

**East China Jiaotong University**

**Submitted to:**

Nishant Kumar

School of Software

ECJTU.

Submitted by:

Name (Chinese)…………….蓝波.....…………

Name (Pinyin)………………LanBo……………

Student ID……2020211001000614……..

*Submission Date*………2022/12/23………..

**Object Oriented Analysis and Design**

**Lab Report**

Lab Assignment 3

Title

Use Case Modeling

Objectives

Familiarize with the concept underlining Use Case Diagram for requirement analysis. Exposure on all elements grouped under UML Case Diagram.

Analyze functional requirements and problem statements of an intended system to be able to translate into use case modeling.

Be able to document use case description and specifications.

Aim

Step 1: Prepare the Problem Statement

Step 2: Identify Major Actor(s)

Step 3: Identify Use Cases

Step 4: Create Initial Use Case Diagram

Step 5: Describe Use Cases

Step 6: Perform Textual Analysis

Step 7: Structuring Use Cases with System Boundary

Design:

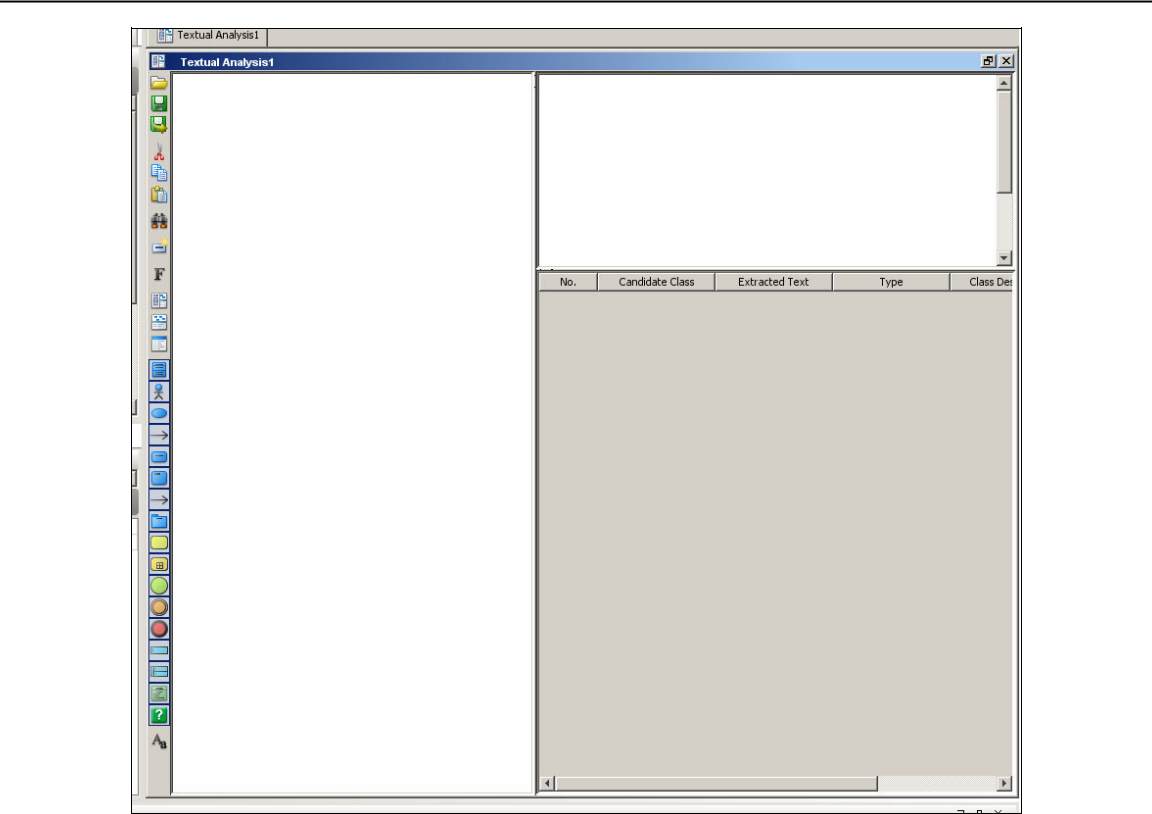
## Step 1: Prepare the Problem Statement

The problem statement is prepared through interviews with the stakeholders of the system. The problem statement can be entered into the VP-UML case tool for further requirement analysis. Simply follow the steps below.

Create a new Textual Analysis by clicking on the “**New Textual Analysis**” on the application toolbar.

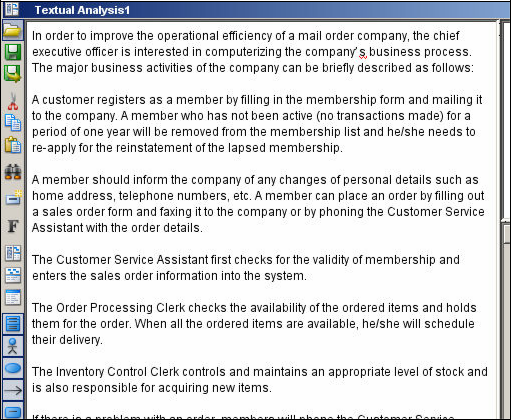
Object Oriented Analysis and Design

Lab#1



Textual Analysis working area

Type in the problem statement from the Mail Order System sample problem statement into the textual analysis text box.



Entering problem statement for Textual Analysis

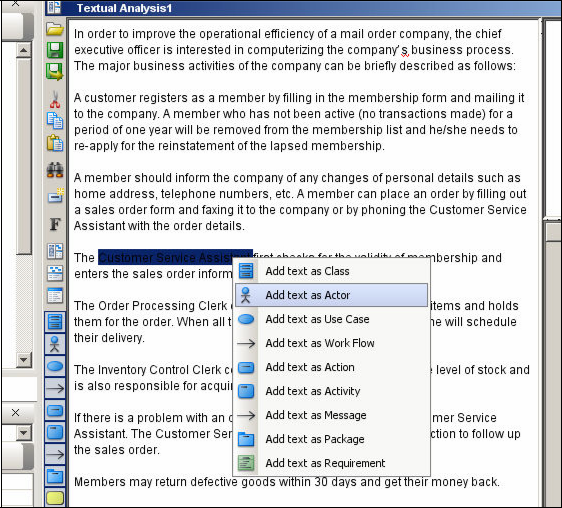
Object Oriented Analysis and Design

Lab#1

## Step 2: Identify Major Actor(s)

Once the problem statement is in the case tool, the next step is to identify actors in the **Textual Analysis** working area.

Highlight the phrase “Customer Service Assistant” in the problem statement as a candidate actor and drag it to the **Candidate Class Container** at the top right-hand corner. Note that all occurrences of the same actor in the problem statement are automatically highlighted.



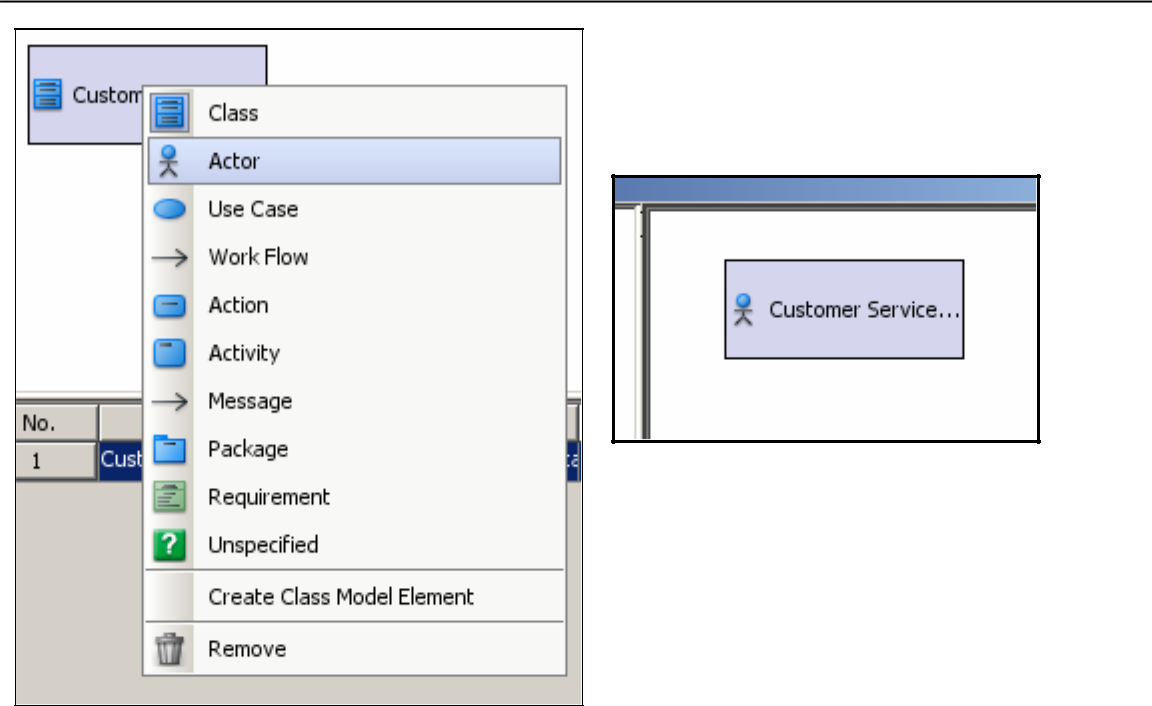
**Identifying major actors**

Now “right click” on the newly created candidate class in the **Candidate Class** **Container**. A pop-up menu will appear > Select the **Actor** option in the pop-up menuto declare the candidate class as an actor.

Note that the icon of the candidate class in the **Candidate Class Container** has changed from class to actor and the type of the candidate class has also changed to **Actor**.

