

CPSC 462 - Software Design

Instructor: Christopher T. Ryu, Ph.D., Professor

Homework #3

Submission Date : 10.31.12

Team: NewGen

Name: Michael Robertson

Email: mirob2005@gmail.com

Name: Xinchao Liu

Email: lookfor@188.com

Name: Wee Siang Wong

Email: willydk@gmail.com

Name: Bryan Tamada

Email: btamada@gmail.com

Name: Wei Jen Lin

Email: littlemike44@hotmail.com

1. CRC models for “Process Sale”

Class: Store	
Responsibility:	Collaborator:
maintains product catalog	ProductCatalog
stores items	Item
logs sale orders	Sale
manages register status	Register

Class: ProductCatalog	
Responsibility:	Collaborator:
manages item id/price	Item, ItemID, Money
stores item descriptions	Item
generates catalog for store	Store

Class: Item	
Responsibility:	Collaborator:
define item id/price	ProductCatalog
define stock status	Store

Class: ItemID	
Responsibility:	Collaborator:
keeps item ID in right format	

Class: Sale	
Responsibility:	Collaborator:
adds SaleLineItem	SaleLineItem
logs sale date/time	Store
logs sale total payment	Payment
defines sale status	Interface, Register

Class: SaleLineItem	
Responsibility:	Collaborator:
shows item id/price	Item
shows item quantity	Sale

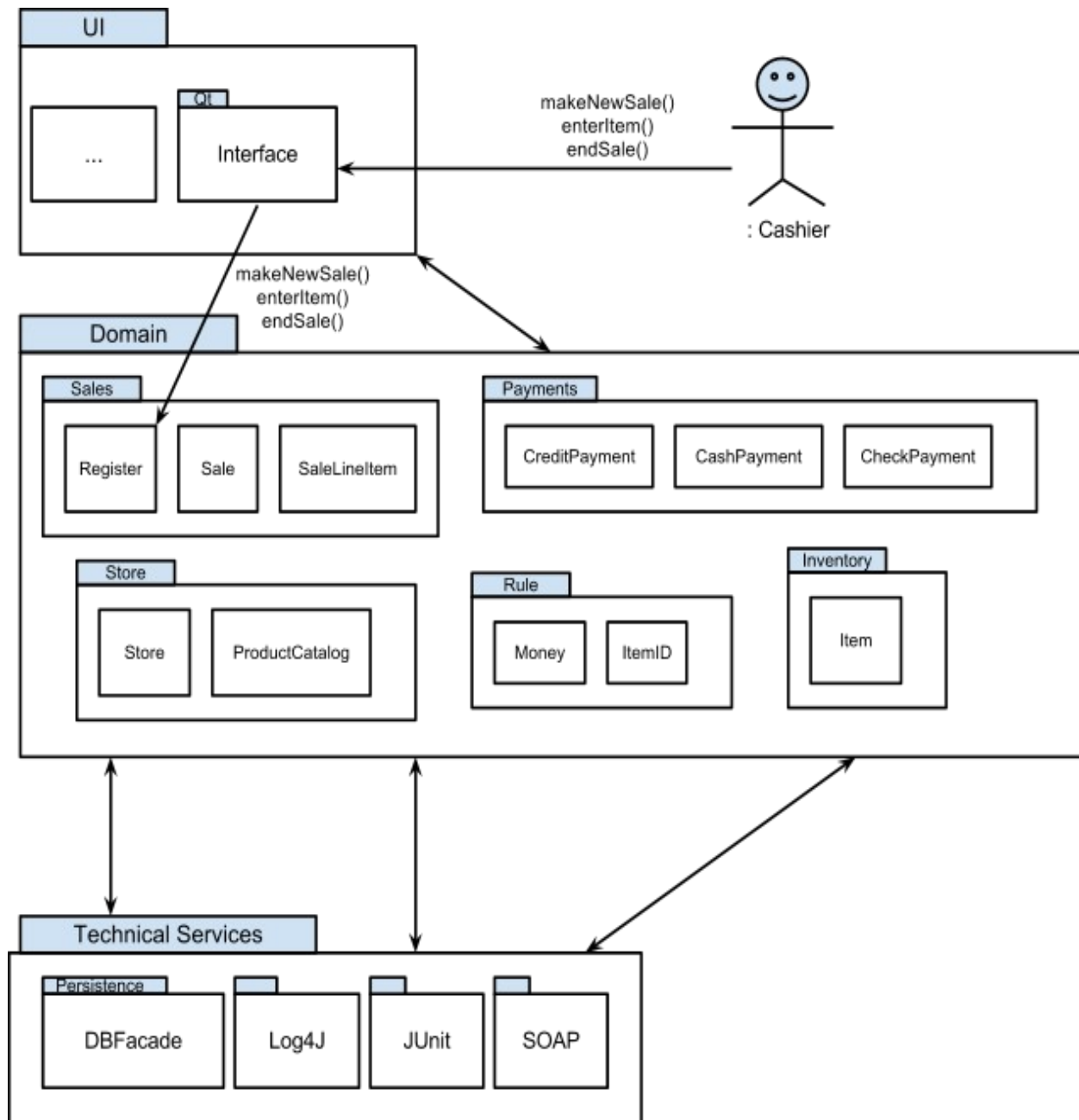
Class: Payment	
Responsibility:	Collaborator:
defines payment total amount	Sale
computes amount for change	Money
verifies payment	Register

