

Chapter 5

Game Design

In this chapter the prototype business is simulated to a game. The study uses object-oriented analysis and design (OOAD) to extract the problem domain. To make the design more understandable, Unified Modelling Language (UML) notations are used to visualise the structure, objects, relations and processes of the game.

5.1 Game Specifications

The game is a strategic-oriented business simulation game, which designed as a helping tool for an instructor in strategic management course. The players could be anyone who would like to practice strategic management skill. The game can host 5 users simultaneously per one session excludes a facilitator. It runs on any computer platform and operating system that support Java Runtime Environment(JRE).

The uses of the system are divided in to two groups, a facilitator and a player. Their roles are described as follows:

5.1.1 Facilitator

A facilitator is a main user who controls the number of system users, assigns game environments.

As illustrated in a use case diagram, a facilitator has the following tasks:

- create a player - in order to play the game, anyone who interested in playing the game have to as a facilitator to create a player account for them.

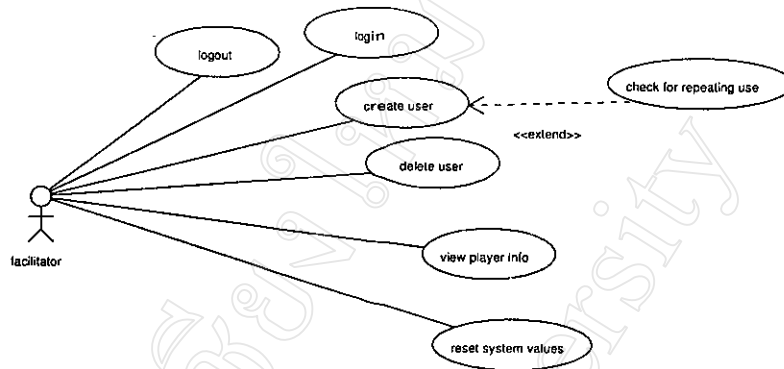


Figure 5.1: Facilitator Use Case

- delete a player - a facilitator is responsible for removing an obsolete player.
- view player information - a facilitator may view all players status.
- reset system values - a facilitator may reset the game values to default values when begin a new game session.

Note that: a facilitator need to login before performing any task.

5.1.2 Player

A player acts as an owner a business unit competing in a market. They must do various strategic management activities that make their business survive and make profit from the market.

As illustrated in a use case diagram, a player may do this following tasks:

- create a new company - a player must firstly create a company in order to run the business.
- design product - in this process, a player need to design their products that will be sold in the market. A player needs to set up all products properties and ingredients.
- produce product- after a design process, a player may submit a design into production.

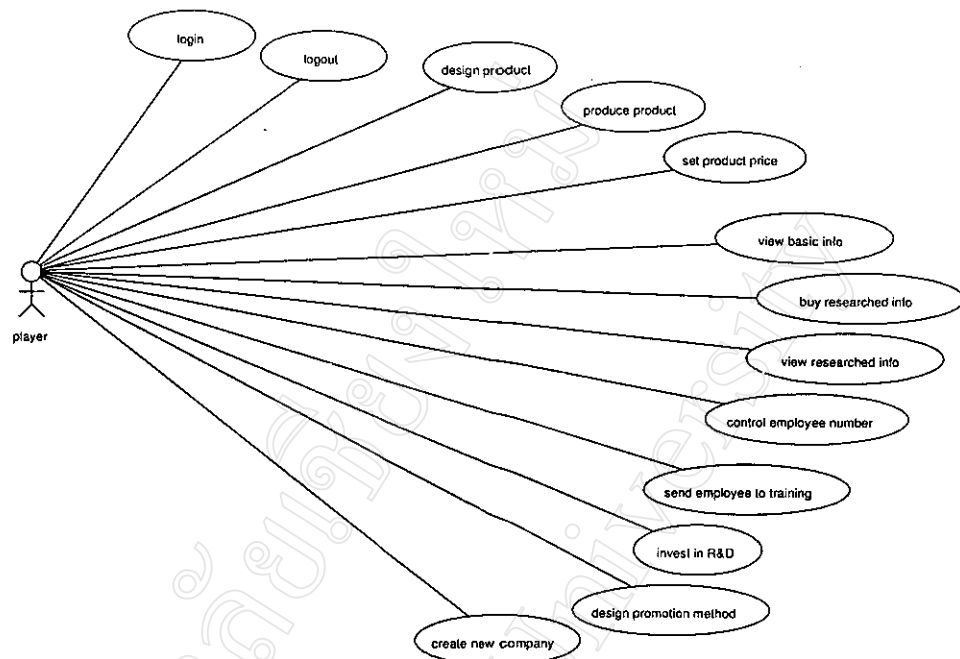


Figure 5.2: Player Use Case

- set product price - a player can freely setup their product price according to their cost and pricing strategy.
- view basic information - there is a set of basic reports that a player can view without any cost. A player gets an industrial information from this set of reports.
- buy researched information - apart from all basic information, a player may buy a specific report that suit their need. A research contains an information that may help a player to decide or plan their strategy.
- view researched information - a player is able to view all researched that they bought during a playing session.
- control employee number - a player need to control an employee number in their company. They may employ more employee or design to down size their organisation. There are two types of employee which will be described later.

- send employee to training - there are human development programmes which a player may send their employee into such training.
- invest in R&D - a player may decided to do a research and development programme which helps to improve a company performance.
- design promotion method - a player may setup promotion to boost their sales.

Note that: a player need to login before performing any task.

5.2 Class Design

After examining use case diagrams, facilitator roles and player roles have been defined. The diagrams also show the interaction between people and a system. The next step is to design all classes that comprise a system.

The study begins with the overall picture which shows the relationship among each class. Then the each class is being described its purpose, properties, and methods.

