# OOAD & UML

วิชาวิเคราะห์และออกแบบระบบเชิงวัตถุ

Object-Oriented Analysis and Deisgn

## บทที่ 4

## แนะนำ UML เบื้องต้น (Introduction to UML)


ผู้ช่วยศาสตราจารย์ .ดร. นัฐพงศ์ ส่งเนียม
http://www.siam2dev.com
siam2dev@hotmail.com

สาขาวิชา วิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์และเทคโนโลยี
มหาวิทยาลัยราชภัฏพระนคร


UNIFIED MODELING LANGUAGE

Last Update : 18/08/2567

# Agenda

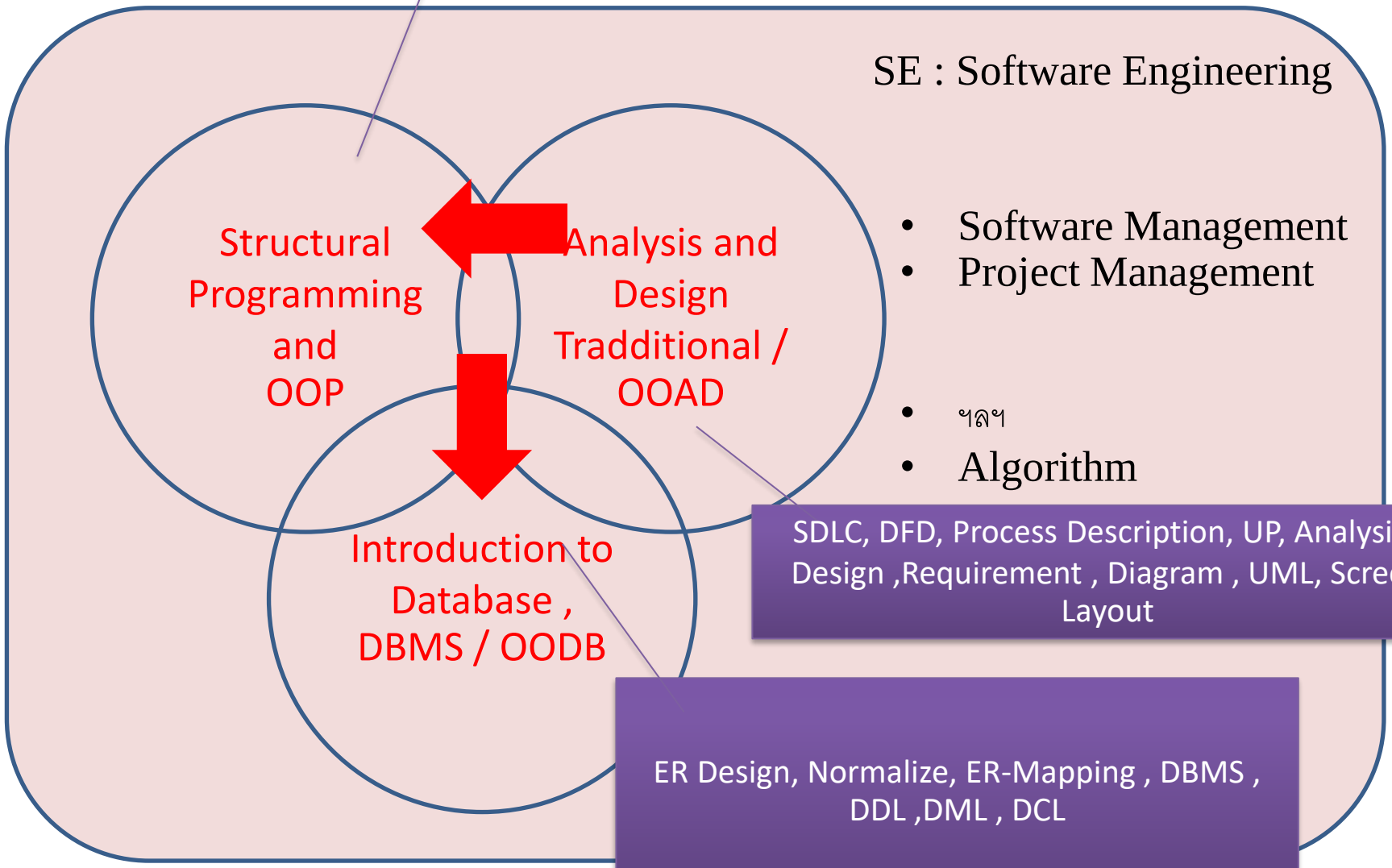
- **ทบทวน (Reviews)**
- **Software Modeling**
- **Require and Domain Analysis Model**
- **Design Model**
- **Brief Overview of Unified Modeling Language (UML)**
- **What is UML**
- **History of UML**
- **Type of UML Model**
- **Benefit of UML**

# งานโปรเจ็กต์ ส่งได้ตั้งแต่

- กำหนดส่ง**ก่อน** สอบปลายภาค 1 สป.
- **ครั้งที่ 2. งานกลุ่มครั้งหน้า  ส่ง Requirement Specifications (10 คะแนน) + VDO**

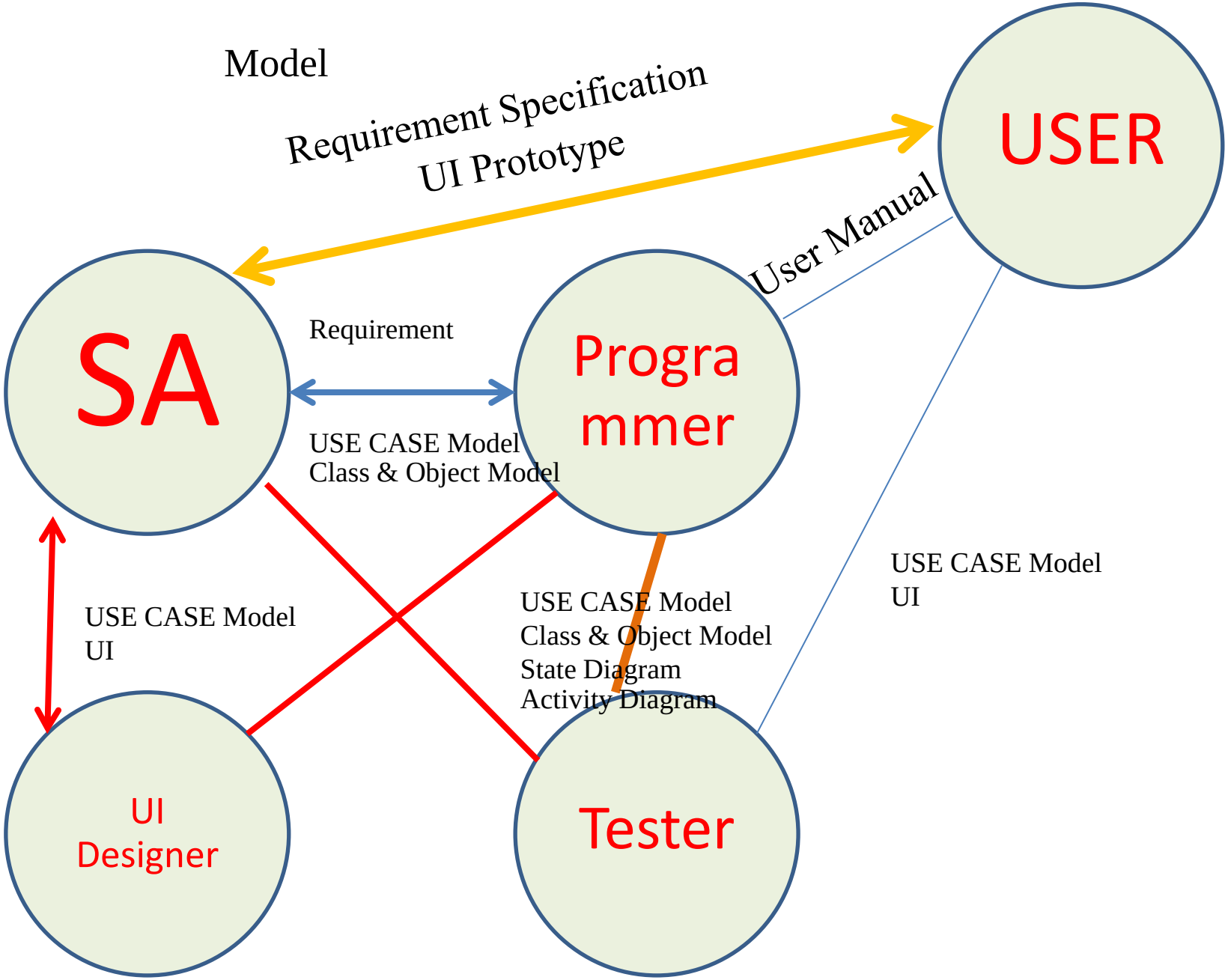
- **ครั้งที่ 3 ส่งครบทุกอย่าง**

ALGORITHM , PSEUDOCODE , SYNTAX, FLOWCHART , SCREEN LAYOUT , Visual Programming ,Event-Driven Programming , Object Oriented Programming




SDLC, DFD, Process Description, UP, Analysis , Design ,Requirement , Diagram , UML, Screen Layout

ER Design, Normalize, ER-Mapping , DBMS , DDL ,DML , DCL



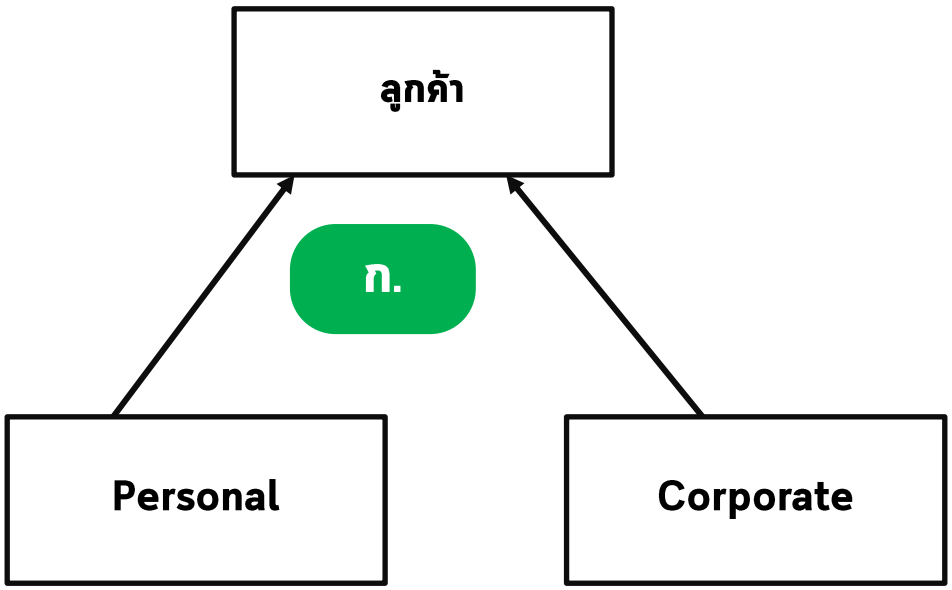

*** ทุกคนต้องเข้าใจตรงกัน ภาษา หรือ สัญลักษณ์ที่ใช้ จะต้องตรงกัน

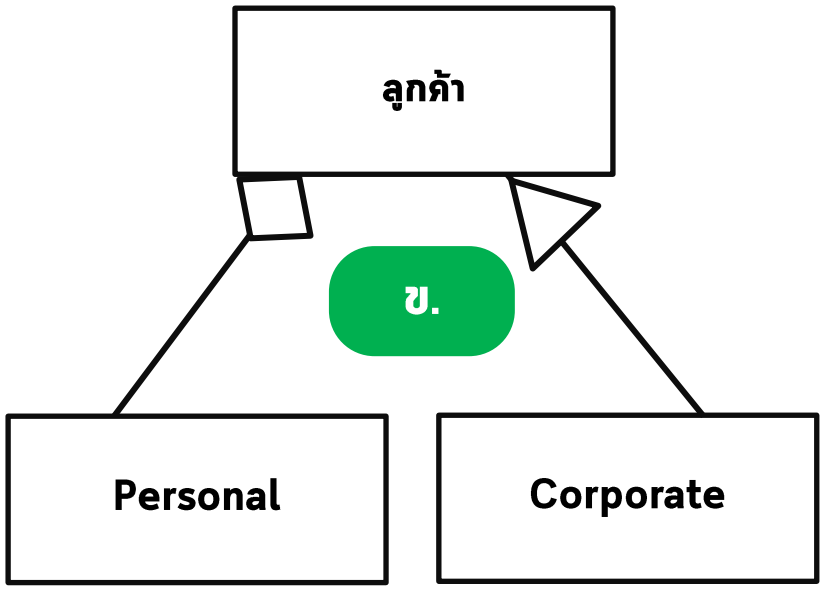
Reviews

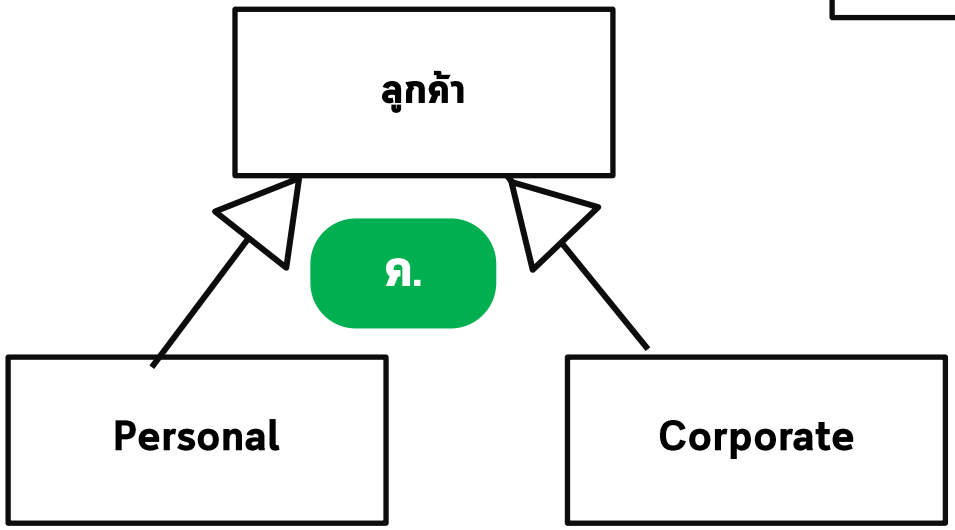
# SQL

- **DDL: Data Definition Language**
  - **CREATE DATABASE ….**
  - **DROP TABLE**
- **DML : Data Manipulation Language**
  - **SELECT , INSERT , UPDATE ,DELETE**
- **DCL : Data Control Language**
  - **GRANT**

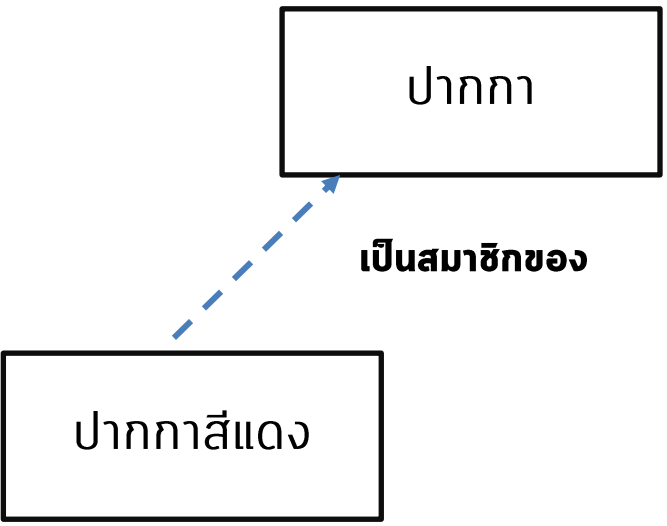
# SA : class & object specified



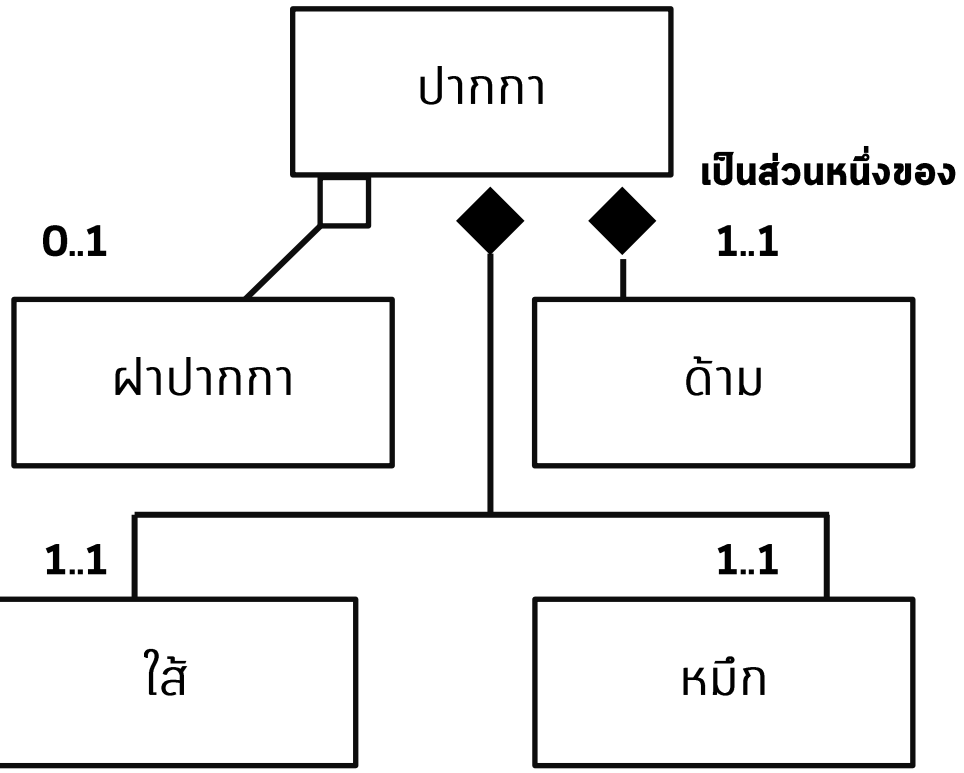



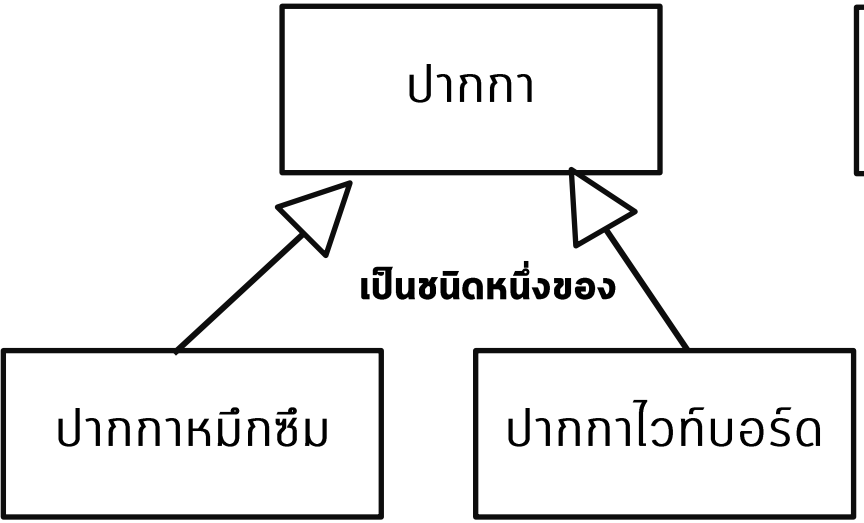
## Classification abstraction



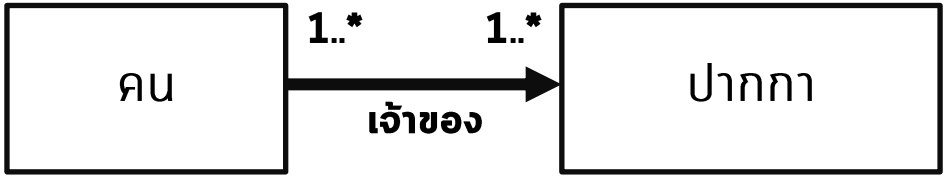

## Aggregation abstraction



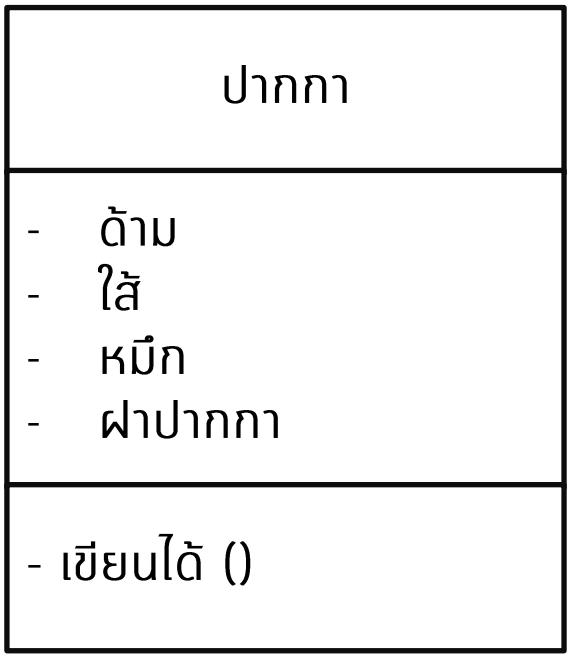

## Generalization abstraction




## Association abstraction



## Reviews




Real world

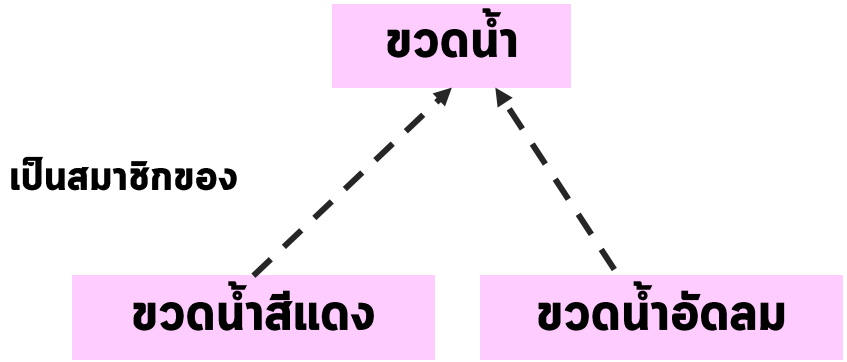

Computer world

Reviews

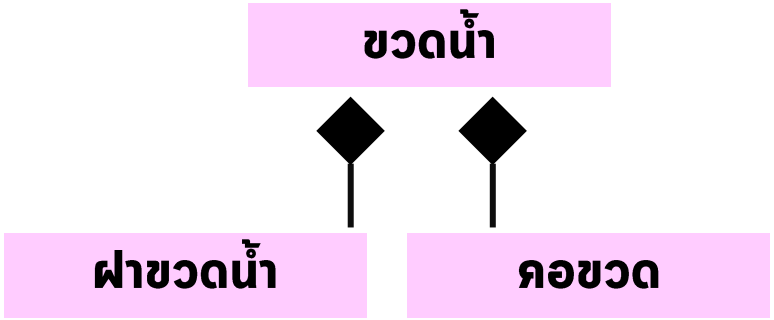
# จงเขียนสิ่งต่อไปนี้ในรูปแบบ abstraction ทั้ง 4 แบบ

- ขวดน้ำ
- โทรศัพท์
- เมาส์
- รถยนต์
- แมว
- ทีวี
- ร้านอาหาร
- ร้านกาแฟ
- โรงพยาบาล
- มหาวิทยาลัย

## Classification abstraction

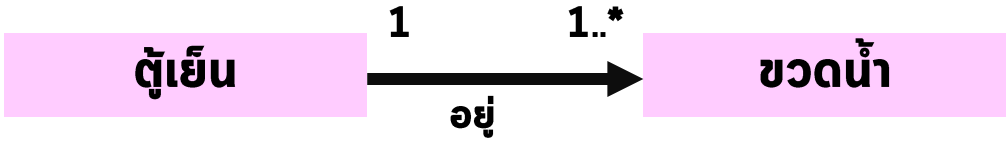


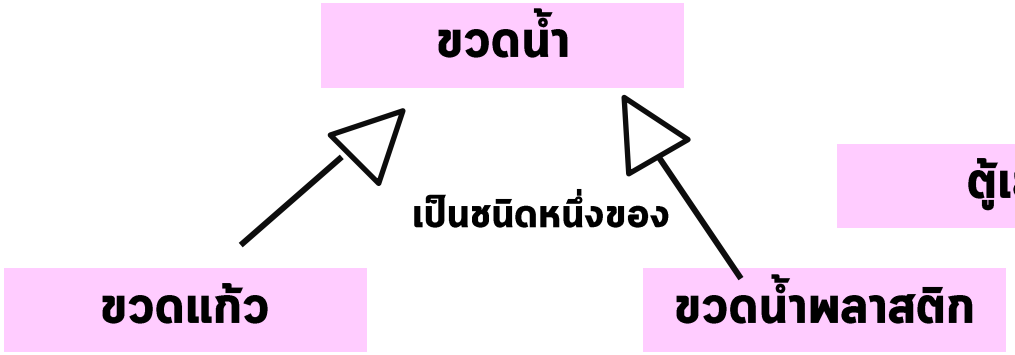

## Aggregation abstraction





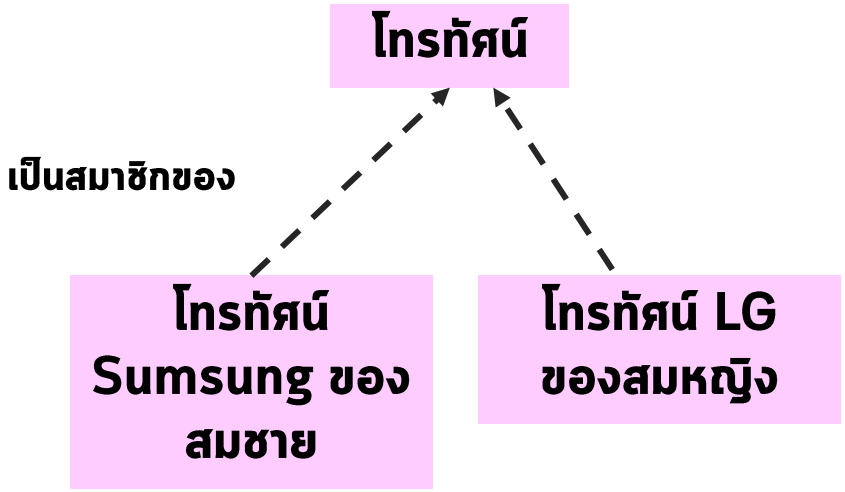
## Association abstraction



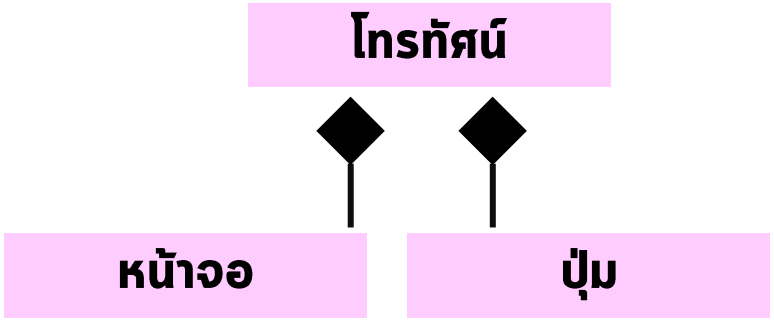



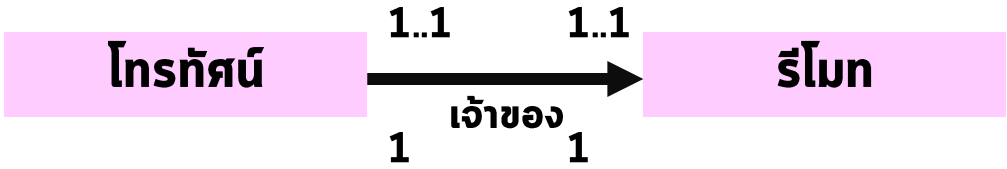

## Generalization abstraction

## Classification abstraction

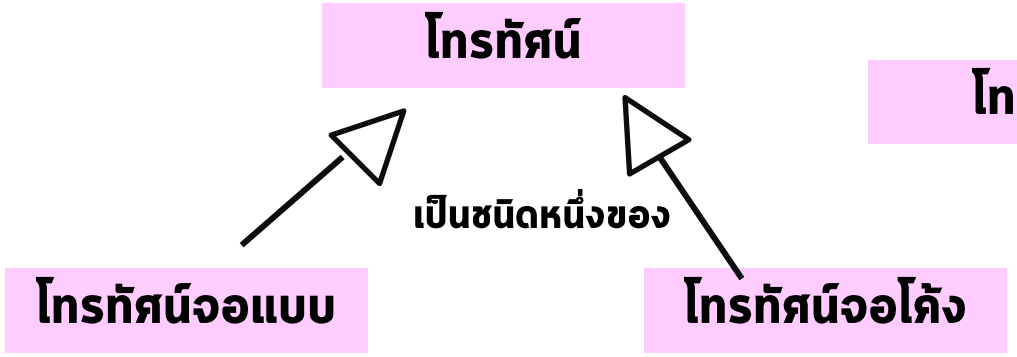



## Aggregation abstraction



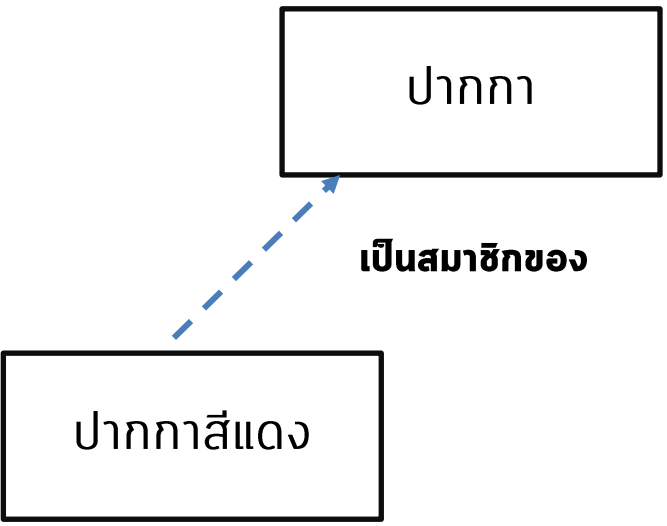




## Association abstraction

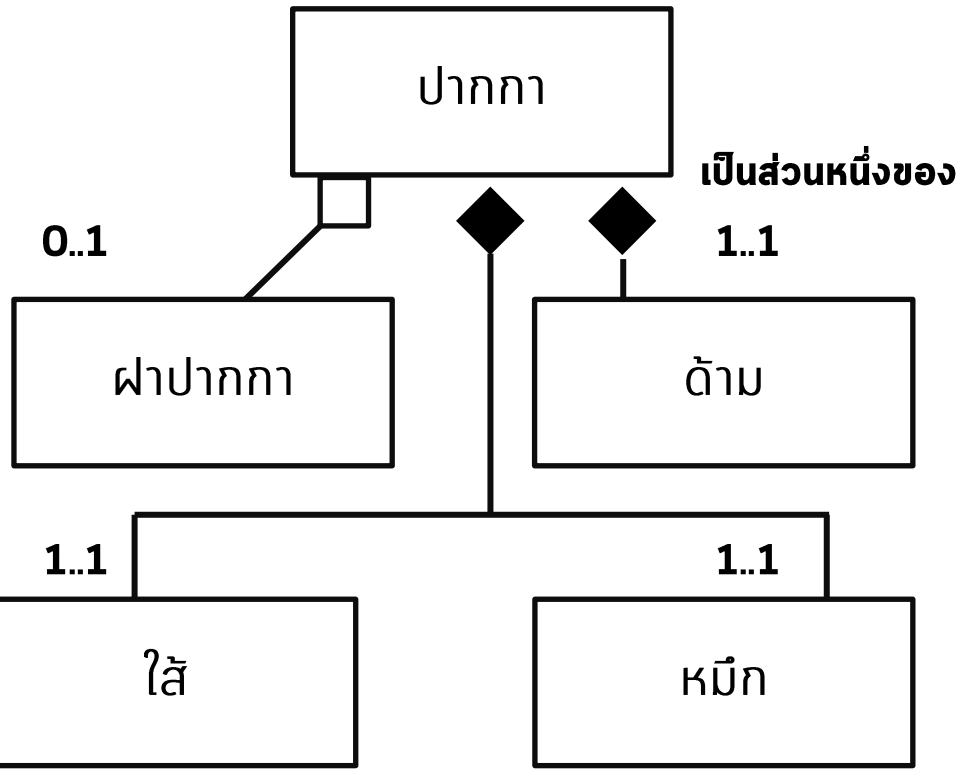




## Generalization abstraction

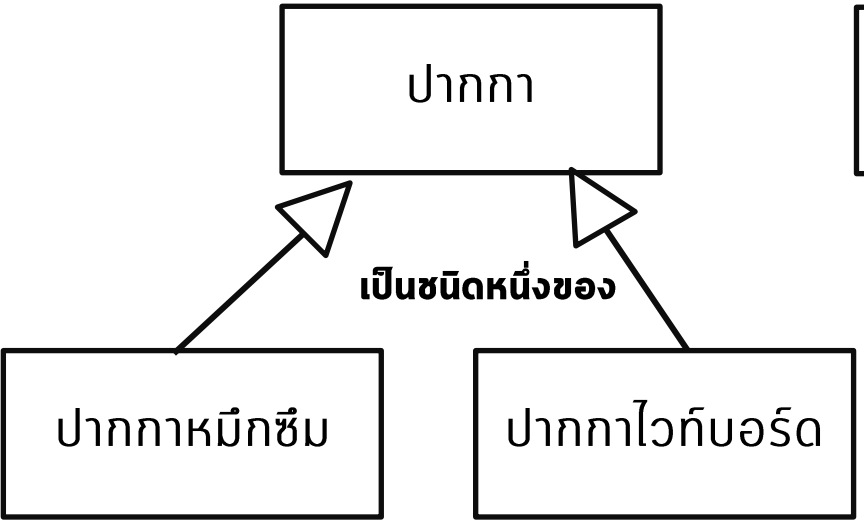
## Classification abstraction



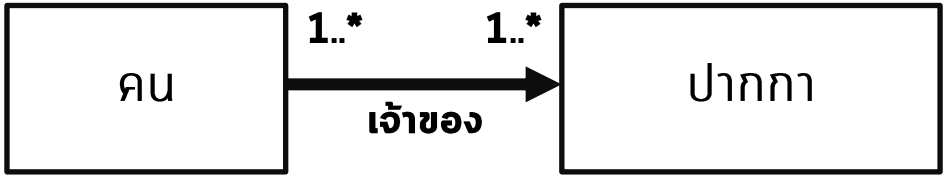

## Aggregation abstraction




## Generalization abstraction




## Association abstraction



บทที่ 4

# What is Modeling?

- การสร้างแบบจำลอง (Modeling)
  - เป็นวิธีการวิเคราะห์ และออกแบบ (Analysis and Design) วิธีการหนึ่งที่เน้นการสร้างแบบจำลอง เพื่อให้สามารถเข้าใจปัญหาของระบบ
  - ใช้เป็นเครื่องมือในการสื่อสาร แนวคิดในการพัฒนาระบบ กับบุคคลอื่นๆ (ระหว่างผู้พัฒนาระบบด้วยกัน และกับบุคคลอื่นๆ)
- Textual Modeling รหัสเทียม รหัสจำลอง
- Mathematics Modeling
  - เป็นโมเดลทางคณิตศาสตร์ เช่น สมการ สูตร $\sum_{i=1}^{n} x$
- Visual Modeling
  - Visual แปลว่า ภาพ
  - Modeling ก็คือแบบจำลอง ดังนั้น Visual Modeling ก็คือ ใช้สัญลักษณ์รูปภาพในการสร้างแบบจำลอง