Class: Register	
Responsibility:	Collaborator:
define register status	Store
starts a new sale	Sale
adds item to sale	SaleLineItem, ItemID
accepts tendered money	Payment, Money

Class: Money	
Responsibility:	Collaborator:
converts price to right format	

Class: Interface	
Responsibility:	Collaborator:
displays sale detail	Sale, SaleLineItem
displays price	Money
clears cart	
accepts register's commands	Register

2. Architecture Diagram



Usability - The POS System is an application that is made to be easier for the user to interact with as well as meet the requirements of the user and consumer. The usability quality attribute focuses in on the overall user experience by providing intuitive, easy to localize and globalize and easy access software.

Maintainability - Our POS system should be able to undergo changes with a certain amount of ease because these changes could impact several parts of the system such components, services, features and the interface. That is why when adding or changing the functionality, fixing errors, and meeting business requirements the system should be easy to accommodate these changes.

Reliability - A system should be able to remain operational over time without much trouble. A POS system running at a grocery store should have very little probability of failing to perform its intended functions over a specific time interval since it is possible that they are running 24 hours per day.

Scalability - In the future it is possible that the system would be required to handle more processes, be able to accommodate new resources (such as hardware), or an expansion of data. To protect against increases in load without sacrificing performance or having the ability to be readily enlarge our system scalability is an important quality attribute to handle these changes and safeguard against future goals while assuring that the system can be utilized.











Security - Due to our system dealing with third-party entities such as the authorization of checks or credit cards during the payment process we want our system to be secure. A secure system has the capability of preventing malicious or accidental actions outside of its usage and to prevent the leakage, retrieval and modification of personal information.

Availability - Similar to reliability, our system must be available for a period of time where it is functional and working. We can measure the system's rate of availability as a percentage of the total system downtime over a predefined period. System availability will be affected by system errors, infrastructure problems, malicious attacks, and system load.

3. Coding for all classes

```
package newgenpos;
class Register {
    private ProductCatalog catalog;
    private Sale currentSale;
    public Register(ProductCatalog pc){
        this.catalog = pc;
    }
    public void endSale(){
        //...
    }
    public void enterItem(ItemID ItemID, int qty){
        //...
    }
    public void makeNewSale(){
        //...
    }
    public void makePayment(Money cashTendered){
        //...
    }
}
```

CS462-Project / HW3 / Q3

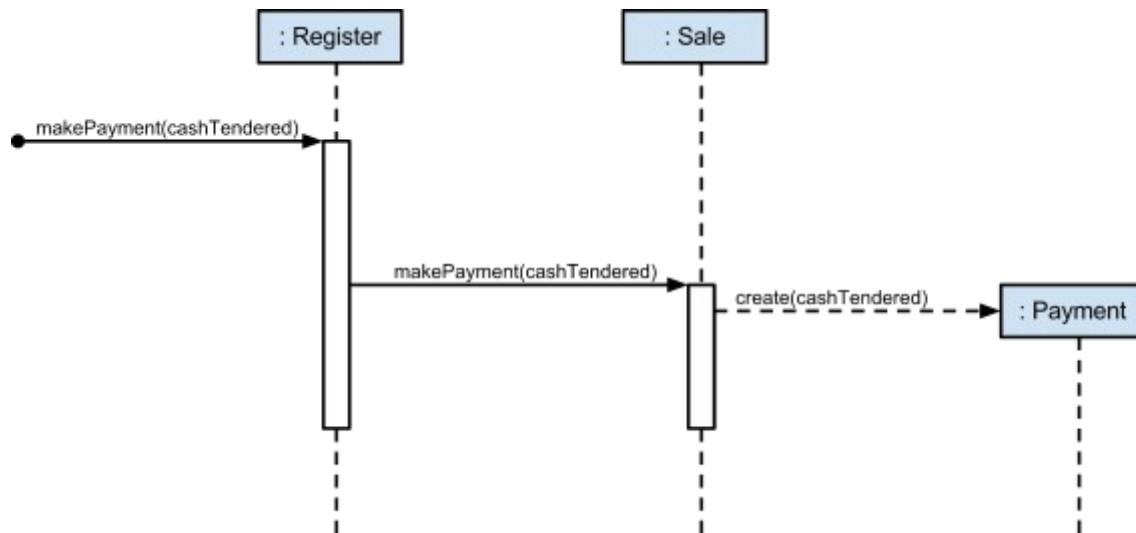
Done with HW3 #5		
 mirob2005 authored 17 hours ago		
..		
 ItemID.java	17 hours ago	Done with HW3 #5 [mirob2005]
 Money.java	17 hours ago	Done with HW3 #5 [mirob2005]
 Payment.java	17 hours ago	Done with HW3 #5 [mirob2005]
 ProductCatalog.java	17 hours ago	Done with HW3 #5 [mirob2005]
 ProductDescription.java	17 hours ago	Done with HW3 #5 [mirob2005]
 Register.java	17 hours ago	Done with HW3 #5 [mirob2005]
 Sale.java	17 hours ago	Done with HW3 #5 [mirob2005]
 SalesLineItem.java	17 hours ago	Done with HW3 #5 [mirob2005]
 Store.java	17 hours ago	Done with HW3 #5 [mirob2005]