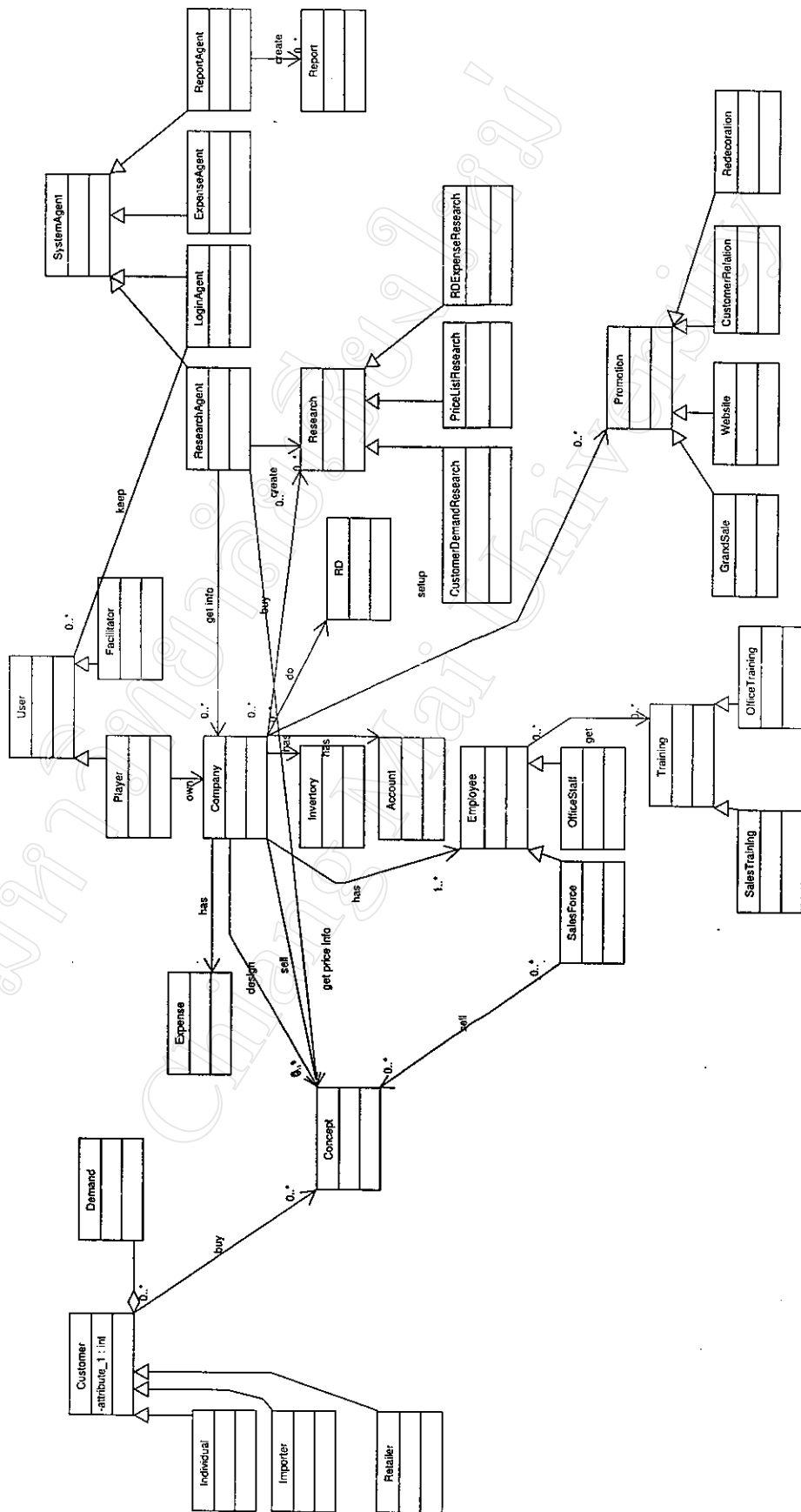


Figure 5.3: Overall Class Diagram

5.2.1 User Class

There are two types of user; a facilitator and a player. Each type of user has distinct roles as previously described by use case diagram. In this section the user class is closely examined.

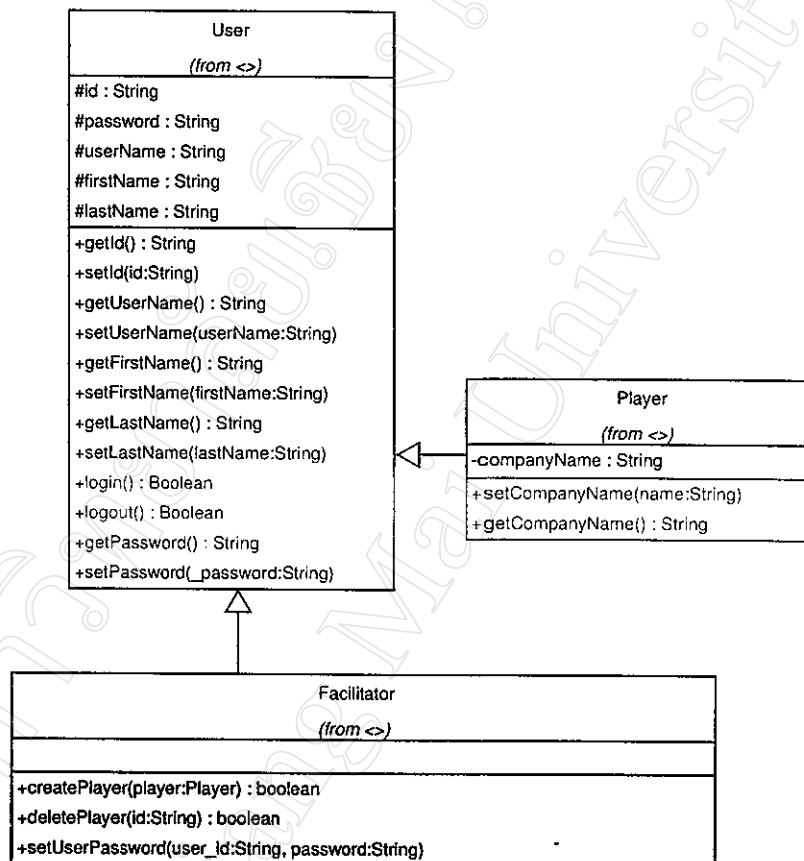


Figure 5.4: User Class Diagram

User Properties and Methods Summary

- id : ID holds user identification number. This number is unique for each user.
- userName : A user name assigned by a facilitator.
- password : User password that used to login to the system.
- firstName : User's first name.

- lastName : User's last name.
- getId : An accessor method that returns user identification number.
- setId : An accessor method that sets user identification number.
- getUsername : An accessor method that returns user name.
- setUsername : An accessor method that sets user name.
- getFirstName : An accessor method that returns user's first name.
- setFirstName : An accessor method that sets user's first name.
- getLastName : An accessor method that returns user's last name.
- setLastName : An accessor method that sets user's last name.
- getPassword : An accessor method that returns user's password.
- setPassword : An accessor method that sets user's password.
- login : A method that logs user into a system.
- logout : A method that logs user out off a system.

Facilitator Properties and Methods Summary

A facilitator class is derived from a user class. It has some methods that only facilitator can perform:

- createPlayer : A method that creates a new user.
- deletePlayer : A method that removes a user.
- setUserPassword : A method that set a user password.

Player Properties and Methods Summary

A player class is another class that also derived from a user class. Its properties and methods are:

- `companyName` : A player's company name.
- `setCompanyName` : An accessor method that sets a company name.
- `getCompanyName` : An accessor method that returns a company name.