Object Oriented Analysis and Design

Lab#1



Candidate Actor in Candidate Class Container

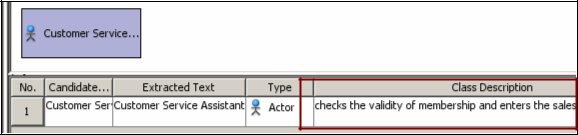
Defining actor type

To enter the description of an actor:

Go to the **Class Description** cell under the multi-line box in the bottom right corner.

Type in a brief description such as the task(s) performed by the actor.

Check the validity of membership and enters the sales



Entering actor description in Class Description

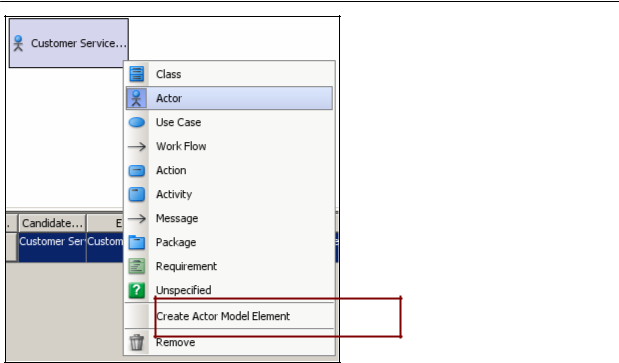
The candidate actors can be added into the model repository. Elements in the model repository can be retrieved for later use, e.g. to draw a use case diagram.

To add Customer Service Assistant (candidate actor) into the model repository, “right click” on the Customer Service Assistant.

A pop-up menu will appear. Select **Create Actor Model** in the pop-up menu

Object Oriented Analysis and Design

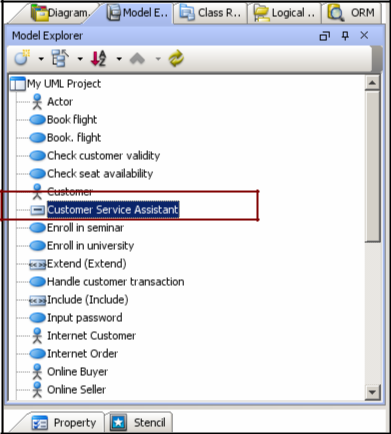
Lab#1



Creating an Actor in model repository

The Customer Service Assistant is now added to the **Model Repository Tree**.

To see the newly created actor model, click on the **Model Explorer** tab in the **Project Explorer Pane** (shown below).



Customer Service Assistant actor in Model Repository Tree

Object Oriented Analysis and Design

Lab#1

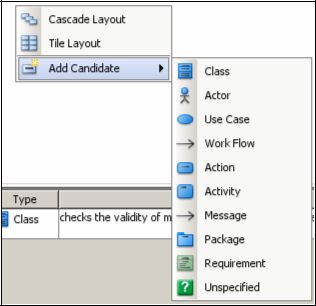
Repeat the above steps to identify and create actor models for the actors: Order Processing Clerk

Inventory Control Clerk

## Step 3: Identify Use Cases

Let us identify a candidate use case from the problem statement. It is necessary to **read through the text carefully to identify a use case.** Follow the steps below to create a use case directly from the **Candidate Class Container**.

“Right click” on the **Candidate Class Container** > A pop-up menu will appear Then select **Add Candidate** in the pop-up menu; a cascading menu will appear > Select **Use Case** in the cascading menu.

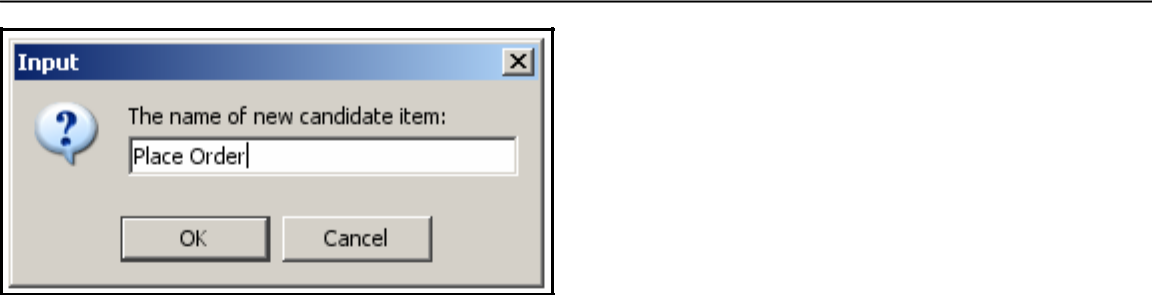


Creating a candidate use case in Candidate Class Container

An input dialog will appear. Enter Place Order in the input dialog

Object Oriented Analysis and Design

Lab#1



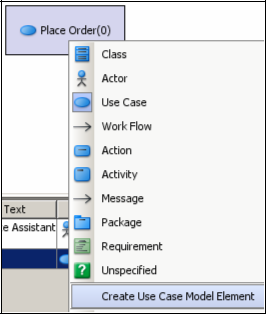
Naming a new candidate use case

“Click **OK** in the input dialog”. A new candidate use case is then created in the

**Candidate Class Container**

Now edit the use case brief description for the candidate use case the same way as you would edit the actor.

To add a candidate use case into the model repository, “right click” on the desired candidate use case in the **Candidate Class Container**. A pop-up menu will appear > Select **Create Use Case Model**

****

A new use case in Model Repository Tree

A new use case is added to the Model Repository Tree**.**

Repeat the above steps to identify all other candidate use cases below:

Check Order Status

Handle Goods Return

Update Membership Record Archive Membership

*ECJTU, NANCHANG* 13

Object Oriented Analysis and Design

Lab#1

Register New Member Process Order

Schedule Delivery Order Goods

Receive Goods Deliver Goods

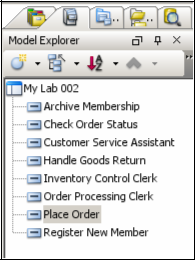
## Step 4: Create Initial Use Case Diagram

Having identified all the use cases, create the use case diagrams with the case tool following the steps below:

Create a new Use Case Diagram.

Click on the “Model” tab in the “Project Explorer”. A list of model elements will be presented.

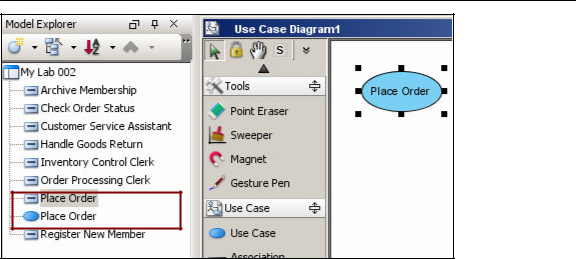
Select the use case Place Order from the **Model Explorer** and drag it to the desired location in the diagram pane. A use case is automatically placed in the diagram with the name Place Order



Models in Model Repository Tree

Object Oriented Analysis and Design

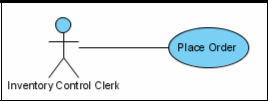
Lab#1



Use case Place Order is automatically created / listed in the model explorer

Select Customer Service Assistant from the **Model Explorer** and drag it to the desired location in the diagram pane. An actor is then placed in the diagram with the name Customer Service.

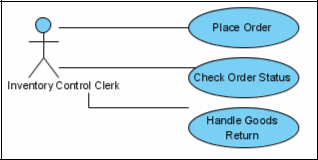
Associate the actor and use case displayed in the diagram.



Creating an association relationship using resource-centric icon

Repeat the above steps to create the following use cases and their association relationships with the Customer Service Assistant actor:

Check Order Status Handle Goods Return



Creating more use cases with Model Repository Tree

*ECJTU, NANCHANG* 15

Object Oriented Analysis and Design

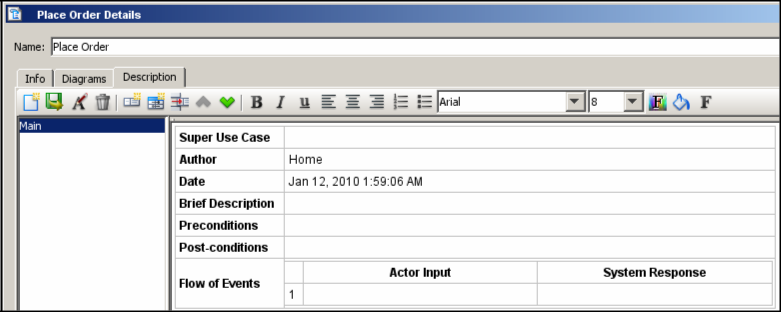
Lab#1

## Step 5: Describe Use Cases

The use cases created require further elaboration so that the next phase of the analysis can be performed. Follow the steps below:

“Right click” the use case Place Order, and choose “Use Case Details”.

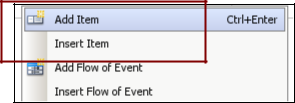
Select the “Description tab”. A Specification Dialog about the files associated to the element will be displayed.



Use case specification and template

Enter the contents for each of the elements in the use case template and click on the **OK** button to confirm the use case description.

You can also add / insert additional heading / element / texts by right clicking anywhere in the Description window.



Try to key in the full use case description for (use case) “Place Order” as given in the table earlier (page 6) and try to translate its flow of event in terms of Actor Input and System Response (2 columns format).

Object Oriented Analysis and Design

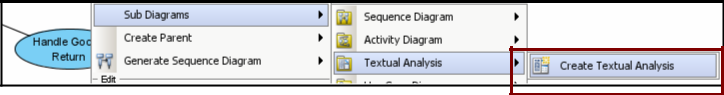
Lab#1

## Step 6: Perform Textual Analysis

A textual analysis needs to be performed for each of the use cases based on their descriptions; this will yield a set of candidate classes in the process. These classes will then be considered for inclusion in the domain class model which serves as a preliminary class model for the future development of the initial class model.

