面向对象程序的分析与设计
Object-Oriented Analysis and Design

Lecture 9

Use Case Realization I

Prof. S. Xu

Projects

- Submission Date: Tuesday (Nov 18th) at 2pm. No late submission is accepted.
- Submission format: a hard copy

Contents

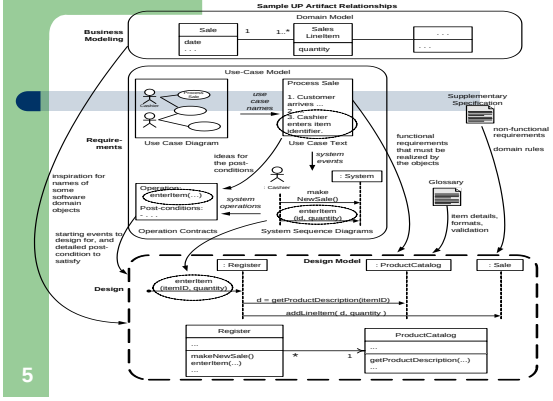
- Use-case realization

Object Design Examples With Grasp

- A use-case realization describes ***how a particular use case is realized within the Design Model***, in terms of collaborating objects.
- We now apply OO design principles and the UML to the case studies, ***to show larger examples of reasonably designed objects with responsibilities and collaborations.***

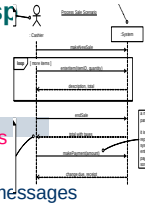
4

Object Design Examples With Grasp



5

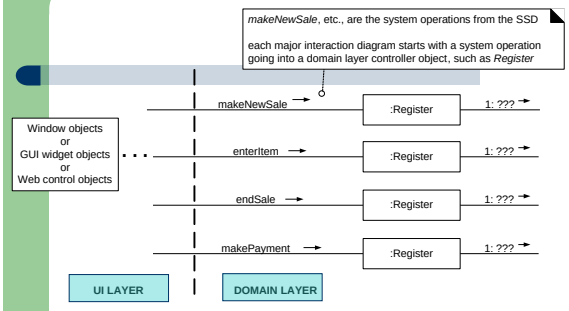
Object Design Examples With Grasp



- Some relevant points:
  - The use case suggests the system operations that are shown in SSDs.
  - The system operations become the starting messages entering the Controllers for domain layer interaction diagrams. See Figure on next slide.
    - ***This is a key point often missed by those new to OOA/D modeling.***
- Domain layer interaction diagrams illustrate how objects interact to fulfill the required tasks - the use case realization.

6

Object Design Examples With Grasp



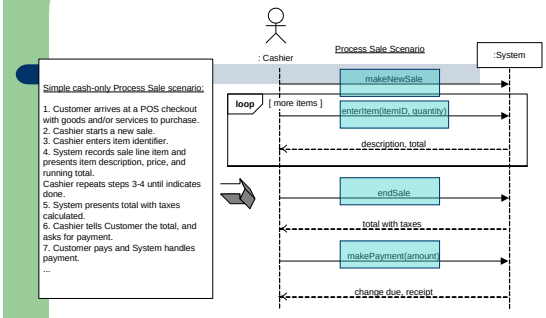
7

Artifact Comments

- In the current NextGen POS iteration we are considering scenarios and system operations identified on the SSDs of the Process Sale use case:
  - makeNewSale
  - enterItem
  - endSale
  - makePayment