5.2.2 Product Class

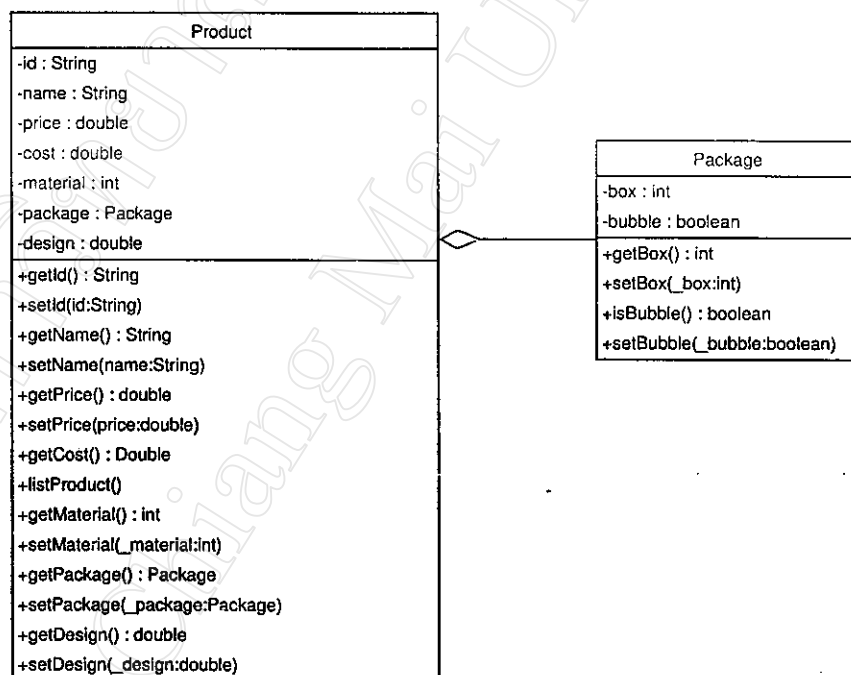


Figure 5.5: Product Class Diagram

Product Properties and Methods Summary

- `id` : A product identification number.
- `name` : A product name.

- price : A product price set by a player.
- cost : A product cost which automatically calculated from product specification.
- material : A material that a product is made from; Terra cotta or Stone ware.
- package : A product package properties.
- design : A design quality of a product. This variable is automatically adjusted based on *employee efficiency* and *R&D investment* levels.
- awareness : Customer brand awareness score. The higher score means there are higher chance that customer will buy the product.
- getId : An accessor method that returns product identification number.
- setId : An accessor method that sets product identification number.
- getName : An accessor method that returns product name.
- setName : An accessor method that sets product name.
- getPrice : An accessor method that returns product price.
- setPrice : An accessor method that sets product price.
- getCost : An accessor method that returns product cost.
- setCost : An accessor method that sets product cost.
- getMaterial : An accessor method that gets material that product is made from.
- setMaterial : An accessor method that sets product material.
- getPackage : An accessor method that calls *Package Class* to return package properties.
- setPackage : An accessor method that calls *Package Class* to set package properties.

- `getDesign` : An accessor method that returns product design score.
- `setDesign` : An accessor method that sets product design score.
- `getAwareness` : An accessor method that returns customer brand awareness.
- `setAwareness` : An accessor method that sets customer brand awareness.

Package Properties and Methods Summary

- `box` : A corrugated box ply values.
- `bubble` : A toggled value whether a product is wrapped by air bubble package or not.
- `getBox` : An accessor class that return the number of ply of a box.
- `setBox` : An accessor class that set the number of ply of a box.
- `isBubble` : A method that examine a product packaging whether it has an air bubble or not.
- `setBubble` : An accessor method that toggle a value of air bubble wrap.

5.2.3 Customer Class and Demand Class

A customer class represents customers in reality. The main objective of this class is to generate demands for the market.

A customer class is as follows:

Customer Properties and Methods Summary

- `type` : A classified customer type: Individual, Importer, or Retailer.
- `demand` : Demand for each type of customer generated by *Demand Class*.
- `level` : A customer level: lower class, middle class, or upper class.
- `floor` : The lowest product score that a customer can accept.
- `ceiling` : The highest product score that a customer can accept.

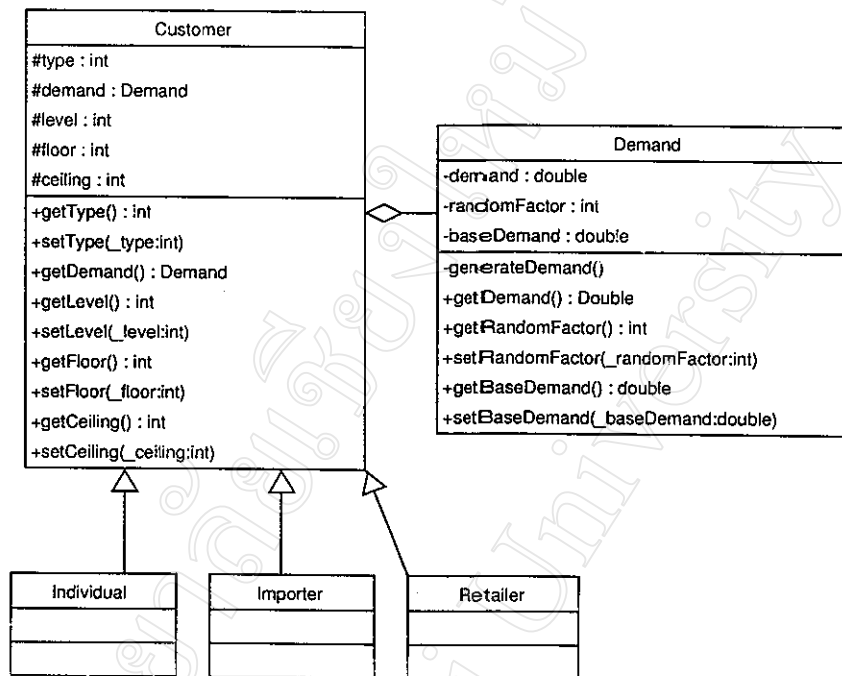


Figure 5.6: Customer Class Diagram

- `getType` : An accessor method that returns customer type.
- `setType` : An accessor method that set a customer type.
- `getDemand` : An accessor method that returns customer demand generated by *Demand Class*.
- `getLevel` : An accessor method that returns customer level.
- `setLevel` : An accessor method that sets a customer level.
- `getFloor` : An accessor method that returns *floor* score.
- `setFloor` : An accessor method that sets *floor* score.
- `getCeiling` : An accessor method that returns *ceiling* score.
- `setCeiling` : An accessor method that sets *ceiling* score.

Demand Properties and Methods Summary

- demand : A customer demand.
- randomFactor : A demand random factor.
- baseDemand : A customer base demand.
- generateDemand : A private method that generates customer demand.
- getDemand : An accessor method that returns customer demand.
- setDemand : An accessor method that sets customer demand.
- getRandomFactor : An accessor method that returns demand random factor.
- setRandomFactor : An accessor method that sets demand random factor.
- getBaseDemand : An accessor method that returns base demand.
- setBaseDemand : An accessor method that sets base demand.