Mathematic Modeling

$$\sum_{i=1}^{n} x$$

- ต.ย. จงหาอายุรวมของ นศ. วิทย์คอม 47
  - 52 53 42 28 47 58 19

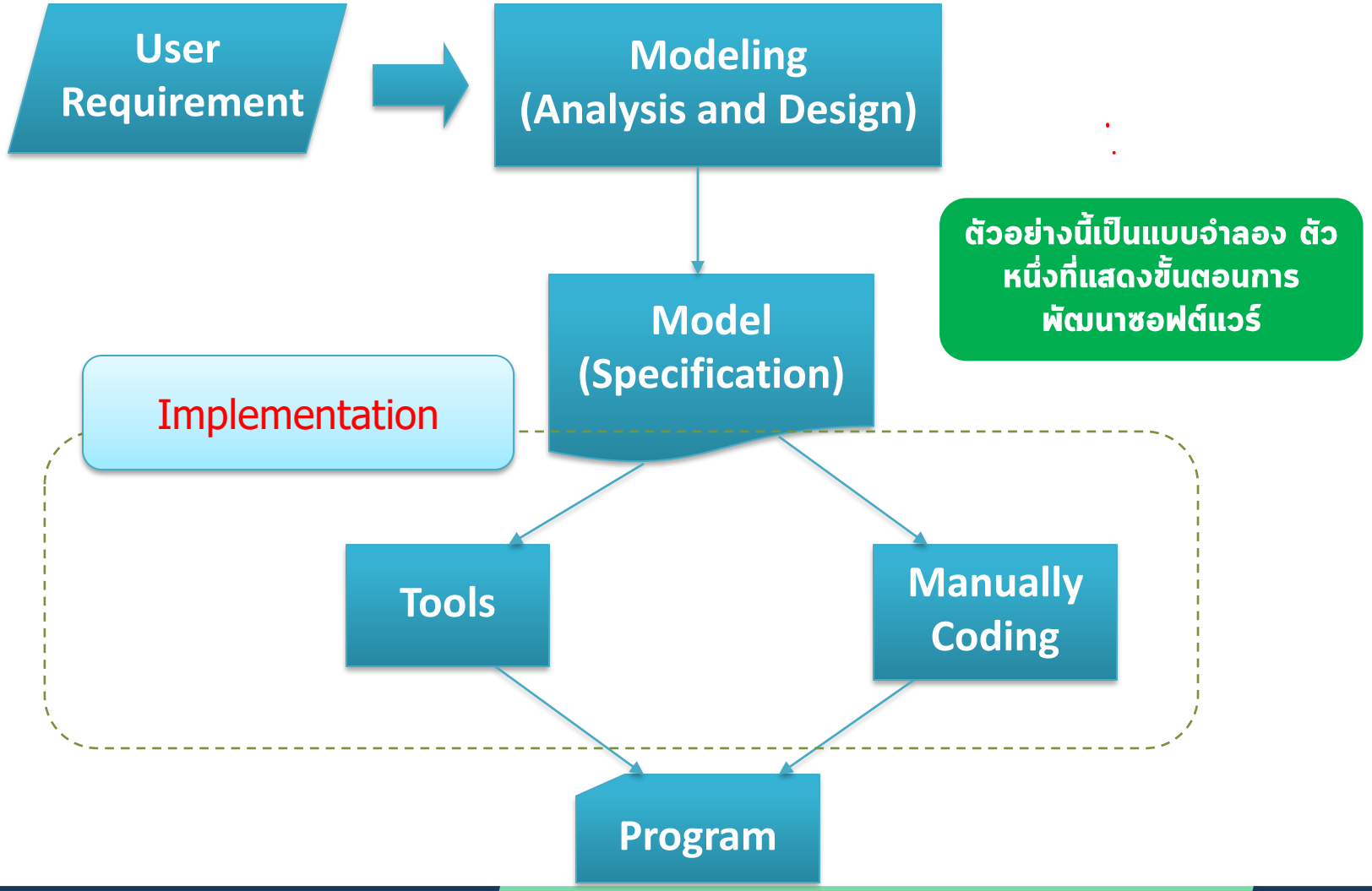
0. Specified problem
1. Requirement
2. การวิเคราะห์ INPUT / PROCESS (algo) / OUTPUT
3. ออกแบบ Flowchart/ Structure chart / PSEUDO-CODE

# เปรียบเทียบ

- **MODEL** ทำงานเสร็จแล้ว แต่อธิบายงานให้ง่ายขึ้นก็ทำโมเดลขึ้นมาเพื่ออธิบายด้วยวิธีการที่ง่าย
- **PROTOTYPE** ยังไม่ได้ทำตัวจริง แต่ทำตัวจำลองขึ้นมาก่อนแล้ว เอาตัวจำลองไปสร้างเป็นของจริง

# Software Modeling
# แบบจำลองของซอฟต์แวร์

บทที่ 4



บทที่ 4

# Models

- Requirement Analysis Models (Requirement Specification)

  ได้จากกระบวนการวิเคราะห์ความต้องการของผู้ใช้ระบบ (Requirement Analysis)

  *โมเดลสำหรับการวิเคราะห์ (SRS)*

- Analysis Model

  ได้จากกระบวนการวิเคราะห์หน้าที่การทำงานของระบบ (System Analysis)

  โมเดลสำหรับการวิเคราะห์ (use case model , class model )

- Design Model

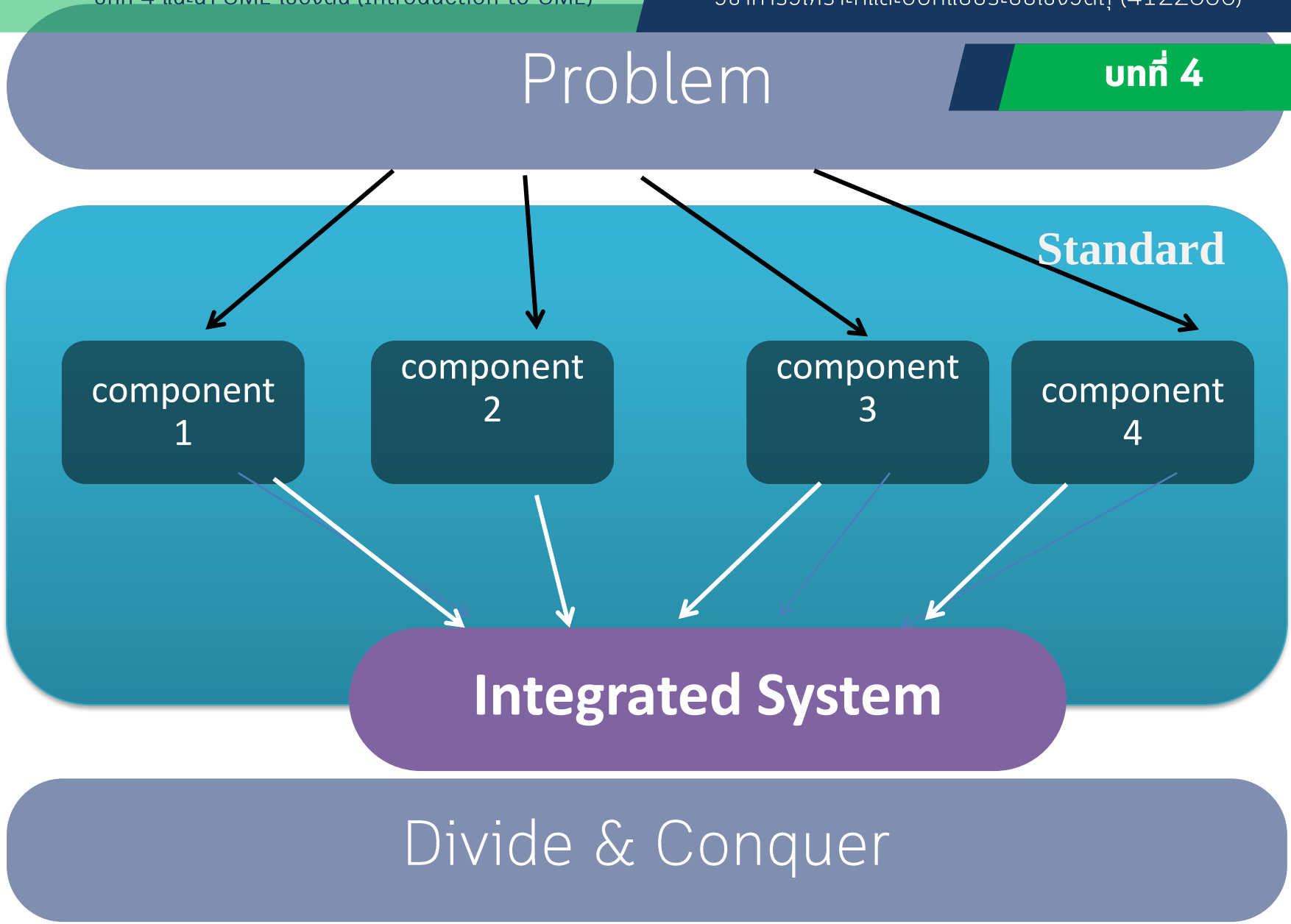
  ได้จากกระบวนการออกแบบการทำงานของระบบ (System Design)

  โมเดลสำหรับการวิเคราะห์ (use case model , class model )

*โมเดลสำหรับการออกแบบ (class model , state model , sequence model , activity model , deployment model)*

# SDLC: Software Development Process

1. **Requirement Specification** : define problem domain
2. **Analysis** : what problem to be solved? (อะไรคือปัญหาที่ต้องแก้)
3. **Design** : how to solve the problem? (แก้อย่างไร)
4. **Implementation** : how to implement the solution?
5. **Testing** : how to ensure that the solution can solve the problem?
    1. ทดสอบในเรื่องความเร็ว ประสิทธิภาพ ความปลอดภัย ความมั่นคง เสถียร
    2. ความเข้ากันได้ Integrated System
6. **Maintenance** : how to adjust the solution to accomodate change?
    1. ในรอบระยะเวลาหนึ่งอาจจะต้องมีการปรับเปลี่ยน
7. **Retirement** : when does the system to be retired?


Problem
บทที่ 4
Standard
component 1
component 2
component 3
component 4
Integrated System
Divide & Conquer

# Web application / Mobile App

- **MVC :**
  - **M : Model** → Data / Database
  - **V : View** → Interface
  - **C : Control/Coding** → Code

# Control : การคอนโทรล

- ควบคุมด้วยซอฟต์แวร์
- ควบคุมด้วย มาตรฐาน / ISO เช่น UML

**จะถูกพูดถึงในรายวิชา Software Project Management หรือ วิชา Software Engineering**

บทที่ 4

## ตัวอย่าง ข้อกำหนดความต้องการของระบบซื้อขายสินค้า

- ปัญหาเดิม / ที่มา ระบบเดิม ขายโดยการจดบันทึกในสมุด ปัญหา มีข้อผิดพลาดเยอะ / ล่าช้า
- ReqID :: 01 ต้องการให้ระบบสามารถบันทึกข้อมูลการขายสินค้าได้
  - ระบบจะต้องทำการบันทึก ข้อมูลสินค้าได้ที่ผู้ซื้อ ซื้อไป
  - ระบบจะต้องจัดเก็บข้อมูลผู้ซื้อได้
  - ระบบจะต้องตรวจสอบได้ว่า พนักงานคนใดขายสินค้าให้กับลูกค้า
  - ระบบจะต้องตรวจสอบจำนวนสินค้าในสต็อกได้

บทที่ 4

# Structured Analysis Models
# โมเดลการวิเคราะห์เชิงโครงสร้าง

- **Data Flow Diagrams :: DFD**
- **Entity Relationship Diagrams : ERD**
- **State-Transition Diagrams : STD**

เป็นโมเดลแบบดั้งเดิมที่ใช้สำหรับการวิเคราะห์ระบบแบบเชิงโครงสร้าง

บทที่ 4

# *Object-Oriented Analysis Models*

- Use-case Diagrams
- Class and Object Diagrams
- Behavioral Diagrams

UML

UP

เป็นโมเดลที่ใช้สำหรับการวิเคราะห์ระบบเชิงวัตถุ

# *The Unified Modelling Language (UML)*

# 4.1.1 what is UML ?

## UML (Unified Modeling Language) ไม่ระบุกระบวนการ (Process)

- **UML (Unified Modeling Language) ไม่ระบุกระบวนการ (Process) ที่ใช้ในการทำงานหรือไม่ระบุภาษาที่ใช้เขียนโปรแกรมหรือการพัฒนาซอฟต์แวร์เฉพาะว่าใช้ภาษาใด โดย UML เน้นการแสดงแผนภาพและแบบจำลอง (models) เพื่อให้ความเข้าใจและสื่อสารเกี่ยวกับโครงสร้างและพฤติกรรมของระบบโปรแกรมหรือระบบในรูปแบบสัญลักษณ์และแผนภาพที่เป็นมาตรฐาน โดยไม่เข้ารายละเอียดถึงกระบวนการหรือภาษาที่ใช้ในการสร้างระบบนั้น นั่นหมายความว่า UML ไม่มีความเกี่ยวข้องกับกระบวนการพัฒนาซอฟต์แวร์หรือการเขียนโปรแกรมแต่อย่างใด ซึ่งใช้กับหลายภาษาโปรแกรมต่าง ๆ เช่น Java, C++, Python, หรือ Ruby ได้.**
- **แทนที่จะระบุกระบวนการหรือภาษาที่ใช้ UML ใช้แผนภาพและแบบจำลองต่าง ๆ แบ่งเป็นหลายลักษณะเพื่ออธิบายและแสดงความสัมพันธ์ระหว่างส่วนต่าง ๆ ของระบบ นอกจากนี้ UML ยังสนับสนุนการสร้างเอกสารและคู่มือที่ช่วยในกระบวนการพัฒนาโปรแกรมและระบบ โดยไม่ได้ระบุลักษณะของกระบวนการเฉพาะหรือภาษาโปรแกรมที่ต้องใช้ในโครงการนั้น ๆ ดังนั้น UML เป็นเครื่องมือที่มีความยืดหยุ่นและสามารถนำมาใช้กับหลายวิธีการและภาษาโปรแกรมต่าง ๆ ได้ตามความเหมาะสมของโครงการและทีมพัฒนา.**

# 4.1.1 what is UML ?

- เป็นเครื่องมือหรือภาษาที่ใช้ในการสร้างแบบจำลอง (Modeling Language) ที่ประกอบด้วยองค์ความรู้ที่ใช้ในการนำเสนอและออกแบบเอกสารประกอบโปรแกรม
  - คู่มือทางเทคนิค Programmer Manual/Technical Manual
    - อธิบายรายละเอียดทางเทคนิค การวิเคราะห์ การออกแบบ การสร้าง การทดสอบ
  - คู่มือสำหรับผู้ใช้ User Manual อธิบาย การใช้งานซอฟต์แวร์ , การป้อนข้อมูล การปรับปรุง แก้ไขขั้นต้น
- รวม 3 แนวคิด/วิธีการเชิงวัตถุที่ ได้แก่ Booch, Rumbaugh และ Jacobson รวมทั้งจากบุคคลอื่น
- จัดรวบรวมออกมาเป็นมาตรฐานสำหรับการแลกเปลี่ยนแนวคิดการออกแบบระบบ และองค์ความรู้ในเชิงเทคนิคในรูปของแบบจำลอง

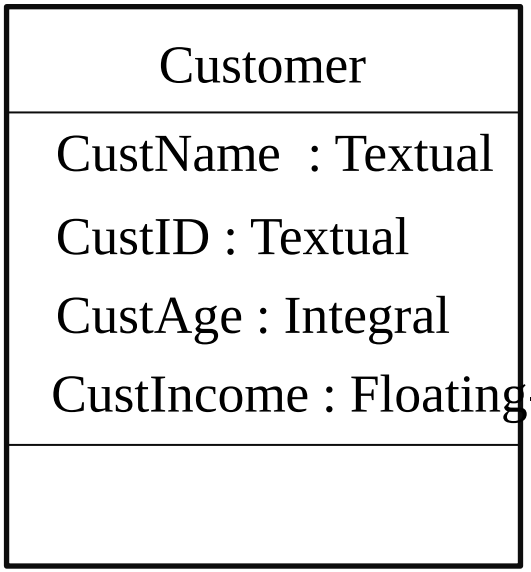
# 4.1.2 what is *not* UML ?

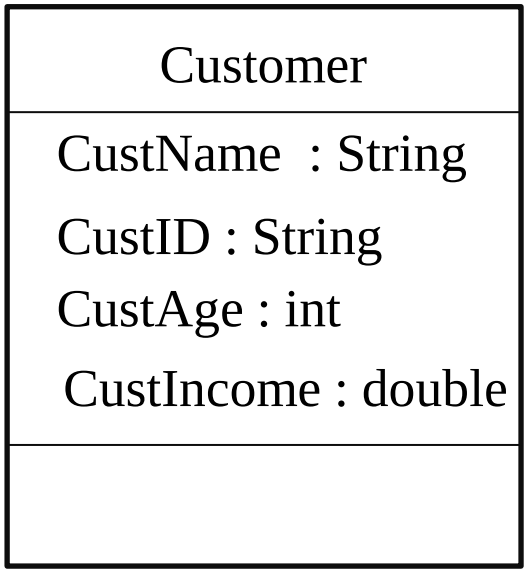
- ไม่ใช่ภาษาโปรแกรม (Programming Language)
  - ไม่ใช่ภาษา VB , Pascal , Delphi , Java , C# etc.
- ใช่ method หรือ methodology {เป็นวิธีการเป็นกระบวนการ}
- ไม่ระบุ Process ที่ใช้ในการทำงาน/จะไม่ระบุเฉพาะว่าใช้ภาษาใด
- UML ใช้ในการสร้างแบบจำลองการวิเคราะห์ และออกแบบระบบ โดยไม่ขึ้นกับ Process
- แต่ละโครงงานสามารถเลือก Process การทำงาน ที่เหมาะสม กับสภาพความจริงของโครงงาน โดยไม่ขึ้นกับ modeling language (สามารถใช้กับภาษาโปแกรมใดก็ได้)

**** *ความหมายไม่กำหนด ภาษาที่ใช้ หรือเครื่องมือที่ใช้ตายตัว ให้ออกแบบกว้างๆ ไว้*

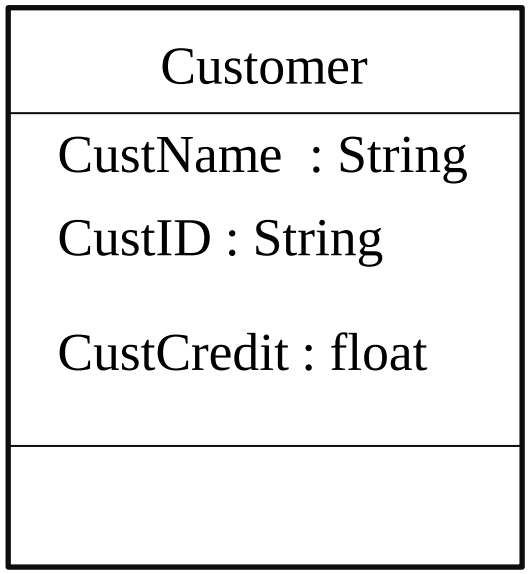
UML Class diagram

Implementation

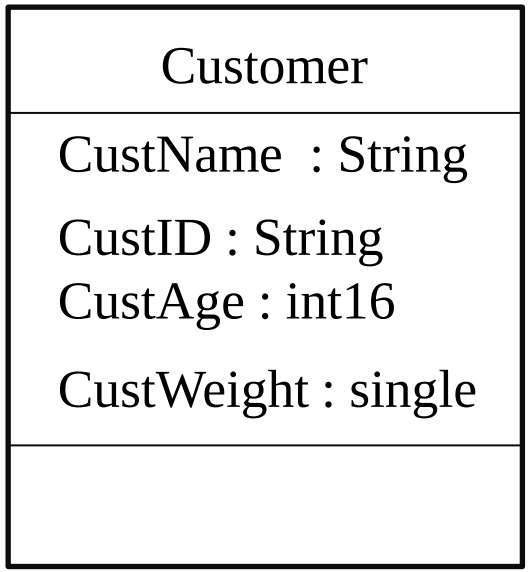



Java, c,c#




Java, c,c#




VB

# 4.1.3 Data Type in UML

## UML Type

- Textual ข้อความ สายอักขระ
- Character ตัวอักษร
- Integral ..... จำนวนเต็ม
- Floating-Point ทศนิยม
- Boolean ตรรกะ
- Date วันที่

# 4.1.4 History of UML ?

- 1970s วิธีเชิงวัตถุ (Object-oriented method)
- 1970s, 1980s, 1990s “Method Wars”

  Grady Booch, Jim Rumbaugh, Ivar Jacobson
- 3 แนวคิด/วิธีการเชิงวัตถุที่เป็นที่นิยมใช้ในช่วงกลางทศวรรษ 1990s ได้แก่
  - OOSE (Object-Oriented Software Engineering) โดย Ivar Jacobson
  - OMT (Object Modelling Technique) โดย Jim Rumbaugh
  - Booch method โดย Grady Booch
- ปัญหาด้านการสื่อสาร สัญลักษณ์ที่ใช้ในแต่ละวิธีการไม่เหมือนกัน

# 4.1.4 History of UML ? *(Cont...)*

- ในปี 1994/5 Booch, Rumbaugh และ Jacobson (เรียกตัวเองว่า “three amigos”) ร่วมกันทำการรวบรวมแนวคิด องค์ความรู้และเทคนิคต่าง ๆ เข้าด้วยกัน ที่ Rational Software
- Three Amigos เสนอ Unified Modelling Language (UML) ไปยัง หน่วยงาน OMG (Object Management Group) เพื่อให้เป็นภาษามาตรฐานสำหรับสร้างแบบจำลองเชิงวัตถุ
- UML Version 1.1 ได้ถูกพัฒนาขึ้นในปี 1997 เพื่อเป็นมาตรฐานสำหรับสร้างแบบจำลองเชิงวัตถุ

# 4.1.4 History of UML ? *(Cont...)*

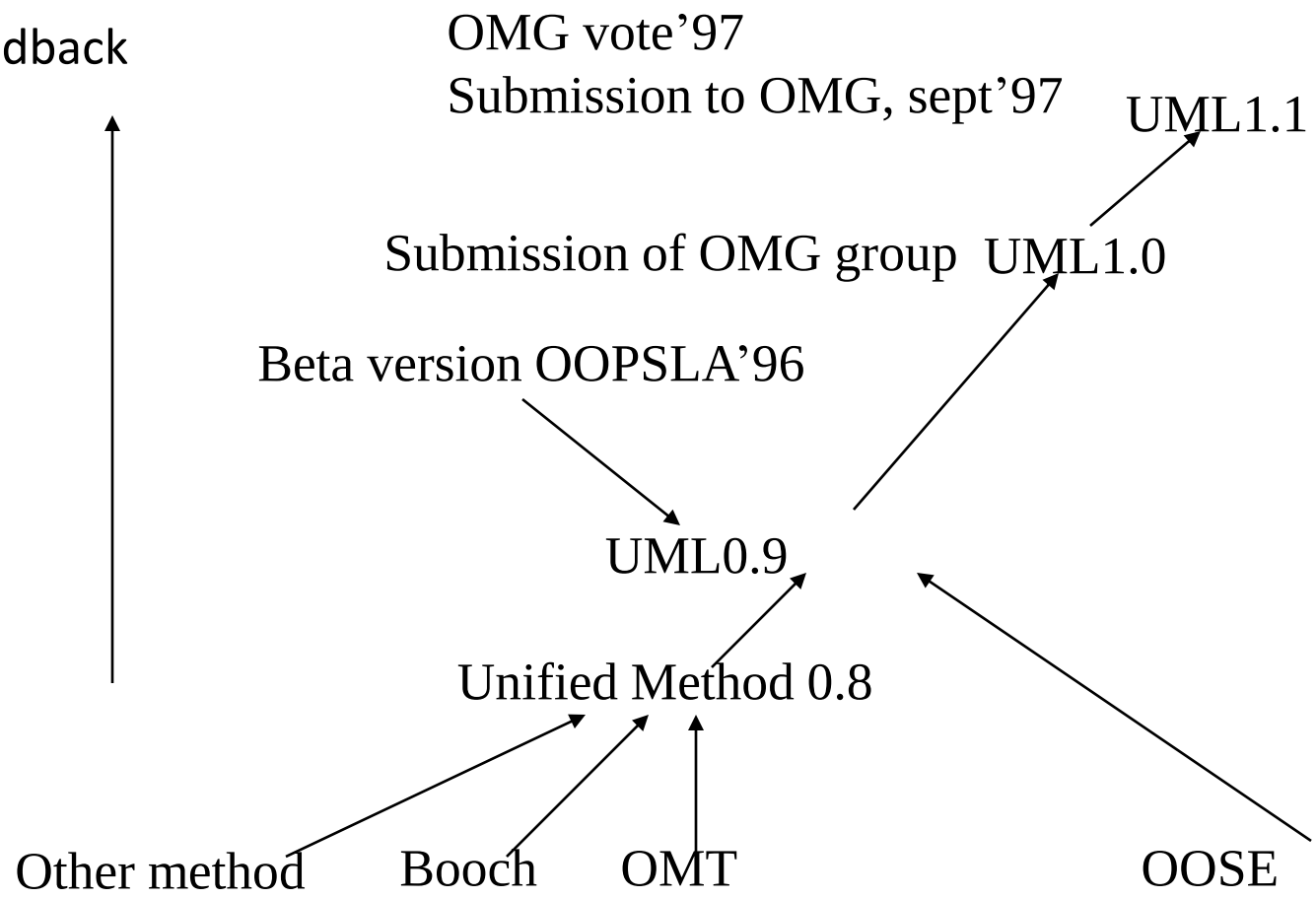
- ปี 1998 พัฒนา UML Version 1.2
- ปี 1999 พัฒนา UML Version 1.3
- ปี 2000 พัฒนา UML Version 1.4
- ปี 2001 พัฒนา UML Version 2.0
- http://www.uml.org/


25 YEARS ANNIVERSARY | WHAT IS UML? | UML VENDOR | UML SPECIFICATIONS | UML CERTIFICATION | TRAINING PAGE

# 4.1.4 History of UML ? *(Cont...)*

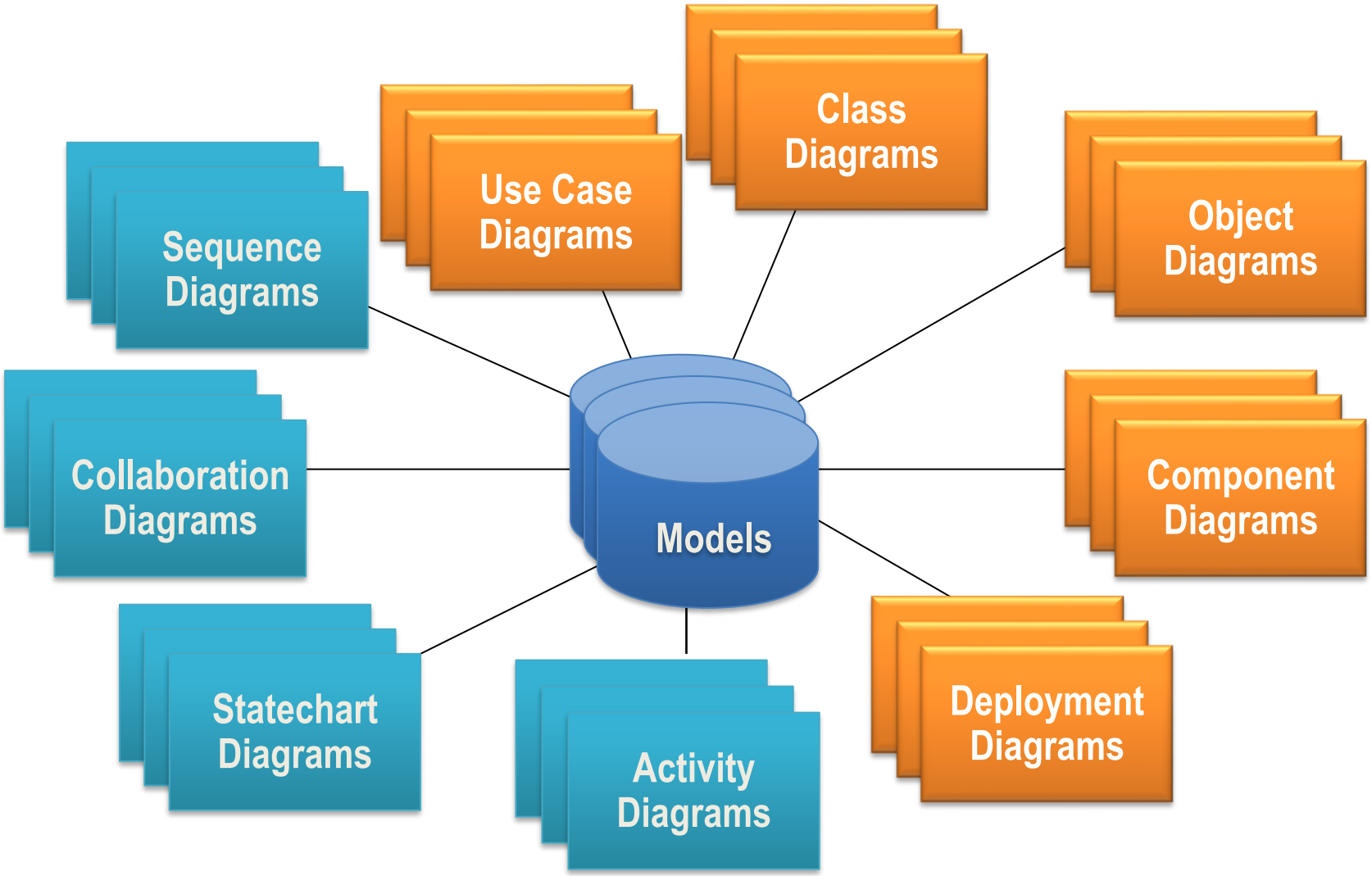
# 4.1.4 History of UML ? *(Cont...)*

The Evolution of the UML:



# Models and Diagrams

A ***model*** is a complete description of a system from a particular perspective



# 4.1.5 UML 2.0

UML 2 defines 14 diagrams (all supported by Enterprise Architect)

1. Package diagrams
2. Class or Structural diagrams
3. Object diagrams
4. Composite Structure
5. Component diagrams
6. Deployment diagrams
7. Use Case Diagrams
8. Activity diagrams
9. State Machine diagrams
10. Communication diagrams
11. Sequence diagrams
12. Timing diagrams
13. Interaction Overview diagrams
14. Profile diagrams

# 4.1.6 ทำไมต้องเป็น UML

- Development life cycle - สนับสนุนทั้ง life cycle ของการพัฒนาระบบ สามารถถูกใช้ตั้งแต่ขั้นตอนการหาความต้องการ (Requirements) จนถึงขั้นตอนการทดสอบติดตั้งระบบ (Test & Installation)
- Application domains - สนับสนุนการสร้าง application ในหลาย domain เช่น information systems {MIS , EIS , ES , DSS,TPS}, embeded real-time systems, technical systems, distributed systems, system software
- Implementation languages and platforms - ไม่ผูกติดกับภาษาและแพลฟอร์ม (platform) ของการ implement สนับสนุนทั้ง pure oo languages เช่น smalltalk java, c#. หรือ hybrid เช่น c++ หรือ non-oo เช่น c,pascal,basic
- Development process – สามารถใช้ UML ร่วมกับกรรมวิธีพัฒนาระบบใดๆก็ได้ (prefer UP)
- Internal concepts – มีหลักการที่ชัด มั่นคง

ระบบสารสนเทศมีหลายประเภทของสถาปัตยกรรมที่สามารถนำมาใช้เพื่อออกแบบและพัฒนาระบบได้อย่างเหมาะสมกับความต้องการและความซับซ้อนของระบบ สิ่งที่เหมาะสมนั้นจะขึ้นอยู่กับลักษณะของธุรกิจหรือองค์กรและวัตถุประสงค์ของระบบสารสนเทศนั้น ๆ ดังนั้น ระบบสารสนเทศสามารถมีสถาปัตยกรรมที่แตกต่างกันได้ตามลักษณะของงานและการทำงาน บางตัวอย่างของประเภทของสถาปัตยกรรมระบบสารสนเทศได้แก่:

1. **สถาปัตยกรรมแบบเบราว์เซอร์ (Browser-Based Architecture)**:
   - สถาปัตยกรรมนี้ใช้เบราว์เซอร์เป็นอินเตอร์เฟซหลักในการเข้าถึงและใช้งานระบบสารสนเทศ โดยส่วนต่าง ๆ ของระบบอาจถูกพัฒนาขึ้นบนเซิร์ฟเวอร์และใช้เทคโนโลยีเว็บเบราว์เซอร์เช่น HTML, JavaScript, และ CSS เพื่อสร้างอินเตอร์เฟซผู้ใช้.

2. **สถาปัตยกรรมแบบมิโครเซอร์วิส (Microservices Architecture)**:
   - สถาปัตยกรรมนี้แยกแยะระบบสารสนเทศออกเป็นบริการเล็ก ๆ ที่อยู่ในคอนเทนเนอร์แยกต่าง ๆ และสามารถทำงานแยกกันได้ บริการเหล่านี้สามารถถูกพัฒนาและปรับปรุงโดยอิสระ เพื่อให้ง่ายต่อการส่งมอบและการขยายขนาดระบบ.

3. **สถาปัตยกรรมแบบเซอร์วิส (Service-Oriented Architecture, SOA)**:
   - สถาปัตยกรรม SOA ใช้การเชื่อมต่อระหว่างบริการที่เป็นองค์ประกอบของระบบสารสนเทศ โดยบริการเหล่านี้สามารถใช้ซ้ำได้และสามารถเรียกใช้งานกันได้ผ่านการสื่อสารมาตรฐาน เช่น SOAP (Simple Object Access Protocol) หรือ REST (Representational State Transfer).

4. **สถาปัตยกรรมแบบซอฟต์แวร์เป็นฐานคลาย (Software as a Service, SaaS)**:
   - สถาปัตยกรรม SaaS ใช้โฮสต์บริการบนคลาวด์ให้กับผู้ใช้ ผู้ใช้สามารถเข้าถึงและใช้งานแอปพลิเคชันหรือระบบสารสนเทศผ่านเบราว์เซอร์ผ่านอินเทอร์เน็ตโดยไม่ต้องติดตั้งซอฟต์แวร์บนคอมพิวเตอร์ของตนเอง.

5. **สถาปัตยกรรมแบบไคลเอนต์-เซิร์ฟเวอร์ (Client-Server Architecture)**:
   - สถาปัตยกรรมนี้แบ่งแยกหน้าที่ระหว่างเครื่องลูกข่าย (Client) และเซิร์ฟเวอร์ (Server) โดยที่เครื่องลูกข่ายใช้งานผ่านเซิร์ฟเวอร์เพื่อเข้าถึงข้อมูลและบริการที่ต้องการ.