Now, let us perform a textual analysis on the Schedule Delivery use case.

“Right click” the use case Schedule Delivery and then > select Sub Diagrams > select **Create Textual Analysis** from the pop-up menu.



Launching textual analysis function with a use case

A new **Textual Analysis** window will appear. Enter the following text in the text pane:

The Order Processing Clerk selects an order from the list of filled sales orders. The system displays the sales order details, together with the member telephone number and address. The Order Processing Clerk enters the delivery date and time after talking with the member over the phone. The system records the delivery date and time of the sales order. The system records the name of the Order Processing Clerk who has handled the sales order.

Now highlight the word order as a candidate class, right click on the word order, and select **Add Text as Class** in the pop-up menu.

[Note all occurrences of the same actor in the problem statement are now automatically highlighted.]

A new candidate class is automatically created in the **Candidate Class Container** on the right-hand side and all occurrences of the same class in the problem statement are automatically highlighted.

Select the **Class Description cell** next to the class Order. Enter a brief description about the Order class.

Object Oriented Analysis and Design

Lab#1

Repeat the above steps to create the following classes: Sales Order

Member Delivery

## Step 7: Structuring Use Cases with System Boundary

Use cases with similar domain can actually be grouped into packages.

In the Mail Order System example, we can identify three packages which are **Inventory**, **Membership** and **Order Processing** which are associated with the actors. The ultimate goal is to organize the use cases into packages to maximize consistency within the individual packages and minimize coupling among these packages.

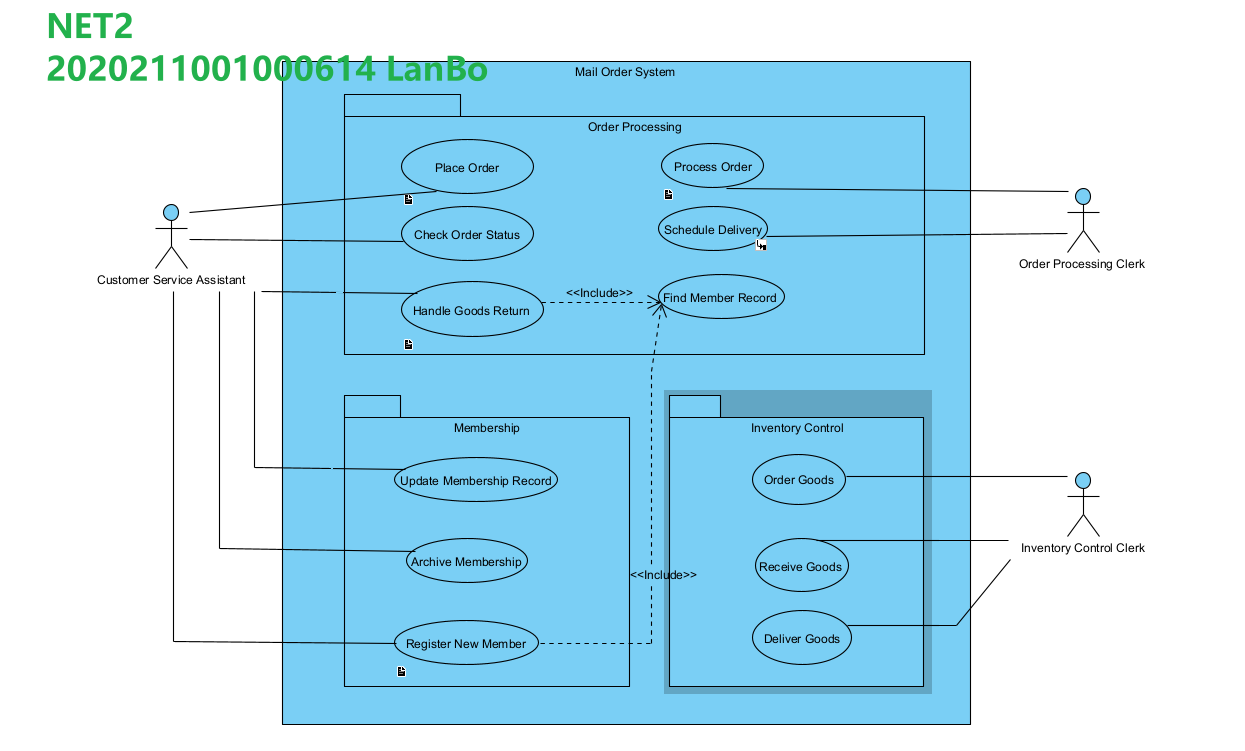
Create a package by first clicking on the **Use Case Diagram** palette.

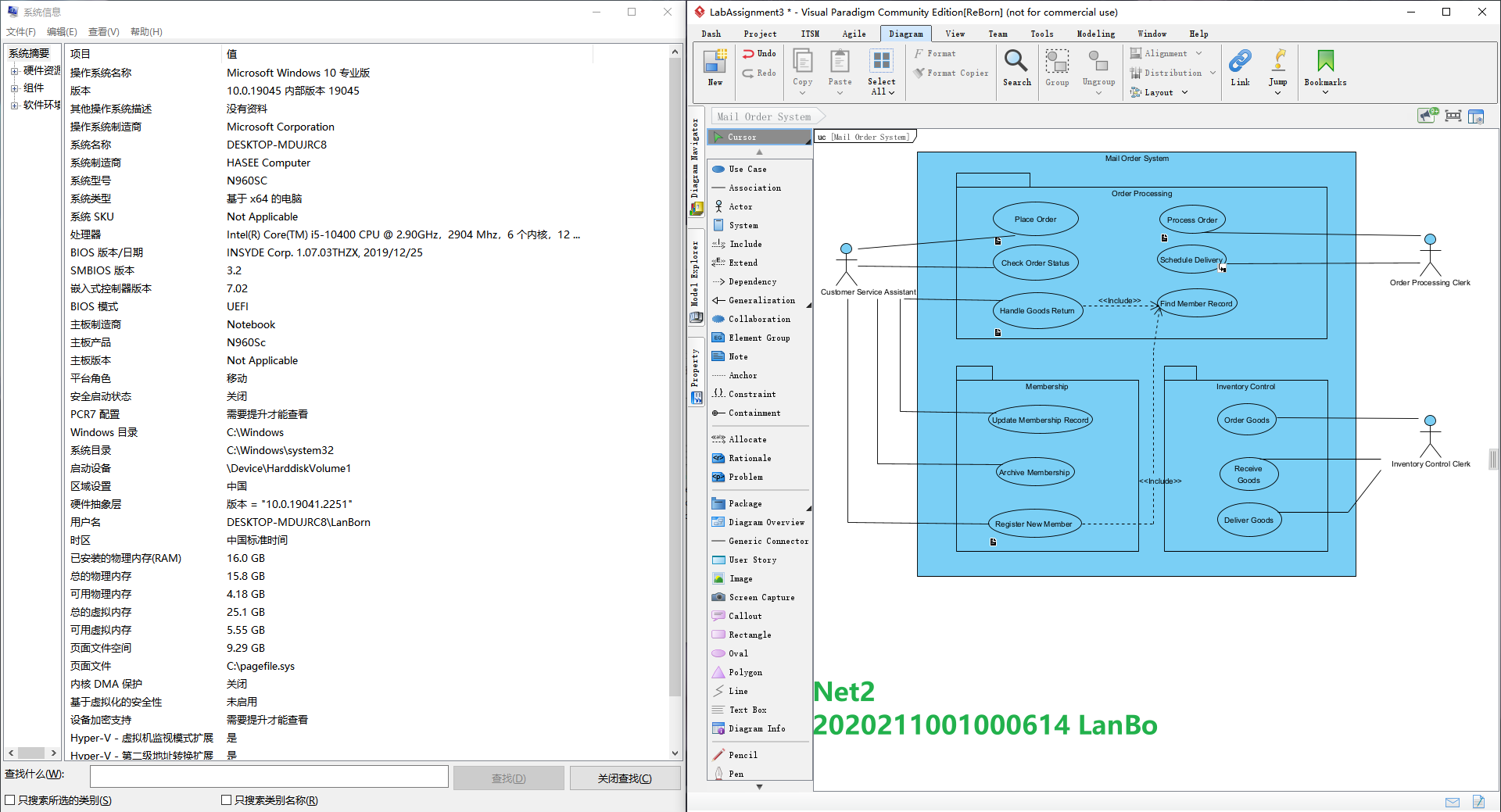
Place the mouse pointer in the design area and click once. A package symbol will then appear in the design area. Rename the new package as Inventory Control. Press Ctrl \_ Enter to finish the operation.

Move each of the use cases by dragging them into the package region where it belongs or where it is a member.

Add the system boundary to the use case model by clicking  on the **Use** **Case Diagram** palette and move the mouse pointer to the desired location on thediagram pane.

Result :





Lab Assignment 4

Title

**Domain Modeling**

Objectives

* + Use the UML notation to represent classes and their properties. Perform domain analysis to develop domain class models.
  + Model the structural aspects of problems with the class model.

Aim

GENERAL STEPS IN DOMAIN MODELING

1.Prepare problem statement for the system being developed.

2.Identify classes and objects.

3.Develop data dictionary.

4.Identify association between classes.

5.Identify attributes of classes and association classes.

6.Structure objects using inheritance.

7.Verify access paths for likely queries.

8.Iterate and refine the model.

Design:

**Step 1: Prepare the Problem Statement(start here)**

Create a new Textual Analysis , and copy the sample stock trading problem statement above.

*ECJTU, NANCHANG*

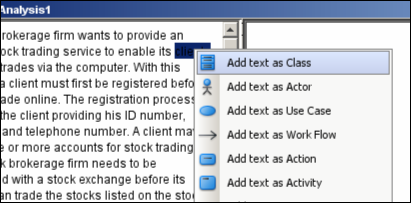
Object Oriented Analysis and Design

Lab#2

**Step 2: Identify Objects and Classes**

Once the problem statement is defined, next is to identify objects and classes in the textual analysis working area.