5.2.4 Company Class, Inventory Class, Employee Class, Expense Class, and Account Class

The following diagram shows a company class and other related classes:

Company Properties and Methods Summary

- name : A company name which is set to be the same as a user name as a default.
- owner : A player object that owns the company.
- product : A product hash table that hold list of product objects with key index.
- expense : A company expense object.
- account : A company account object.
- inventory : A company inventory object.

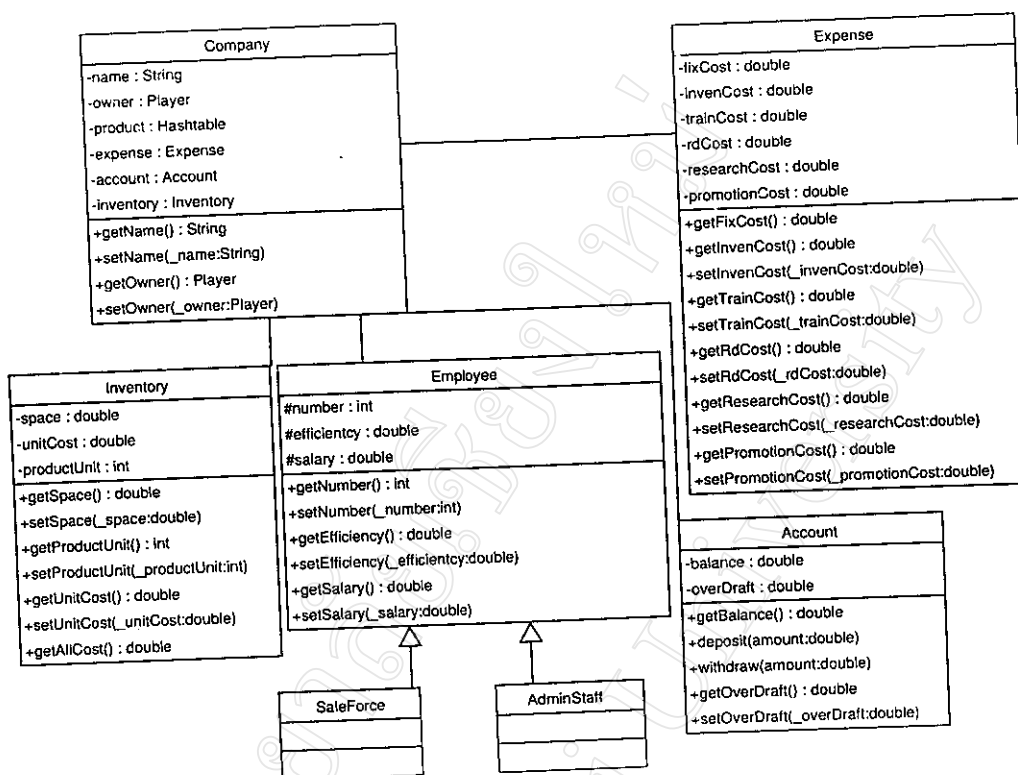


Figure 5.7: Company & Related Classes Diagram

- `getName` : An accessor method that returns a company's name.
- `setName` : An accessor method that set a company's name.
- `getOwner` : An accessor method that returns an owner of a company object.
- `setOwner` : An accessor method that set an owner of a company.

Inventory Properties and Methods Summary

- `space` : Warehouse space in unit. The default value when company starts is 1,500 units.
- `unitCost` : Cost per month for each product unit stored in a warehouse.
- `productUnit` : It shows how many product units are in a warehouse.
- `getSpace` : An accessor method that returns a warehouse capacity.
- `setSpace` : An accessor method that sets a warehouse capacity.

- **getProductUnit** : An accessor method that returns product quantity stored in a warehouse.
- **setProductUnit** : An accessor method that set product quantity stored in a warehouse.
- **getUnitCost** : An accessor method that returns an inventory cost for each unit.
- **setUnitCost** : An accessor method that set an inventory cost for each unit.
- **getAllCost** : A method that returns cost of inventory : $\text{UNIT_COST} \times \text{PRODUCT_UNIT}$.

Employee Properties and Methods Summary

- **number** : A total employee number.
- **efficiency** : A number that indicates an efficiency level of employee.
- **salary** : Salary amount for an employee.
- **getNumber** : An accessor method that returns an employee number.
- **setNumber** : An accessor method that sets an employee number.
- **getEfficiency** : An accessor method that returns efficiency factor.
- **setEfficiency** : An accessor method that set efficiency factor.
- **getSalary** : An accessor method that returns salary of an employee.
- **setSalary** : An accessor method that set salary of an employee.

Expense Properties and Methods Summary

- **fixCost** : Fix cost values. Fix cost is comprise of: salary, depreciation, facility expense (electricity, running water, etc), inventory management. This value is fix at 500,000 Baht per month for every company.

- **invenCost** : Inventory cost. Each set of product set inventory cost is 10 Baht/month. The cost may vary depends on the number of product sets stored in a warehouse.
- **trainCost** : Total training cost for each company.
- **rdCost** : Total R&D cost.
- **researchCost** : Total research expense.
- **promotionCost** : Total promotion expense.
- **getFixCost** : An accessor method that returns fix cost value.
- **getInvenCost** : An accessor method that returns inventory cost value.
- **setInvenCost** : An accessor method that sets inventory cost.
- **getTrainCost** : An accessor method that returns training cost value.
- **setTrainCost** : An accessor method that sets training cost.
- **getRdCost** : An accessor method that returns R&D cost.
- **setRdCost** : An accessor method that sets R&D cost.
- **getResearchCost** : An accessor method that returns report expense.
- **setResearchCost** : An accessor method that sets report expense.
- **getPromotionCost** : An accessor method that returns promotion expense.
- **setPromotionCost** : An accessor method that sets promotion expense.

Account Properties and Methods Summary

- **balance** : Total amount of cash on hand. However, the highest amount that a company has is: `BALANCE + OVER_DRAFT`.
- **overDraft** : Total amount of over draft allowed.
- **getBalance** : An accessor method that returns balance amount.

- deposit : A method that deposit the amount of money into an account.
- withdraw : A method that withdraw the amount of many out of an account.
- getOverDraft : An accessor method that returns an over draft amount.
- setOverDraft : An accessor method that set an over draft amount.