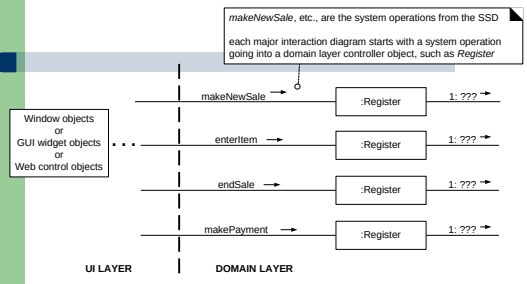
8

Artifact Comments



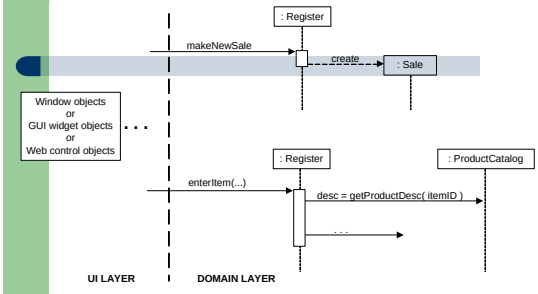
9

Artifact Comments



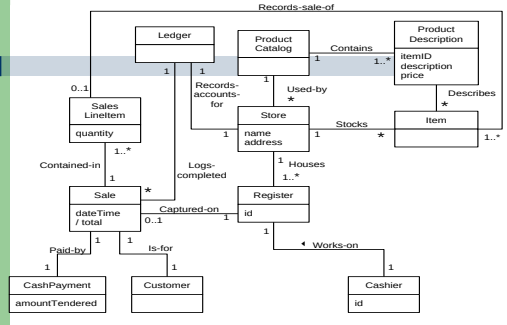
10

Artifact Comments



11

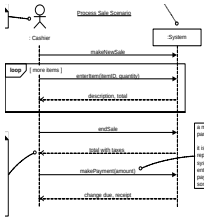
Domain Concept Model for NextGen POS



12

How to Design makeNewSale?

- The *makeNewSale* system operation occurs when a cashier initiates a request to start a new sale, after a customer has arrived with things to buy.



How to Design makeNewSale?

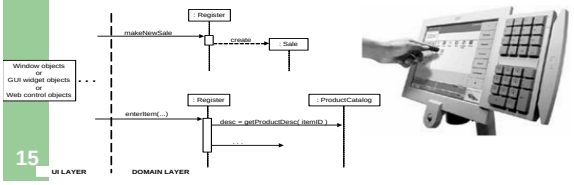
Contract CO1: makeNewSale

Operation:	makeNewSale()
Cross References:	Use Cases: Process Sale
Preconditions:	none
Postconditions:	- A Sale instance s was created (instance creation). - s was associated with the Register (association formed). - Attributes of s were initialized.

How to Design *makeNewSale*?

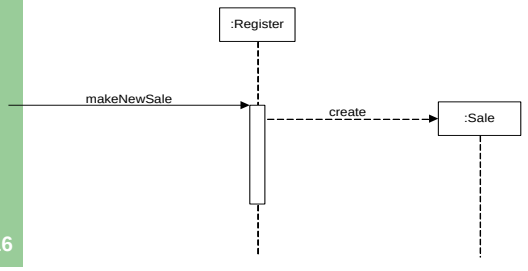
- The first design choice.

- By the Controller pattern, as we discussed previously, Register object represents the first object in Domain Layer



How to Design *makeNewSale*?

- Based on the Controller pattern, the interaction diagram shown below begins by sending the system operation ***makeNewSale*** message to a Register software object.



How to Design makeNewSale?

Contract CO1: makeNewSale

Operation:	makeNewSale()
Cross References:	Use Cases: Process Sale
Preconditions:	none
Postconditions:	- A Sale instance s was created (instance creation). - s was associated with the Register (association formed). - Attributes of s were initialized.

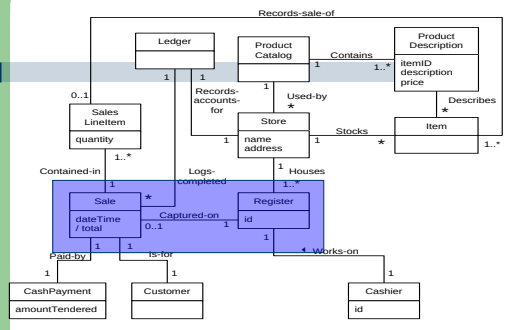
Creating a New Sale

- The GRASP Creator pattern

- Look at the domain model

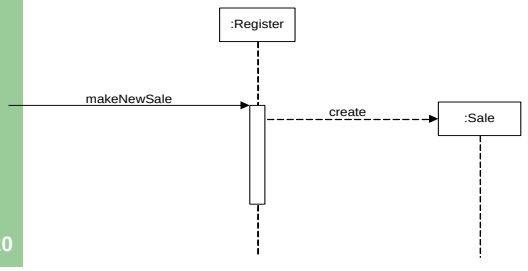
- It reveals that a Register may be thought of as recording a Sale;
  - indeed, the word "register" in business has for hundreds of years meant the thing that recorded (or registered) account transactions, such as sales.

Use Case Realization for NextGen POS



How to Design *makeNewSale*?

- Thus, Register is a reasonable candidate for creating a Sale.