Highlight the term “client” as a candidate class from the textual analysis (Use ctrl+f).



Highlighting the term client as class candidate

Repeat the above steps to identify the remaining classes: Trade Order

Account

Stock Exchange Buy Order

Transaction Sell Order

Stock Trading System Execution Result

Stock

**Step 3: Develop Data Dictionary**

To further define the candidate classes selected as in step 1, enter the following description in the Class Description cell (under the Object Description sub-window).

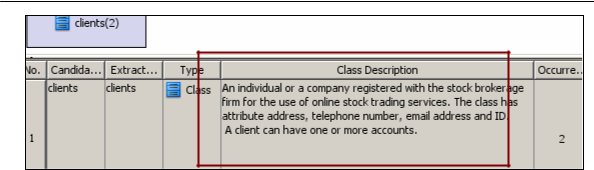
An individual or a company registered with the stock brokerage firm for the use of online stock trading services. The class has attribute address, telephone number, email address and ID. A client can have one or more accounts.

Class Description cell for Client

*ECJTU, NANCHANG* 8

Object Oriented Analysis and Design

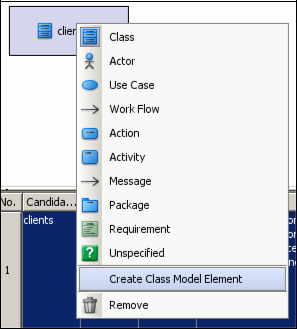
Lab#2



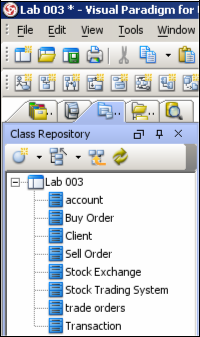
Candidate class Client description

**Repeat** the above steps to complete the dictionary for all remaining candidate classes (Refer [Appendix B](#page17)).(click link with ctrl)

We can now create the models from the candidate classes. “Right click” on the candidate class and select “Create Class Model Element”.



Creating a class model element from candidate class.



Candidate classes listed under Class Repository.

*ECJTU, NANCHANG* 9

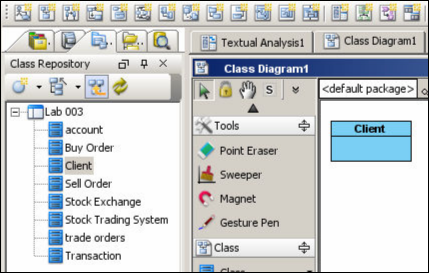
Object Oriented Analysis and Design

Lab#2

**Step 4: Identification of Associations between Classes**

Create a new Class Diagram (refer to lab 1).

Drag class client (displayed under Class Repository) into the Class Diagram pane.

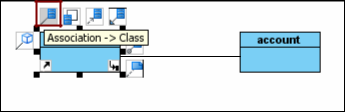


Using defined class Client in Class Diagram

Repeat the previous steps to create class Account in the Class Diagram.

Now create an association between classes “Client” and “Account”.

“Left click” the class Client and click the association icon from the Resource centric interface and drag it to class Account.



Association between classes example

Repeat the above steps to complete all other associations (Refer to [Appendix C](#page18) for description of all other classes)

Now double click the association line connecting the classes Account and Client. An in-line text editing area will be displayed to enter a text as shown below. Type the text “has”.

*ECJTU, NANCHANG* 10

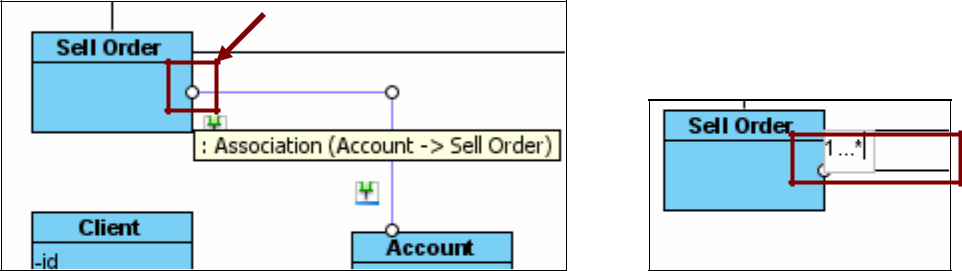
Object Oriented Analysis and Design

Lab#2



Defining association between classes

To add numeric associative (multiplicity) relationship between classes, do this by “double clicking” the end connector dots of the association line to add numerical relationship status as shown below:



**Step 5: identify Attributes of Classes and Association Classes**

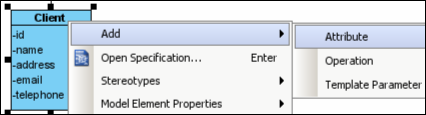
We will now concentrate on adding attributes of the classes.

Select class Client. Right click on the class Client, and then select New Attribute. Add the following as attributes for class Client:

Id

Name

Telephone address



Right click the class to access this menu

*ECJTU, NANCHANG* 11

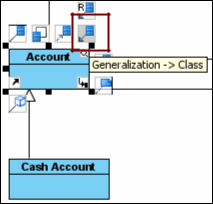
Object Oriented Analysis and Design

Lab#2

**Step 6: Structure Object Classes using Inheritance**

By adopting a top-down approach, we discover that the class Account has 2 subtypes, “Cash Account” and “Margin Account”. To structure them using inheritance, we need to create them as subclasses.

Select the class “Account”. On the resource centric interface, click and drag the icon for “Generalization -> Class” to create a new subclass. Name it as Cash Account. Repeat the step again by including subclass Margin Account.



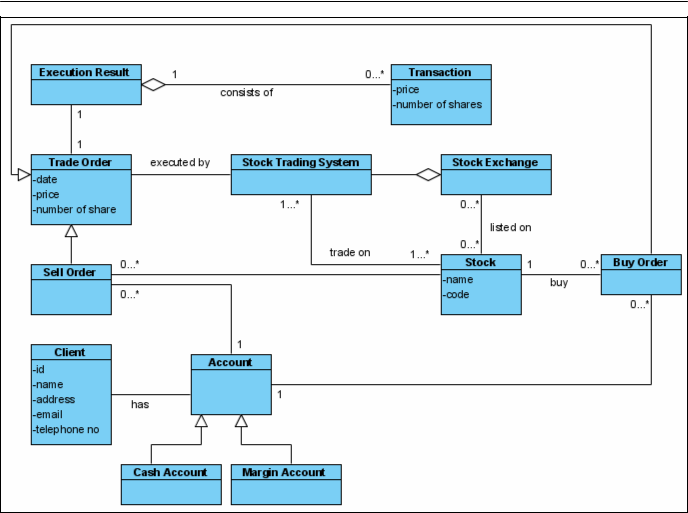
Generalization option under Resource centric option

The restructured domain class diagram is shown below:

*ECJTU, NANCHANG* 12

Object Oriented Analysis and Design

Lab#2



Restructured Domain Class Diagram of an Online Stock Trading System

**Step 7: Verifying Access Paths for Likely Queries**

At this stage, we usually need to enhance the domain class diagram by including support for typical queries of the application domain. Sometimes during the modeling phase, there will be other type of queries / between associated classes. Consider a query based on our example above:

How does a client find out the stock balance of his account?

The Domain Model cannot directly answer this as it only show transactions between classes (and remember we can’t put in operations under the classes at this stage). To do this:

We need to associate class Account with Stock.

Create an association between class Account and Stock as in Step 4.

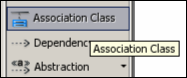
Then, we need to create an “Association Class” between these 2 classes.

Click the “Association Class” icon in the Diagram Palette, which will create a class. Name it as class “Stockline”.

*ECJTU, NANCHANG* 13

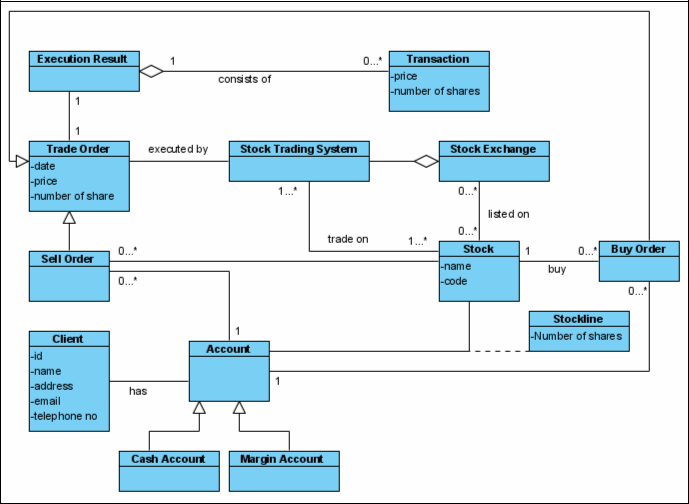
Object Oriented Analysis and Design

Lab#2



The Association Class option under Diagram Palette

Highlight the class “Stockline” and drag it to the association line between Account and Stock.