5.2.5 Promotion Class

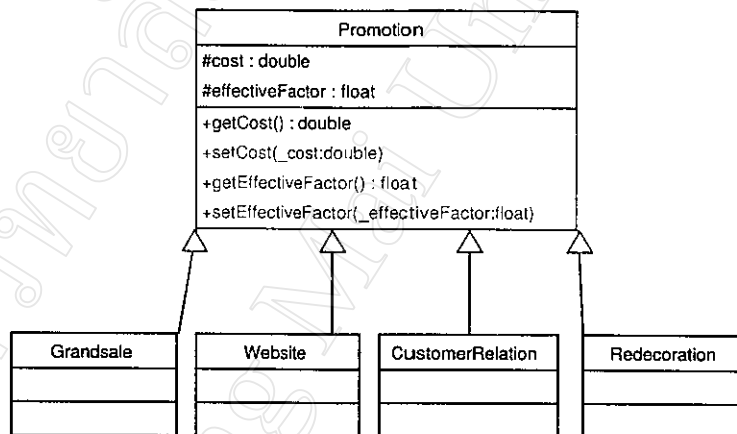


Figure 5.8: Promotion Class Diagram

Promotion Properties and Methods Summary

- cost : Cost of each promotion type.
- effectiveFactor : An effective factor value which affects product score.
- getCost : An accessor method that returns cost of each promotion.
- setCost : An accessor method that sets cost of each promotion.
- getEffectiveFactor : An accessor method that returns an effective factor value.
- setEffectiveFactor : An accessor method that set an effective factor value.

5.2.6 Training Class

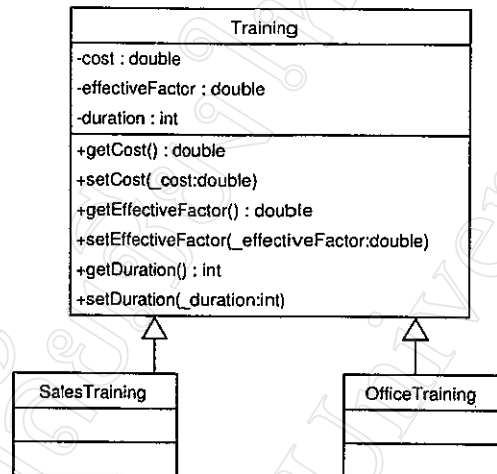


Figure 5.9: Training Class Diagram

Training Properties and Methods Summary

- `cost` : Cost of each training type.
- `effectiveFactor` : An effective factor value of each training.
- `duration` : Training duration (in month).
- `getCost` : An accessor method that returns a training cost.
- `setCost` : An accessor method that sets a training cost.
- `getEffectiveFactor` : An accessor method that returns an effective factor value.
- `setEffectiveFactor` : An accessor method that set an effective factor value.
- `getDuration` : An accessor method that returns a training duration.
- `setDuration` : An accessor method that sets a training duration.

5.2.7 R&D Class

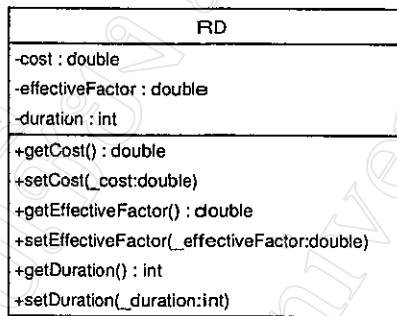


Figure 5.10: R&D Class Diagram

R&D Properties and Methods Summary

- **cost** : Cost of each training type.
- **effectiveFactor** : An effective factor value of each training.
- **duration** : Training duration (in month).
- **getCost** : An accessor method that returns a training cost.
- **setCost** : An accessor method that sets a training cost.
- **getEffiveFactor** : An accessor method that returns an effective factor value.
- **setEffectiveFactor** : An accessor method that set an effective factor value.
- **getDuration** : An accessor method that returns a training duration.
- **setDuration** : An accessor method that sets a training duration.

5.2.8 Report Class

At the end of each month, a company is getting a summary report which contains the following information:

- Average product quality.
- Average price.
- Inventory capacity.
- Total expense.
- Cash on hand.

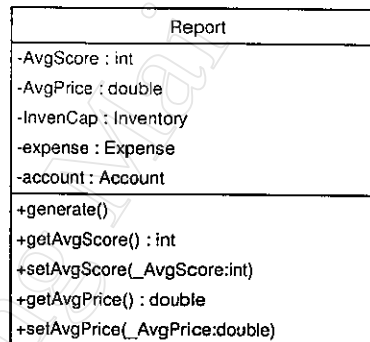


Figure 5.11: Report Class Diagram

Report Properties and Methods Summary

- avgScore : An average product score from every company.
- avgPrice : Product average price.
- invenCap : An inventory capacity of a company.
- expense : An expense object that reports company expense.
- account : An account object that reports a company financial status.

- generate : A method that generate a whole report.
- getAvgScore : An accessor method that returns an average product score.
- setAvgScore : An accessor method that sets an average product score.
- getAvgPrice : An accessor method that returns a product average price.
- setAvgPrice : An accessor method that sets a product average price.

5.2.9 System Agent Class

A System Agent Class is a class that acts a work horse for overall information collector. The functionality of a system agent is implemented in only one method, *doAgentAction*. There are 5 types of system agents as follows:

Research Agent: A research agent is responsible for collecting data needed to report in a research ie., each product price, customer demand from the “customer matrix” and an average industry R&D expense.

Report Agent: A report agent is responsible for summing up each company data and directly reports the information back to a company.

Expense Agent: An expense agent is responsible for collecting expense data from each company and submits the results to report agent.

Login Agent: An login agent keeps track of active users in a system. It also has to register a user into a system list.

Timer Agent: A timer agent acts as a system clock. It generates a signal at the end of each interval.

System Agent Properties and Methods Summary

- name : A system agent name.
- setName : An accessor method that sets a system agent name.
- getName : An accessor method that returns a system agent name.

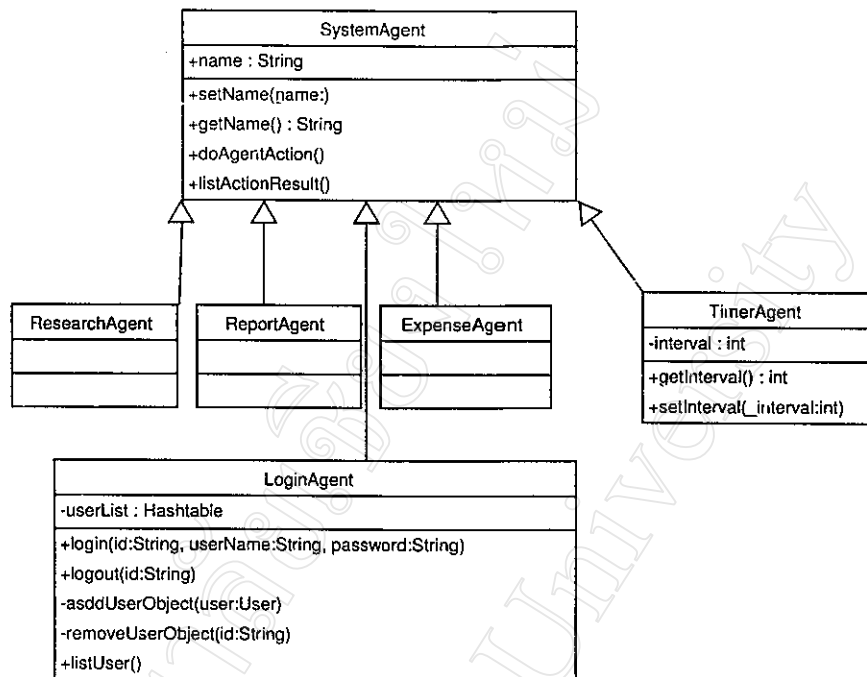


Figure 5.12: System Agent Class Diagram

- doAgent : A method that implements a agent functionality.
- listActionResult : A method that returns a result from a system agent.