นอกจากนี้ยังมีสถาปัตยกรรมอื่น ๆ ที่สามารถนำมาใช้กับระบบสารสนเทศตามความต้องการของโครงการและองค์กร เลือกสถาปัตยกรรมที่เหมาะสมสำหรับระบบส

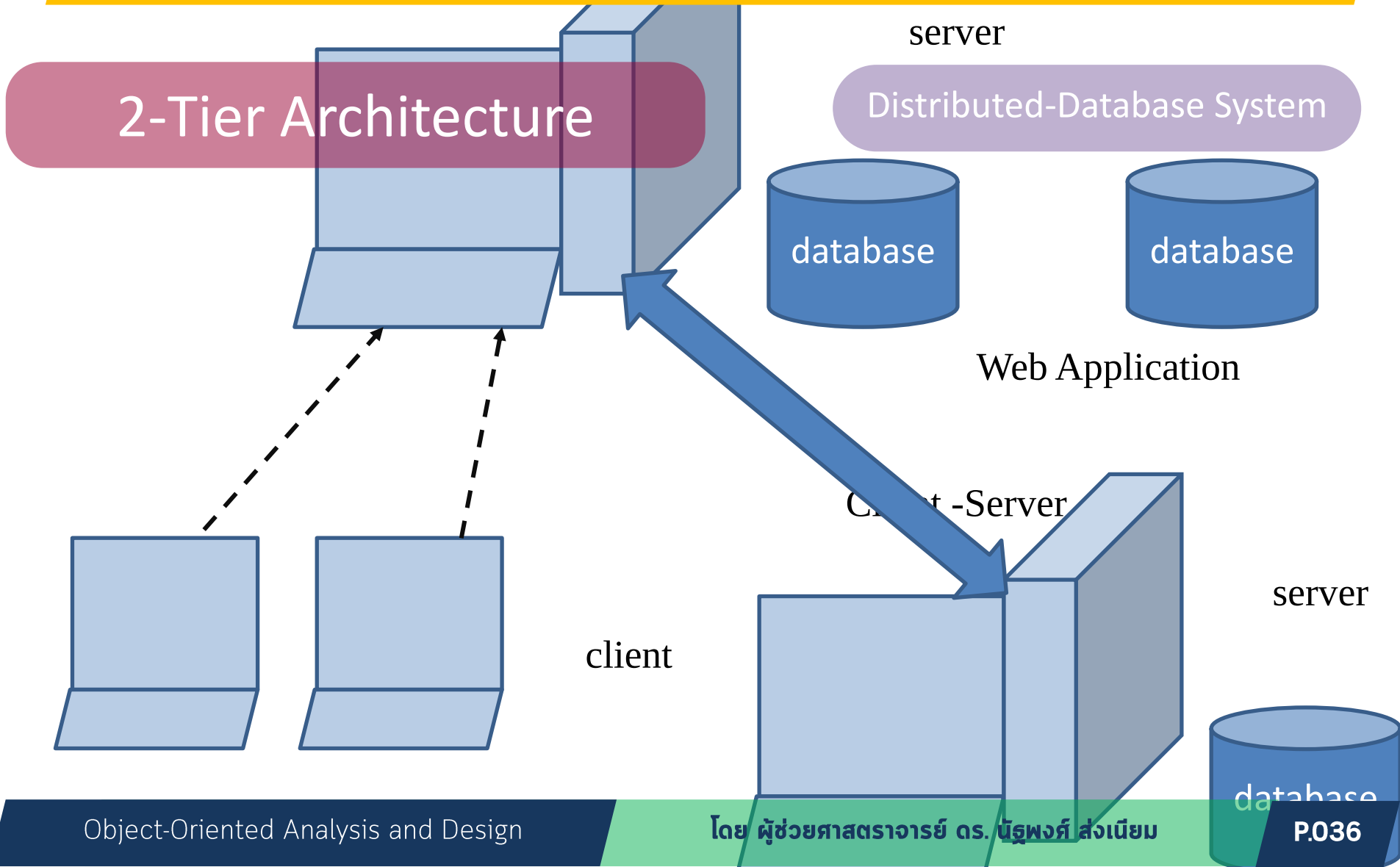
# 4.1.6 ทำไมต้องเป็น UML *(Cont...)*

## สถาปัตยกรรมแบบเบราว์เซอร์

**1. สถาปัตยกรรมแบบเบราว์เซอร์ (Browser-Based Architecture):**

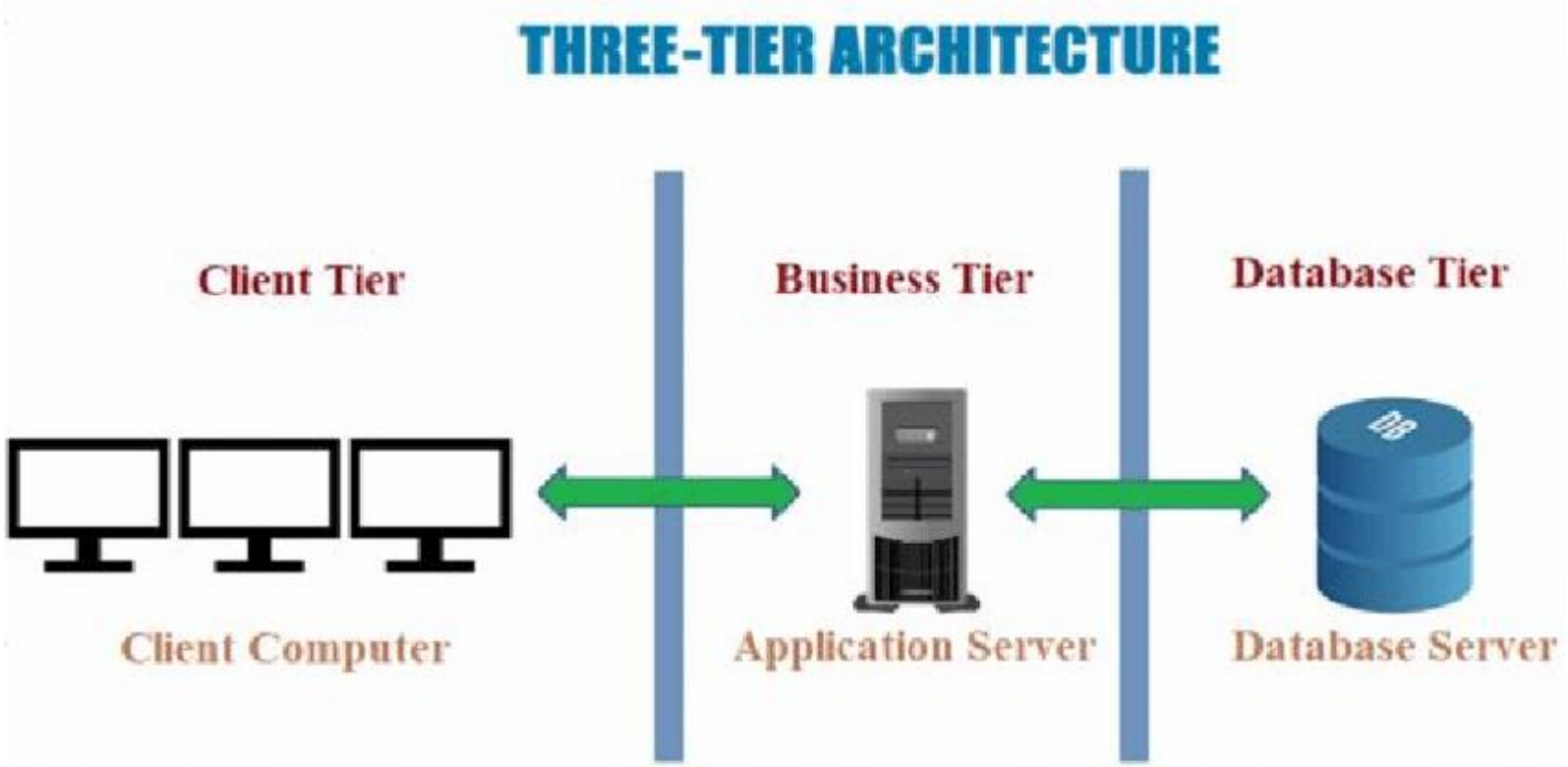
**- สถาปัตยกรรมนี้ใช้เบราว์เซอร์เป็นอินเตอร์เฟซหลักในการเข้าถึงและใช้งานระบบสารสนเทศ โดยส่วนต่าง ๆ ของระบบอาจถูกพัฒนาขึ้นบนเซิร์ฟเวอร์และใช้เทคโนโลยีเว็บเบราว์เซอร์เช่น HTML, JavaScript, และ CSS เพื่อสร้างอินเตอร์เฟซผู้ใช้.**

# 4.1.6 ทำไมต้องเป็น UML *(Cont...)*



# 4.1.6 ทำไมต้องเป็น UML *(Cont...)*

## 3-Tier Architecture



# 4.1.7 มุมมอง UML (UML Views)

**สามารถใช้ UML สร้างเป็นโมเดลแสดงมุมมองระบบได้เป็น 3 กลุ่ม**

| Major Area | **มุมมอง** (View) | **แผนภาพ** (Diagrams) |
|---|---|---|
| **โครงสร้าง** (structural) | static view | class diagram |
| | use case view | use case diagram |
| | implementation view<br>deployment view | component diagram |
| | | deployment diagram |
| **พลวัตร** (dynamic) | state machine view | state chart diagram |
| | activity view | activity diagram |
| | interaction view | sequence diagram |
| | | collaboration diagram |
| **การจัดการโมเดล** (model management) | model management view | class diagram (package, subsystem) |

# 4.1.7 มุมมอง UML (UML Views)

- 5 มุมมองหลักของ UML
  - Use-case view : หน้าที่การทำงานของระบบซอฟต์แวร์ โดยพิจารณาจากมุมมองของผู้ใช้ภายนอก หรือ ระบบภายนอก
    - use-case diagram
  - Logical view : หน้าที่การทำงานของระบบมีโครงสร้างอย่างไร มองในรูปของ static structure และ dynamic behavior
    - class diagram, object diagram, state, sequence, collaboration, activity diagrams

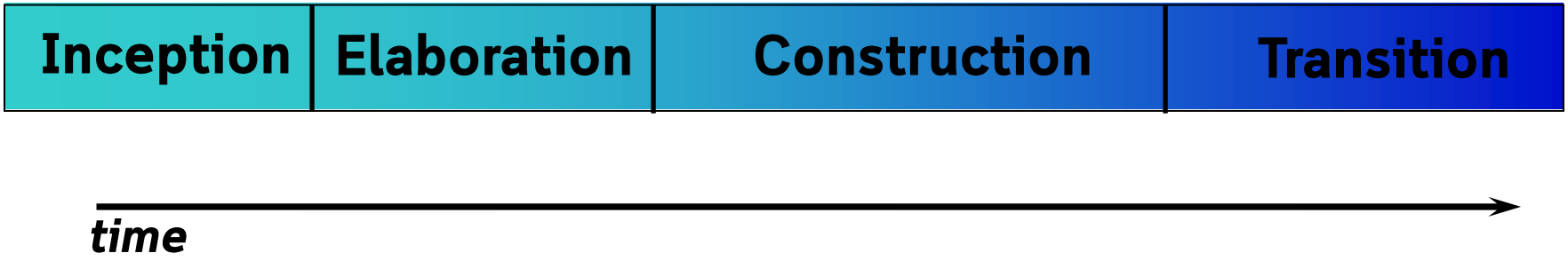
# 4.1.7 มุมมอง UML (UML Views)

- **Component view** : องค์ประกอบย่อยในการ implement ที่ประกอบเป็นระบบ และ **dependency** ระหว่างองค์ประกอบเหล่านั้น
  - component diagram

- **Concurrency view** : การแบ่งแยก process และ processors โดยพิจารณาทั้ง communication และ synchronization
  - dynamic diagrams (state, sequence, collaboration activity)
  - implementation diagrams(component และ deployment)

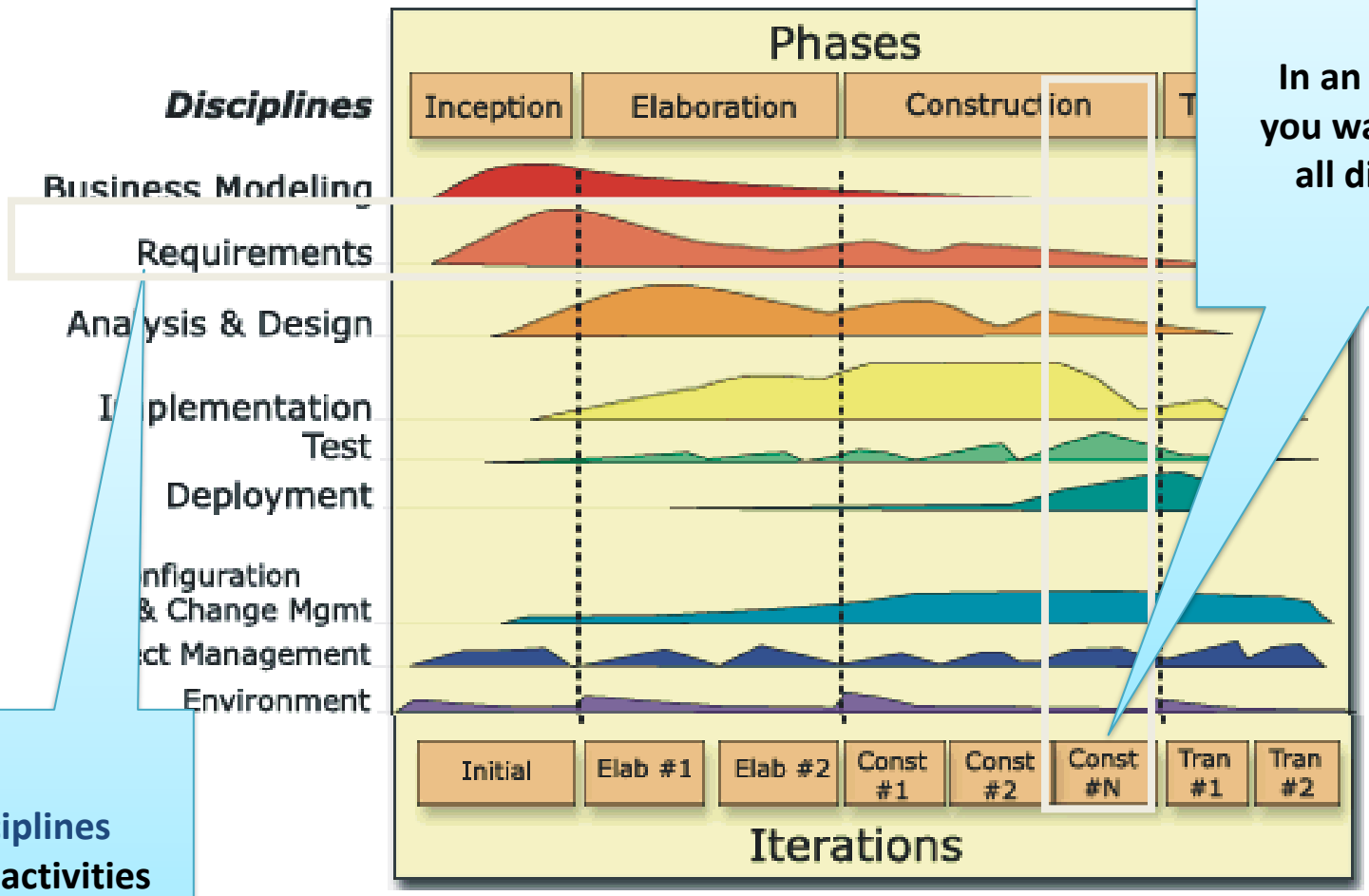
- **Deployment view** : โครงสร้างทางกายภาพเกี่ยวกับ การติดตั้ง และใช้งานระบบ
  - deployment diagram

# 4.1.8 UP: Life Cycle Phases




- **Inception** — Define the scope of the project and develop business case
- **Elaboration** — Plan project, specify features, and baseline the architecture
- **Construction** — Build the product
- **Transition** — Transition the product to its users

# 4.1.8 UP: Life Cycle Phases (*Cont...*)



# 4.2 UML has 9 kinds of diagrams

1. Class Diagram
2. Object Diagram
3. Component Diagram
4. Deployment Diagram

**Structural Diagrams (static diagrams)**

6. Use case Diagram
6. Sequence Diagram
7. Collaboration Diagram
8. State Transition Diagram
9. Activity Diagram

**Behavioral Diagrams (dynamic diagrams)**

# 4.3 แผนภาพคลาส(Class Diagrams)

**Class diagram (แผนภาพชั้นคลาส)** เป็นหนึ่งในแผนภาพของ UML (Unified Modeling Language) ที่ใช้ในการแสดงและอธิบายโครงสร้างของคลาสและองค์ประกอบของระบบหรือโปรแกรมซอฟต์แวร์ โดย Class diagram จะแสดงคลาส (Class) และความสัมพันธ์ระหว่างคลาสที่ใช้ในระบบโปรแกรม แผนภาพนี้มักถูกใช้ในกระบวนการวางแผนและออกแบบโครงสร้างของระบบ รวมถึงสื่อสารและแสดงความเข้าใจระหว่างทีมพัฒนาและผู้เกี่ยวข้องอื่น ๆ เกี่ยวกับโครงสร้างของระบบ.

**Class diagram** เป็นเครื่องมือที่สำคัญในการวางแผนและออกแบบโครงสร้างของระบบซอฟต์แวร์ และมีประโยชน์ในการเพิ่มความเข้าใจและความยืดหยุ่นในกระบวนการพัฒนาซอฟต์แวร์ นอกจากนี้ยังช่วยในการสื่อสารระหว่างทีมพัฒนาและผู้ใช้งาน โดยทำให้คำสั่งและความต้องการของผู้ใช้งานเป็นที่เข้าใจและสามารถแปลงเป็นโครงสร้างโปรแกรมได้อย่างมีประสิทธิภาพ.

# 4.3 แผนภาพคลาส(Class Diagrams)

- **Class diagrams**
  - แสดงรายละเอียดของ Class และความสัมพันธ์ระหว่าง Class ในมุมมองแบบ logical view

- **ส่วนประกอบของ UML class diagrams ได้แก่**
  - Class, โครงสร้างของ Class และพฤติกรรมของ Class
  - ตัวบ่งชี้ Multiplicity และ navigation
  - ชื่อของ Role
  - ความสัมพันธ์แบบ Association, Aggregation, Dependency, และ Generalization (Inheritance)

*** แผนภาพที่อธิบายเกี่ยวกับคลาสในบทที่ ผ่านมา เป็นเพียงแค่ใช้สำหรับการเรียน แต่ต่อไปนี้จะเป็นแผนภาพและสัญลักษณ์ตามมาตรฐาน UML

# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.1 รายละเอียดส่วนประกอบของแผนภาพคลาส

1. คลาส (Class): แสดงคลาสหรือชนิดของอ็อบเจกต์ที่มีคุณลักษณะและเมธอดที่สมบูรณ์.
2. คุณลักษณะ (Attributes): แสดงข้อมูลหรือตัวแปรที่เกี่ยวข้องกับคลาสและมีค่าเป็นส่วนหนึ่งของอ็อบเจกต์ ในรูปแบบชื่อและประเภทของคุณลักษณะ.
3. เมธอด (Methods): แสดงการดำเนินการหรือฟังก์ชันที่คลาสสามารถทำได้ ในรูปแบบชื่อและพารามิเตอร์ของเมธอด.
4. ความสัมพันธ์ (Relationships): แสดงความสัมพันธ์ระหว่างคลาสต่าง ๆ ซึ่งสามารถเป็นความสัมพันธ์ที่ต่างกัน เช่น ความสัมพันธ์แบบเป็นส่วนหนึ่ง (association), ความสัมพันธ์แบบการสืบทอด (inheritance), หรือ ความสัมพันธ์แบบเส้นเชื่อม (dependency).
5. หมวดหมู่ (Packages): แสดงการจัดหมวดหมู่คลาสเพื่อแบ่งแยกและจัดระเบียบโครงสร้างของระบบ.

# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.1 รายละเอียดส่วนประกอบของแผนภาพคลาส

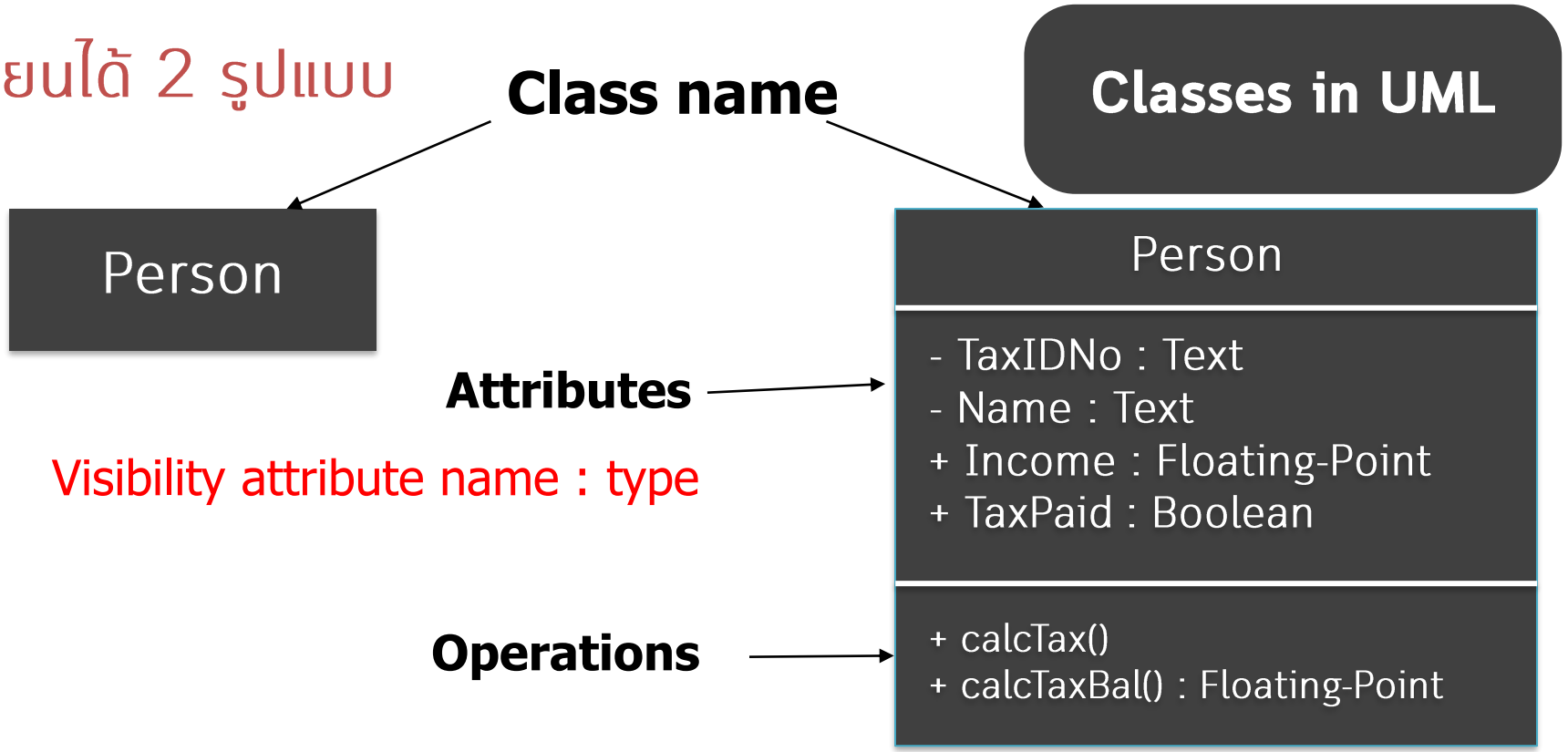
### มุมมอง

- Logical View เชิงตรรกะ แนวคิด use case , Class diagram
- Physical View เชิงกายภาพ component , deployment

# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.2 หลักการเขียนแผนภาพคลาส

- เขียนได้ 2 รูปแบบ

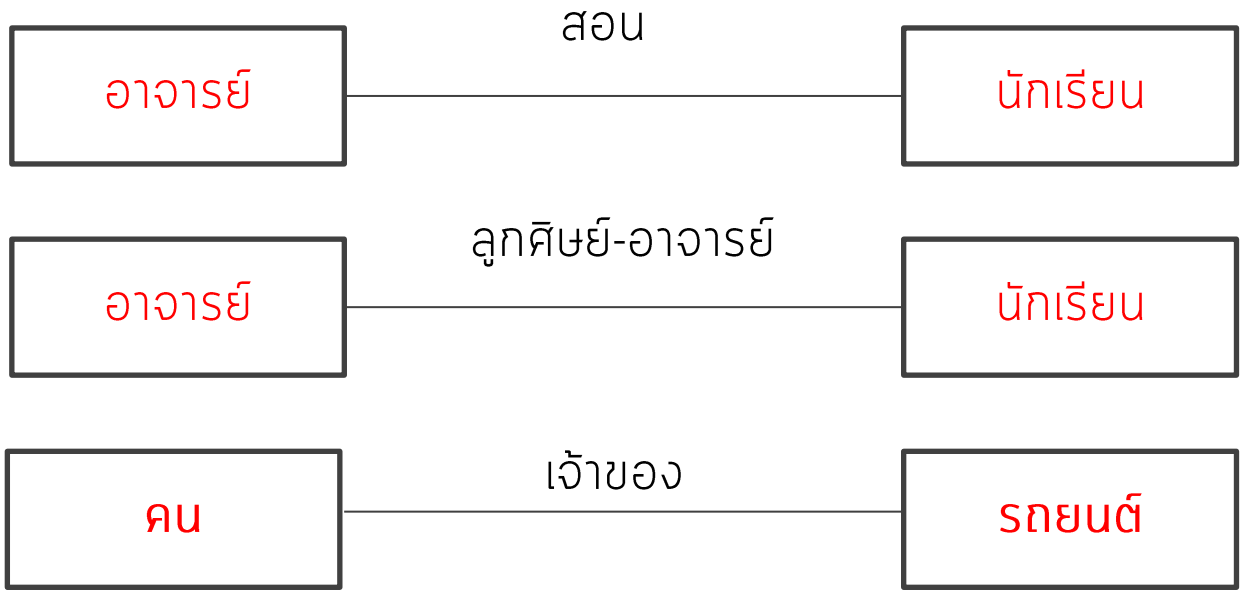



Visibility operation name(parameter : type) : result type

# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.2 หลักการเขียนแผนภาพคลาส

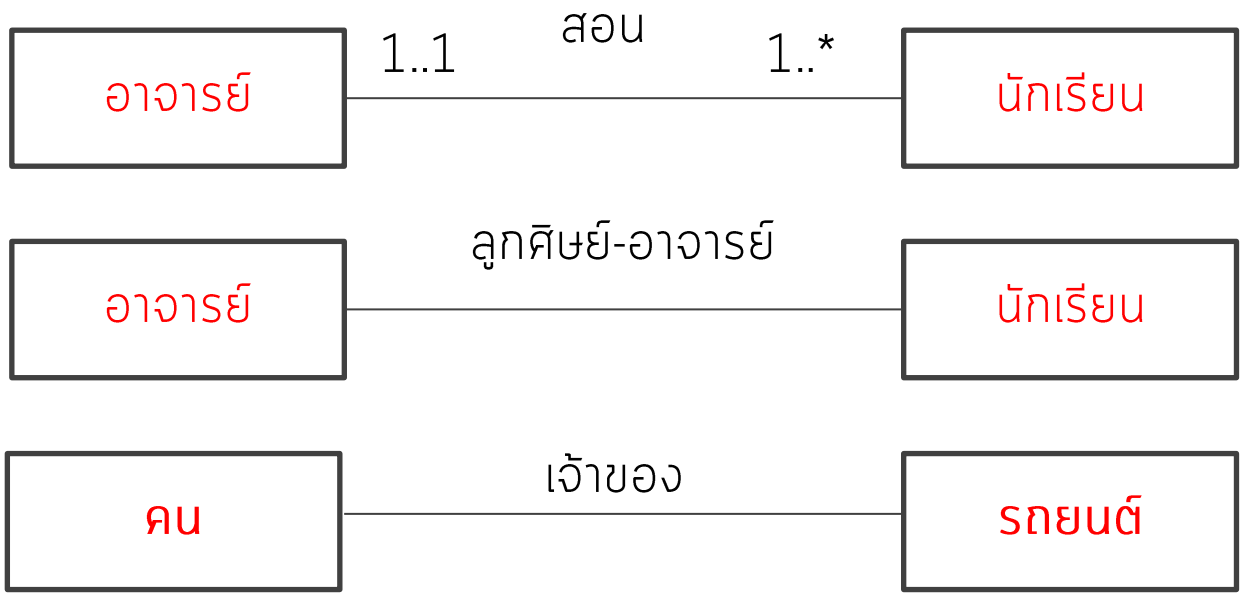
**กรณีที่ต้องการแสดงภาพรวมของระบบ ให้เขียนแบบย่อ คือ มีแค่ชื่อคลาส และความสัมพันธ์ระหว่างคลาสเท่านั้น**



# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.2 หลักการเขียนแผนภาพคลาส

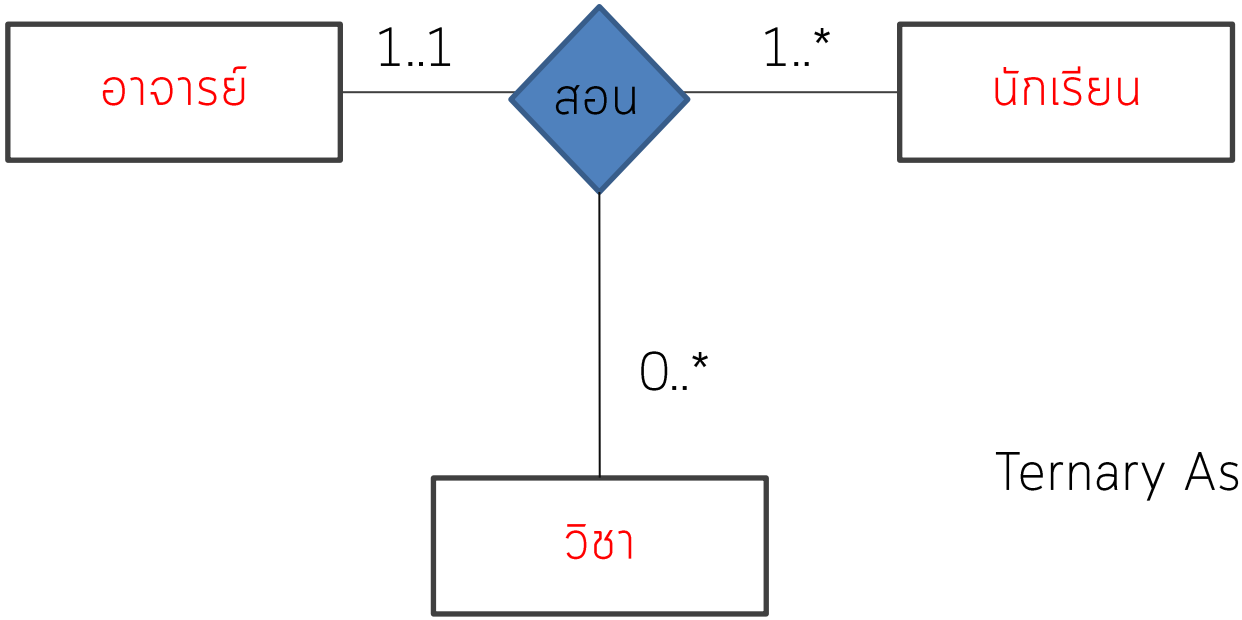
**กรณีที่ต้องการแสดงภาพรวมของระบบ ให้เขียนแบบย่อ คือ มีแค่ชื่อคลาส และความสัมพันธ์ระหว่างคลาสเท่านั้น**



# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.2 หลักการเขียนแผนภาพคลาส

**กรณีที่ต้องการแสดงภาพรวมของระบบ ให้เขียนแบบย่อ คือ มีแค่ชื่อคลาส และความสัมพันธ์ระหว่างคลาสเท่านั้น**

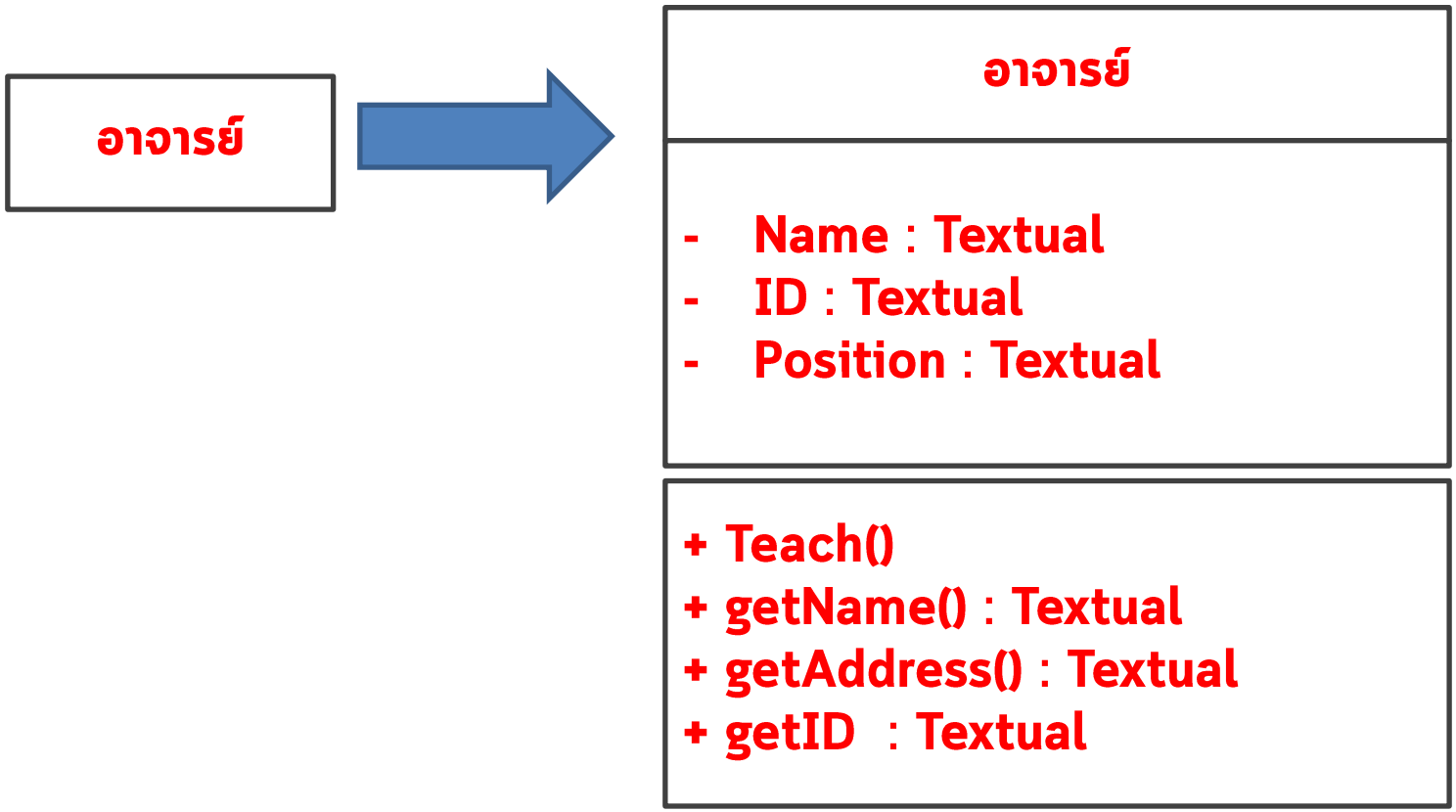



Ternary Association

# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.2 หลักการเขียนแผนภาพคลาส

UP : Construction phase



# 4.3 แผนภาพคลาส(Class Diagrams)

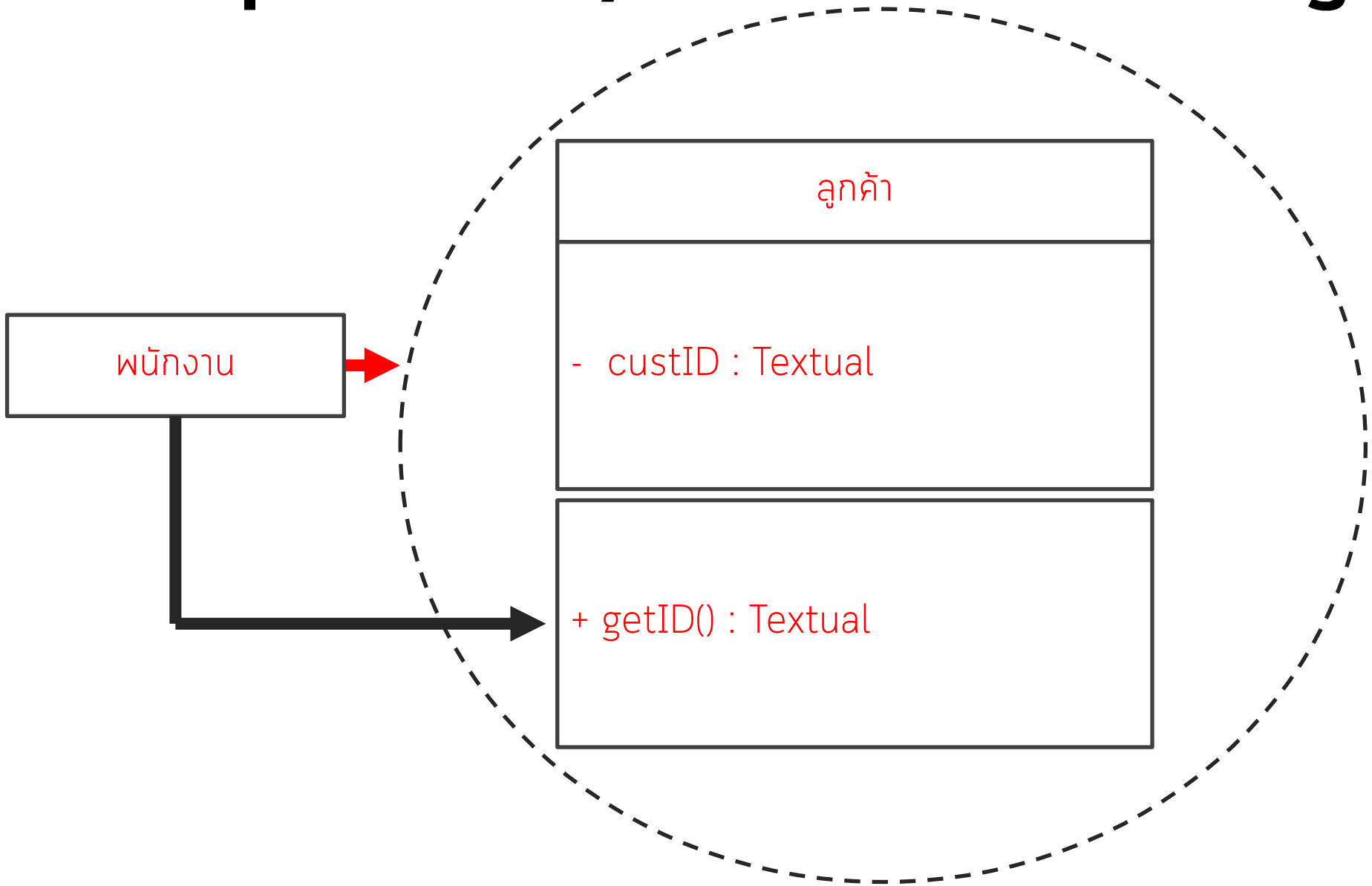
## 4.3.3 Syntax for attributes

**รูปแบบ**
**Visibility attribute_name : type**
**Visibility : public (+) , private(-) , protected(#)**

**ตัวอย่าง**

**+ custID : Textual**
**+ custName : Textual**
**- color : Textual**
**- price : Floating-Point**

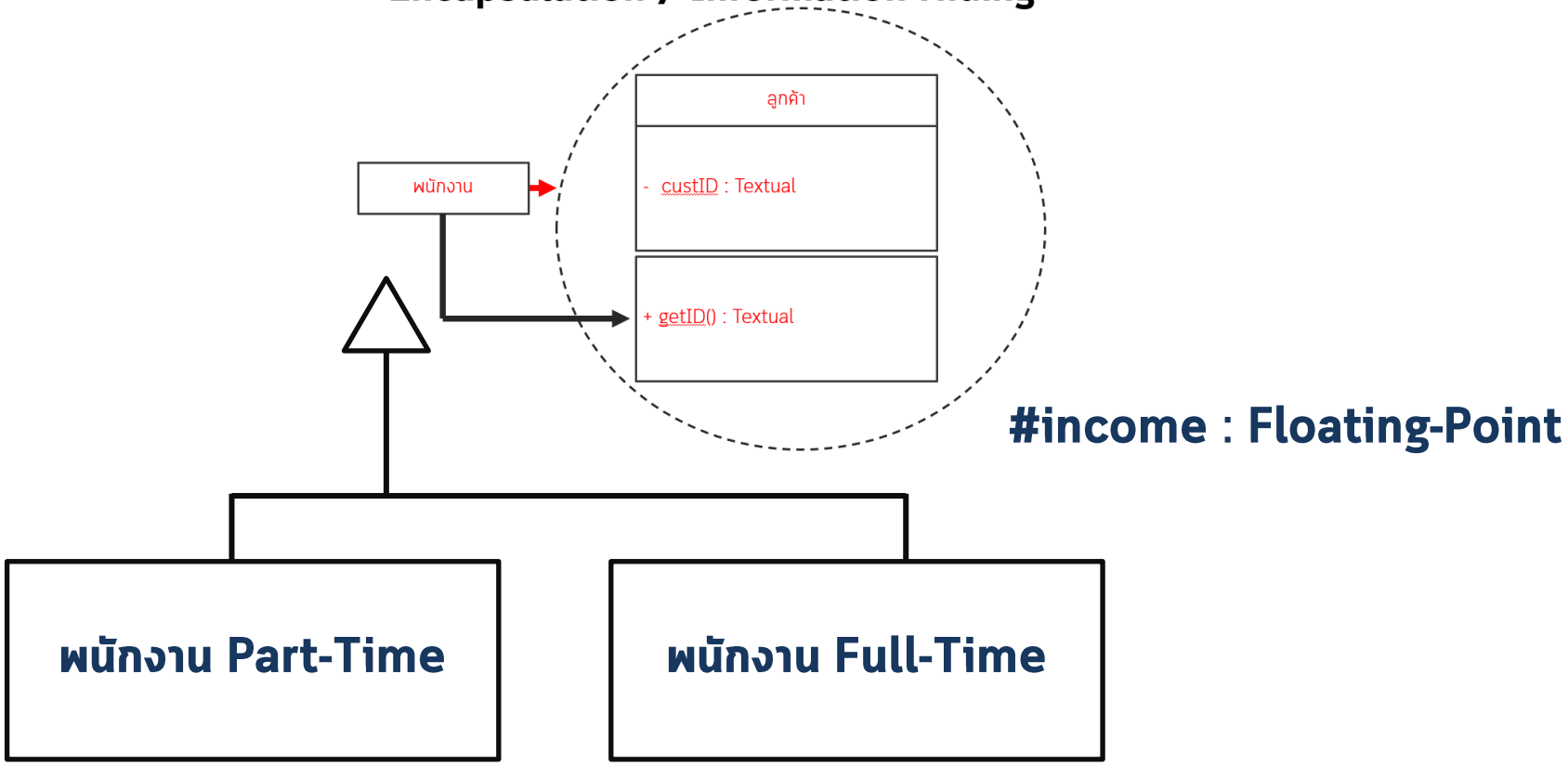
# การห่อหุ้ม / ซ่อนรายละเอียด
# Encapsulation / Information Hiding



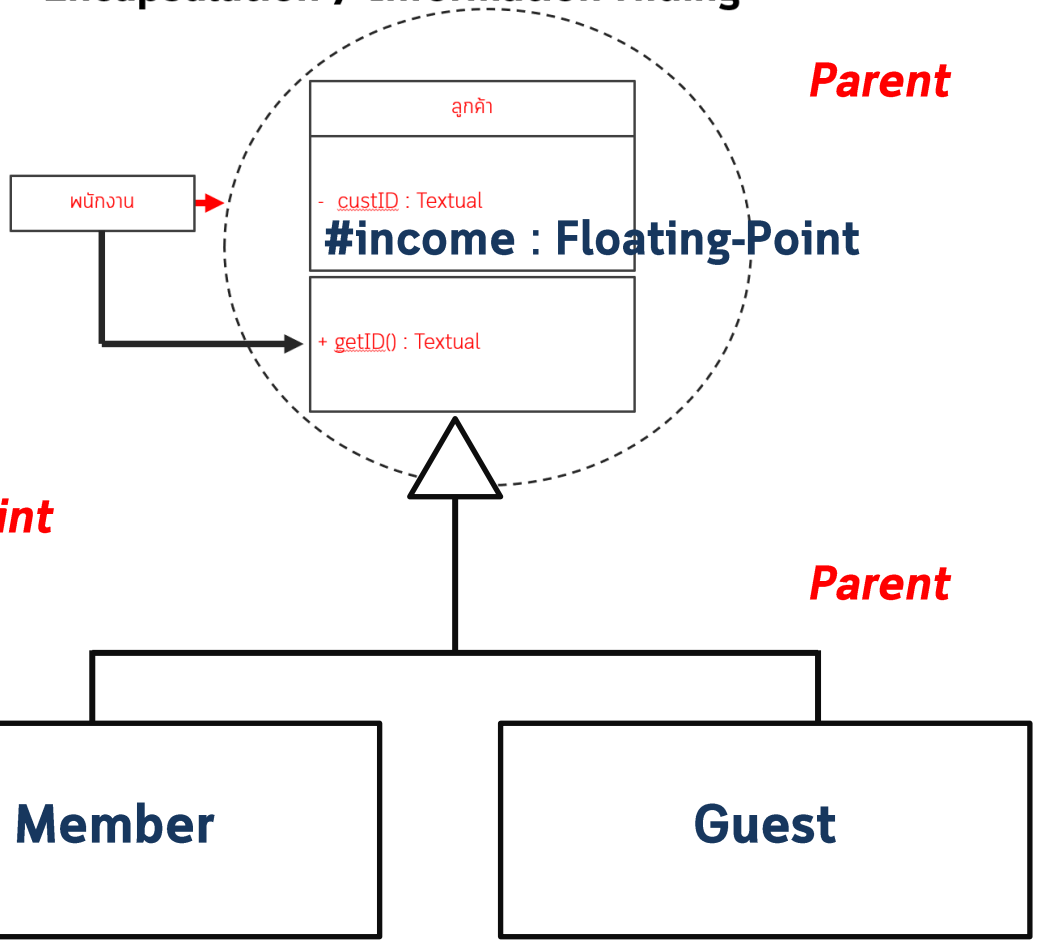
# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.3 Syntax for attributes

**การห่อหุ้ม / ซ่อนรายละเอียด**
**Encapsulation / Information Hiding**



# การห่อหุ้ม / ซ่อนรายละเอียด
# Encapsulation / Information Hiding




***Parent***

***+getIncome() : Floating-Point***

***Parent***

***- id***

***- name***

***- ส่วนลด***

***- id***

***- name***

# 4.3 แผนภาพคลาส(Class Diagrams)

## 4.3.4 Syntax for Operation/function

**รูปแบบ**

**Visibility operation name(parameter : type) : result type**

**ตัวอย่าง**

**+ getCustName() : Textual**
**+ setCustName(n : Textual)**
**+ deposit(amt : Floating-Point)**
**+ withdraw(amt : Floating-Point)**
**+ getBalance() : Floating-Point**
**+ getName() : Textual**
**+ setIncome(income: Floating-Proint)**

# แผนภาพคลาส แสดงโครงสร้างทั้งระบบ ว่าประกอบไปด้วยอะไรบ้าง

ตัวอย่างคลาสเช่น Person, Address, Student, Teacher, และ Course ซึ่งแต่ละคลาสจะมีคุณลักษณะ (attributes) และเมธอด (methods) ดังนี้:

| Class Name | Attributes | Methods |
|---|---|---|
| **Person** | - firstName: String | + getFullName(): String |
| | - lastName: String | + getAddress(): String |
| | - age: int | |
| **Address** | - street: String | + getFullAddress(): String |
| | - city: String | |
| | - zipCode: String | |
| **Student** | - studentID: String | + enrollInCourse(course: Course): void |
| | - coursesEnrolled: List<Course> | + dropCourse(course: Course): void |
| **Teacher** | - teacherID: String | + assignGrade(student: Student, course: Course, grade: Grade): void |
| | - coursesTaught: List<Course> | |
| **Course** | - courseCode: String | + getCourseName(): String |
| | - courseName: String | + getCourseCode(): String |

จากตัวอย่างก่อนหน้านี้:

มีคลาสหลายคลาสเช่น Person, Address, Student, Teacher, และ Course ซึ่งแต่ละคลาสจะมีคุณลักษณะ (attributes) และเมธอด (methods) ดังนี้:

Person: คลาส Person มีคุณลักษณะ firstName, lastName, และ age รวมถึงเมธอด getFullName() และ getAddress() เพื่อเรียกดูข้อมูลเกี่ยวกับชื่อและที่อยู่ของบุคคล.

Address: คลาส Address มีคุณลักษณะ street, city, และ zipCode รวมถึงเมธอด getFullAddress() เพื่อรวบรวมข้อมูลที่อยู่เต็ม.

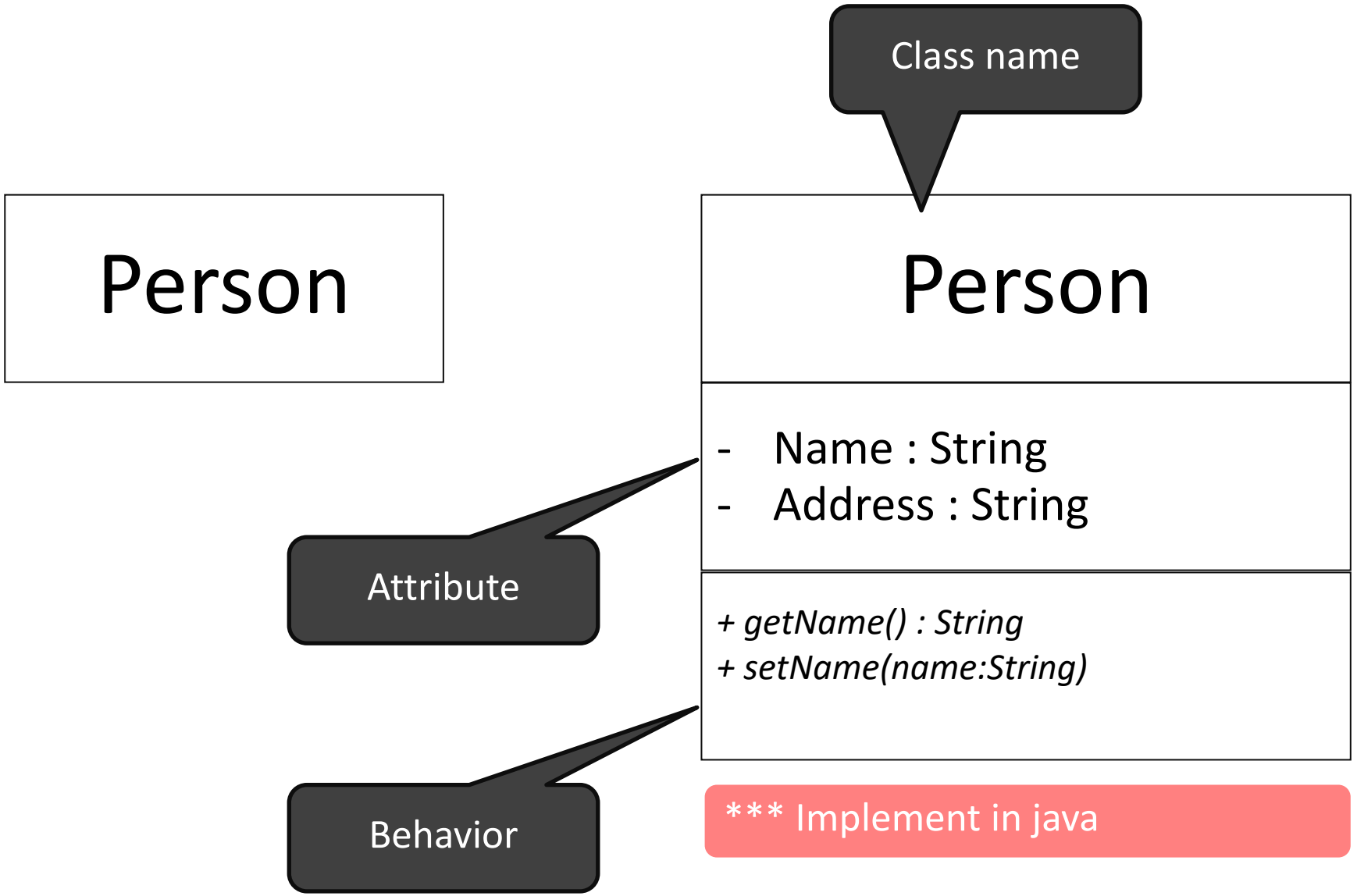
Student: คลาส Student มีคุณลักษณะ studentID และ coursesEnrolled (ลิสต์ของคอร์สที่ลงทะเบียน) รวมถึงเมธอด enrollInCourse() และ dropCourse() เพื่อทำการลงทะเบียนและยกเลิกการลงทะเบียนคอร์ส.

Teacher: คลาส Teacher มีคุณลักษณะ teacherID และ coursesTaught (ลิสต์ของคอร์สที่สอน) รวมถึงเมธอด assignGrade() เพื่อกำหนดเกรดให้กับนักเรียน.

Course: คลาส Course มีคุณลักษณะ courseCode และ courseName รวมถึงเมธอด getCourseName() และ getCourseCode() เพื่อดึงข้อมูลเกี่ยวกับชื่อและรหัสคอร์ส.

แผนภาพคลาสแบบตารางนี้ช่วยในการแสดงความสัมพันธ์และโครงสร้างของคลาสแต่ละคลาสในระบบอย่างชัดเจนและสามารถนำมาใช้ในกระบวนการออกแบบและพัฒนาระบบซอฟต์แวร์ได้อย่างมีประสิทธิภาพ.

| Class Name | Student |
|---|---|
| Attributes | |
| - studentID | String |
| - firstName | String |
| - lastName | String |
| - age | int |
| - gender | String |
| - major | String |
| - credits | int |
| - coursesEnrolled | List<Course> |
| - address | Address |
| - contactInfo | ContactInfo |
| | |
| Methods | |
| + enrollInCourse(course: Course): void | |
| + dropCourse(course: Course): void | |
| + getFullName(): String | |
| + getAddress(): Address | |
| + getAge(): int | |
| + getGender(): String | |
| + getMajor(): String | |
| + getCredits(): int | |
| + getCourseList(): List<Course> | |
| + updateContactInfo(info: ContactInfo): void | |
| + calculateGPA(): double | |
| | |

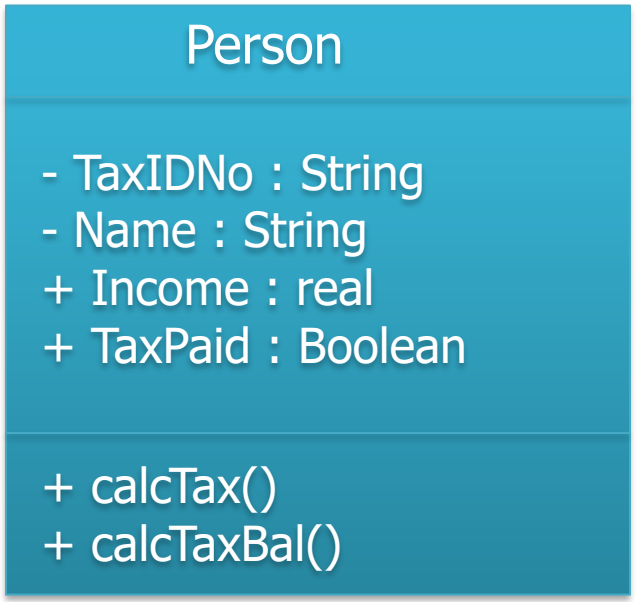

Person
Class name
Person
- Name : String
- Address : String
Attribute
+ getName() : String
+ setName(name:String)
Behavior
*** Implement in java

# อย่าไปเอา Type ในภาษาใดภาษาหนึ่งมาใช้กับ UML class diagram

**Real เป็นชนิดตัวแปรทศนิยม ในภาษาปาสคาลเท่านั้น ภาษาอื่นไม่รู้จัก**
**UML ไม่ผูกติดกับภาษา**

Real → Pascal
float,Float → c, c++, java
double,Double → c, c++, java
Single,Double →VB

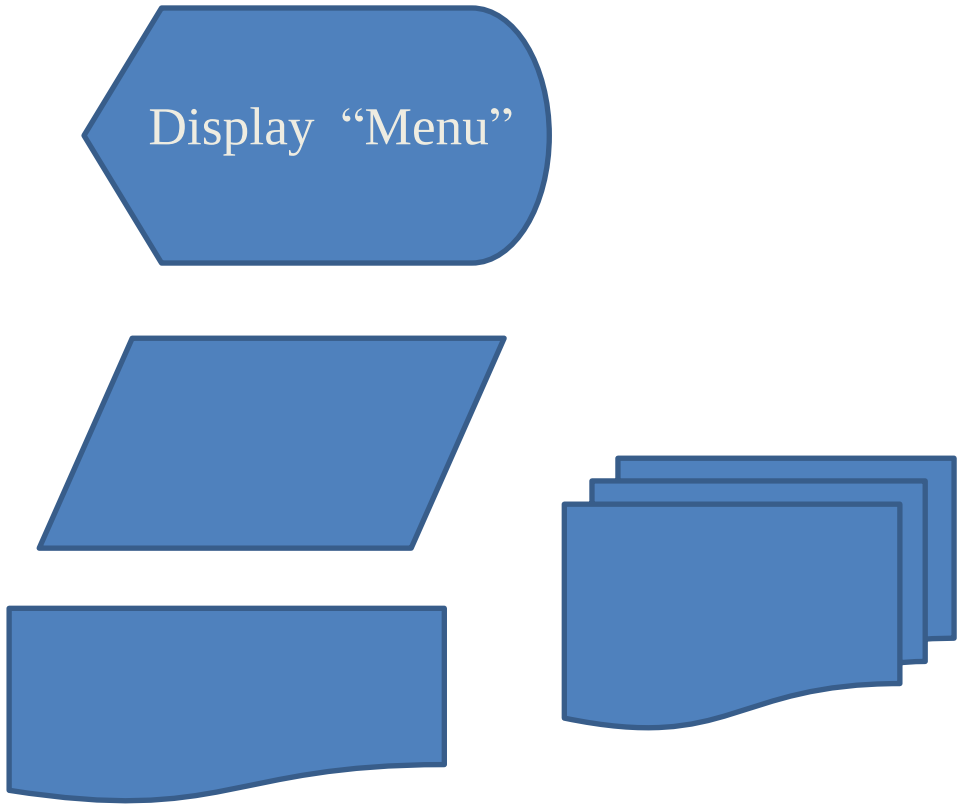
Floating,String,Boolean



# แสดงผลบนหน้าจอ

1. Echo
2. Print
3. Printf
4. Scanf
5. System.out.print
6. System.out.println
7. Write
8. Writeln
9. Display
10. Input
11. Read

**แต่ละคำสั่งคือภาษาอะไร แล้วไว้ทำอะไร**

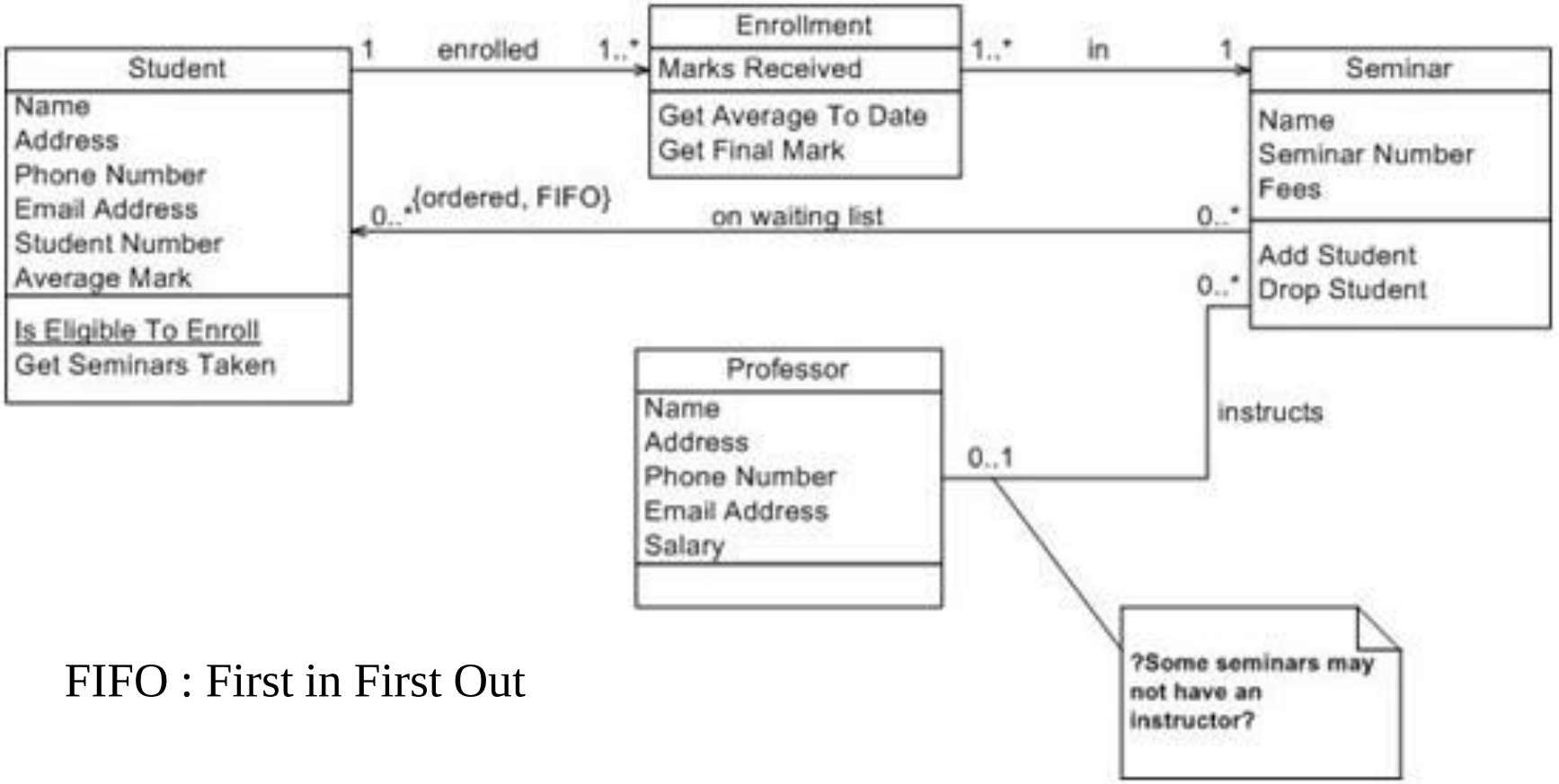


# ในโลกนี้มีภาษาคอมพิวเตอร์ 160 กว่าภาษา

คุณรู้จักแค่ 2-3  ภาษา
เขียนได้แค่ 1 ภาษา
เขียนเป็น 0 ภาษา

จงเขียนโปรแกรม Paint
ด้วยภาษาที่ ถนัด
 สร้างรูป
บันทึกได้
เปิดได้

# ตัวอย่างแผนภาพคลาสของระบบลงทะเบียนเรียน




FIFO : First in First Out

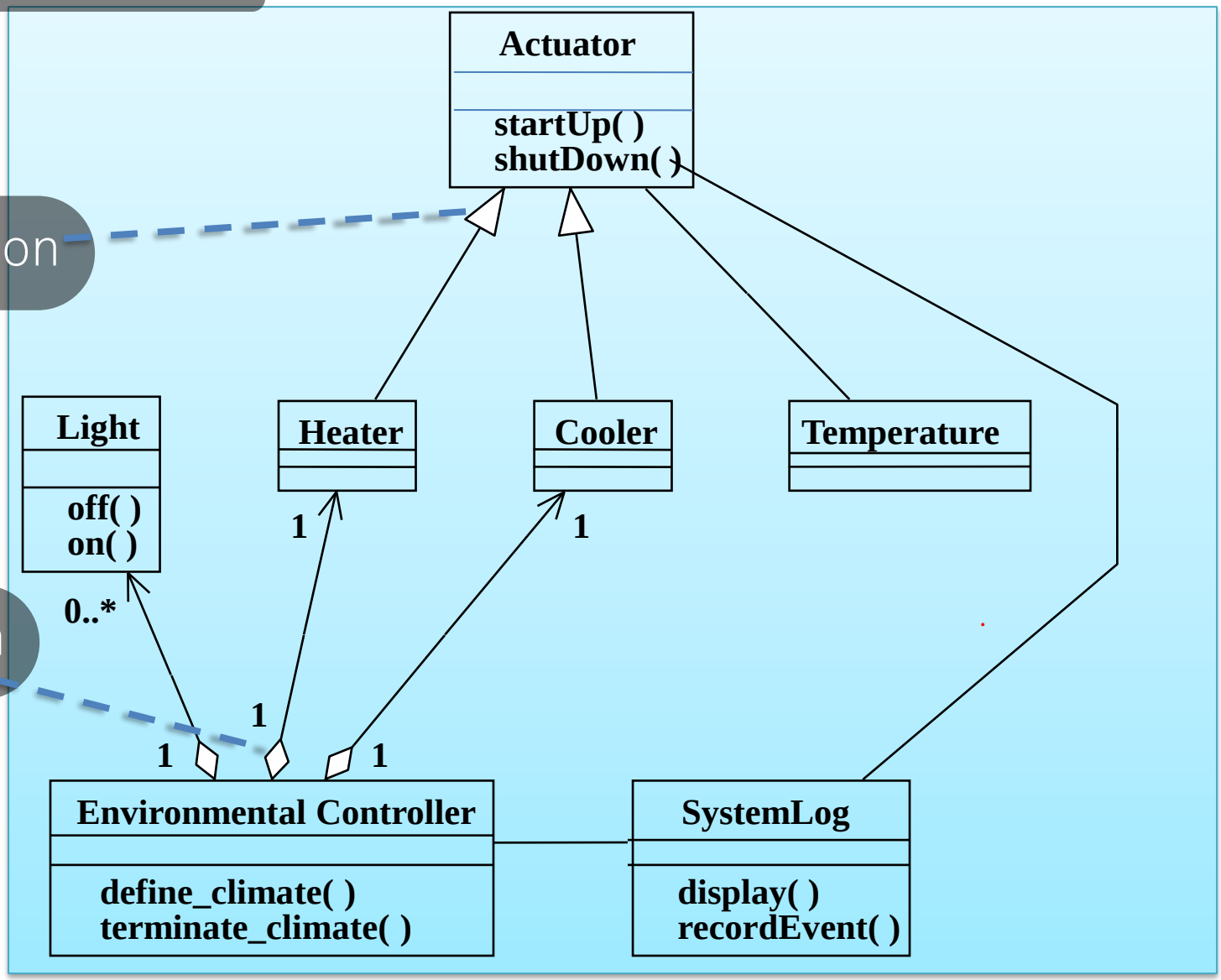
# แผนภาพคลาสนี้แทนโครงสร้างการทำงานของระบบเครื่องปรับอากาศ