Full organized code for all the classes can be seen at:

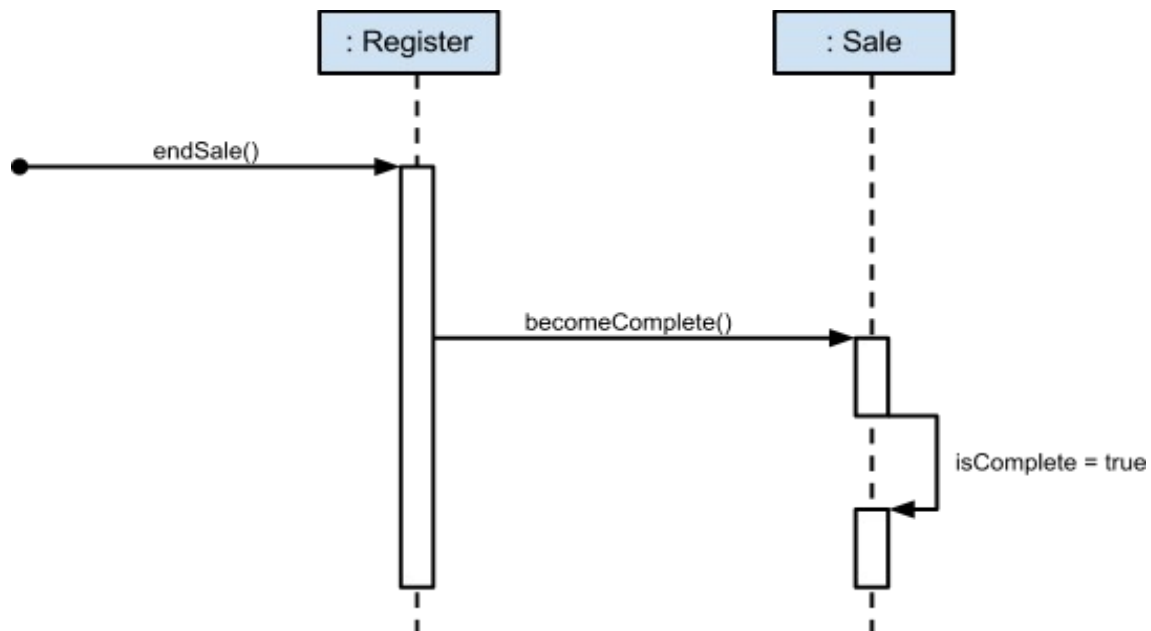
<https://github.com/miro2005/CS462-Project/tree/master/HW3/Q3>

4. Object Design Models

1. Sequence diagram for makePayment



2. Sequence diagram for endSale



5. Codes for two operations in #4

```
package newgenpos;
import java.util.Date;

class Register {
    private ProductCatalog catalog;
    private Sale currentSale;
    private ProductDescription description;

    public Register(ProductCatalog pc){
        this.catalog = pc;
    }
    public void endSale(){
        currentSale.becomeComplete();
    }
    public void enterItem(ItemID ItemID, int qty){
        description = catalog.getProductDescription(ItemID);
        currentSale.makeLineItem(description, qty);
    }
    public void makeNewSale(){
        currentSale = new Sale();
    }
    public void makePayment(Money cashTendered){
        currentSale.makePayment(cashTendered);
    }
}

class Sale {
    private boolean isComplete;
    private Date time;
    private Payment payment;
    private SalesLineItem cart;
    private Money subTotal;
    private Money total;
    private double tax;

    public Sale(){
        isComplete = false;
        time = new Date();
    }

    public boolean isComplete(){
        return isComplete;
    }

    public void becomeComplete(){
        isComplete = true;
    }
}
```

```

    public void makeLineItem(ProductDescription desc, int qty){
        cart = new SalesLineItem(desc, qty);
    }

    public void makePayment(Money cashTendered){
        payment = new Payment(cashTendered);
    }

    public Money getTotal(){
        subTotal = cart.getSubtotal();
        total = subTotal.calcTotal(tax);

        return total;
    }
}

class Payment {

    private Money amount;

    public Payment(Money Amount) {
        this.amount = Amount;
    }
}

```