Creating a New Sale

How to Design *makeNewSale*?

- By having the Register create the Sale, ***the Register will have a reference to the current Sale instance.***
- ***How?***

Contract CO1: makeNewSale

Operation:	makeNewSale()
Cross References:	Use Cases: Process Sale
Preconditions:	none
Postconditions:	- A Sale instance s was created (instance creation). - s was associated with the Register (association formed). - Attributes of s were initialized.

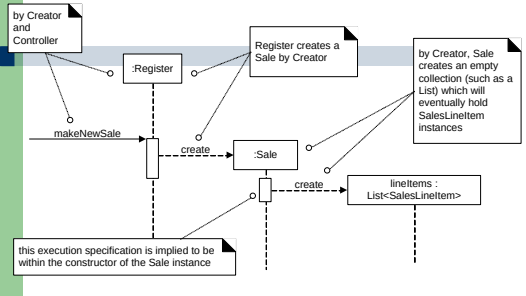
Creating a New Sale

How to Design *makeNewSale*?

- When the Sale is created, it must create an empty collection (such as a Java List) to record all the future SalesLineItem instances that will be added.
- This collection will be contained within and maintained by the Sale instance, which implies ***by Creator that the Sale is a good candidate for creating the collection.***

Creating a New Sale

How to Design *makeNewSale*?

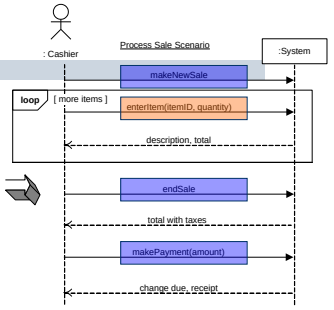


- Future Wearable Devices

Use Case Realization for NextGen POS

Simple cash-only Process Sale scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.
Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
...



25

How to Design enterItem?

- The *enterItem* system operation occurs when a cashier enters the itemID and (optionally) the quantity of something to be purchased.

Contract CO2: enterItem

Operation:	enterItem(itemID : ItemID, quantity : integer)
Cross References:	Use Cases: Process Sale
Preconditions:	There is an underway sale.
Postconditions:	- A SalesLineItem instance sli was created (instance creation). - sli was associated with the current Sale (association formed). - sli.quantity became quantity (attribute modification). - sli was associated with a ProductDescription, based on itemID match (association formed).

26

How to Design enterItem?

- The first design choice - Who is controller?
- Based on the Controller pattern, as for *makeNewSale*, we will continue to use *Register* as a controller.

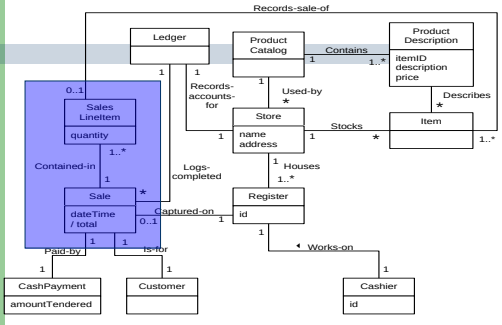
Contract CO2: enterItem

Operation:	enterItem(itemID : ItemID, quantity : integer)
Cross References:	Use Cases: Process Sale
Preconditions:	There is an underway sale.
Postconditions:	- A SalesLineItem instance sli was created (instance creation). - sli was associated with the current Sale (association formed). - sli.quantity became quantity (attribute modification). - sli was associated with a ProductDescription, based on itemID match (association formed).

27

Creating a New SalesLineItem - How to Design enterItem?

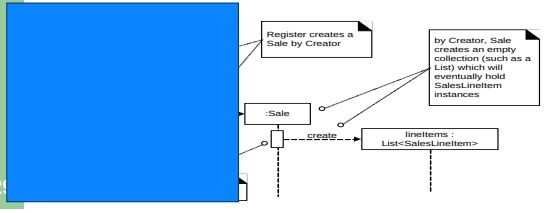
- Look at the domain model,



28

Creating a New SalesLineItem - How to Design enterItem?

- A ***Sale*** contains ***SalesLineItem*** objects.
- Hence, by Creator, ***a software Sale is an appropriate candidate to create a SalesLineItem.***



29

Creating a New SalesLineItem - How to Design enterItem?

