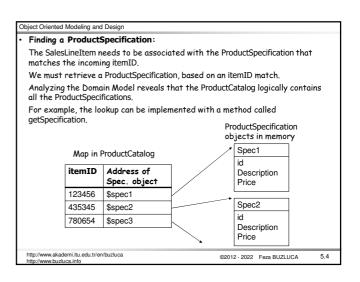


Object Oriented Modeling and Design Design Example: enterItem The enterItem system operation occurs when a cashier enters the itemID and $\,$ (optionally) the quantity of an item the customer buys. Actually, we could design this operation right after makeNewSale, but as this operation is more complicated than others, I left it to this chapter. Remember the contract of the operation. Contract CO2: enterItem Operation: enterItem(itemID: ItemID, quantity: integer) Cross References: Use Cases: Process Sale Preconditions: There is a sale underway 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 ProductSpec. based on itemID match (association formed) Remember, postconditions give responsibilities. w.akademi.itu.edu.tr/en/buzluca ©2012 - 2022 Feza BUZLUCA 5.2

Object Oriented Modeling and Design Responsibilities and assignments: Choosing the Controller: We will continue to use Register as the facade controller. Displaying Item Description and Price: Because of the principle of Model-View Separation, it is not the responsibility of non-GUI objects. Therefore, we ignore the design of the display at this time. Creating a New SalesLineItem: Analysis of the Domain Model reveals that a Sale contains SalesLineItem objects. Therefore, 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. ademi.itu.edu.tr/en/buzluca 5.3 ©2012 - 2022 Feza BUZLUCA



Sending a message to a ProductCatalog to get the specification:

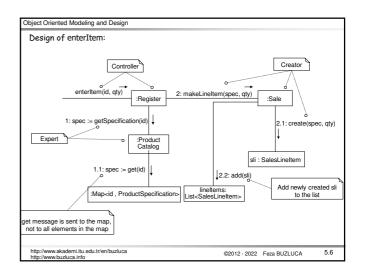
Who will get the specification from the catalog; Register or Sale?

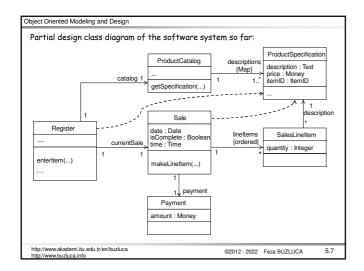
It is reasonable to assume that Register and ProductCatalog instances were created during the initialization of the system (Store) and that the Register object is permanently connected to the ProductCatalog object.

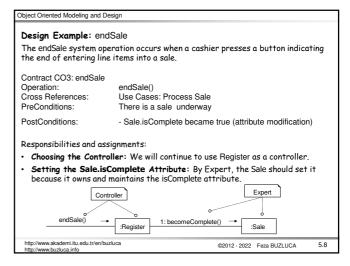
With that assumption (we must remember this during the design of initialization operations), we know that the Register can send the getSpecification message to the ProductCatalog.

Another possibility is that the Sale sends the getSpecification message to the ProductCatalog.

Which assignment is better? Coupling, cohesion...







Design Example: Calculating the balance

The "Process Sale" use case implies that the balance due from payment should be displayed somehow.

Because of the Model-View Separation principle, we do not concern ourselves with how the balance will be displayed or printed, but we must ensure that it is known.

Responsibility:

Who is responsible for knowing the balance?

To calculate the balance, we need the sale total and payment cash tendered. Therefore, Sale and Payment are partial Experts on solving this problem.

Solution 1:

If we assign the responsibility for knowing the balance to Payment, it needs visibility (coupling) to the Sale to ask the Sale for its total.

Since it does not currently know about the Sale (class diagram in 5.7), this approach would increase the overall coupling in the design (violates the Low Coupling pattern).

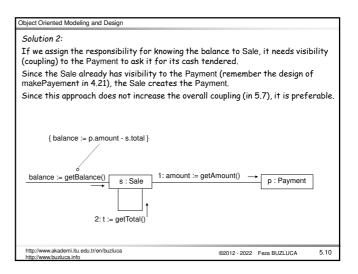
In this case, a new arrow from Payment to Sale would be necessary.

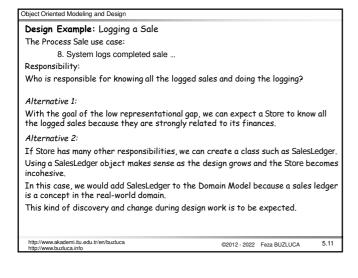
5.9

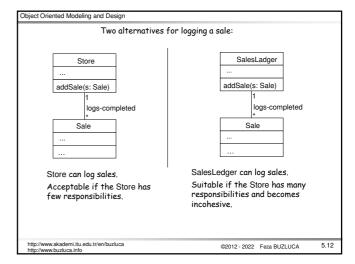
©2012 - 2022 Feza BUZLUCA

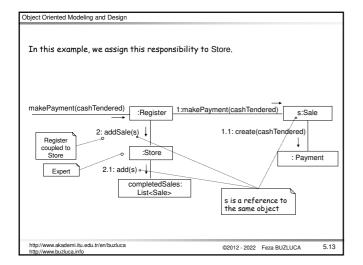
Object Oriented Modeling and Design

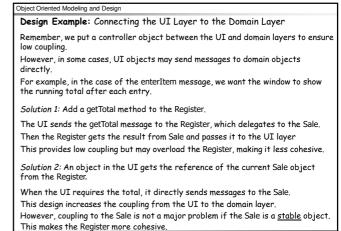
akademi.itu.edu.tr/en/buzluca



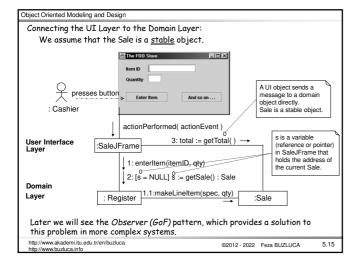


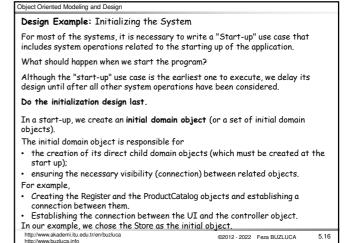


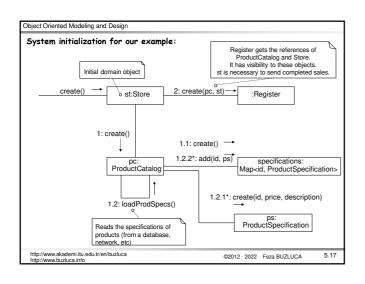




©2012 - 2022 Feza BUZLUCA







```
Object Oriented Modeling and Design
 Initialization in Java
 public class Main
                                        // Java
 public static main( String[] args)
   // Store is the initial domain object
   Store store = new Store();
   Register register = store.getRegister();
                                                    // register is created by Store
   ProcessSaleJFrame frame = new ProcessSaleJFrame(register); // Frame is connected
                                                                     // to Register
  Initialization in C++:
   int main()
                               // C++
     // Store is initial domain object
     Store store;
     Register *register = store.getRegister();
     ProcessSaleJFrame *frame = new ProcessSaleJFrame(register);
     ://www.akademi.itu.edu.tr/en/buzluca
                                                          ©2012 - 2022 Feza BUZLUCA
                                                                                       5.18
```