LoginAgent Properties and Methods Summary

- userList : A hash table that holds user objects.
- login : A method that logins a user into a system.
- logout : A method that log outs a user off a system.
- addUserObject : A private method that add user into a user list.
- removeUserObject : A private method that remove user out off a user list.
- listUser : A method that list all user in a user list.

TimerAgent Properties and Methods Summary

- interval : Timer interval in second.

- `getInterval` : An accessor method that returns an interval value.
- `setInterval` : An accessor method that sets an interval value.

5.2.10 Research Class

A research system agent provide few types of researches which a company can buy. Therefore, a company can use such information to make decision or create a suitable strategy to compete in an industry. A research class design is as follows:

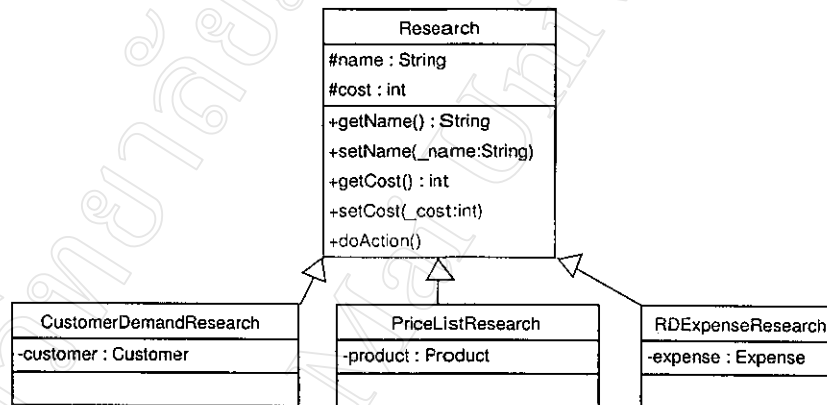


Figure 5.13: Research Class Diagram

Research Properties and Methods Summary

- `name` : A research name
- `cost` : A research cost
- `getName` : An accessor method that returns a research name.
- `setName` : An accessor method that sets a research name.
- `getCost` : An accessor method that returns a research cost.
- `setCost` : An accessor method that sets a research cost.

- doAction : A method that get information from customer object, product object, and expense object.

มหาวิทยาลัยเชียงใหม่
Chiang Mai University

5.3 Persistent Data Design

All classes that motioned previously do all game business logics. There is another type class that does an object data persistence. In other words, this kind of class is responsible for writing and retrieving information from a database back-end to keep object states. The study uses database adapter classes to handle data persistence for the game. The overall class view is as follows:

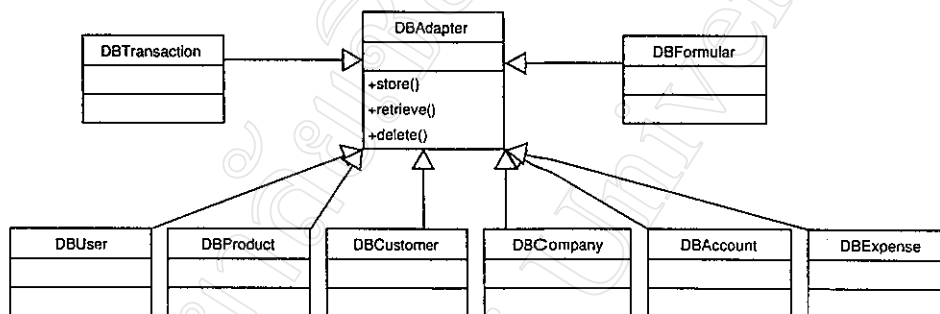


Figure 5.14: Database Logic Classes

All sub classes implement only three methods:

- store - A method that writes data into database back-end.
- retrieve - A method that retrieve data from database back-end.
- delete - A method that deletes data from database back-end.

The following data tables are used to store persistence data:

User Table

A user table stores system users information

User Table			
id	Char(10)	Primary Key	Index
firstName	VarChar(100)		First name
lastName	VarChar(100)		Last name
password	Char(20)	Password	User password
type	VarChar(15)		User type

Table 5.1: User Table

Company Table

A company table stores company information and the owner(user) of a company.

Company Table			
name	Char(20)		Company name
owner	Char(10)	Primary Key	User ID

Table 5.2: Company Table

Employee

An employee table store employee numbers and their efficiency of a company.

Employee Table			
id	Char(10)	Primary Key	User ID
saleNumber	Integer		Number of salesperson
staffNumber	Integer		Number of staff
saleEfficiency	Float		Sale efficiency factor
staffEfficiency	Float		Staff efficiency factor

Table 5.3: Company Table

Session Table

A session table keeps track of a system users to prevent a double login.

Session Table			
counter	Integer	Auto Increment, Primary Key	Index
id	Char(10)		User ID
inTime	Date		Login time
outTime	Date		Logout time

Table 5.4: Session Table

Product Table

A product table stores a product information including its quality factors and package quality.

Product Table			
counter	Int	Auto Increment, Primary Key	Index
id	Char(10)		Product ID
ownerId	Char(10)		User ID
name	VarChar(50)		Product name
price	Float		Product price
cost	Float		Product cost
material	Integer		Material used
packBox	Integer		Packaging
bubble	Boolean		Air bubble (yes/no)
design	Float		Product design score
amount	Integer		Product amount

Table 5.5: Product Table

Account Table

An account table mainly store a company cash amount.

Account Table			
id	Char(10)	Primary Key	Index
cash	Float		Cash on hand
overDraft	Float		Overdraft

Table 5.6: Account Table

Customer Demand Table

A customer demand table store demands for each type of customer generated in each month of a specific year.

Customer Demand Table		
type	Integer	Customer type
class	Integer	Customer class
demand	Float	Customer demand
month	Integer	Month
year	Integer	Year

Table 5.7: Customer Demand Table

Sale Transaction Table

A sale transaction table keeps records of sales that happen in each month in a specific year.