- We can associate the ***Sale*** with the newly created ***SalesLineItem*** by storing the new instance in its collection of line items.
- Also the postconditions indicate that the new ***SalesLineItem*** needs a quantity when created;

Contract CO2: enterItem

Operation:	enterItem(itemID : ItemID, quantity : integer)
Cross References:	Use Cases: Process Sale
Preconditions:	There is an underway sale.
Postconditions:	- A SalesLineItem instance sli was created (instance creation). - sli was associated with the current Sale (association formed). - sli.quantity became quantity (attribute modification). - sli was associated with a ProductDescription, based on itemID match (association formed).

30

Creating a New SalesLineItem - How to Design enterItem?

- Therefore, the ***Register*** must pass it along to the ***Sale***, which must pass it along as a parameter in the create message.
  - In Java, that would be implemented as a constructor call with a parameter.

Contract CO2: enterItem

Operation:	enterItem(itemID : ItemID, quantity : integer)
Cross References:	Use Cases: Process Sale
Preconditions:	There is an underway sale.
Postconditions:	- A SalesLineItem instance sli was created (instance creation). - sli was associated with the current Sale (association formed). - sli.quantity became quantity (attribute modification). - sli was associated with a ProductDescription, based on itemID match (association formed).

Creating a New SalesLineItem - How to Design enterItem?

- To summary, by Creator, a ***makeLineItem*** message is sent to a ***Sale*** for it to create a ***SalesLineItem***.
- The ***Sale*** creates a ***SalesLineItem***, and then stores the new instance in its permanent collection.
- The parameters to the ***makeLineItem*** message include the quantity, so that the ***SalesLineItem*** can record it, **and the *ProductDescription* that matches the itemID.**

Creating a New SalesLineItem - How to Design enterItem?

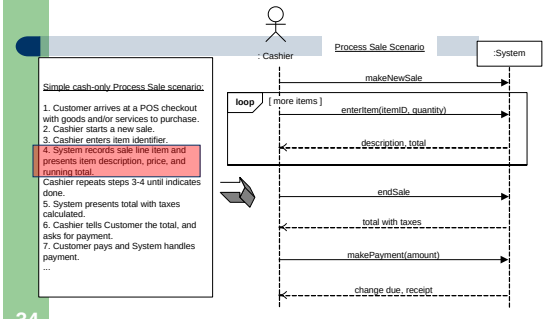
- **The *ProductDescription* that matches the itemID.**
- **Why bother with *ProductDescription*?**
- **Two reasons (1):**

Contract CO2: enterItem

Operation:	enterItem(itemID : ItemID, quantity : integer)
Cross References:	Use Cases: Process Sale
Preconditions:	There is an underway sale.
Postconditions:	- A SalesLineItem instance sli was created (instance creation). - sli was associated with the current Sale (association formed). - sli.quantity became quantity (attribute modification). - sli was associated with a ProductDescription, based on itemID match (association formed).

Creating a New SalesLineItem - How to Design enterItem?

- **Two reasons (2):**



Creating a New SalesLineItem - How to Design enterItem?

- Display Item Description and Price?
- Because of a principle of Model-View Separation, it is not the responsibility of non-GUI objects (such as a Register or Sale) to get involved in output tasks.

Finding a ProductDescription - How to Design enterItem?

- The ***SalesLineItem*** needs to be associated with the ***ProductDescription*** that matches the incoming itemID.
- This implies that we must retrieve a Product-Description, based on an itemID match.
- Before considering how to achieve the lookup, we want to consider who should be responsible for it. Thus, a first step is:

  Start assigning responsibilities by clearly stating the responsibility.

Finding a ProductDescription - How to Design enterItem?

- To restate the problem:
- Who should be responsible for knowing a ***ProductDescription***, based on an itemID match?
- Which pattern?

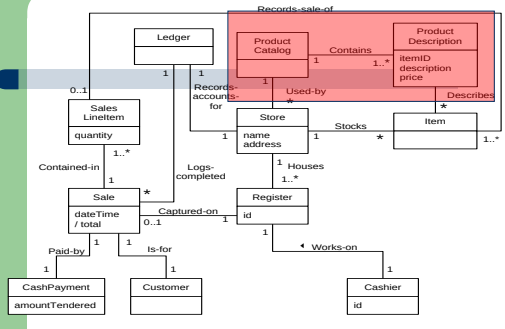
37

Finding a ProductDescription - How to Design enterItem?