A revised Domain Model for Online Stock Trading System

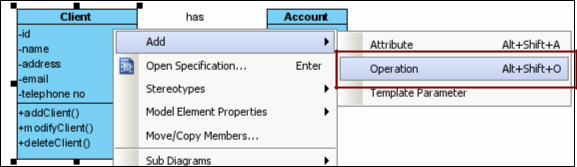
**Step 8: Add Operations in the Model**

*ECJTU, NANCHANG* 14

Object Oriented Analysis and Design

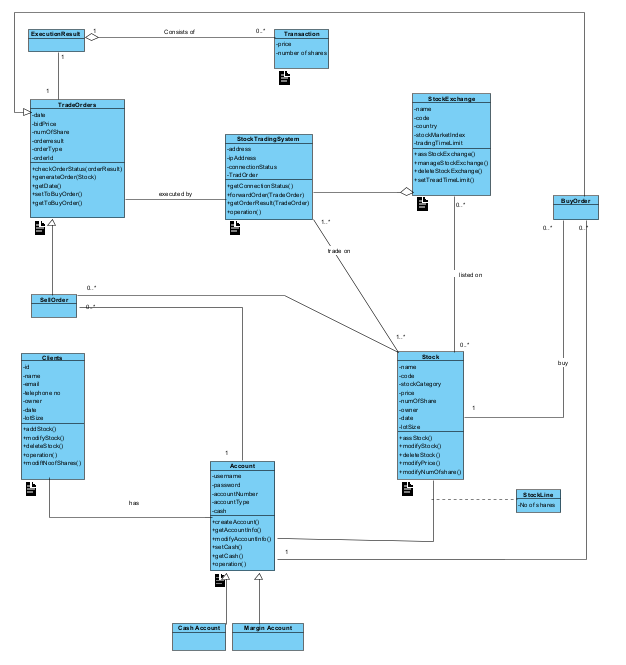
Lab#2

Add operations [ addClient(), modifyClient() and deleteClient() ] under class Client by “Right clicking” the class and choosing “ Add > Operation”.

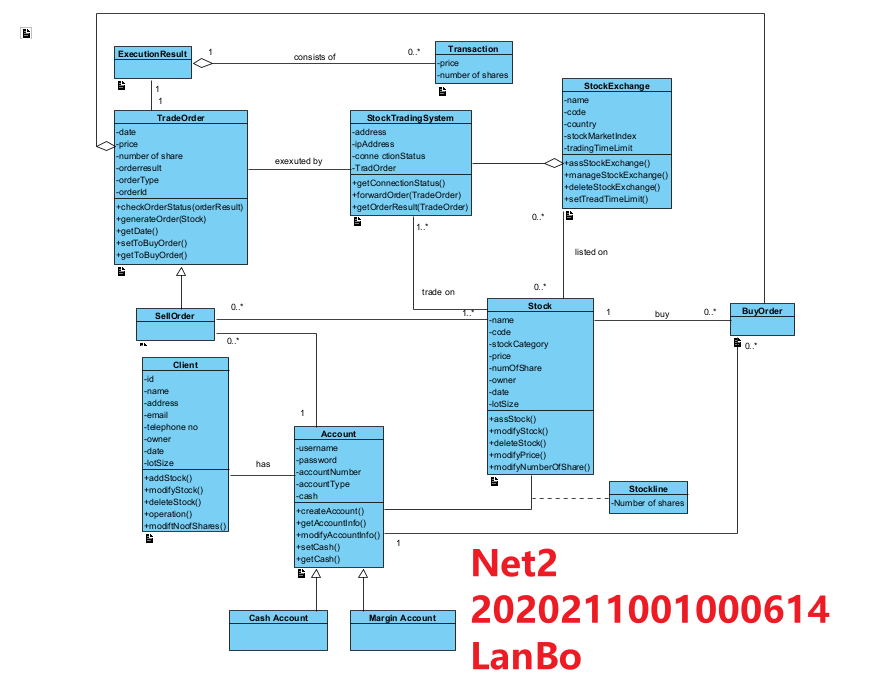


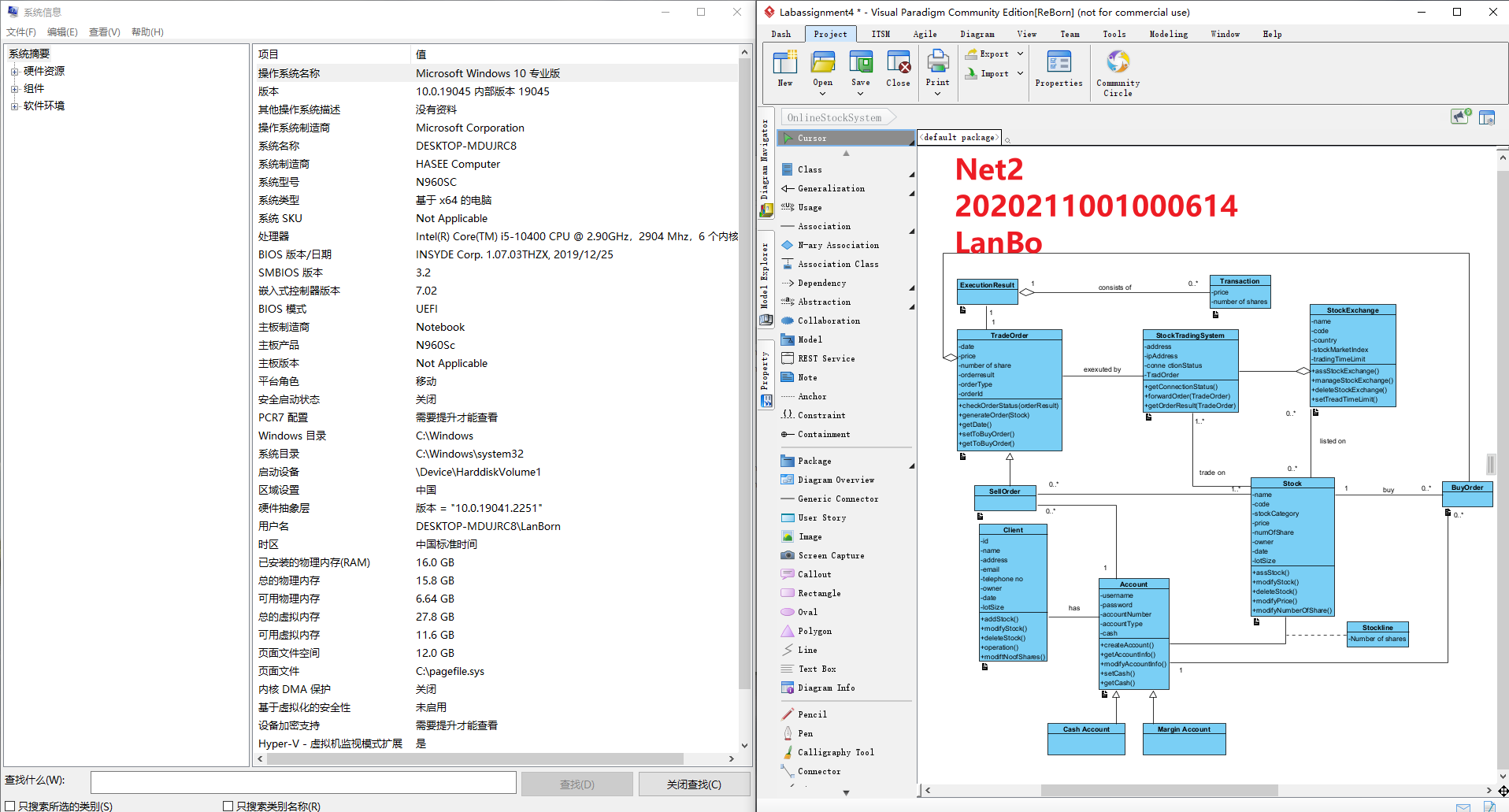
Adding operations under class

Now try to complete your Class Diagram by following the a well described Class Diagram shown below:



Result :





Lab Assignment 5

Title

**Interaction Diagram- UML Sequence Diagram**

Objectives

* + To understand the role of dynamic models in requirement analysis by reading and constructing UML Sequence Diagram.
  + To identify sequences of interactions between the objects of a system.
  + To identify the distinction between a Sequence Diagram and System SequenceDiagram.

Aim

* + 1.Describe the main objects.
  + 2.Describe the main flow of events / scenario.
  + 3.Translate them into system events for communication between the objects.
  + 5.Create the SSD.
  + 6.Create a new Sequence Diagram named “Make New Sale”.

Design:

To illustrate the working flow of SSD and Sequence Diagram, we will base our guidelines to the problem sample below by creating a System Sequence Diagram for Process Sale:

Consider a Sale Process Scenario for a cashier processing sale activities in a

Object Oriented Analysis and Design

Lab#3

supermarket. The cashier is required to process any purchasing (sale) activity by the customer. The customer will firstly put their goods on the cash counter. The cashier will then enter the item ID (could be the barcode) and its quantity, so that the system can give the description of the item and total price. The process is repeated until all items are scanned and recorded as sale. The process will end afterwards, and will require the customer to pay the amount indicated.

Step1:Describe the main objects (2 in this case):

Step2: Describe the main flow of events / scenario.

Step3:Translate them into system events for communication between the objects:

Step 4:Create the SSD.

The Use Case has been thoroughly analyzed and identified below:

1. Cashier starts a new sale.
2. Cashier enters item identifier (should be barcode of item).
3. System records sale item and presents item description, price and running total.
4. Cashier repeats steps 2 to 3 until all items processed.
5. System presents total.
6. Cashier informs the customer the total and asks for payment.
7. Customer pays and System handles payment.

The Use Case 2 column format is shown below:

|  |  |
| --- | --- |
| **Actor** | **System Response** |
| Cashier starts a new sale. |  |
| Cashier enters item identifier |  |
|  | System records sale item and presents |
|  | item description, price and running total. |
|  | System presents total. |
| Cashier process the payment done on |  |
| behalf of customer. |  |
|  | Customer pays and System handles |
|  | payment. |

Create a new Sequence Diagram and name the diagram as “Process Sale”.

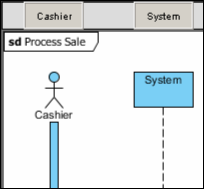
Create the actors “Customer” and “Cashier”. To do this, click on the Actor icon under the Diagram Pane.