# A Class Diagram

Embedded –real time System

generalization

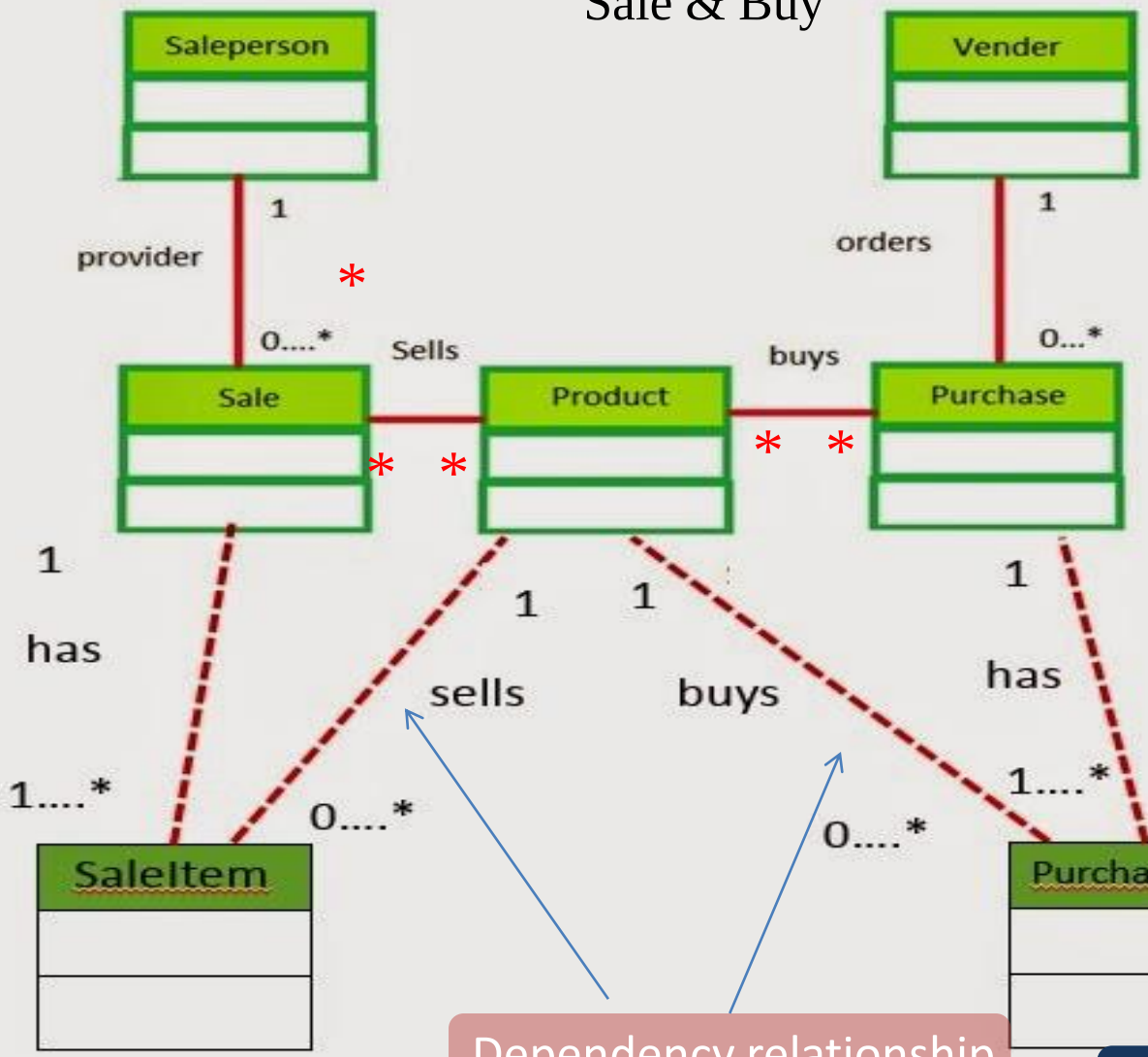
aggregation

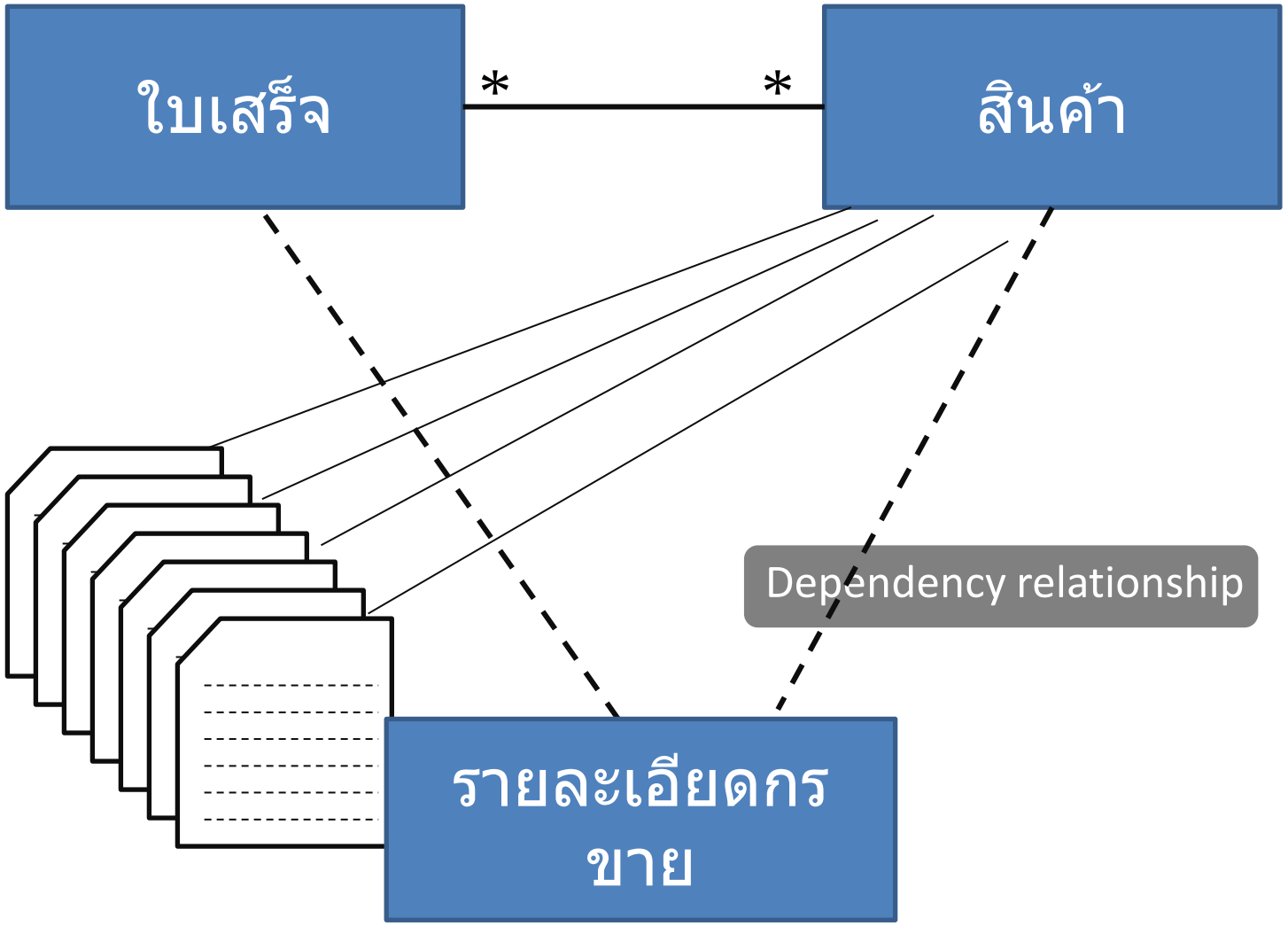


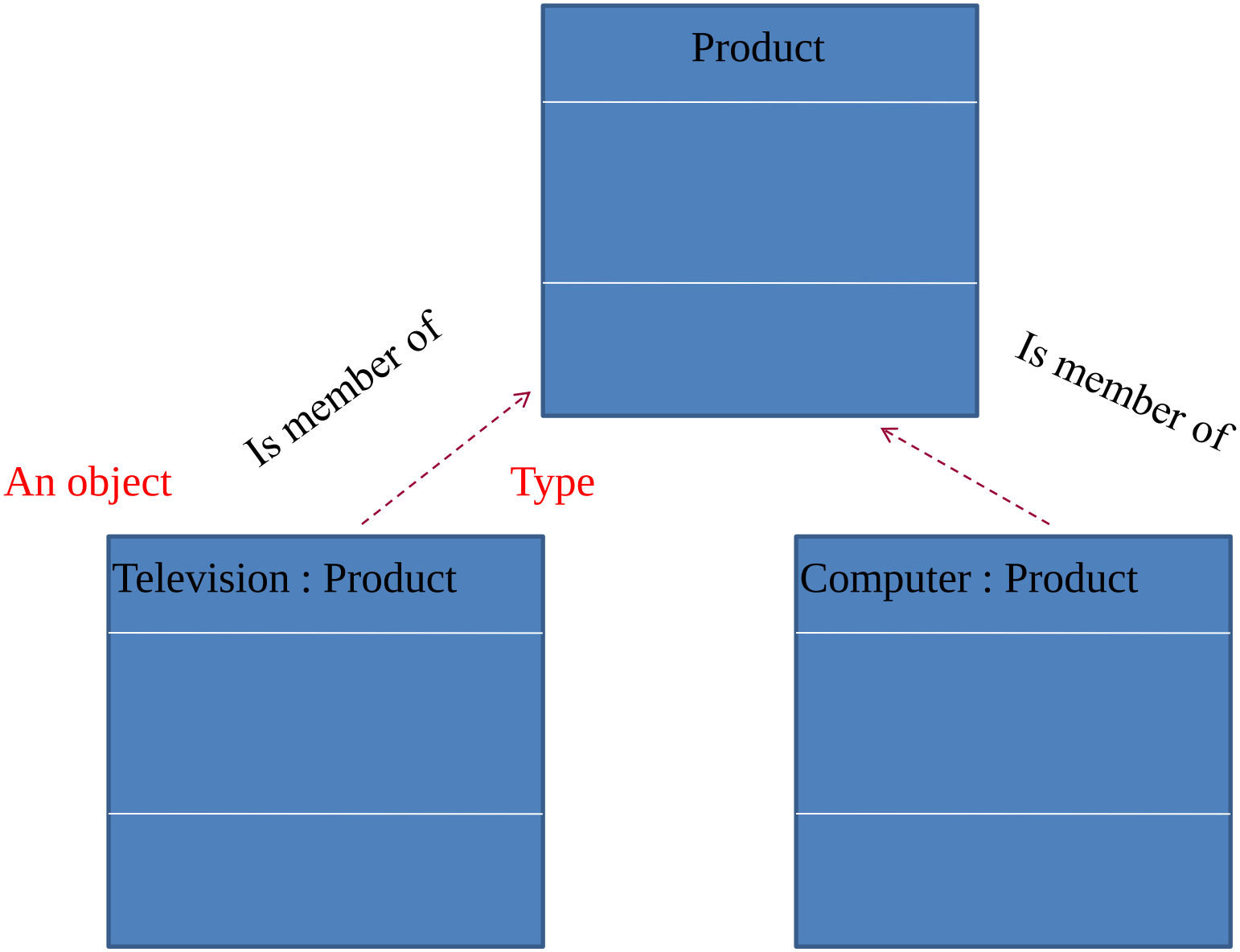
# POS: Purchase Order System

Vender / Dealer

Sale & Buy



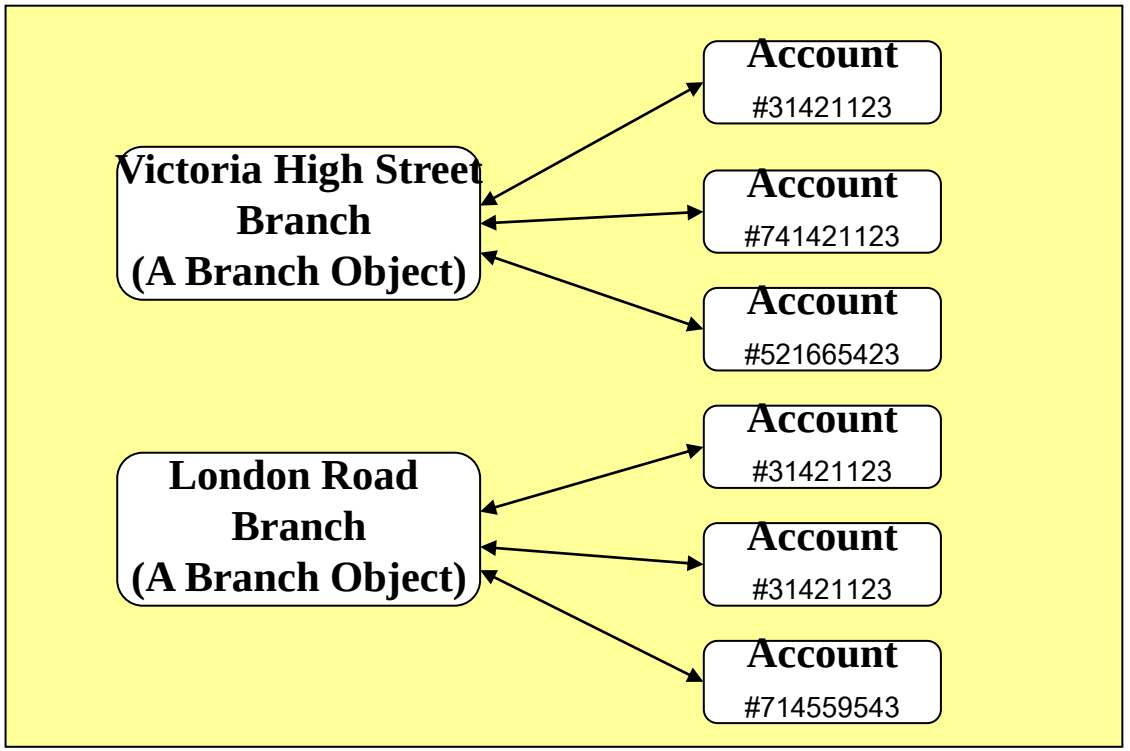

ใบเสร็จ
*
*
สินค้า
Dependency relationship
รายละเอียดการขาย


Product
Is member of
Is member of
An object
Type
Television : Product
Computer : Product

# 4.4 แผนภาพวัตถุ(Object Diagrams

## 4.4.1 ความหมายของแผนภาพวัตถุ

**แผนภาพวัตถุ (Object Diagram)** เป็นหนึ่งในแผนภาพใน Unified Modeling Language (UML) ที่ใช้ในการแสดงอ็อบเจกต์ (objects) และความสัมพันธ์ระหว่างอ็อบเจกต์ในระบบโปรแกรมหรือระบบสารสนเทศ. แผนภาพวัตถุช่วยในการแสดงสถานะปัจจุบันของอ็อบเจกต์และความสัมพันธ์ที่มีอยู่ระหว่างอ็อบเจกต์ในเวลาที่กำหนด โดยทั่วไปแผนภาพวัตถุนิยามคุณลักษณะและค่าของอ็อบเจกต์แต่ละตัวในระบบ



# 4.4 แผนภาพวัตถุ(Object Diagrams

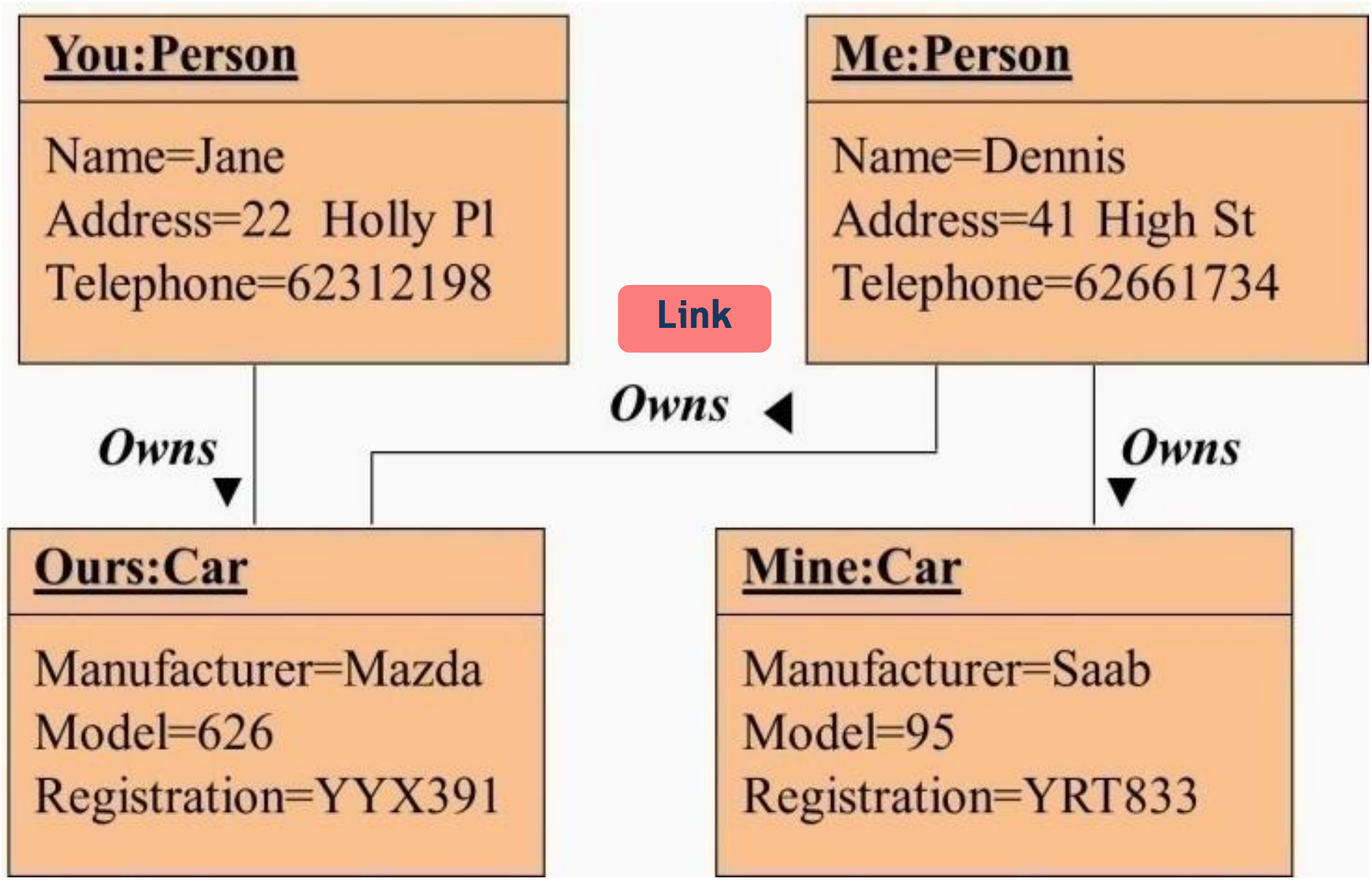
## 4.4.1 ความหมายของแผนภาพวัตถุ

- **Object Diagram**
- **Object Diagram คือหนึ่งในประเภทของ UML (Unified Modeling Language) Diagram ซึ่งใช้แสดงโครงสร้างของวัตถุ (Objects) และความสัมพันธ์ของวัตถุเหล่านั้นในช่วงเวลาหนึ่ง (snapshot of instances at a specific point in time)**

**องค์ประกอบของ Object Diagram**

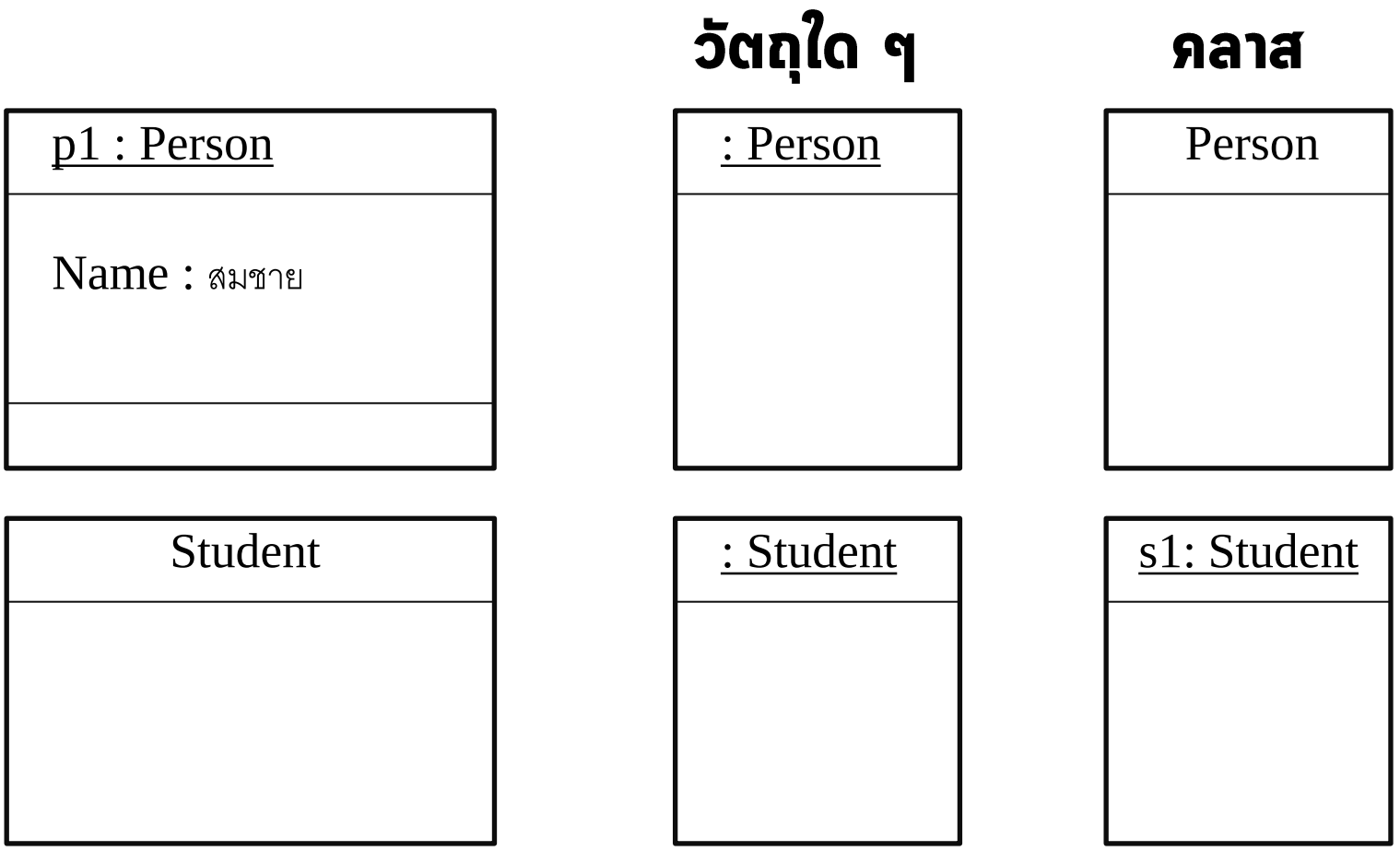
1. **Object (วัตถุ)**
    1. **อินสแตนซ์ของคลาสในระบบ**
    2. **แสดงชื่อและค่าของคุณสมบัติ (Attributes)**
2. **Links (ความสัมพันธ์)**
    1. **แสดงความเชื่อมโยงระหว่างวัตถุ**
    2. **เป็นเหมือนตัวอย่างการใช้งาน Association ใน Class Diagram**

# Object diagram



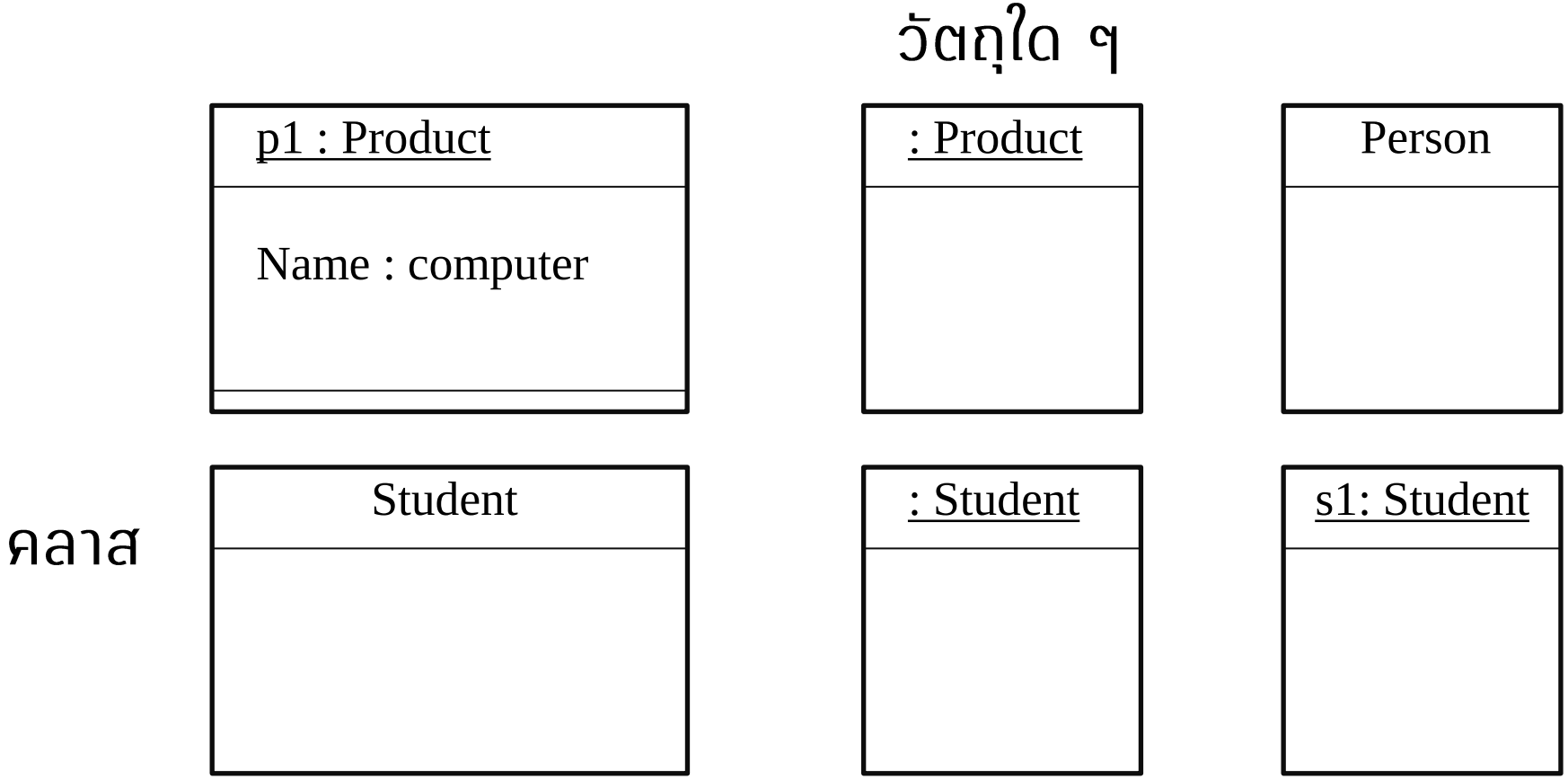
# 4.4 แผนภาพวัตถุ(Object Diagrams

## 4.4.2 วัตถุใด ๆ



# 4.4 แผนภาพวัตถุ(Object Diagrams)

## 4.4.2 วัตถุใด ๆ



# 4.4 แผนภาพวัตถุ(Object Diagrams)

## 4.4.3 หลักการเขียนแผนภาพวัตถุ

นำไปเขียนเป็น class diagram *เอกสิทธิ์*

- ลูกค้าสมัครบัตรเคดิต ( Association )
- นางรุ่งทิพย์ สมัครบัตรเคดิต กรุงศรี plattinum ...ของ ธนาคาร กรุงศรี Scenario
- นายสมชาย สมัครบัตรเครดิต ของ ธ. กสิกรไทย Scenario

นำไปเขียนเป็น objects diagram *รุ่งทิพย์*

# 4.4 แผนภาพวัตถุ(Object Diagrams)

## 4.4.3 หลักการเขียนแผนภาพวัตถุ

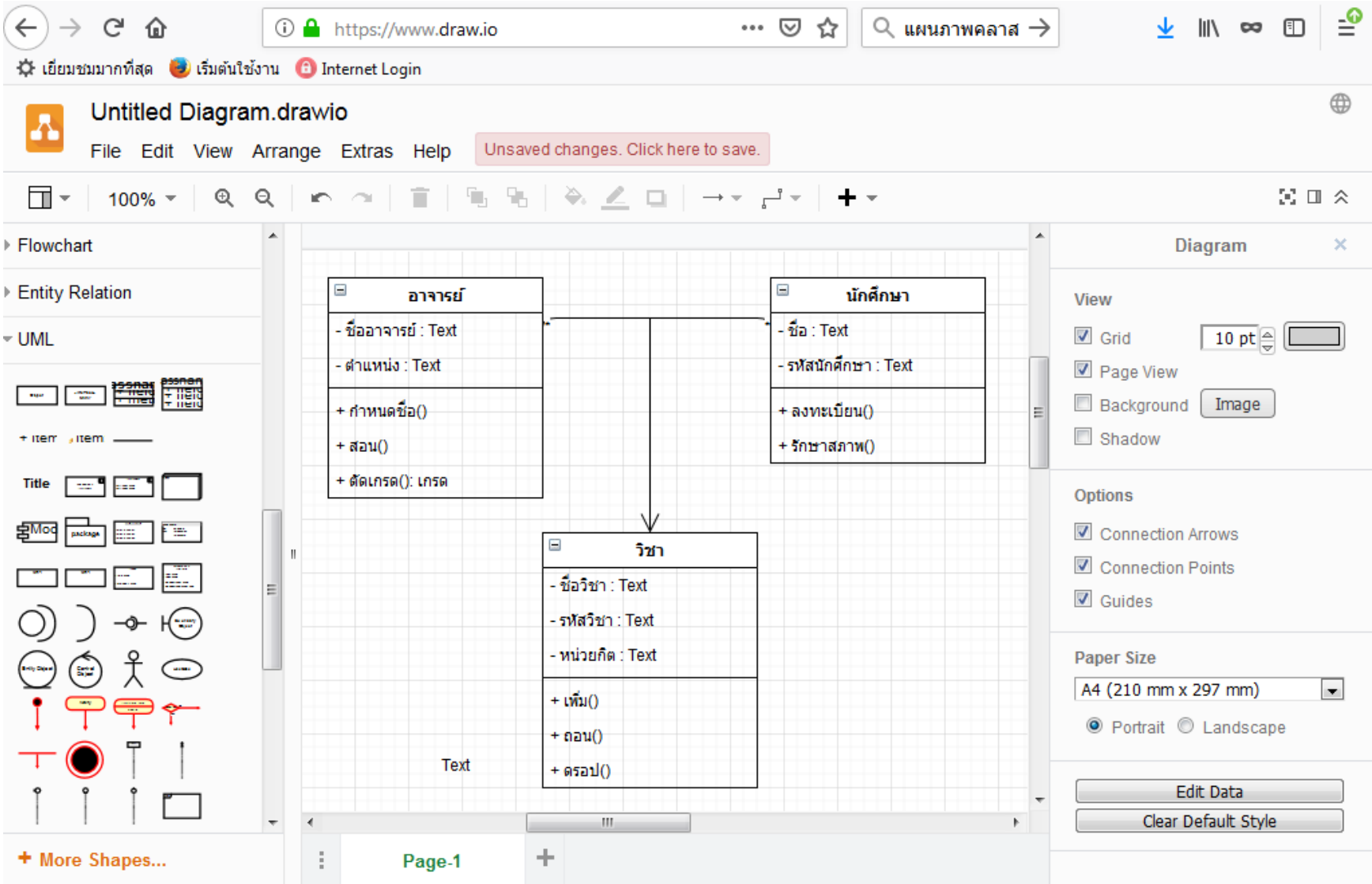
นำไปเขียนเป็น Class diagram

- อาจารย์ สอน นักศึกษา ในรายวิชา OOAD ( Association )
- อาจารย์นัฐพงศ์ สอน นักศึกษาไอที 63/1 ในรายวิชา OOAD …Scenario

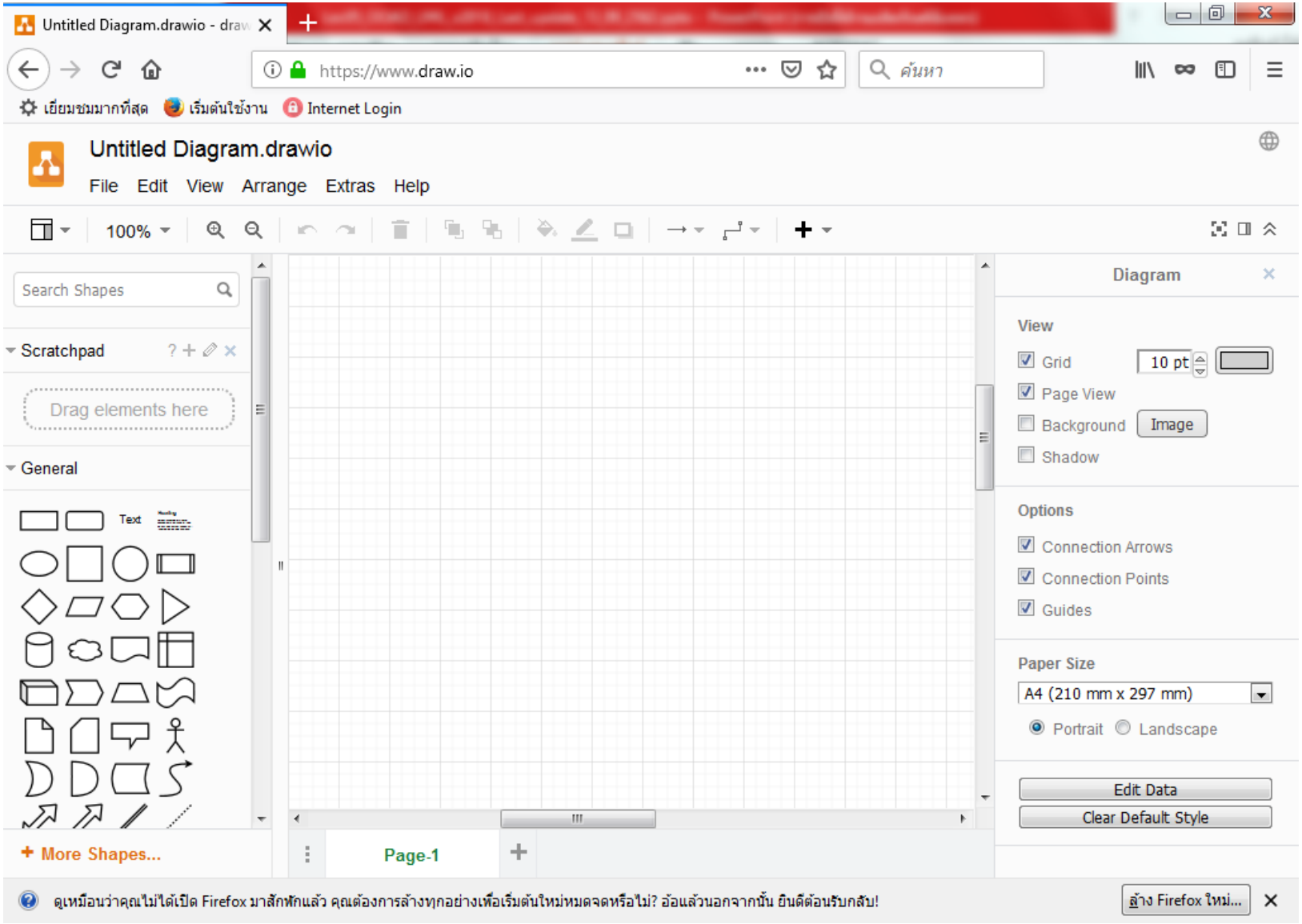
นำไปเขียนเป็น Objects diagram

ให้ใช้ CASE Tool : draw.io

# ให้ใช้ CASE Tool : draw.io



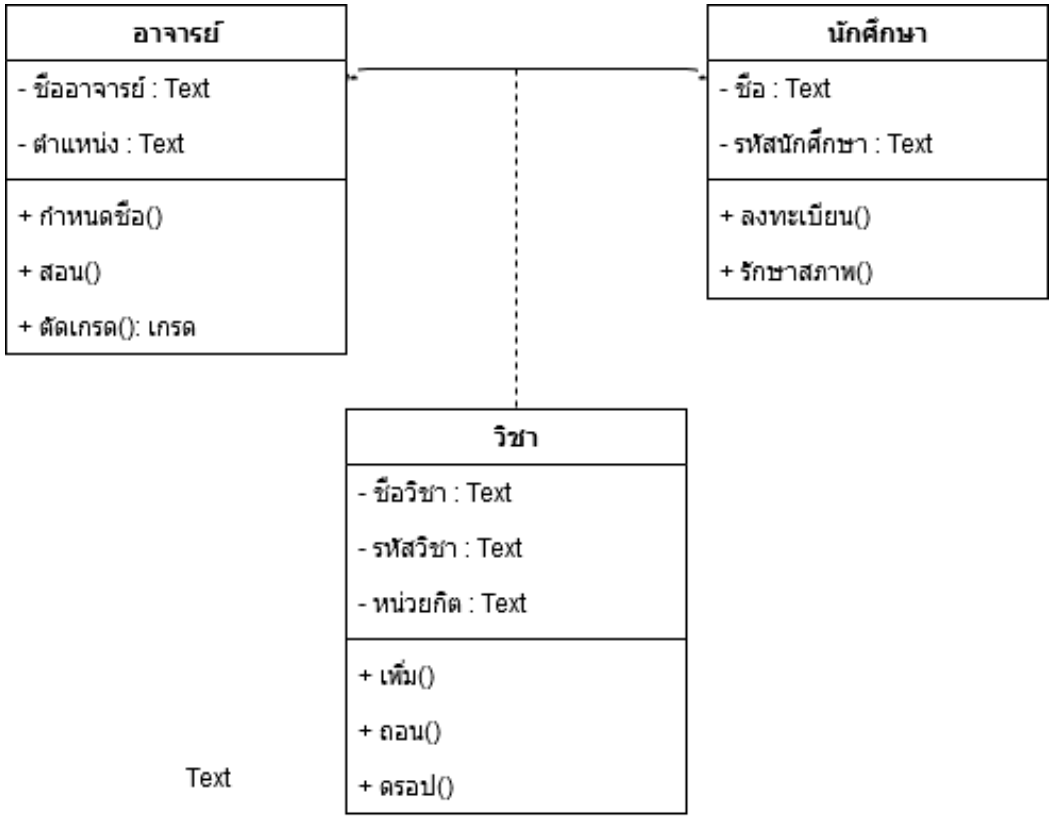
# CASE Tool : draw.io



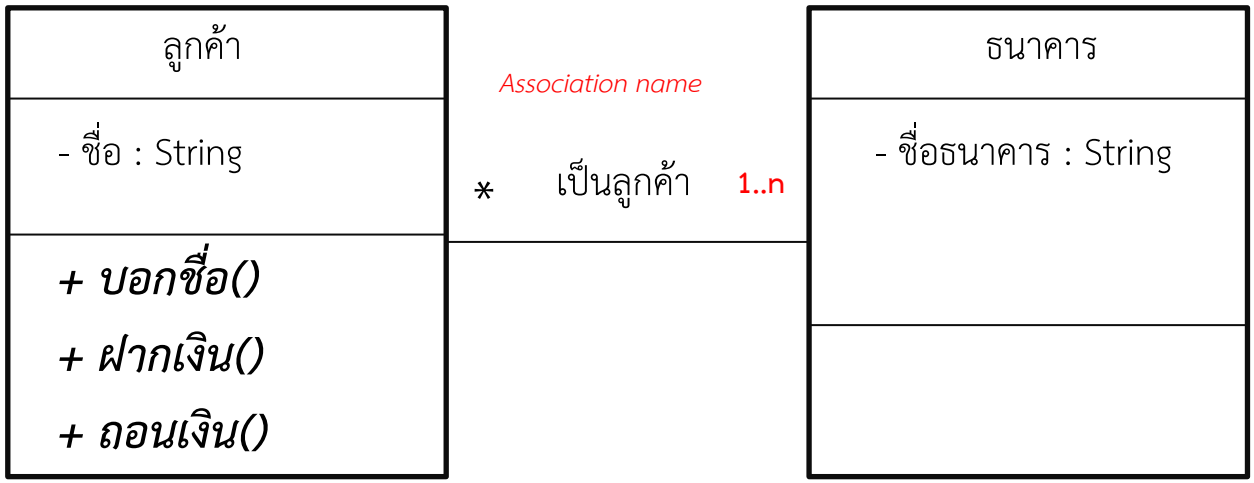
# 4.4 แผนภาพวัตถุ(Object Diagrams)