Sale_Transaction Table			
counter	Integer	Auto Increment, Primary Key	Index
productId	Char(10)		Product ID
price	Float		Sale price
cost	Float		Product cost
amount	Integer		Sale amount
month	Integer		Month
year	Integer		Year

Table 5.8: Sale_Transaction Table

Formula Table

A formula table stores all factors and system default values. This table acts as a value pair values storage.

Formula Table		
capital	Float	Initial cash on hand
overDraft	Float	Initial overdraft amount
saleEmployee	Integer	Number of salesperson
staffEmployee	Integer	Number of staff
inventory	Float	Initial warehouse capacity
random	Float	Demand random factor
interval	Integer	Timer interval
totalDemand	Float	Customer total demand
fixCost	Flat	A company fix cost
rdCost	Float	R&D Cost
rDemand	Float	Customer demand research cost
rPrice	Float	Price list research cost
rRD	Float	Average R&D expense research cost
pGS	Float	Grand sale promotion cost
pWS	Float	Web site cost
pCR	Float	Customer relation cost
pRD	Float	Store re-decorate cost
saleTraining	Float	Salesperson training cost (per head)
staffTraining	Float	Staff training cost (per head)

Table 5.9: Formular Table

5.4 Game System Architecture

The study uses a client-server architecture to design the game. The necessary components needed to deploy the games are:

Client Computer: A client computer runs a user interface part. A player uses this interface to interact with other parts of the game.

JBoss Application Server: JBoss is a J2EE compliance application server. All enterprise Java beans are deployed onto this server. This includes all business logic classes and database adapter classes.

PostgreSQL Database Back-End: PostgreSQL is a free database server that used to store all persistent data.

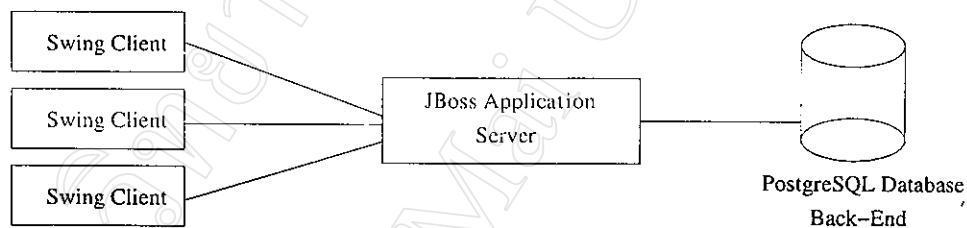
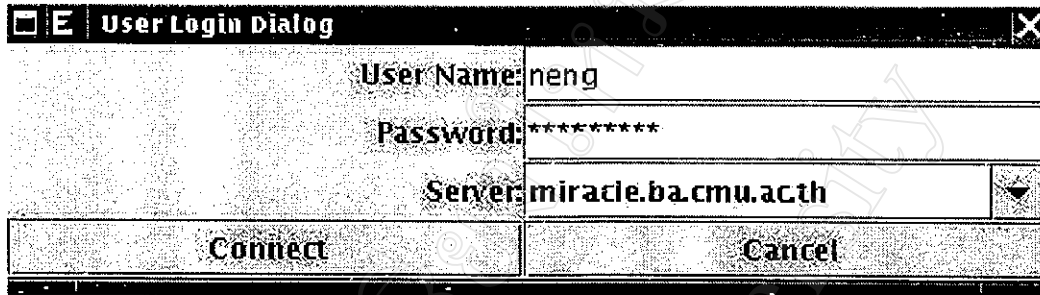


Figure 5.15: Game System Architecture

5.5 Game User Interface Examples

Before a player can play the game, they need to login to the system.



User Login Dialog	
User Name:	neng
Password:	*****
Server:	miracle.ba.cmu.ac.th
Connect	Cancel

Figure 5.16: User Login Dialog

Main application frame that lets use select an operation from menu system.

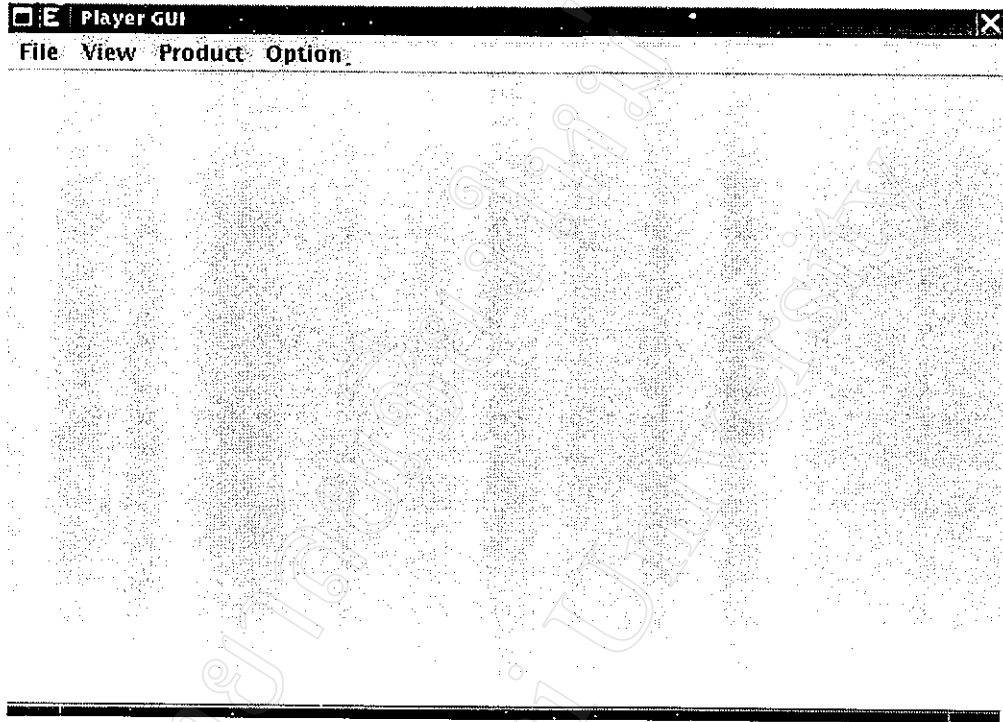


Figure 5.17: Main Application Space

A player may choose to view a report displays a company status.

Company Data	
Company Name:	Test Company
Owner:	Test Player
Employee Data	
Employee Number:	40
Sale Force Efficiency:	1.5
Office Staff Efficiency:	1.0
	Person(s)
	Point(s)
	Point(s)
Inventory Data	
Maximum Capacity:	1500
Stored Product:	0
	Unit(s)
	Unit(s)
Financial Data	
Cash On Hand(Baht):	5,000,000

Close

Figure 5.18: Company Information Report Dialog

A player can create their product, set up product price, and select a packaging type.