Object Oriented Analysis and Design

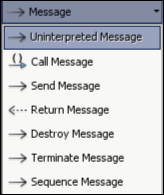
Lab#3

Subsequently, create an object called “System”. This can be done by clicking the “Lifeline” icon under the Diagram Pane.

You should get the following display:



To create messages between objects / actors, click on the “Message” icon under the Diagram Pane.



Create message “makeNewSale()” and “enterItem(itemID, quantity)” linking between the actor Cashier and object System.

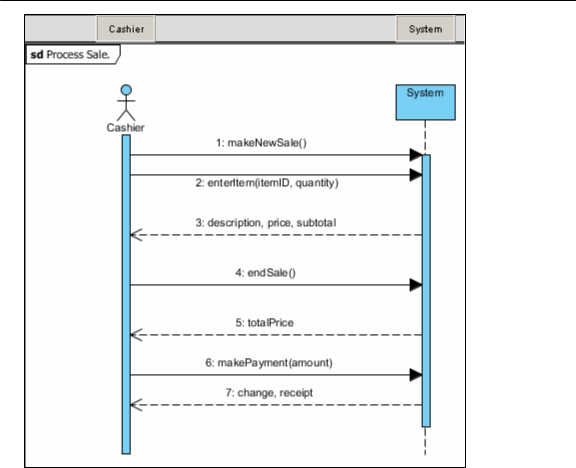
To create a return message, choose “Return Message” under the Message tree in the Diagram Pane.

Create a return message “description, price, subtotal” from System to Cashier.

**Repeat the above steps to come up with a complete SSD shown below:**

Object Oriented Analysis and Design

Lab#3

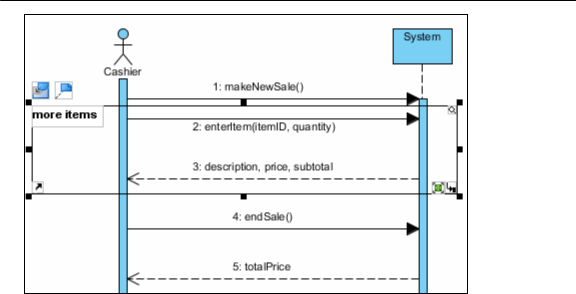


Consider that the cashier is required to process all sale items until completed. We need to model this with a **loop fragmen**t.

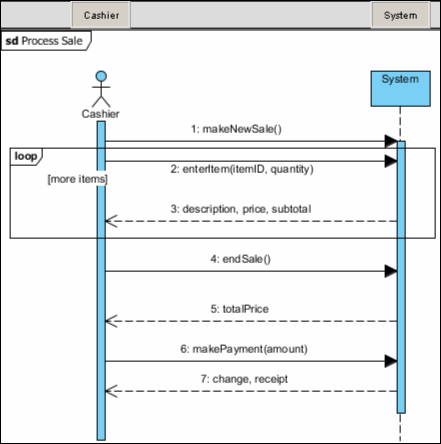
To create a loop scenario fragment, click on the “Loop combined Fragment” icon under the Diagram Pane. Left click and drag to cover required selected events as shown below.

Object Oriented Analysis and Design

Lab#3



A modified / updated SSD of Process Sale is shown below:



Updated System Sequence Diagram for Process Sale

Object Oriented Analysis and Design

Lab#3

**Now** we will model a more **detailed Sequence Diagram** for the process sale use case scenario.

Analyzing the situation concerned with processing sales items, we have come up with the following use case description:

**Step1:Identified Actors / Objects**: Cashier, Cash Register System, Sale, Product Item

**Step: Identified Main Flow:( from use case follow of event)**

1. Cashier initiates a new sales record.
2. Cashier checks in product.
3. System retrieves product description and price.
4. System add product to sales record.
5. System repeats check in and adding until all products is checked-in.
6. System calculates total price of sale and indicates to cashier.
7. Cashier process payment by customer.

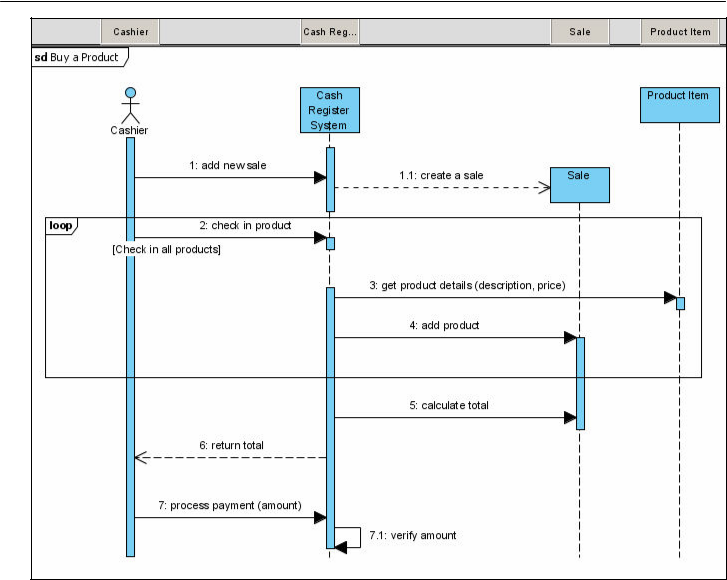
The Use Case Description 2 column format is shown below:

Create a new Sequence Diagram named “Make New Sale”. Fill it in with the identified objects and messages as shown below.

|  |  |
| --- | --- |
| **Actor** | **System Response** |
| Cashier initiates a **new sales** record. |  |
| Cashier **checks in product**. |  |
|  | System retrieves product **description and** |
|  | **price**. |
|  | System **add product** to sales record. |
|  | System r**epeats check i**n and adding until |
|  | all products is checked-in. |
|  | System calculates **total price** of sale and |
|  | indicates to cashier. |
| Cashier **process** **payment** by customer. |  |

Object Oriented Analysis and Design

Lab#3



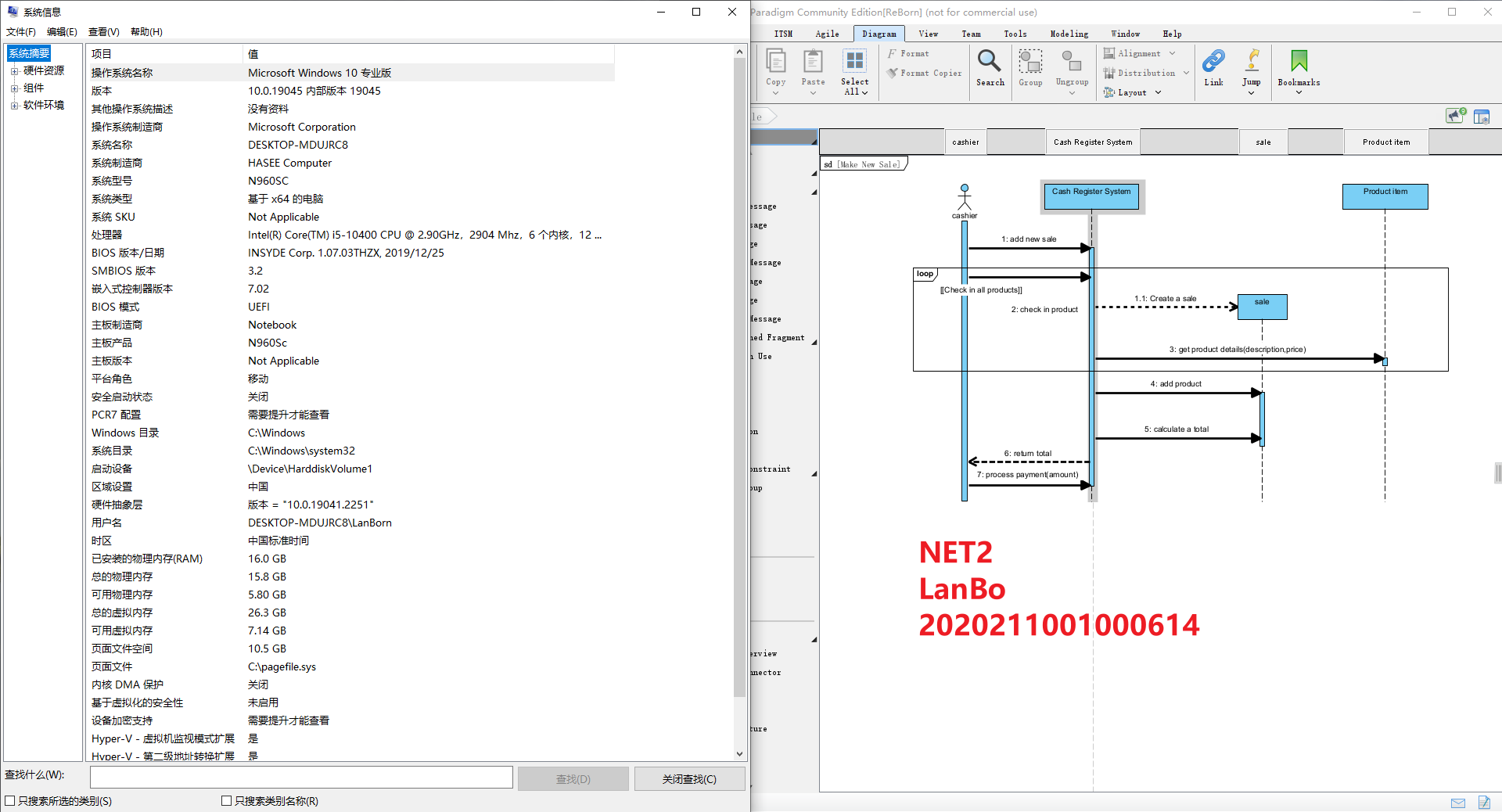
Sequence Diagram for Make New Sale scenario