## 4.4.3 หลักการเขียนแผนภาพวัตถุ

**จงเขียนแผนภาพคลาสที่ถูกต้องและสมบูรณ์**



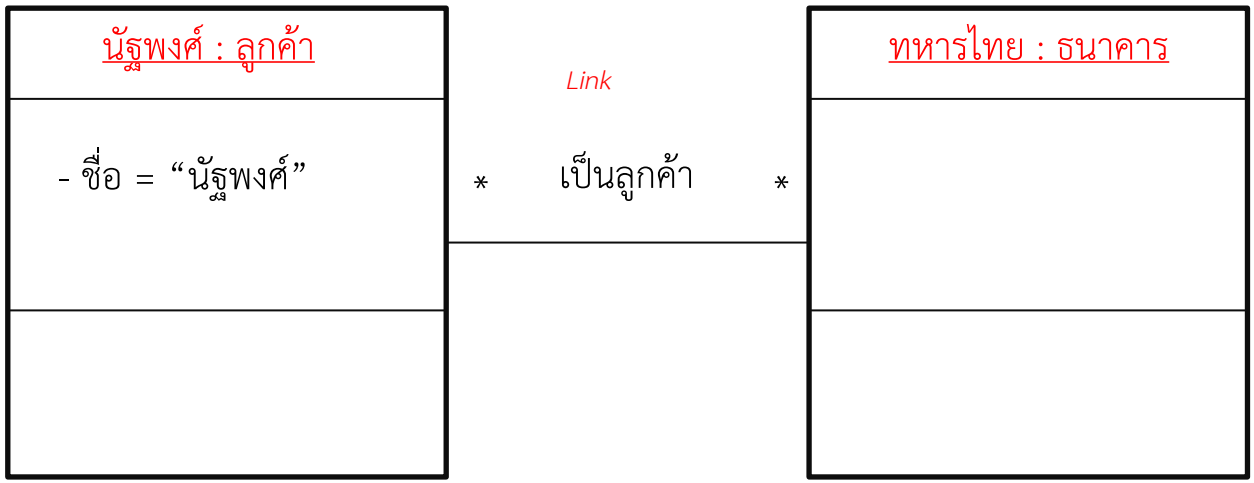
Class diagram




*

0..n

Object diagram



นำไปเขียนเป็น class diagram *นายนพณัฐ*

- ลูกค้าไปถอนเงินจากธนาคาร Association
- อาจารย์นัฐพงศ์ ถอนเงิน 500 บาท จากตู้ธนาคารกสิกร …Senario

นำไปเขียนเป็น objects diagram *นางสาววชิราภรณ์*

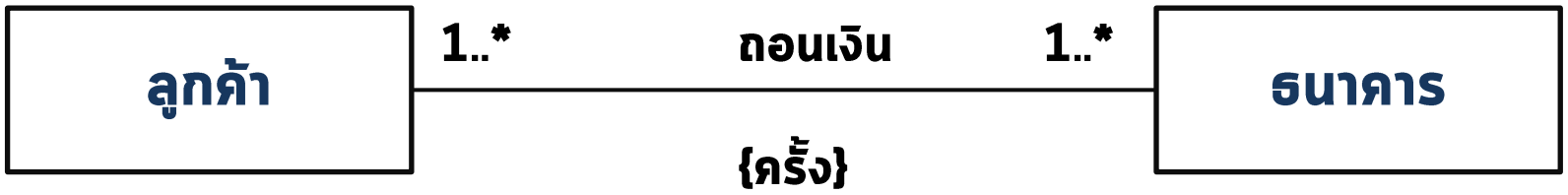
ลูกค้าไป<u>เป็นลูกค้า</u>จากธนาคาร Association

# 4.4 แผนภาพวัตถุ(Object Diagrams)

## 4.4.3 หลักการเขียนแผนภาพวัตถุ

**ลูกค้าไปถอนเงินจากธนาคาร Association {นำไปเขียนเป็นแผนภาพคลาส class diagram}**

***Association name***



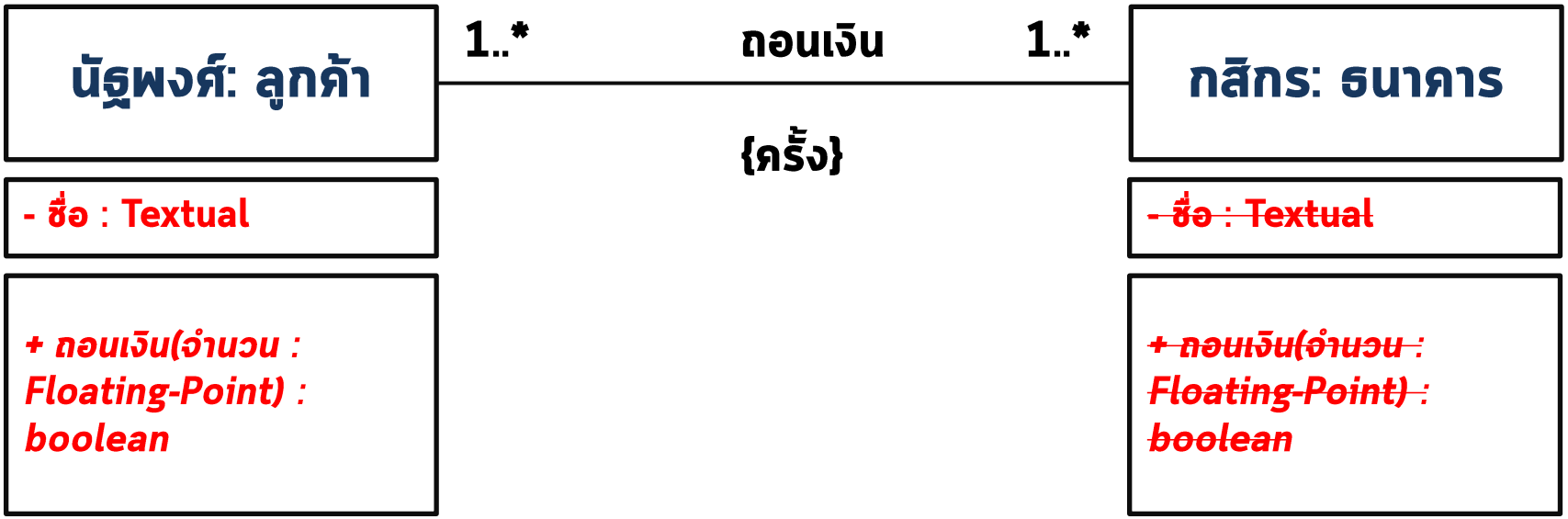

# 4.4 แผนภาพวัตถุ(Object Diagrams)

## 4.4.3 หลักการเขียนแผนภาพวัตถุ

**อาจารย์นัฐพงศ์ ถอนเงิน 500 บาท จากตู้ธนาคารกสิกร ...Senario**

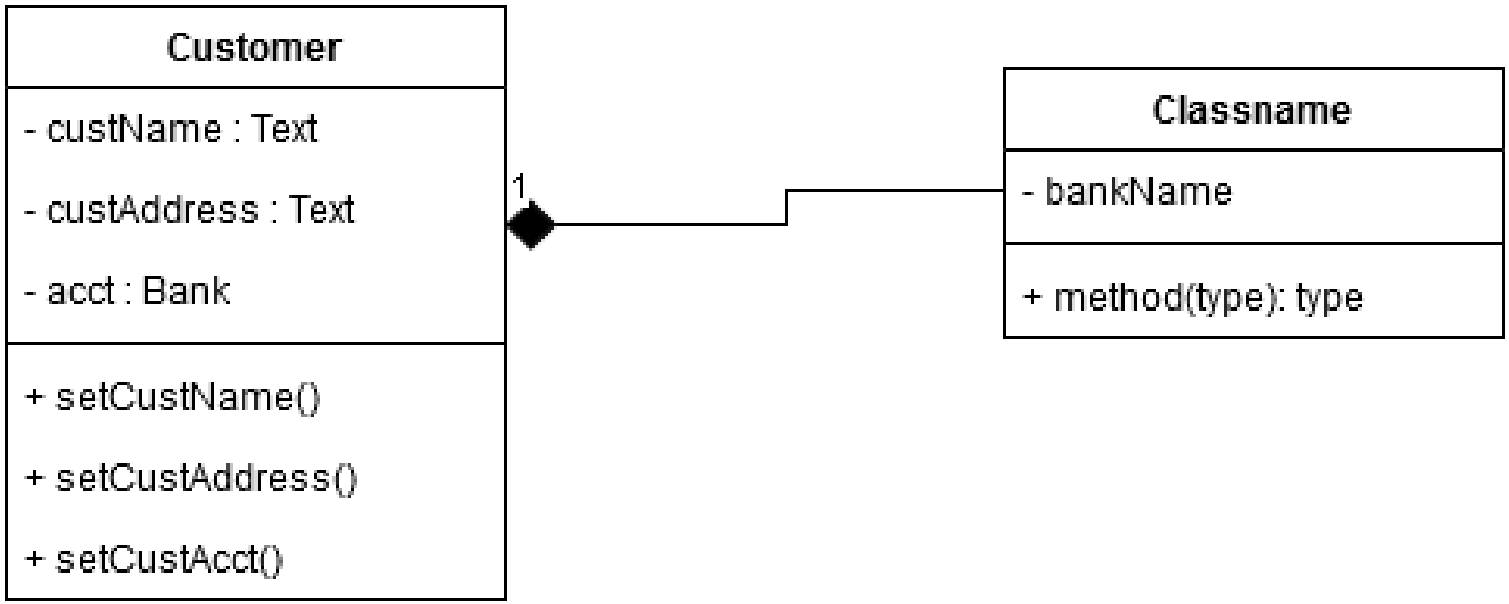
**{นำไปเขียนเป็นแผนภาพวัตถุ object diagram}**

***Link***



# 4.4 แผนภาพวัตถุ(Object Diagrams)

## 4.4.3 หลักการเขียนแผนภาพวัตถุ



# 4.5 แผนภาพองค์ประกอบ (Component Diagram

## 4.5.1 ความหมายของแผนภาพองค์ประกอบ

- Component Diagram เป็นไดอะแกรมที่ใช้อธิบายถึงซอฟต์แวร์ต่าง ๆ ที่เป็นองค์ประกอบ (Component) ของระบบนั้น ๆ ซึ่งแสดงโครงสร้างทางกายภาพของ(physical view)
- เป็นแผนภาพสำหรับ ขั้นตอน การออกแบบระบบ
- Elaboration Phase และ Construction Phase

เป็นการอธิบายว่าระบบนั้น ๆ จะประกอบไปด้วยซอฟต์แวร์ ไฟล์ต้นฉบับ (source code .java , .vb , .cpp , .pas) โปรแกรมประมวลผล(.exe) ไฟล์ฐานข้อมูล (database .mdb) ไฟล์ไลบลารี (.dll) และรายงาน (.rpt) อะไรบ้าง หรือ มีอะไรเป็นองค์ประกอบบ้าง

# 4.5 แผนภาพองค์ประกอบ (Component Diagram

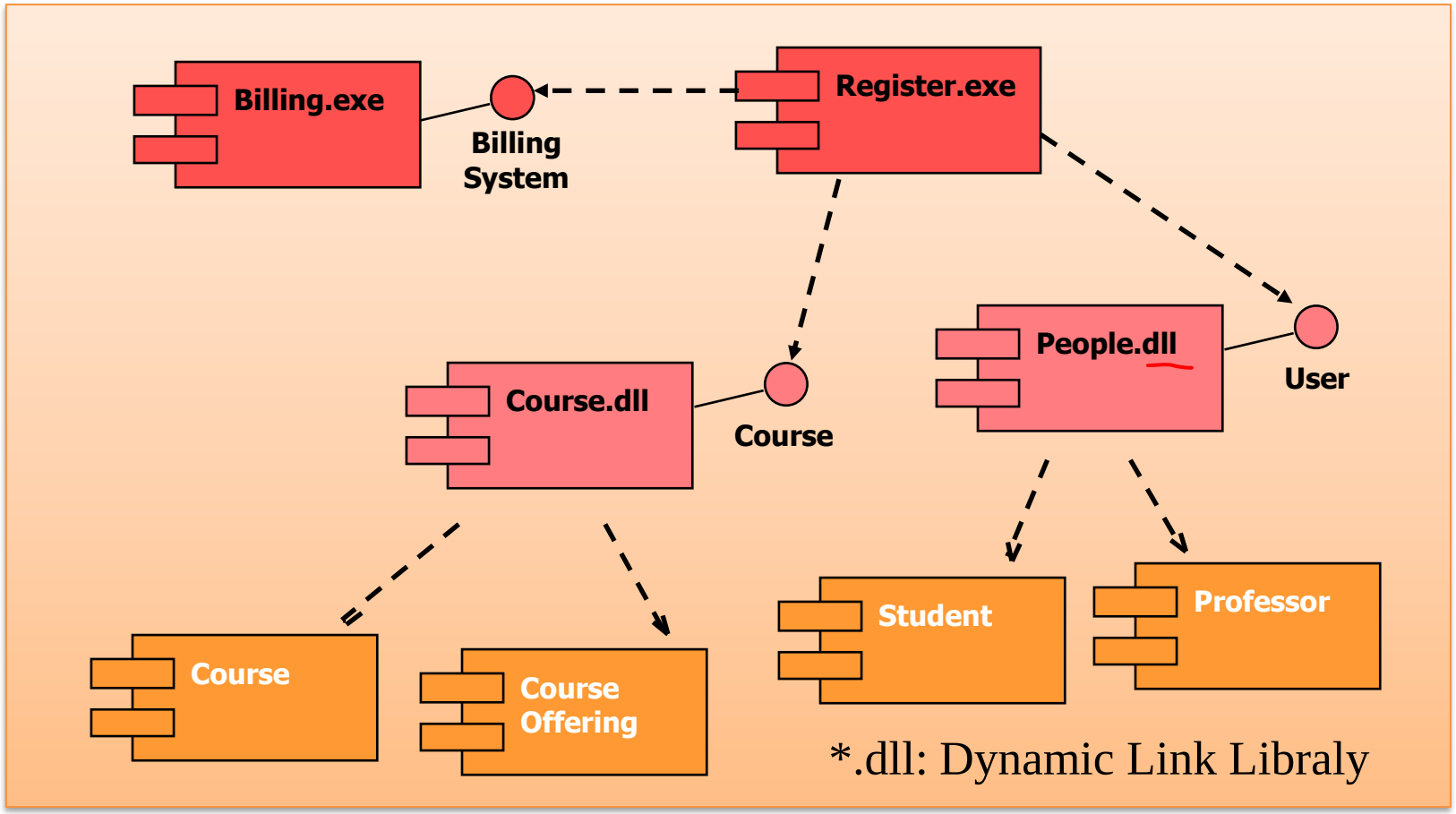
## 4.5.1 ความหมายของแผนภาพองค์ประกอบ

- **แผนภาพคอมโพเนนต์ (Component Diagram) ใน UML ใช้ในการแสดงคอมโพเนนต์และความสัมพันธ์ระหว่างคอมโพเนนต์ในระบบซอฟต์แวร์หรือระบบสารสนเทศ โดยทั่วไปแผนภาพคอมโพเนนต์นี้แสดงองค์ประกอบที่มีความอิสระในการติดตั้งและประกอบรวมกันเพื่อสร้างระบบที่ใช้งานได้**

# A Component Diagram

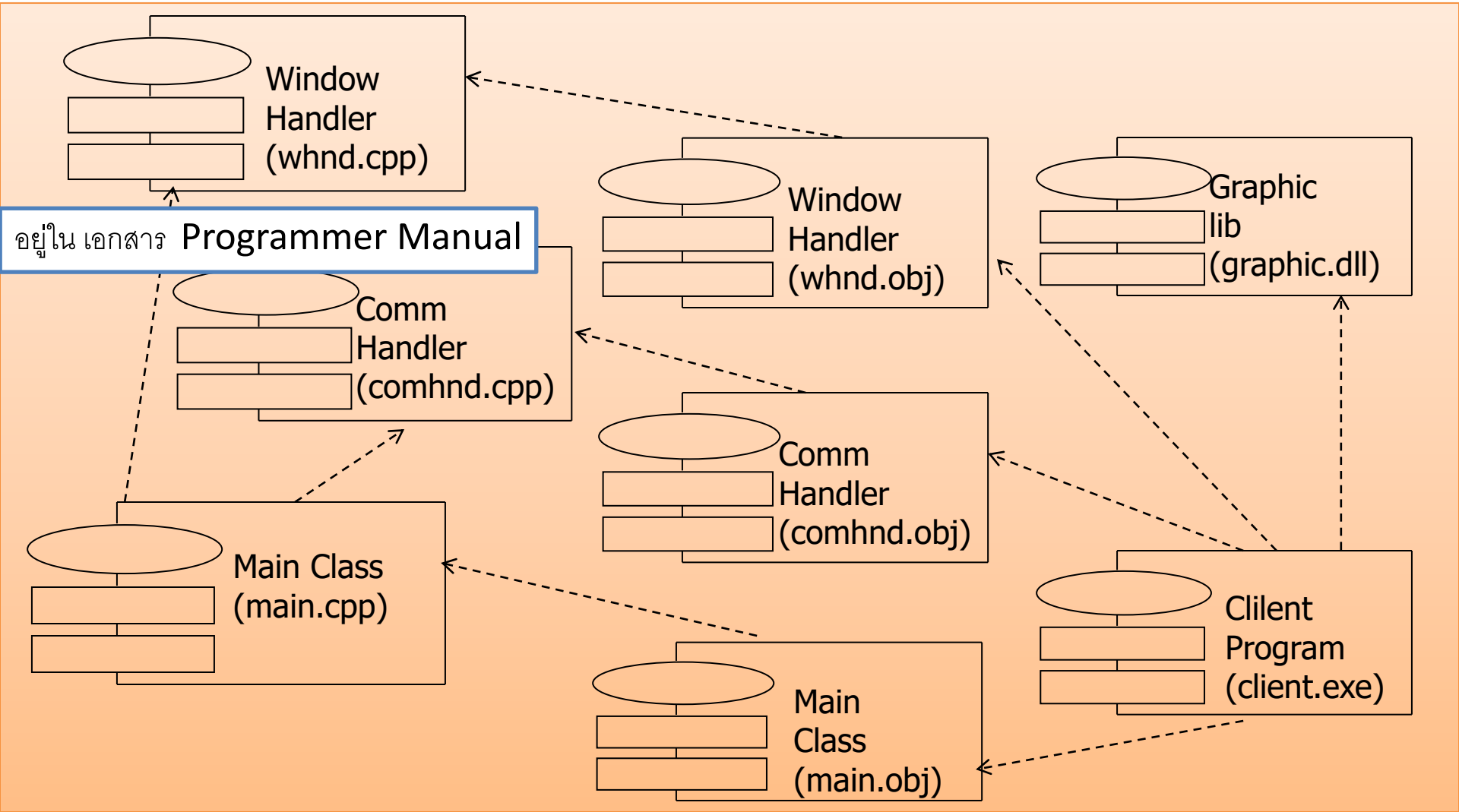
อยู่ใน เอกสาร Programmer Manual
เพื่อให้ คนที่จะพัฒนาต่อเข้าใจองค์ประกอบของระบบไฟล์ ที่คุณพัฒนา

## Billing เป็นส่วนหนึ่งการลงทะเบียนเรียน



# A Component Diagram

component diagram แสดงโครงสร้างทางกายภาพของ(physical view) code ในเทอมของ code components (component อาจเป็น source code component, a binary component, or an executable component)



# 4.5 แผนภาพองค์ประกอบ (Component Diagram)

## 4.5.2 ตัวอย่างแผนภาพคอมโพเนนต์

**Component Diagram Example**

**ในตัวอย่างนี้:**

**คอมโพเนนต์ "Web Server" แสดงถึงเซิร์ฟเวอร์เว็บที่ให้บริการแก่ผู้ใช้ ซึ่งอาจประกอบด้วยซอฟต์แวร์หลายชิ้นส่วน เช่น เซิร์ฟเล็ต, เพจจองสินค้า, และฐานข้อมูล**

**คอมโพเนนต์ "Mobile App" แสดงถึงแอปพลิเคชันมือถือที่ใช้งานบนโทรศัพท์มือถือของผู้ใช้.**

**คอมโพเนนต์ "Database" แสดงถึงฐานข้อมูลที่ใช้เก็บข้อมูลของระบบ เช่น ข้อมูลลูกค้า, ข้อมูลสินค้า, และอื่น ๆ.**

**ลูกศูนย์การชำระเงินและการส่งข้อมูลถูกเชื่อมต่อกับเซิร์ฟเวอร์เว็บผ่านการเรียกใช้บริการ (Service) ที่ให้บริการโดยเซิร์ฟเวอร์เว็บ.**

**คอมโพเนนต์ "Mobile App" สามารถเรียกใช้บริการจากเซิร์ฟเวอร์เว็บและอ่านข้อมูลจากฐานข้อมูลเพื่อแสดงผลให้กับผู้ใช้.**

**แผนภาพคอมโพเนนต์ช่วยในการแสดงโครงสร้างและการทำงานของระบบซอฟต์แวร์หรือระบบสารสนเทศโดยเน้นคอมโพเนนต์แยกต่างหากและการสื่อสารระหว่างพวกเขาในระบบ**

# 4.5 แผนภาพองค์ประกอบ (Component Diagram)

## 4.5.2 ตัวอย่างแผนภาพคอมโพเนนต์

**Physical Model ทางกายภาพ**
**Logical Model ทางลอจิก**

# 4.6 แผนภาพการปรับใช้ (Deployment Diagram)

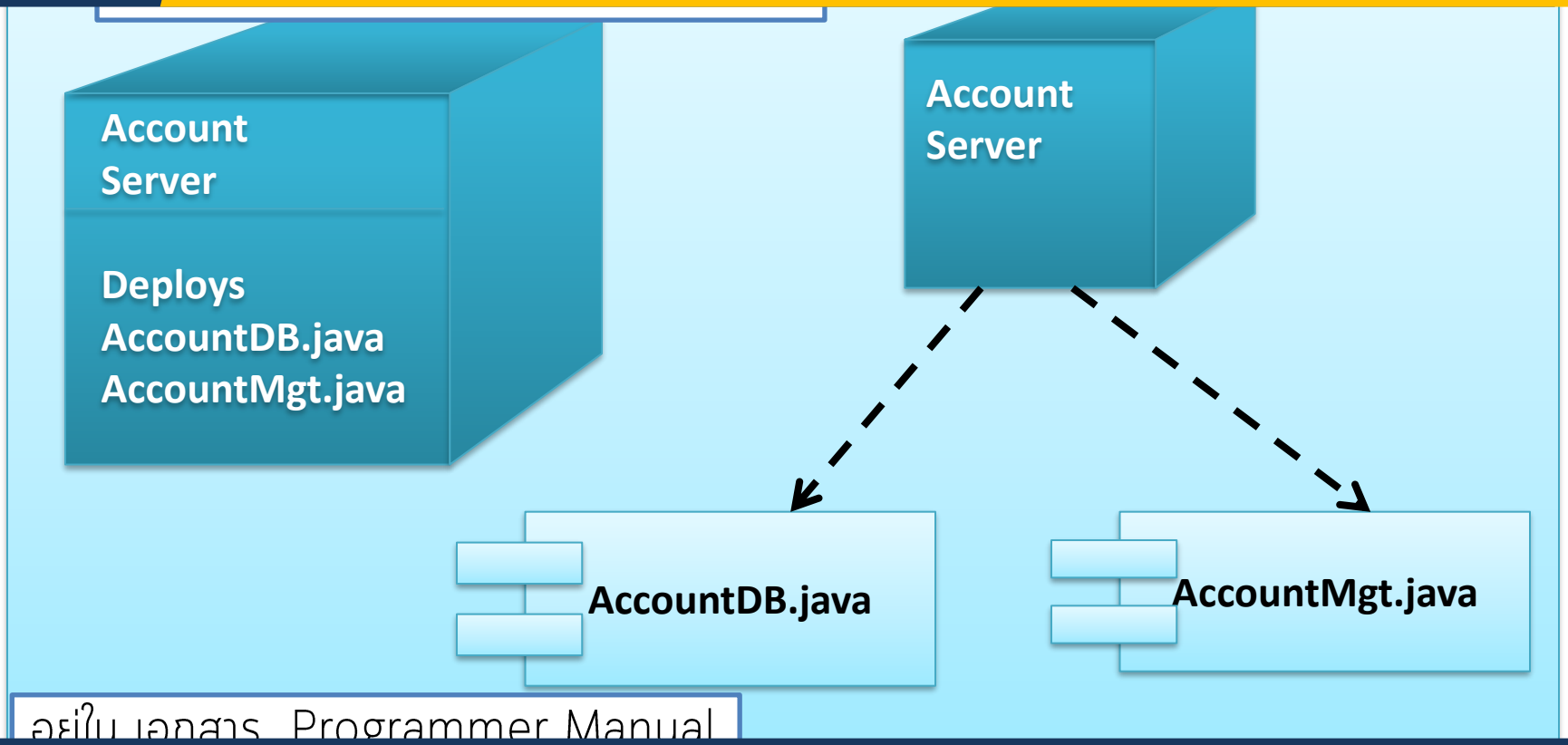
## 4.6.1 ความหมายของแผนภาพการปรับใช้

- **Deployment Diagram ใช้สำหรับแสดงสถาปัตยกรรมของระบบในลักษณะที่เป็น Physical Architecture คือแสดงว่ามีคอมพิวเตอร์และอุปกรณ์อะไรบ้าง และระบุเครือข่ายที่ต้องการใช้ในระบบ**
- **เวลาที่จะนำระบบไปติดตั้ง ตัว deployment diagram จะอยู่ในคู่มือ Technical Manual เป็นคู่มือให้ผู้ติดตั้งระบบเข้าใจการติดตั้งและองค์กอบต่าง ๆ**
- **โดยใช้รูปลูกบาศก์แทน โดย 1 ลูกบาศก์จะแทน 1 โหนด และแต่ละโหนดก็จะมี component ที่เป็นองค์ประกอบของโหนดนั้น**

**แสดงโครงสร้างทางกายภาพของ(physical view)**
**เป็นแผนภาพที่ใช้ในขั้นตอนการออกแบบระบบ Elaboration Phase Construction Phase เป็นหน้าที่ของ System Architecture Engineering**

# 4.6 แผนภาพการปรับใช้ (Deployment Diagram)

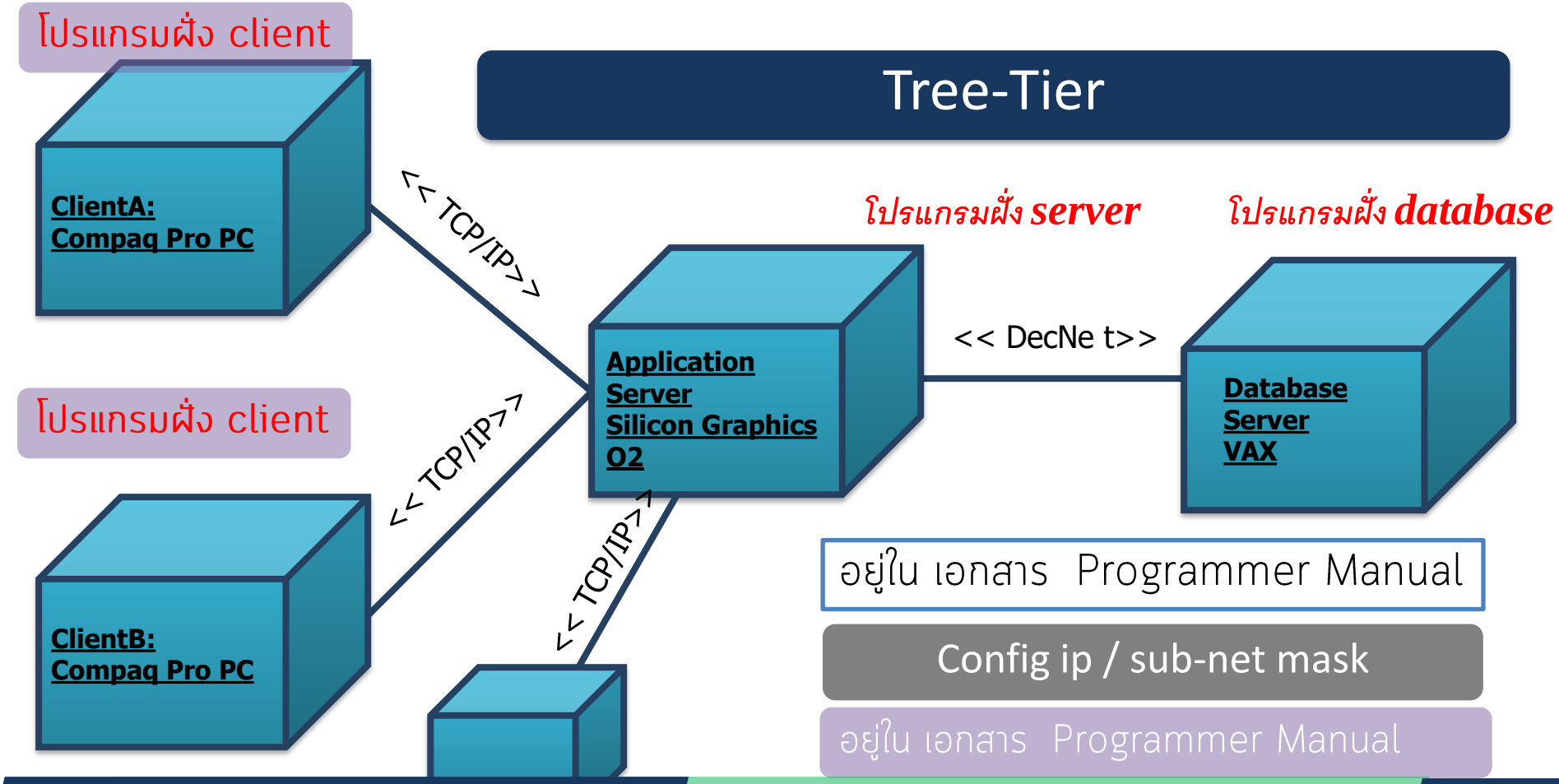
## 4.6.2 ตัวอย่างของแผนภาพการปรับใช้




**จากรูป จะแสดงแผนภาพดีพลอยด์อย่าง ที่มี แค่โหนดเดียว ประกอบไปด้วย 2 คอมโพเน้นท์ *จะถูกใช้งานตอนนำไปติดตั้ง จะอ่านคู่มือในการติดตั้ง ว่าควรติดตั้งอย่างไร***

# 4.6 แผนภาพการปรับใช้ (Deployment Diagram)

## 4.6.2 ตัวอย่างของแผนภาพการปรับใช้



# 4.7 Use Case Model

## 4.7.1 ความหมายของแบบจำลองยูสเคส

- Use Case Model เป็นหนึ่งในแผนภาพสำคัญใน Unified Modeling Language (UML) ที่ใช้ในการแสดงฟังก์ชันหรือการกระทำที่ระบบซอฟต์แวร์หรือระบบสารสนเทศต้องทำเพื่อรองรับความต้องการและการใช้งานของผู้ใช้. Use Case Model จะแสดงกิจกรรมหรือฟังก์ชันในมุมมองของผู้ใช้ ซึ่งช่วยในการเข้าใจความต้องการและส่วนประสมที่สำคัญของระบบ
- Use case จะใช้ในขั้นตอนการเก็บ Requirement
- จะอยู่ในเฟส Inception , Elaboration Phase
- Use case จะอยู่ในคู่มือทางด้านเทคนิค เช่นเดียวกันกับ component และ deployment

# 4.7 Use Case Model

## 4.7.1 ความหมายของแบบจำลองยูสเคส

- Use Case อาจหมายถึง การอธิบายฟังก์ชันการทำงานต่าง ๆ ของระบบนั่นเอง
- Use Case Diagram จะแสดงแผนผังการใช้งานของระบบ โดยมีองค์ประกอบ 2 ส่วนคือ actor และ use case

Use case จะใช้ในการเก็บ Requirement
*** จะอยู่ในเฟส Inception , Elaboration Phase
Use case จะอยู่ในคู่มือทางด้านเทคนิค เช่นเดียวกันกับ component และ deployment

# 4.7 Use Case Model

## 4.7.2 ประเภทของแบบจำลองยูสเคส

**แบบจำลองยูสเคสแบ่งออกเป็น 2 ประเภทด้วยกันดังนี้**

1. **Use case diagram แผนภาพยูสเคส**
2. **Use case description คำอธิบายยูสเคส**

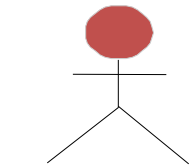
**โดย Use Case แปลตรงตัวมีความหมายว่า กรณีของการใช้งานที่เกิดจากมุมมองของผู้ใช้ระบบ ตัวอย่างง่ายๆ ของ Use Case ก็คือ**

- **Create Order (ทำการลงใบสั่งซื้อสินค้า)**
- **Modify Order (ทำการเปลี่ยนแปลงหรือแก้ไขใบสั่งซื้อสิ้นค้า)**
- **Delete Order (ทำการลบใบสั่งซื้อสินค้า)**

# 4.7 Use Case Model

## 4.7.3 ส่วนประกอบของแผนภาพยูสเคส

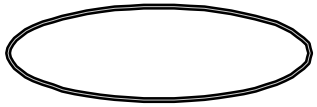
แสดงความสามารถหรือพฤติกรรมของระบบที่ผู้ใช้คาดหวัง



Actor

*Actor* (**ผู้ใช้ระบบ**)