Product Creator Dialog	
Product Name:	Test Product
Material:	Terra Cotta
Set Price (Baht):	5000.00
Package	
Cortered Box Ply:	2
Bubble Wrapper	<input checked="" type="radio"/> Yes <input type="radio"/> No
Create Cancel	

Figure 5.19: Product Creator Dialog

There are 4 types of promotion schemes that a player may launch to increase their sale volume.

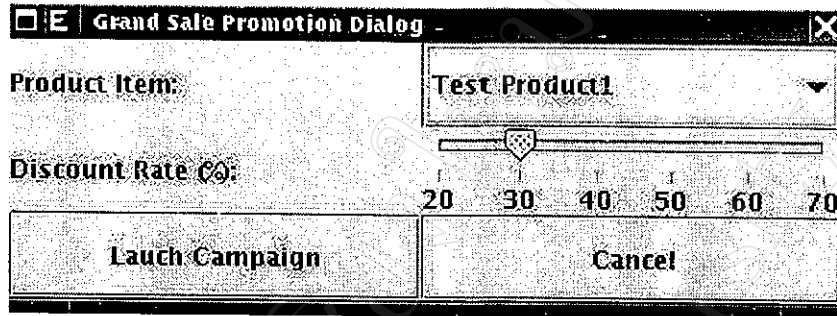


Figure 5.20: Grand Sale Promotion Dialog

5.6 Game Playing Procedures

Game playing procedures are as the following figure:

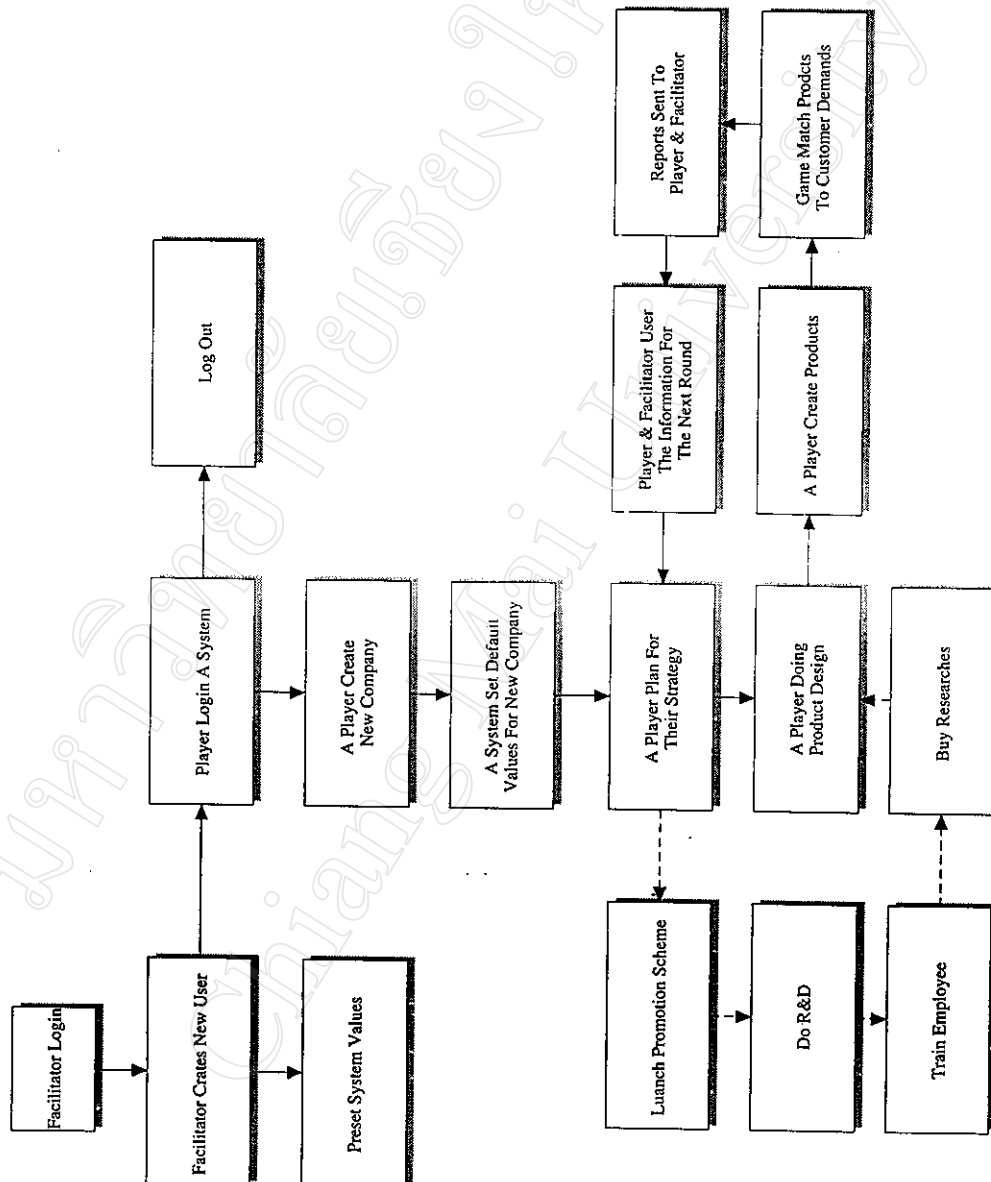


Figure 5.21: Playing Procedures

1. A facilitator login a system and create new players.
2. A player must login first in order to play the game by specifying user name, password, and game server.
3. After login a user must create a new company in order to start a new business.
4. All company default values are set when a company is newly started.
5. A player may plan what market segment should they be in.
6. A player must design a product to serve customer demands.
7. After a product design, a player must create product and specifies its price, package and volume.
8. At the end of each month(control by game clock), each product will be matched to customer demands.
9. A player gets a sale report from a system.
10. A play may analyse their status from a company basic information and sale report in order to provide a plan for next month.
11. A facilitator also have company performance report for every company. They may use the information to instruct/advise a player before a next month begins.
12. During each month, a player may choose to launch a promotion to increase a sale volume or customer brand awareness.
13. A player may send employee into training in order to increase employee efficiency.
14. A player may choose to do a research and development programme that helps to improve product quality and design.
15. A player can buy numbers of studied researches. These research helps a player getting to know customer demand for each segment, industrial standard, and competitors' prices.

16. After finish a session, user should log out the systems.

This chapter simulated the prototype business into a game design using object-oriented system analysis approach. All business logic and persistent data logic classes were visualised using UML notation. Client-server architecture was used to host the game “Enterprise Java Beans” to promote a network capable concept. At the end of the chapter, there were few game screens illustrated what a player would see during a game session. The game playing procedures also explained.