上关键的参数可见性。

[实验结果]

完善的设计类图（采用 StarUML 绘制的设计类图），包括类间可见性、关联的多重性、类中重要属性和方法等。

【及涉及模型组织、绘制要点等重要步骤】。



给出分析类向设计类转换的详细过程和理由。

同构聚合在设计时，A 类要依赖 B 类，实现时，可以将 A 类设计成拥有 B 类对象的容器，或者，A 类的实现属性中含有一个这样的容器。异构聚合在设计时，A 类要依赖于这些被包含的类。实现时，可以在 A 类的实现属性中分别增加含有这些类对象作为元素的容器。

关联关系在设计时，在关联的两个类的属性中分别说明另一个类对象的指针或引用，适用于比较固定的关系。也可以设计出一个新的类，该类对象是个容器，容器的每个元素是类 A 与类 B 对象的关联对。

[实验总结]

① 对重点实验结果进行分析。

【比如如何从顺序图中找出依赖关系】

有链连接的对象间存在依赖关系

② 实验中的问题和提高：对自己的分析或设计进行评价，指出合理和不足之处，提出改进的方案。

【例如：结合具体架构、语言、数据库，是否需要对分析类进行适当的拆分或组合；是否可以运用 GoF 模式对局部进行优化】

对分析类进行适当的拆分或组合，可以运用 GoF 模式对局部进行优化。

③ 收获与体会。

【例如：分析类向设计类转换的要点、分析依赖关系的要点；绘制设计类图的要点】

设计类图描述软件的接口部分，而不是软件的实现部分。面向对象开发方法非常重视区别接口与实现之间的差异，可以用一个类型（Type）描述一个接口，这个接口可能因为实现环境、运行特性或者用户的不同而具有多种实现方式。设计类图更易于开发者之间的相互理解和交流。设计类图通常是在分析类图的基础上进行细化和改进的。

附录 1：采用 StarUML 完善设计类图。