Someone/something ที่อยู่นอกระบบที่โต้ตอบกับระบบ



Use Case name

*Use case*

ชุดของการกระทำที่ทำโดยระบบซึ่งให้ผลลัพธ์ที่มีคุณค่าต่อผู้ใช้

Communicates-Association

*Communicates-Association*

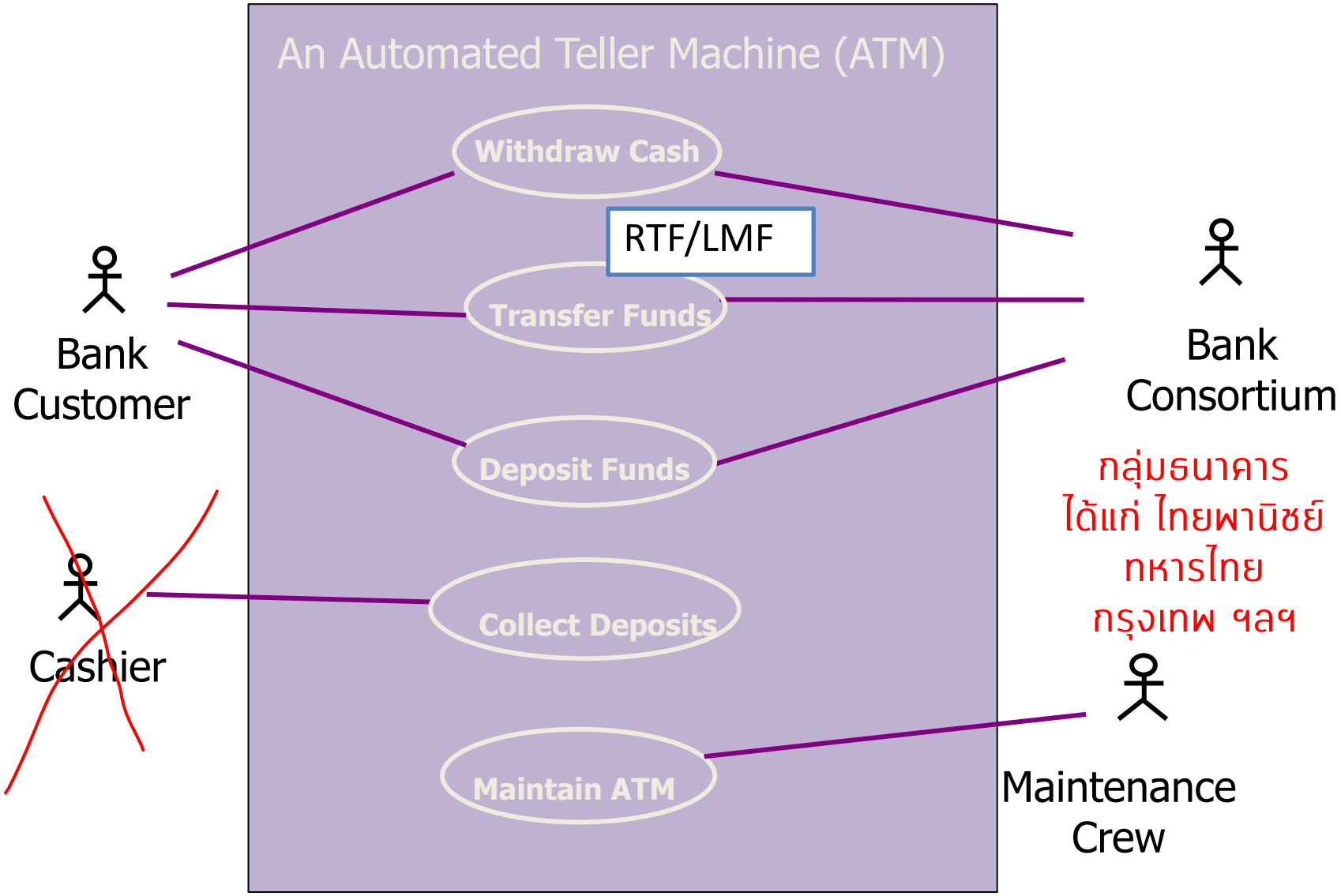
สายการสื่อสาร แทนชุดการกระทำทั้งหมด

# 4.7 Use Case Model

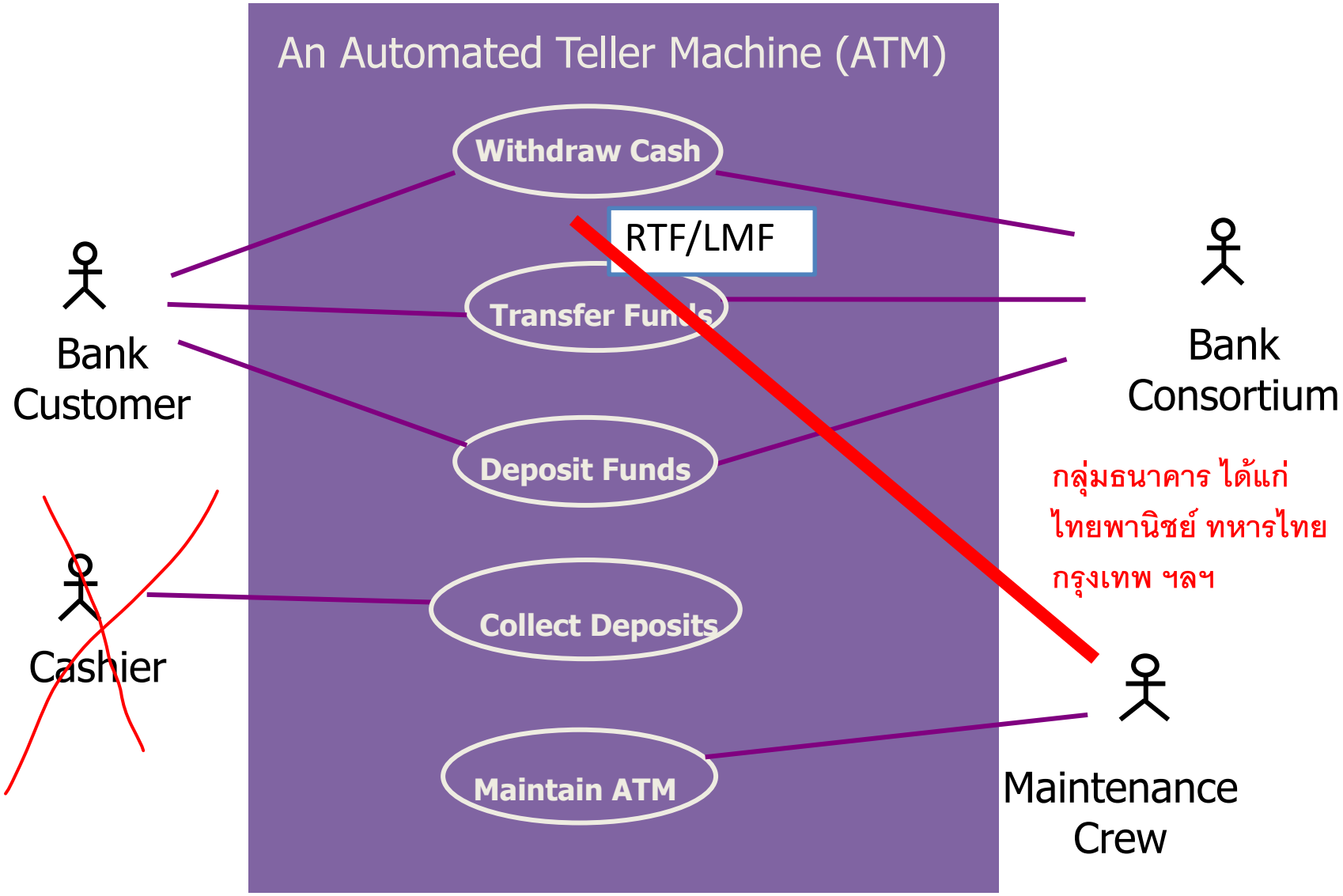
## 4.7.3 ส่วนประกอบของแผนภาพยูสเคส

### ขอบเขตของระบบ (System Boundary)

# 4.7.4 ตัวอย่างของแผนภาพยูสเคส

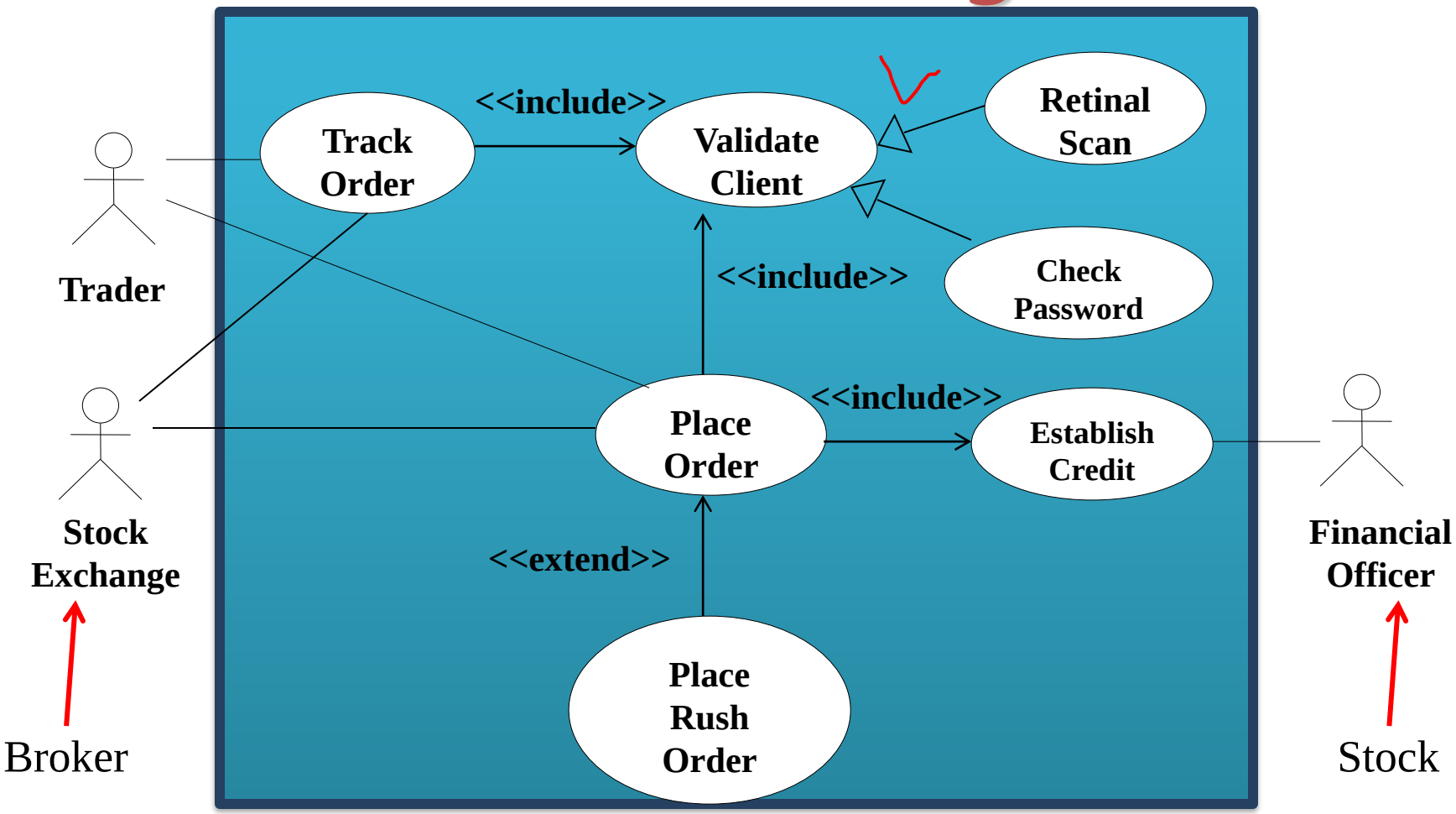


# 4.7.4 ตัวอย่างของแผนภาพยูสเคส



# 4.7.4 ตัวอย่างของแผนภาพยูสเคส

## A Use Case Diagram




ตย. การเขียนยูสเคสในการซื้อขายหุ้นในตลาดหลักทรัพย์

# 4.8 แผนภาพลำดับ Sequence Diagram

## 4.8.1 ความหมายของแผนภาพลำดับ

- Sequence Diagram จะแสดงการทำงานของออบเจ็กต์ต่าง ๆ เมื่อเกิดการส่งข่าวสารหรือ message และเมื่อเกิดเหตุการณ์ต่าง ๆ โดยทิศทางของลูกศรจะเป็นการบ่งบอกถึงทิศทางการส่ง message ระหว่างออบเจ็กต์

- แผนภาพลำดับ (Sequence Diagram) เป็นหนึ่งในแผนภาพของ Unified Modeling Language (UML) ที่ใช้ในการแสดงและอธิบายการโต้ตอบระหว่างอ็อบเจกต์หรือคลาสในระบบซอฟต์แวร์หรือระบบสารสนเทศ. แผนภาพลำดับช่วยในการเข้าใจและแสดงลำดับของข้อมูลหรือการโต้ตอบที่เกิดขึ้นระหว่างอ็อบเจกต์หรือคลาสต่าง ๆ ในระบบในแต่ละขั้นตอนหรือทำความเข้าใจการทำงานของระบบในรูปแบบลำดับขั้นตอน.

# 4.8 แผนภาพลำดับ Sequence Diagram

## 4.8.2 ส่วนประกอบของแผนภาพลำดับ

1. **อ็อบเจกต์ (Objects):** แผนภาพลำดับแสดงอ็อบเจกต์หรือคลาสที่มีการเปลี่ยนแปลงสถานะหรือทำงานในแต่ละขั้นตอน
2. **เส้นลำดับ (Sequence Lifelines):** เส้นลำดับหรือแวดล้อมแสดงอ็อบเจกต์หรือคลาสแต่ละตัว ในแต่ละขั้นตอนของแผนภาพ แสดงถึงช่วงเวลาที่อ็อบเจกต์หรือคลาสทำงาน
3. **ข้อความ (Messages):** ข้อความแสดงการโต้ตอบระหว่างอ็อบเจกต์หรือคลาส โดยระบุชื่อของเมธอดหรือการกระทำที่ถูกเรียกใช้และค่าที่ถูกส่งไปมาระหว่างอ็อบเจกต์
4. **เงื่อนไข (Conditions):** เงื่อนไขหรือการตัดสินใจสามารถแสดงในแผนภาพลำดับเพื่อแสดงว่าการทำงานมีการแยกสาขาเมื่อต้องการตัดสินใจตามเงื่อนไข
5. **การกลับ (Return Messages):** การกลับแสดงถึงการโต้ตอบกลับจากอ็อบเจกต์หรือคลาสหลังจากที่ได้รับข้อความ

# 4.8 แผนภาพลำดับ Sequence Diagram

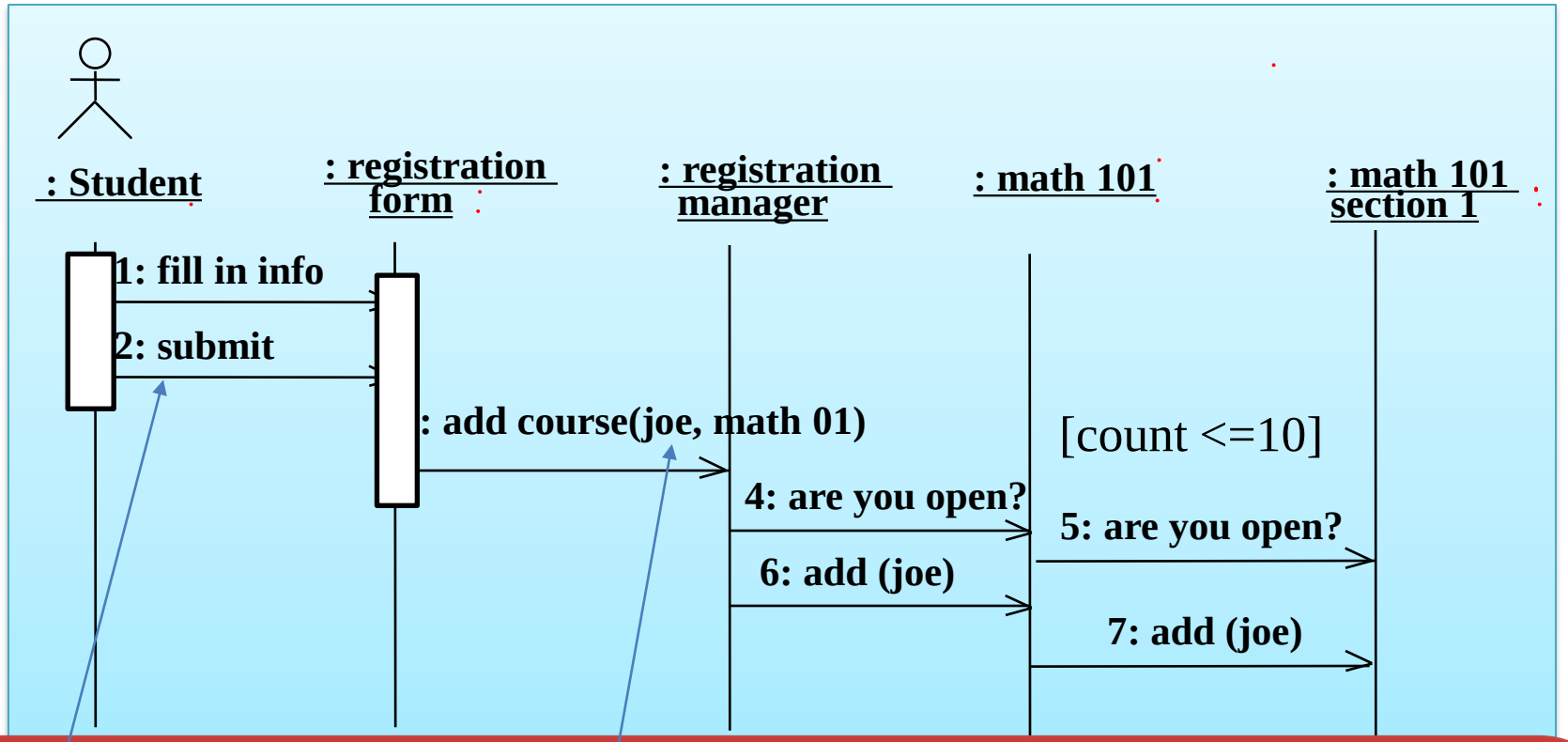
## 4.8.3 ตัวอย่างของแผนภาพลำดับ

1. ผู้ใช้เริ่มต้นโดยการเลือกสินค้าในเว็บไซต์.
2. เว็บไซต์ส่งคำขอ (Request) ไปยังเซิร์ฟเวอร์เว็บเพื่อเรียกดูรายการสินค้า.
3. เซิร์ฟเวอร์เว็บดึงข้อมูลรายการสินค้าจากฐานข้อมูลและส่งข้อมูลกลับไปยังเว็บไซต์.
4. เว็บไซต์แสดงรายการสินค้าให้กับผู้ใช้.
5. ผู้ใช้เลือกสินค้าและทำการสั่งซื้อ.
6. เว็บไซต์ส่งคำขอการสั่งซื้อไปยังเซิร์ฟเวอร์เว็บ.
7. เซิร์ฟเวอร์เว็บตรวจสอบคำขอการสั่งซื้อและบันทึกรายการสั่งซื้อในฐานข้อมูล.
8. เซิร์ฟเวอร์เว็บส่งการยืนยันการสั่งซื้อกลับไปยังผู้ใช้.
9. ผู้ใช้ได้รับการยืนยันและสามารถดาวน์โหลดใบสั่งซื้อ.
10. กระบบส่งคำขอให้เซิร์ฟเวอร์เว็บส่งอีเมล์แจ้งการสั่งซื้อถึงผู้ขาย.
11. เซิร์ฟเวอร์เว็บส่งอีเมล์แจ้งการสั่งซื้อถึงผู้ขาย.
12. ผู้ขายตอบกลับด้วยข้อมูลการจัดส่ง.
13. เซิร์ฟเวอร์เว็บส่งอีเมล์แจ้งข้อมูลการจัดส่งถึงผู้ใช้.
14. ผู้ใช้ได้รับอีเมล์แจ้งข้อมูลการจัดส่ง.
15. ผู้ขายจัดส่งสินค้า.
16. เซิร์ฟเวอร์เว็บส่งอีเมล์แจ้งการจัดส่งสินค้าถึงผู้ใช้.
17. ผู้ใช้ได้รับอีเมล์แจ้งการจัดส่งสินค้า.

# A Sequence Diagram

Create new student()
...
Destroy

หน้าที่ เป็นของโปรแกรมเมอร์ กับ Tester



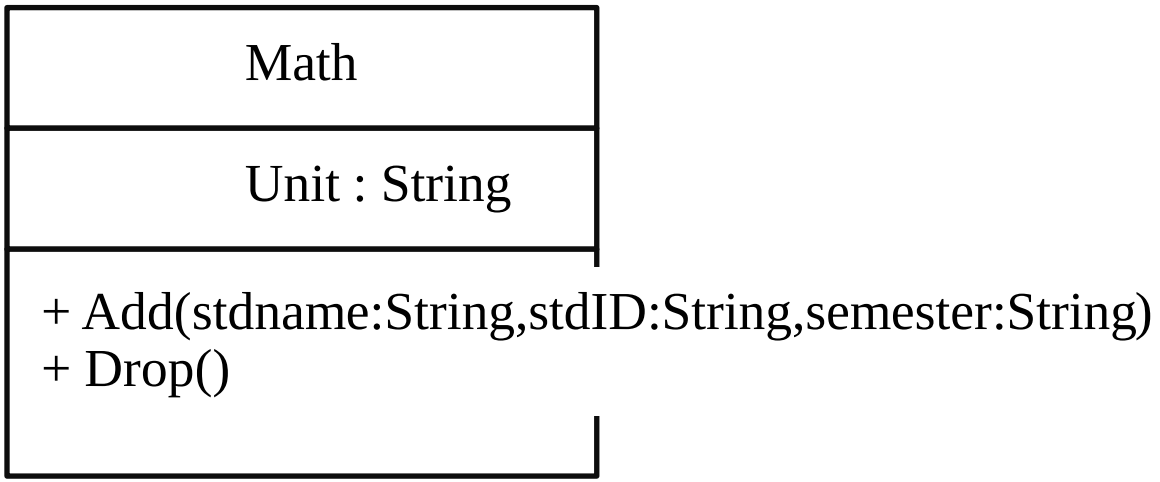

ถ้างานใดที่เน้น ลำดับ เวลา ทำก่อนหลัง หรือ Requirement ใด อะไรทำก่อน-หลัง ให้ ใช้ Sequence Diagram

กรณีผ่านเว็บก็ คลิกปุ่ม submit

Message : joe , math101 if(Math101.isOpen()) { …}

# 4.8 แผนภาพลำดับ Sequence Diagram

## 4.8.3 ตัวอย่างของแผนภาพลำดับ




1/2561

3(3-0-6)

3(2-2)

# 4.9 แผนภาพคอลาบอเรชัน Collaboration Diagram

## 4.9.1 ความหมายของแผนภาพคอลาบอเรชัน

แผนภาพคอลลาโบเรชัน (Collaboration Diagram) เป็นหนึ่งในแผนภาพของ Unified Modeling Language (UML) ที่ใช้ในการแสดงความสัมพันธ์และการทำงานระหว่างอ็อบเจกต์หรือคลาสในระบบซอฟต์แวร์หรือระบบสารสนเทศในมุมมองที่มุ่งเน้นการร่วมทำงานระหว่างสมาชิกของกลุ่มหรือคอลลาโบเรชัน

Collaboration Diagram จะแสดงการติดต่อสื่อสารระหว่างออบเจ็กต์ต่าง ๆ และความสัมพันธ์ระหว่างที่แต่ละออบเจ็กต์ติดต่อสื่อสารกัน

คอลลาโบเรชันแสดงถึงความสัมพันธ์ระหว่างอ็อบเจกต์หรือคลาสในรูปแบบของการส่งข้อความระหว่างกันและแสดงอ็อบเจกต์หรือคลาสในลักษณะของกลุ่มที่ทำงานร่วมกันเพื่อกระทำกิจกรรมบางอย่าง.

ในยูเอ็มแอลรุ่น 2.0 เป็นต้นไป จะเรียก Collaboration Diagram >> Communication Diagram

แผนภาพคอลลาโบเรชันช่วยในการเข้าใจและอธิบายความสัมพันธ์และการทำงานระหว่างอ็อบเจกต์หรือคลาสในระบบโปรแกรมหรือระบบสารสนเทศ โดยเน้นการร่วมทำงานและการสื่อสารระหว่างสมาชิกในระบบอย่างชัดเจน. แผนภาพนี้มักถูกใช้ในกระบวนการออกแบบระบบและการโต้ตอบของระบบก่อนการสร้างระบบจริง

# 4.9 แผนภาพคอลลาบอเรชัน Collaboration Diagram

## 4.9.2 ลักษณะของแผนภาพคอลลาโบเรชัน

**ลักษณะหลักของแผนภาพคอลลาโบเรชัน:**

1. **อ็อบเจกต์หรือคลาส (Objects or Classes): แผนภาพคอลลาโบเรชันแสดงรายชื่อของอ็อบเจกต์หรือคลาสที่เกี่ยวข้องกับกระบวนการหรือกิจกรรมที่กำลังแสดงในแผนภาพ**
2. **ส่งข้อความ (Messages): ส่งข้อความระหว่างอ็อบเจกต์หรือคลาสเพื่อแสดงการสื่อสารและการโต้ตอบในกระบวนการ**
3. **ส่วนประกอบคอลลาโบเรชัน (Collaboration Occurrences): แสดงอ็อบเจกต์หรือคลาสที่เข้าร่วมในการคอลลาโบเรชันนี้และสร้างความสัมพันธ์ระหว่างพวกเขา**
4. **ความสัมพันธ์ (Relationships): แสดงความสัมพันธ์ระหว่างอ็อบเจกต์หรือคลาสที่เกี่ยวข้องในแผนภาพคอลลาโบเรชัน**
5. **ข้อความเงื่อนไข (Conditional Messages): แสดงการส่งข้อความในกรณีที่มีเงื่อนไขหรือการตัดสินใจ**

# 4.9 แผนภาพคอลาบอเรชัน Collaboration Diagram

## 4.9.3 ข้อแตกต่างระหว่างแผนภาพลำดับกับแผนภาพคอลลาโบเรชัน

- **แผนภาพลำดับ (Sequence Diagram) และแผนภาพคอลลาโบเรชัน (Collaboration Diagram)** เป็นสองประเภทของแผนภาพใน Unified Modeling Language (UML) ที่ใช้ในการแสดงและอธิบายความสัมพันธ์และการทำงานระหว่างอ็อบเจกต์หรือคลาสในระบบซอฟต์แวร์หรือระบบสารสนเทศ โดยมีลักษณะและการใช้งานที่แตกต่างกันตามลักษณะของการสื่อสารและการกระทำ:
- **Sequence Diagram** ใช้ในการแสดงการสื่อสารและการโต้ตอบระหว่างอ็อบเจกต์หรือคลาสในรูปแบบของลำดับขั้นตอนที่เกิดขึ้นในเวลาจริง.
- **Collaboration Diagram** ใช้ในการแสดงความสัมพันธ์และการทำงานระหว่างกลุ่มของอ็อบเจกต์หรือคลาส โดยเน้นความร่วมทำงานและคุณลักษณะของการร่วมทำงาน.

การเลือกใช้แผนภาพที่เหมาะสมขึ้นอยู่กับวัตถุประสงค์และแง่มุมมองของการแสดงข้อมูลในการออกแบบและวางแผนระบบของคุณ.

# 4.9 แผนภาพคอลาบอเรชัน Collaboration Diagram

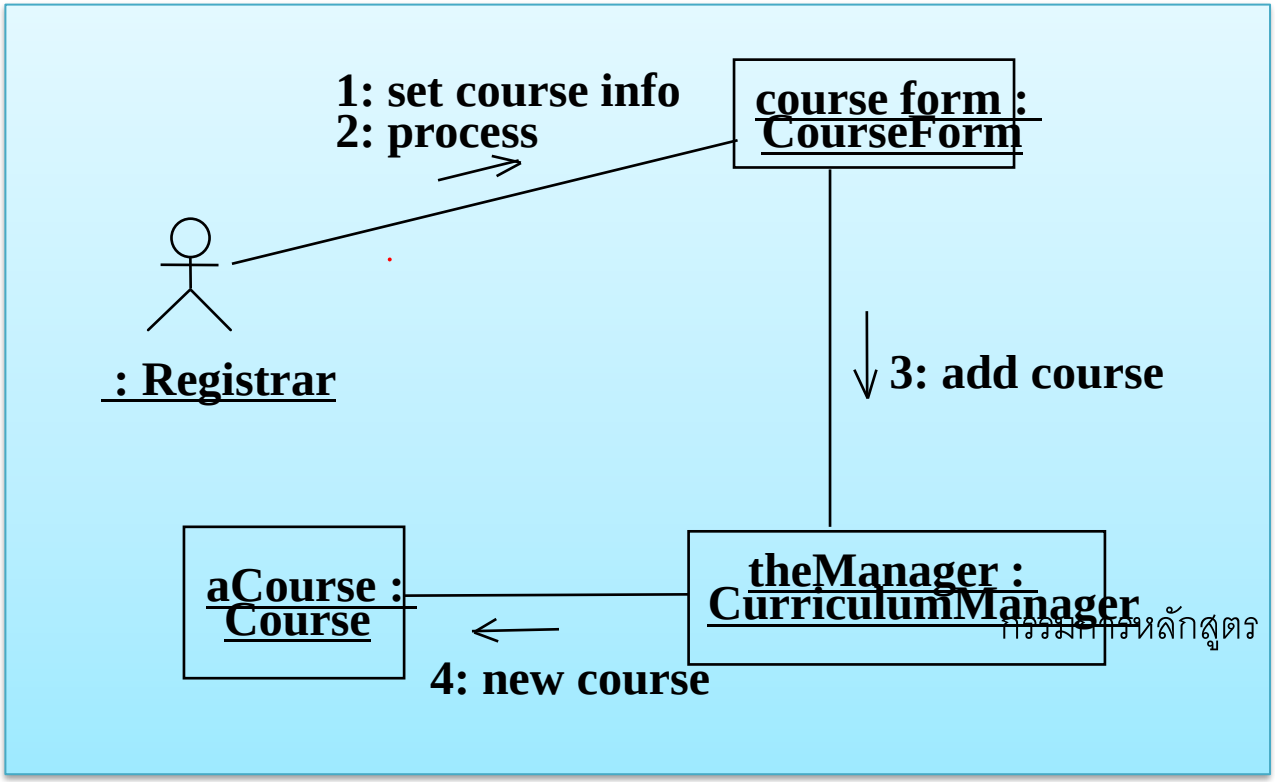
## 4.9.3 ข้อแตกต่างระหว่างแผนภาพลำดับกับแผนภาพคอลลาโบเรชัน

1. Sequence Diagram (แผนภาพลำดับ):
    1. การสื่อสาร: Sequence Diagram เน้นการสื่อสารและการโต้ตอบระหว่างอ็อบเจกต์หรือคลาสในแบบลำดับเวลา โดยแสดงลำดับของข้อความที่ถูกส่งระหว่างอ็อบเจกต์.
    2. การใช้งาน: Sequence Diagram มักถูกใช้ในกระบวนการวางแผนและออกแบบการทำงานของระบบ และช่วยในการเข้าใจและทดสอบการทำงานของระบบ.
    3. การเน้น: Sequence Diagram เน้นลำดับขั้นตอนและการโต้ตอบในรูปแบบลำดับที่เกิดขึ้น.
2. Collaboration Diagram (แผนภาพคอลลาโบเรชัน):
    1. การสื่อสาร: Collaboration Diagram เน้นความสัมพันธ์และการทำงานระหว่างอ็อบเจกต์หรือคลาสในมุมมองของคอลลาโบเรชัน โดยแสดงแบบคุณลักษณะของการร่วมทำงาน.
    2. การใช้งาน: Collaboration Diagram มักถูกใช้ในการแสดงความสัมพันธ์และการทำงานระหว่างกลุ่มของอ็อบเจกต์หรือคลาส ซึ่งทำงานร่วมกันเพื่อกระทำกิจกรรมบางอย่าง.
    3. การเน้น: Collaboration Diagram เน้นการร่วมทำงานและความสัมพันธ์ระหว่างอ็อบเจกต์หรือคลาสในมุมมองของการทำงานร่วมกัน.

# 4.9 แผนภาพคอลาบอเรชัน Collaboration Diagram

## 4.9.4 ตัวอย่างของแผนภาพคอลลาโบเรชัน

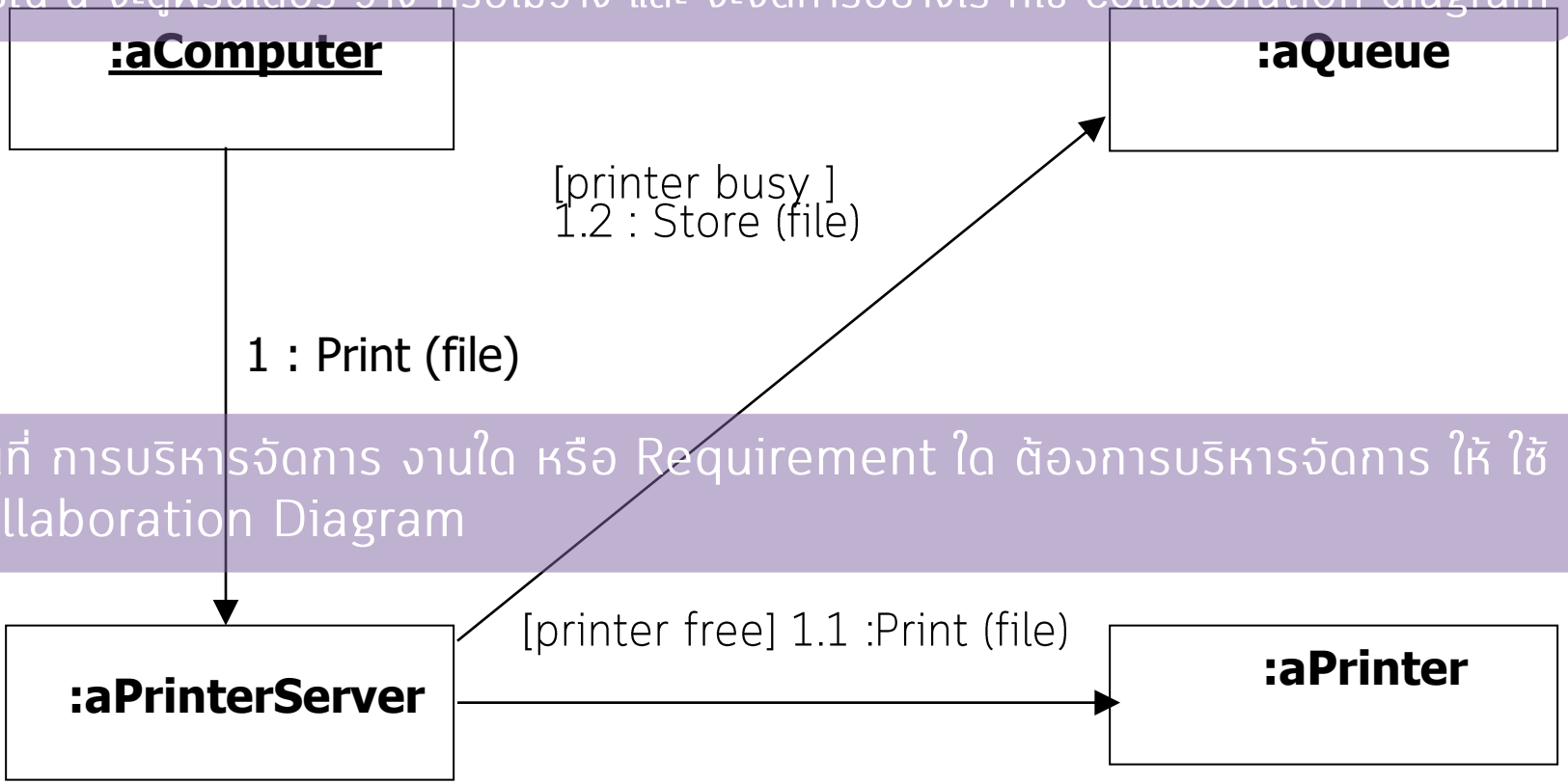
- จนท ดำเนินการ สร้าง Course เรียนขึ้นมา โดยระบุรายละเอียดของคอร์ส ตามแบบฟอร์มเน้นที่ การบริหารจัดการ
- งานใด หรือ Requirement ใด ต้องการบริหารจัดการ ให้ ใช้ Collaboration Diagram



# 4.9 แผนภาพคอลาบอเรชัน Collaboration Diagram

## 4.9.4 ตัวอย่างของแผนภาพคอลลาโบเรชัน

collaboration diagram แสดงความเกี่ยวพันของ objects ต่าง ๆ ที่ต้องทำงานร่วมกัน กรณี นี้ จะดูพรินเตอร์ ว่าง หรือไม่ว่าง และ จะจัดการอย่างไร ก็ใช้ collaboration diagram



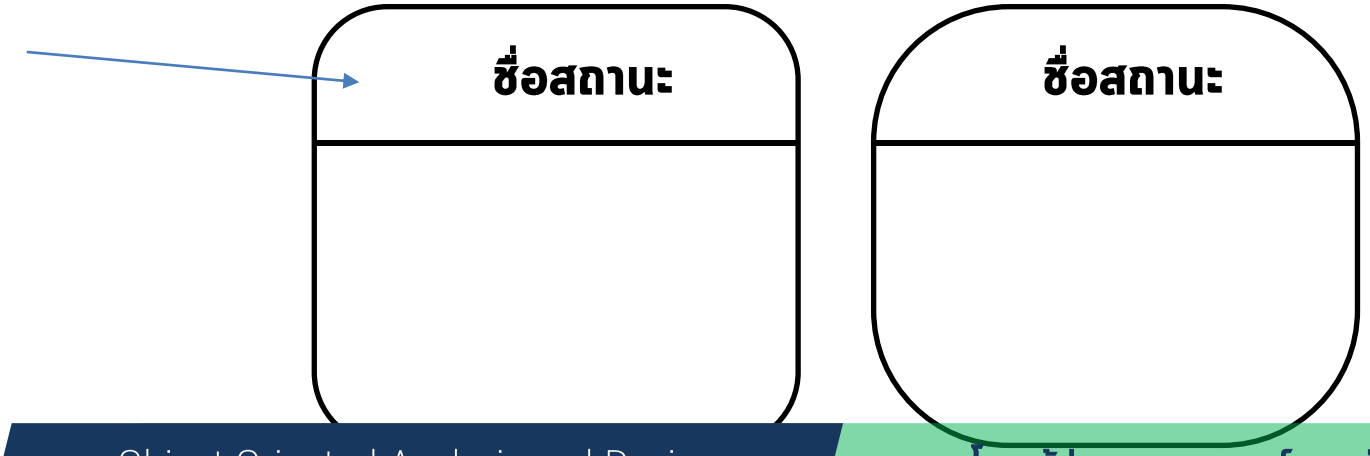

เน้นที่ การบริหารจัดการ งานใด หรือ Requirement ใด ต้องการบริหารจัดการ ให้ ใช้ Collaboration Diagram