- To restate the problem:
- Who should be responsible for knowing a ***ProductDescription***, based on an itemID match?
- This is neither a creation problem nor one of choosing a controller for a system event. It is about ***Information Expert***.

38

Finding a ProductDescription - How to Design enterItem?



39

Finding a ProductDescription - How to Design enterItem?

- Analyzing the Domain Model reveals that the ***ProductCatalog*** logically contains all the ***ProductDescriptions***.
- Taking inspiration from the domain, we design software classes with similar organization: a software ***ProductCatalog*** will contain software ***ProductDescriptions***.

40

Finding a ProductDescription - How to Design enterItem?

- Then by Information Expert, ***ProductCatalog*** is a good candidate for this lookup responsibility since it knows all the ***ProductDescription*** objects.
  - The lookup can be implemented, for example, with a method called ***getProductDescription*** (abbreviated as getProductDesc in some of the diagrams).
- Who should send the ***getProductDescription*** message to the ***ProductCatalog*** to ask for a ***ProductDescription***? (Register?Sales? SaleLineItem?...)
- Some reasonable assumptions have to be made…

41

Finding a ProductDescription - How to Design enterItem?

- It is reasonable to ***assume*** that a long-life ***Register*** and a ***ProductCatalog*** instance were created when the application is first executed.
- And that the ***Register*** object is permanently connected to the ***ProductCatalog*** object.
- We know that the ***Register*** can send the ***getProductDescription*** message to the ***ProductCatalog***.

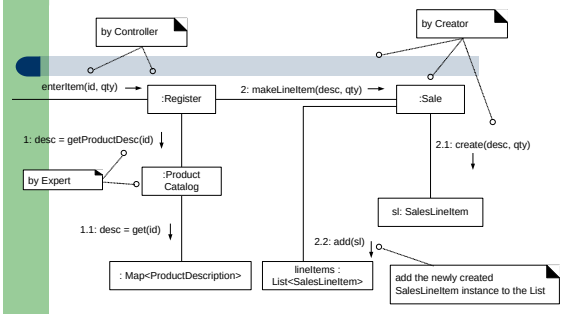
42

Finding a ProductDescription - How to Design enterItem?

- **Visibility** is the ability of one object to "see" or have a reference to another object.
- For an object to send a message to another object, it must have visibility to it.
  - Since we assume that the ***Register*** has a permanent connection or reference to the ***ProductCatalog***, ***it has visibility to it***, and hence can send it messages such as ***getProductDescription***.

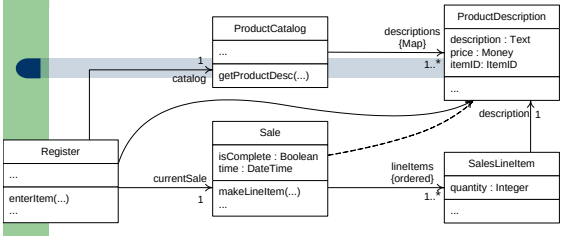
43

The Final Design - How to Design enterItem?



44

The Final Design - How to Design enterItem?



45

Review

- What is use case realization?

Review

- If A is closely related to B then B will be the creator of A **[True/False]**
- High cohesion is usually supported by Information Expert **[True/False]**
- Reusability is an advantage of Low coupling **[True/False]**
- Maintenance become difficult if high cohesion is followed. **[True/False]**

Review

- What are the GRASP patterns? Name and briefly describe those studied so far
- What is use case realization? Examples?
- How to make the decision for controller object to accept the system operations?
- Given the class diagram of a pattern, identify the pattern being used in the design.
- Given a design model, identify which design patterns (GRASP) would motivate or justify certain aspects of the design.

Review

- Explain what the **Creator pattern** is and the **questions** and the **solution** it solves.

Name: Creator
Problem:
Solution:

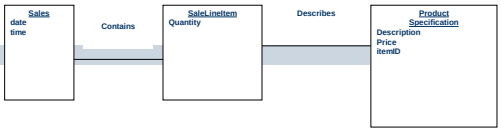
Review

- Explain what the **Low Coupling** is and the **questions** and the **solution** it solves.

Name: Low Coupling
Problem:
Solution:

Review

-



Considering the above diagram, who should be responsible for knowing the grand total amount of a sale? Explain. Also, identify the pattern you have applied here.

-

- Transparent phone concept

-

- Mobile Computing