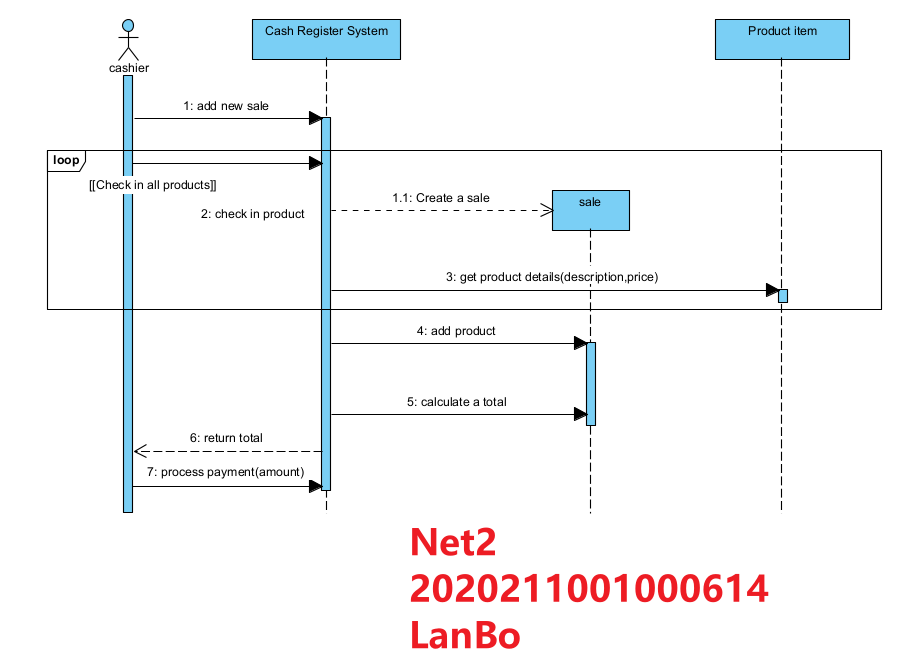
Consider that the cash register system must verify the amount being paid by the customer (which is processed by the cashier). This should require the system to invoke a self request for amount verification (done by self messaging).

To create a self message, click on the “Self Message” icon under the Diagram Pane. Left click under the object “Cash Register System” lifeline to create a new self message.



Result :





Lab Assignment 6

Title

**UML Sequence Diagram- Model the scenario of the ATM withdraw money use case**

Objectives

* + Model the scenario of the Withdraw Money use case in more detail.
  + Where the SSD showed only input and output generated by external actors and the system object.
  + thisSequence Diagram will elaborate further by including new internal objects under the withdraw money use case such as Card Controller (to control card management) and Bank (the issuing bank).

Aim

* General steps in Sequence Diagram
  + 1.Describe the main objects.
  + 2.Describe the main flow of events in this scenario.
  + 3.Translate the flows into corresponding system events.

4.Construct a Sequence Diagram based on the information

Design:

**INTRODUCTION**

**Problem Statement**

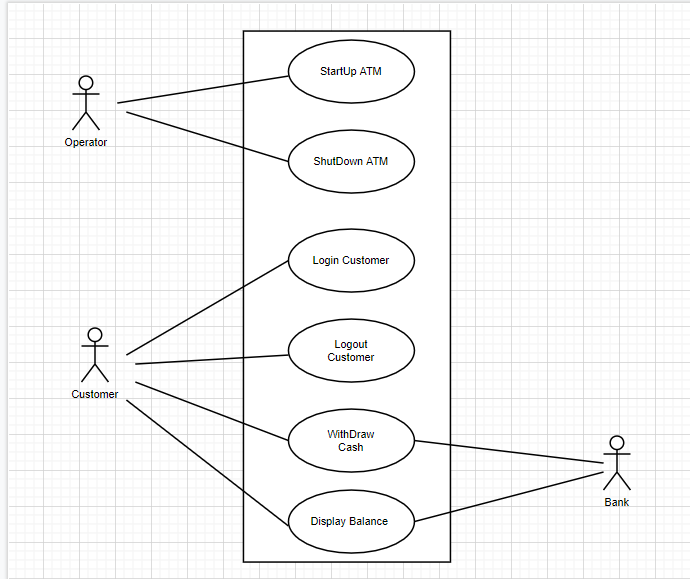
The software ATM version1.0 is to be developed for Automated Teller Machines (ATM). An automated teller machine (ATM) is computerized telecommunications device that provides a financial institution's customers a secure method of performing financial transactions, in a public space without the need for a human bank teller. Through ATM, customers interact with a user-friendly interface that enables them to access their bank accounts and perform various transactions.

The network enables customers to complete simple bank account services via automatedteller machines (ATMs) that may be located off premise and that need not be owned andoperated by the customer’s bank. The ATM identifies a customer by a cash card and password. It collects information about a simple account transaction (e.g.,deposit, withdrawal,transfer, bill payment),communicates the transaction information to the customer’s bank,and dispenses cash to the customer. The banks provide their own software for their own computers

**Enumerated Functional Requirements**

|  |  |
| --- | --- |
| IDENTIFIER | REQUIREMENT |
| REQ1 | A customer must be able to make a cash **withdrawal** from any suitable account linked to the card, in multiples of $20.00. Approval must be obtained from the bank before cash is dispensed. |

**Use Case Diagram**



**Fully-Dressed Description**

|  |  |
| --- | --- |
| **Use Case UC-1: Withdraw Cash** | |
| **Related Requirements:** | A customer who has logged in can withdraw up to $500 cash in $20 bills. |
| **Initiating Actor:** | The customer |
| **Actor’s Goal:** | Customer withdraws cash from ATM. |
| **Participating Actors:** | The bank |
| **Preconditions:** | The ATM has been started up.  See use case “Start up ATM”.  The customer has inserted a valid bank card.  The customer has entered a correct PIN. |
| **Post conditions:** | The customer receives the desired amount of cash.  The amount is deducted from the customer’s account.  The customer sees the new account balance.  ***OR:*** The customer receives no cash.  The customer’s account is unchanged. |
| Flow of Events for Main Success Scenario:  **Primary sequence:**   1. The system prompts the customer for the amount. 2. The customer chooses from a list of amounts or enters a amount. 3. The customer confirms and submits the amount. 4. (The ATM communicates with the bank to check the customer’s account.) 5. The system dispenses the amount in $20 bills. 6. (The bank deducts the amount from the customer’s balance.) 7. The system displays the customer’s balance See use case “Display balance”.   **Alternate sequences**:  3.1 The customer entered an amount that is not a multiple of $20.  3.1.1 The system displays a message to the customer .  3.1.2. The system prompts the customer for a new amount.  3.2 The customer’s bank balance is insufficient.  3.2.1 *eject* | |

**Task 1 : System sequence diagram (do your self)**

**Step 1: Describe the main objects**

Cardholder (or Customer)

ATM System

**Step 2: Describe the main flow of events in this scenario.**

The first 2 events are listed below:

1. Customer arrives at the ATM machine and inserts a bank card.
2. The system requests for user authentication (password).
3. .
4. ..
5. …
6. …
7. …
8. ..

**Step 3:**Translate the flows into corresponding system events.

|  |  |
| --- | --- |
| **Actor** | **System Response** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Step 4:**Construct a SSD based on the information above.

Object Oriented Analysis and Design

Lab#3

**Task 2: ATM Withdraw money sequence diagram( do your self)**

Now model the scenario of the Withdraw Money use case in more detail. Where the SSD showed only input and output generated by external actors and the system object, this Sequence Diagram will elaborate further by including new internal objects under the withdraw money use case such as Card Controller (to control card management) and Bank (the issuing bank).

**Step 1: Describe the main objects** :

User

ATM System Card Controller

**Step 2**: Describe the main flow of events in this scenario.

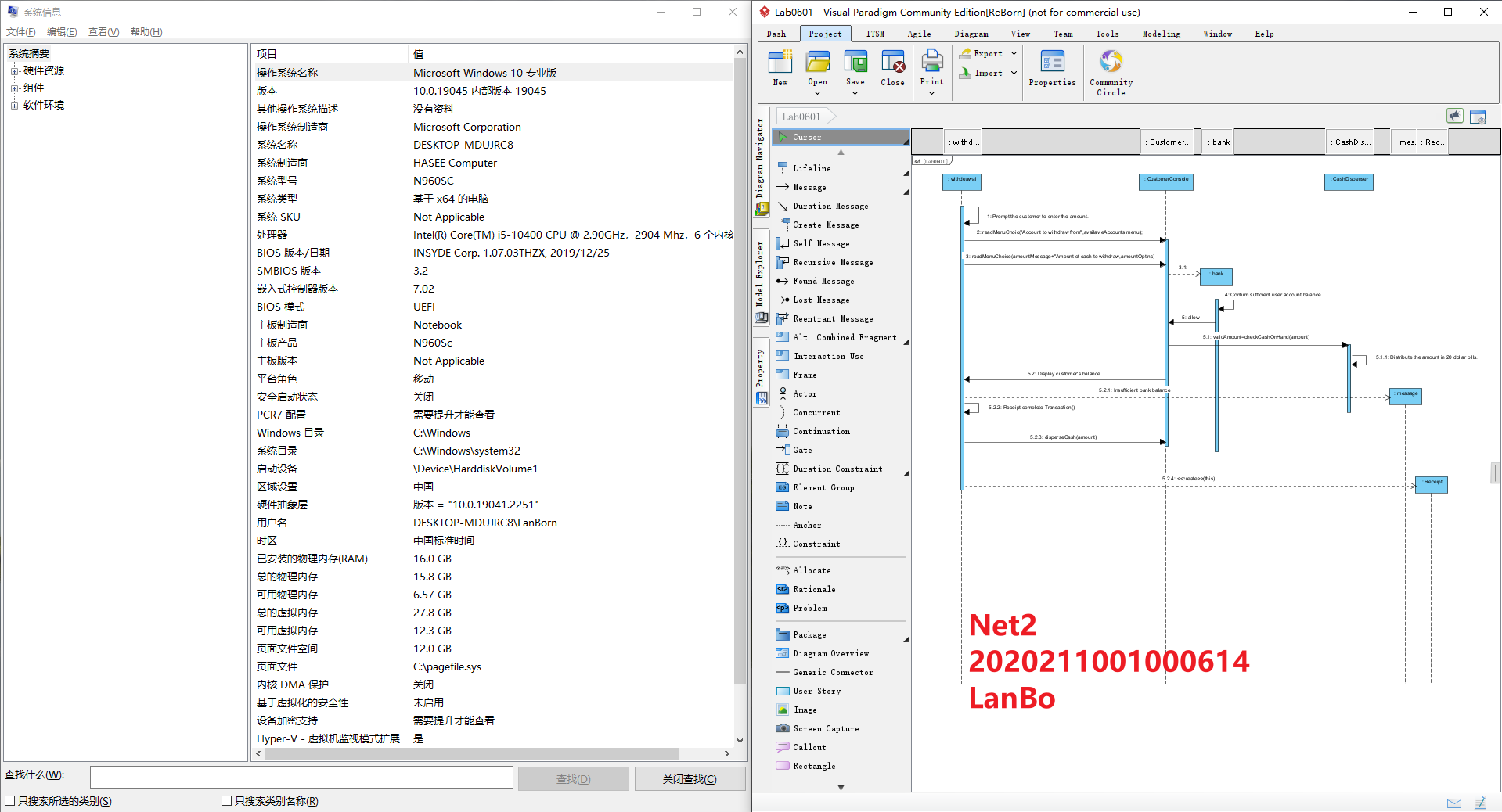
The first 2 events are listed below:

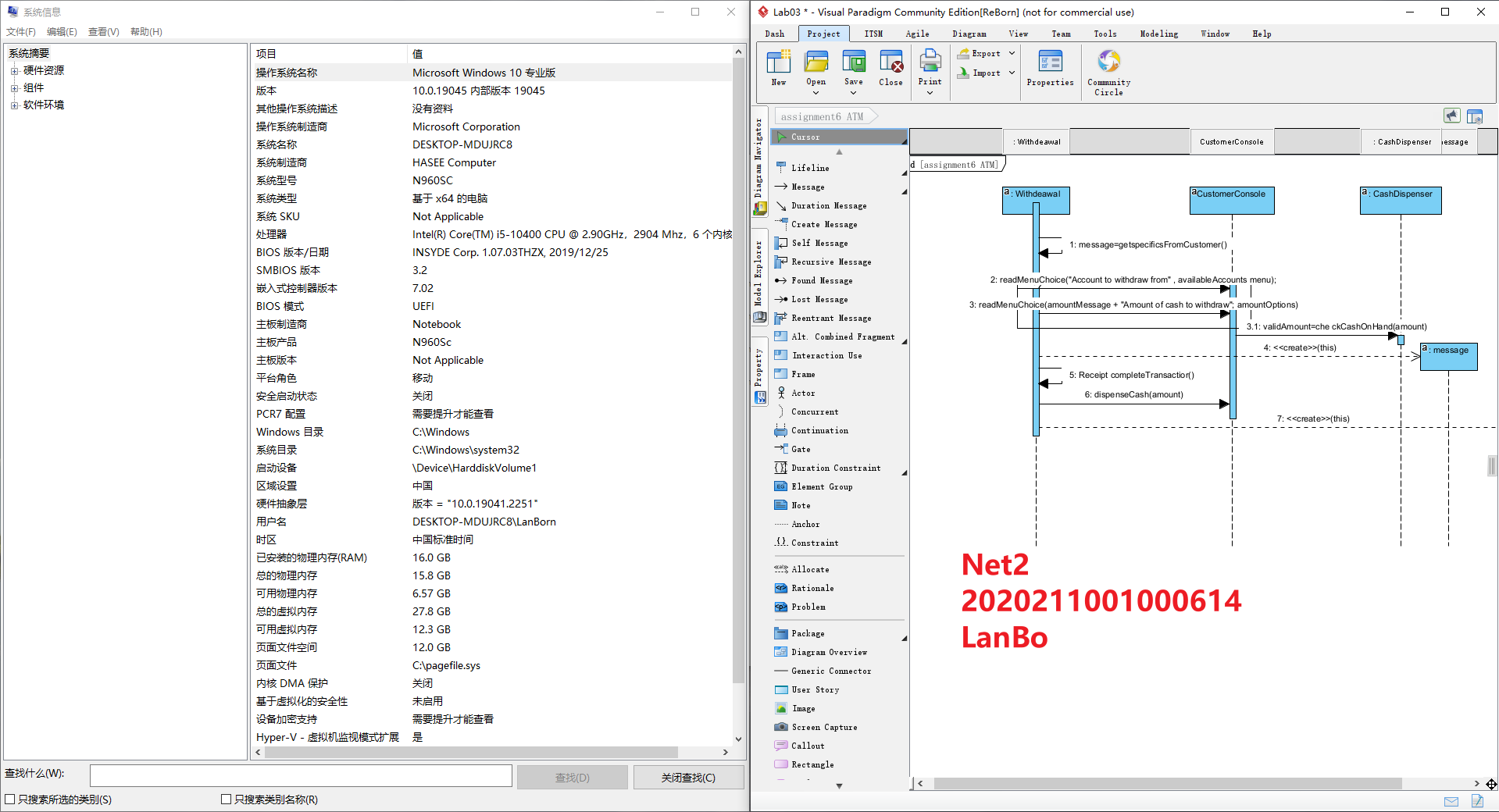
1. Customer arrives at the ATM machine and inserts a bank card.
2. The card is verified by the Card Controller.
3. .
4. .
5. .
6. .
7. .
8. .
9. .

**Step 3:**Translate the flows into corresponding system events (input and response)

**Step 4:**Construct a **Sequence Diagram** based on the information above.

Result :





Lab Assignment 7

Title

**Interaction Diagram - UML Communication Diagram**

Objectives

* + To understand the role of dynamic models in requirement analysis by reading and constructing UML Communication Diagram.

Aim

* + 1.Creating and Filling in Objects in Communication Diagram.
  + 2.Editing Sequence Number of Messages.
  + 3.Quick Conversion to Sequence Diagram.

Design:

**Step1:Creating and Filling in Objects in Communication Diagram**

To illustrate how to operate within Communication Diagram, refer to the case study of “**Make New Sale**” scenario in Lab 4 (page 7).

Begin by creating an actor called Cashier. Do this by clicking on the Actor icon in the Communication Diagram toolbar.

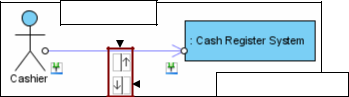
To create a linked message between an actor and object, 2 ways are possible: First Method:

Create a lifeline called “**Cash Register System**”.

Next, link the 2 object **“Cash Register System**” and actor “**Cashier**” by using the link icon under the Diagram toolbar.

Now create a message on the created link. Do this by clicking on the created link. 2 icons will appear. Choose one that represents the flow intended.



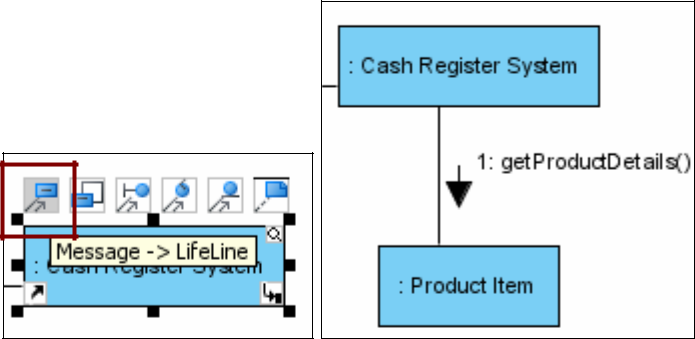


Create Message

Create Reverse Message

Second Method:

Using the Resource Centric option available as usual.

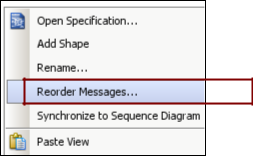


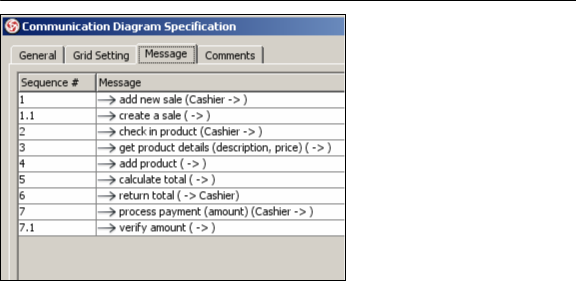
The message is automatically created, but can be **edited / added** for later revisions.

**Step 2: Editing Sequence Number of Messages**

The steps below show how to edit sequence number of messages:

**Right click** anywhere on the diagram (not any object or link), select “**Reorder Message**s” from the popup menu.



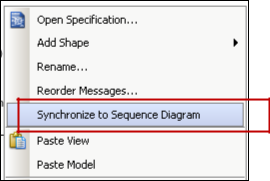


Reorder Message Dialog

**Step 3: Quick Conversion to Sequence Diagram**

In case the user requires modeling of a Sequence Diagram, VP-UML allows for instant conversion from and to (**Sequence Diagram and Communication Diagram**). Follow the steps below:

Right click anywhere on the diagram and choose “Synchronize to Sequence Diagram”.



A new Sequence Diagram is now created, although usually most of the items / elements need to be **rearranged correctly**.

Notice that some messages are unified under an arrow display when they have the same destination and origin. We can separate them into individual messaged display by:

Select the Arrow display with the unified messages. Right click and choose **“Decompose Messages**”.

Result :