# 4.10 State-Transition Diagram

## 4.10.1 ความหมายของแผนภาพการเปลี่ยนสถานะ

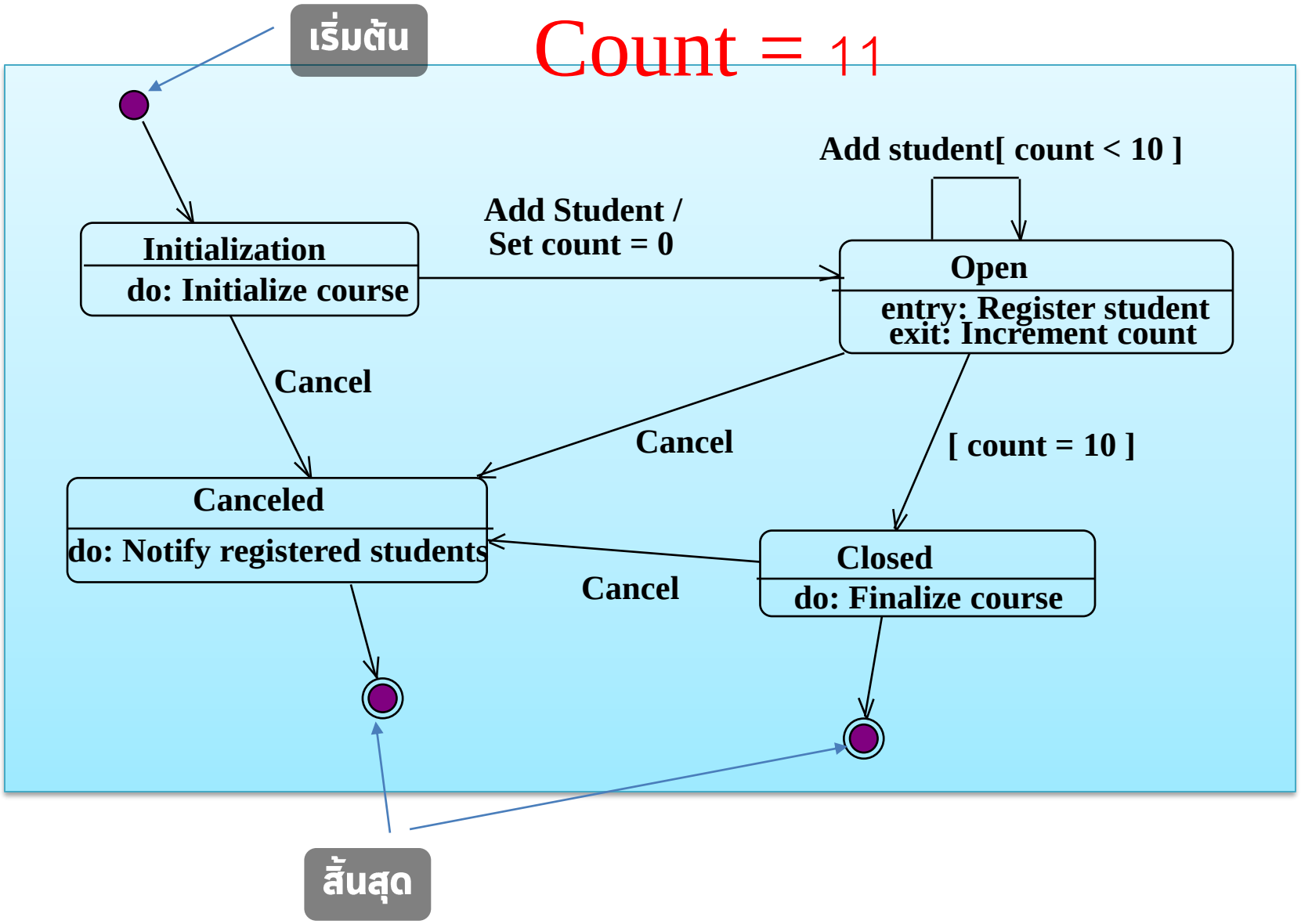
**State-Transition Diagram ทำหน้าที่ต่อไปนี้**

- แสดงวงจรชีวิตของออบเจ็กต์ ระบบย่อยต่าง ๆ และระบบโดยรวม
- บ่งบอกว่าเหตุการณ์ต่าง ๆ จะส่งผลกระทบให้เกิดอะไรขึ้นบ้างในระบบ
- อาจมีจุดเริ่มต้นและจุดจบได้หลายจุด



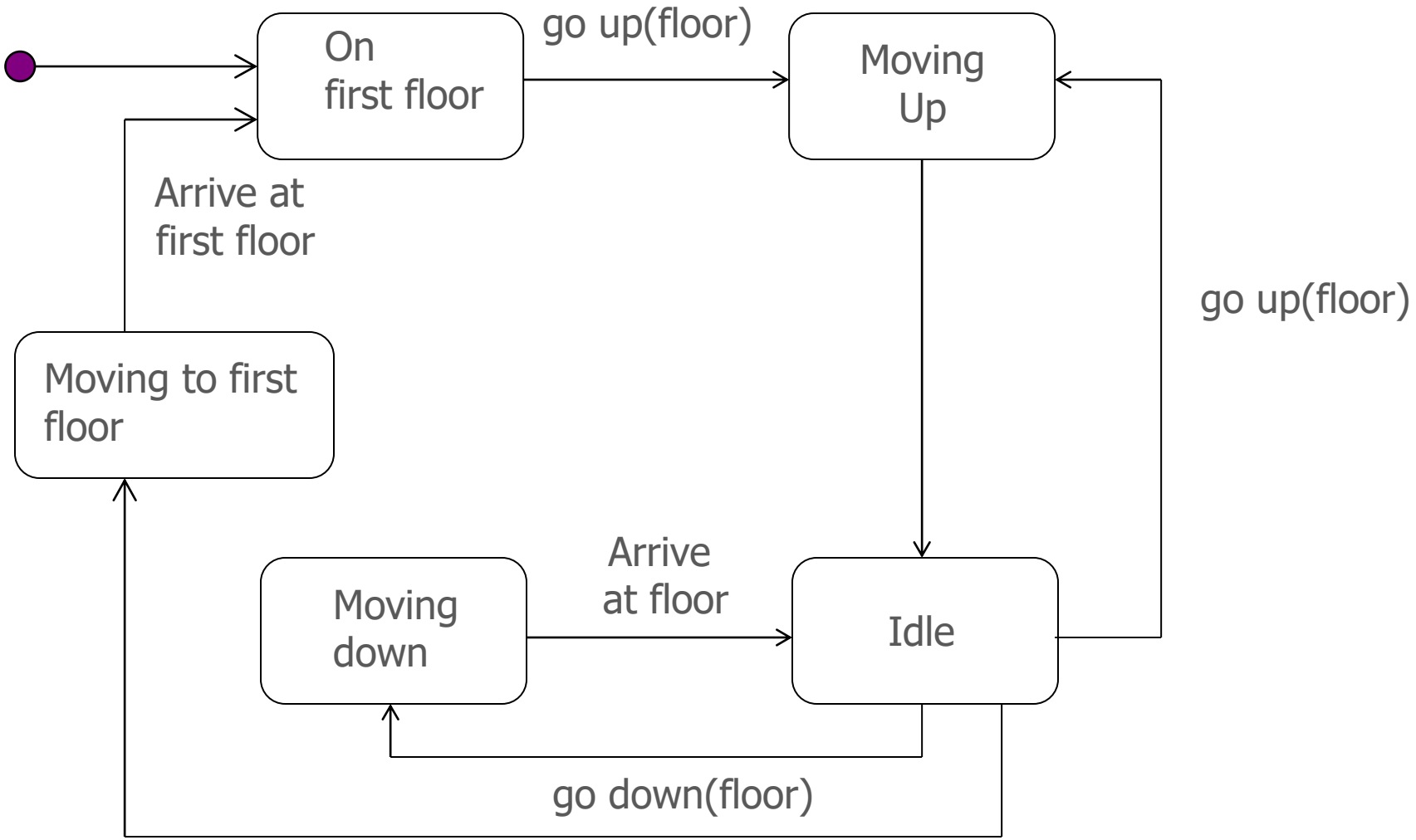
จากรูปนี้ สถานะของคอร์สเรียนมี 4 สถานะ

# A State-Transition Diagram



# 4.10.1 ความหมายของแผนภาพการเปลี่ยนสถานะ

**state diagram แสดงสถาณภาพ (state) ที่เป็นไปได้ของ object ใด ๆ รวมทั้งแสดงเหตุการณ์ (event) ที่ทำให้เกิดการเปลี่ยนสภาพของ object**



# 4.11 Activity diagram

## 4.11.1 ความหมายของแผนกิจกรรม

- ใช้สำหรับ
  - อธิบาย กระแสการไหลของการทำงาน (workflow)
  - แสดงขั้นตอนการทำงานของระบบ
- แต่ละขั้นตอนการทำงาน เรียกว่า Activity ตัวอย่าง ได้แก่
  - การคำนวณผลลัพธ์บางอย่าง
  - การเปลี่ยนแปลงสถานะ (State) ของระบบ
  - การส่งค่ากลับคืน
  - การส่งสัญญาณ
  - การเรียกให้โอเปอร์เรชันอื่นๆ ทำงาน
  - การสร้าง หรือ ทำลายวัตถุ

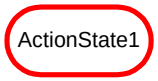
Flowchart in traditional approaches

# 4.11 Activity diagram

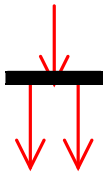
**activity diagram แสดงลำดับกระแสของกิจกรรมของการทำงานใดๆเช่น ขั้นตอนการทำ operation ขั้นตอนการลงทะเบียน**

Initial State

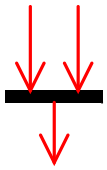
Final State




Action State



Transition (Fork)

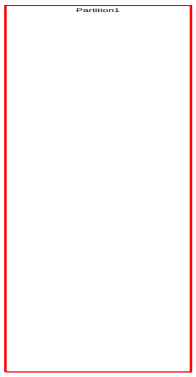


Transition (Join)



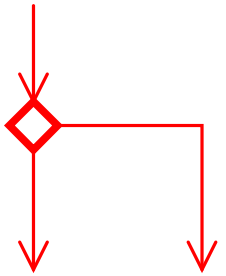
Control Flow

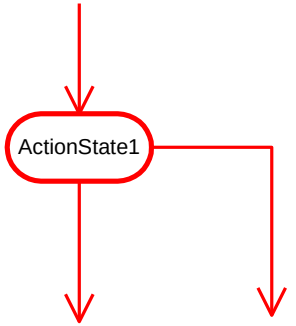
Decision




Swimlane

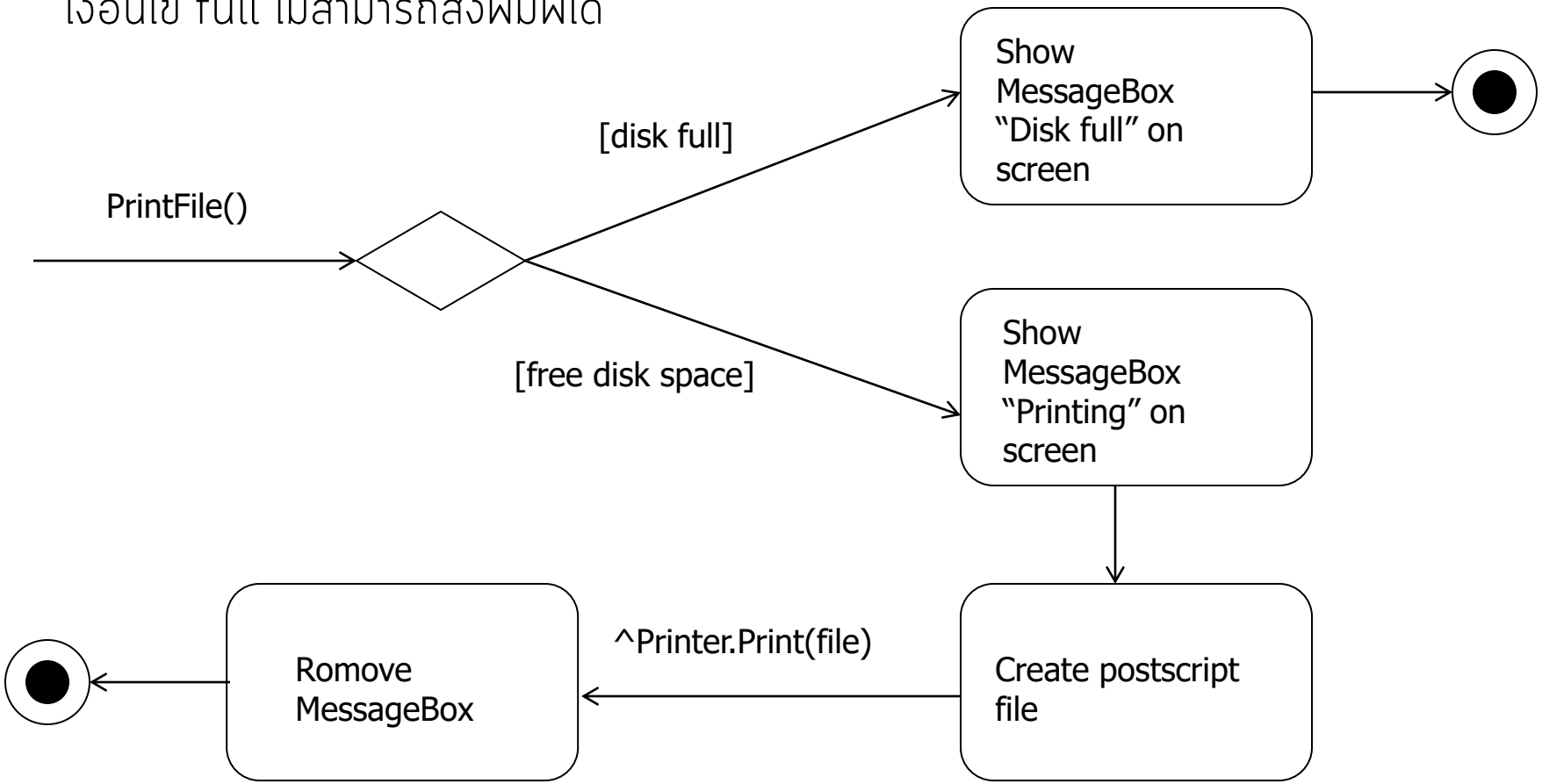
Note





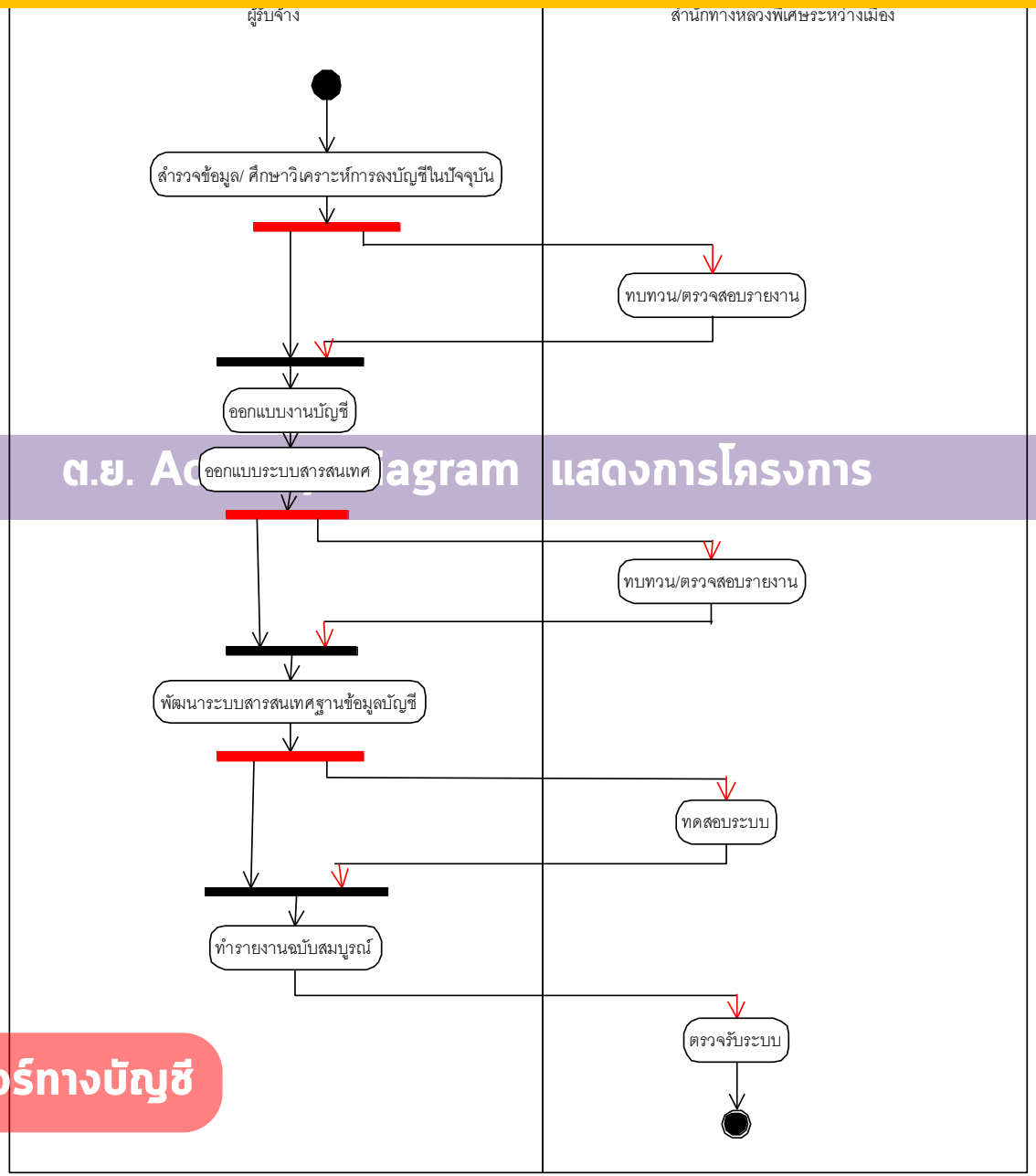
# ระบบ แบบ Realtime / ระบบ control ซึ่งจากตัวอย่าง จะเป็นระบบ control การสั่งพิมพ์ไฟล์เอกสาร ในเครื่องพิมพ์

เงื่อนไข full ไม่สามารถสั่งพิมพ์ได้




ตย. Activity Diagram แสดงการทำงานของ operation 'print'

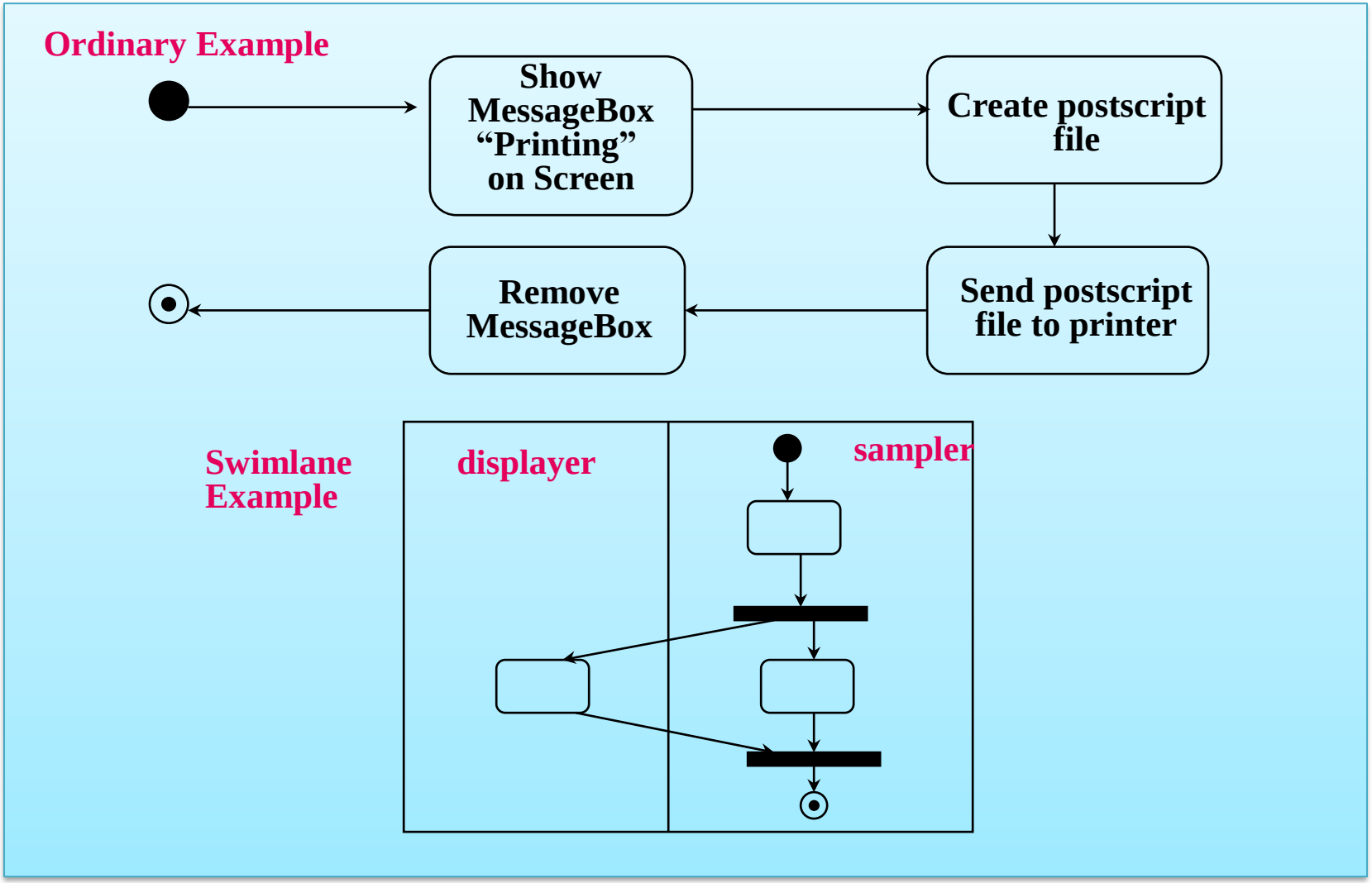
# 4.11 Activity diagram



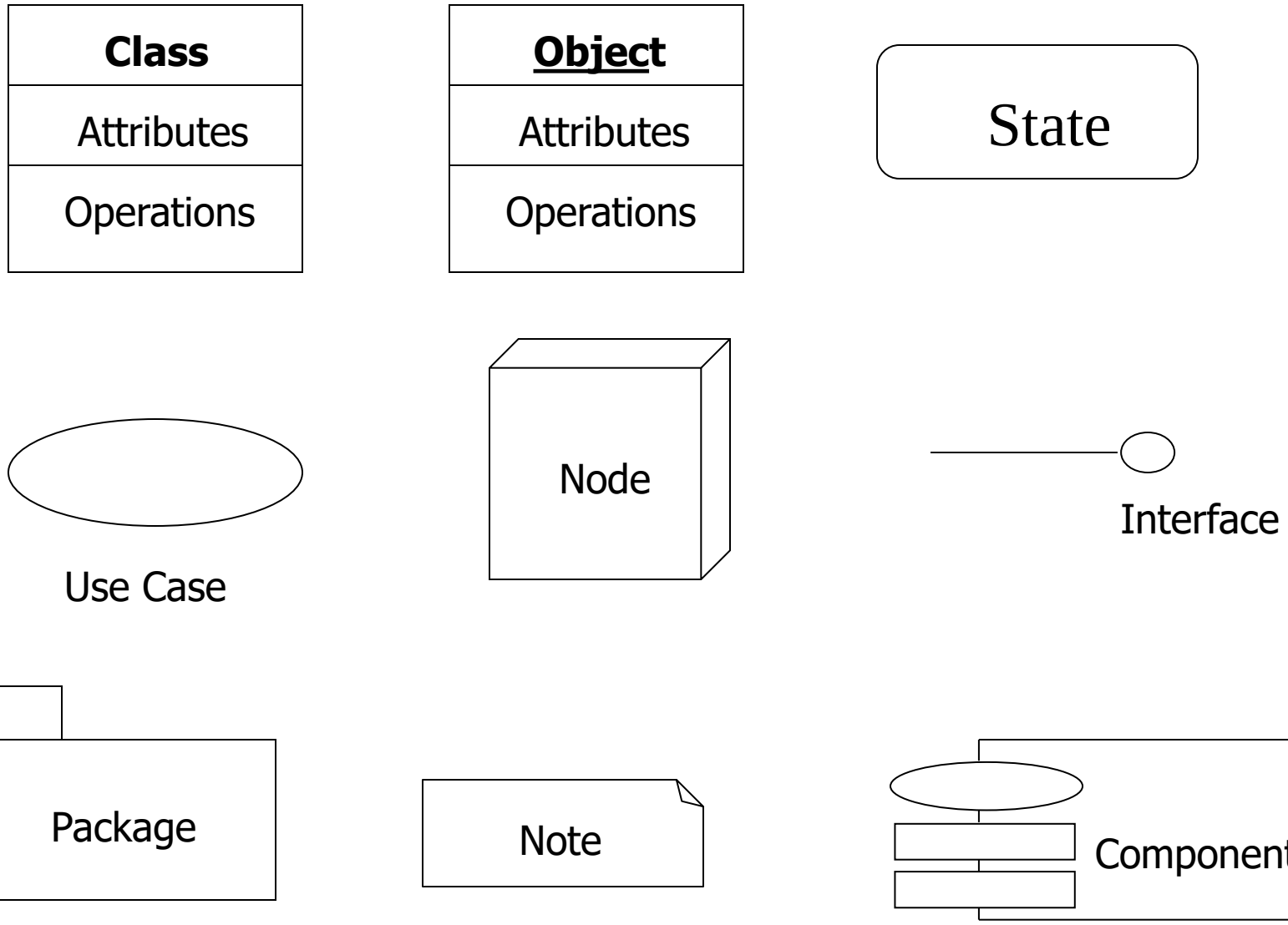

ต.ย. Activity diagram แสดงการโครงการ

**รับจ้างพัฒนาซอฟต์แวร์ทางบัญชี**

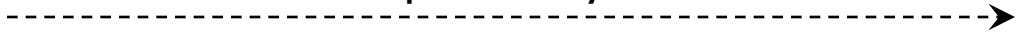
# 4.11 Activity diagram



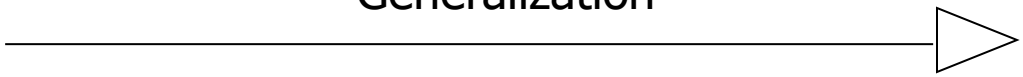
# Model Elements



Dependency



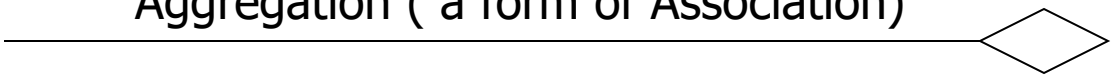
Generalization



Association



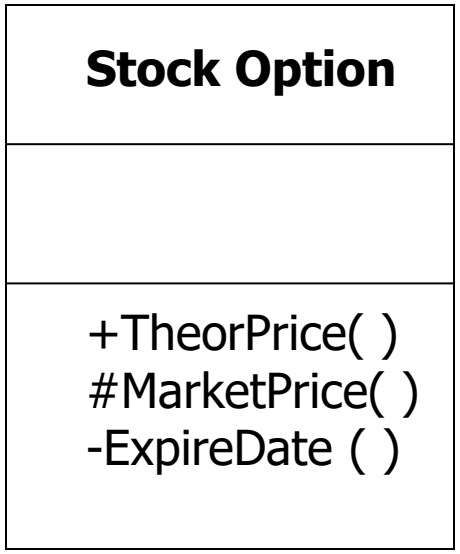
Aggregation ( a form of Association)



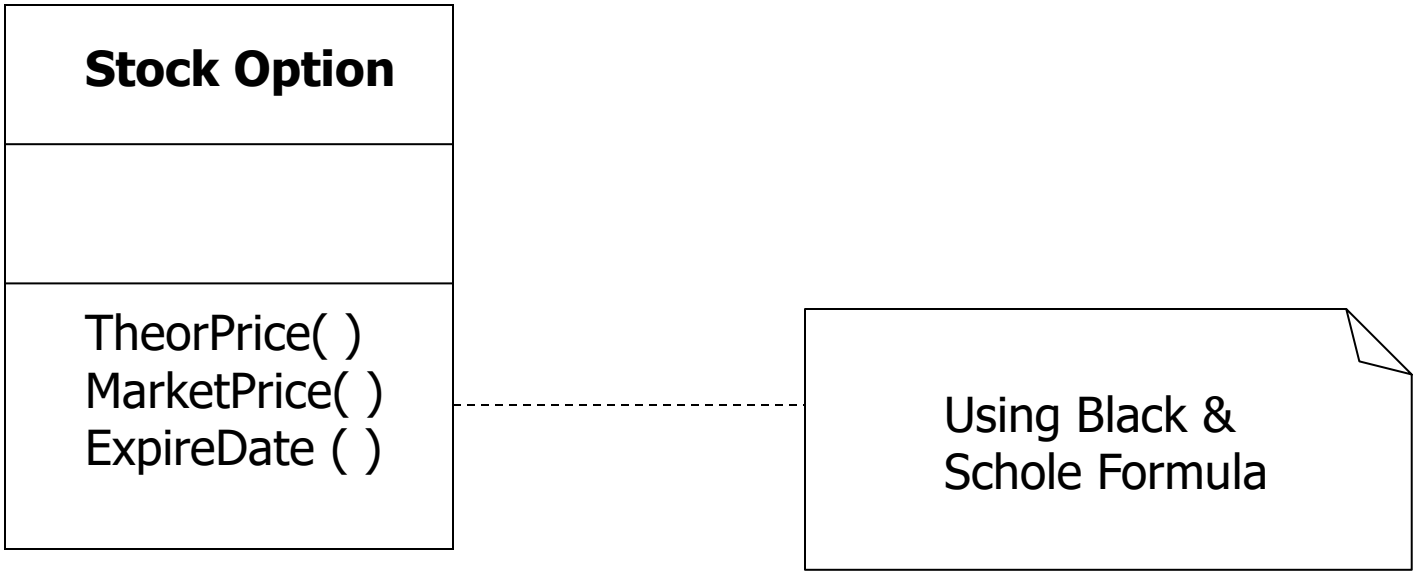
# General Mechanisms

UML มีกลไกสำหรับการให้ information เพิ่มเติมเข้าไปในทุกประเภทของแผนภาพ โดยเฉพาะ information นั้นไม่สามารถสื่อสารได้โดยใช้สัญลักษณ์พื้นฐานของ UML ที่มี

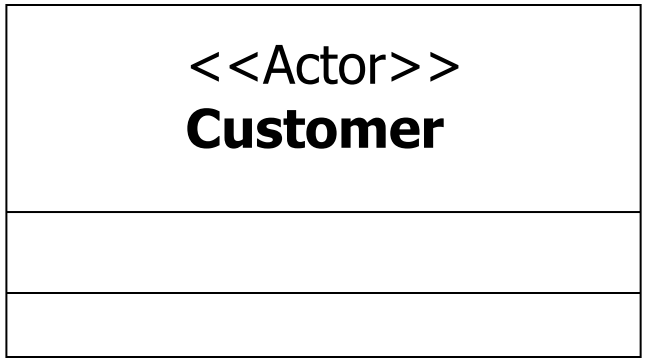
- เครื่องประดับ (Adornments)
- โน้ต (Notes)
- ข้อกำหนด (Specifications)
- Extensibility Mechanism
  - แม่พิมพ์ (Stereotypes)
  - ติดป้าย (Tagged value)
  - ข้อจำกัด (Constraints)

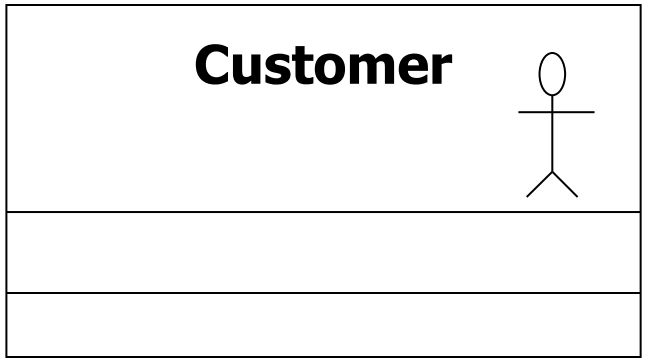


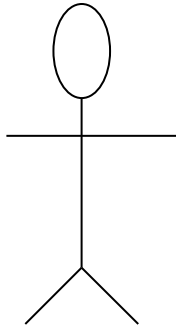

**Adornments** เพิ่ม information เข้าไปในสัญลักษณ์ปกติ




A **note** contains any additional information such as a simple comment

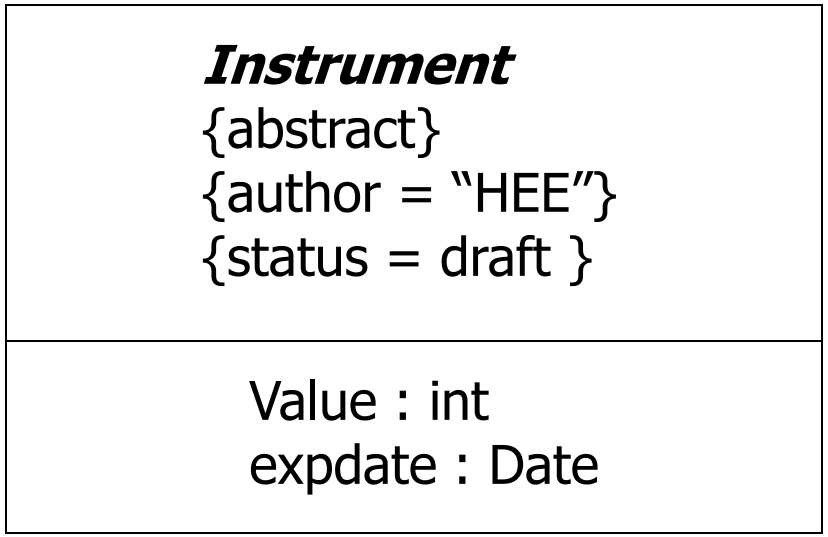


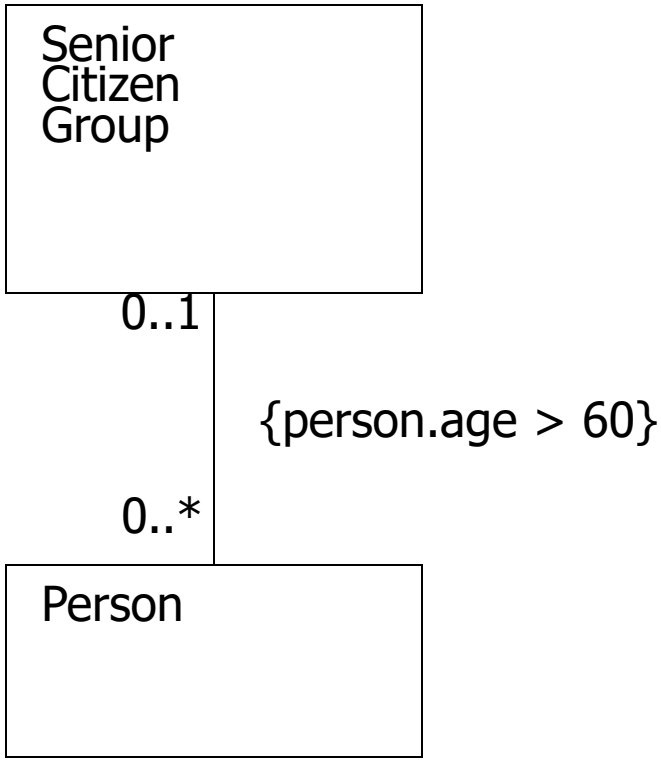



Customer

The customer is a class with the stereotype <<Actor>>. The stereotype adds extra semantics to the class; in this case, that the class represents an external user of the system




Properties on an Instrument class. Abstract is a predefined property; author and status are user defines tagged values




A constraint restricts which Person objects may participate in the association

# *Benefits of UML*

- เป็นภาษามาตรฐานที่ใช้ในการสร้างแบบจำลองเชิงวัตถุ โดยใช้รูปภาพ (Standard Visual Modeling Language)
- ใช้ในการแลกเปลี่ยนข้อมูล แบบจำลองกันระหว่างทีมผู้พัฒนา และระหว่างผู้ใช้ กับทีมผู้พัฒนา
- นำเสนอ และ สนับสนุนหลักการเชิงวัตถุได้ครบถ้วน ชัดเจน
- ไม่ผูกติดกับภาษาโปรแกรม (Programming Language) ภาษาใดภาษาหนึ่ง
- สนับสนุนการขยายขอบเขต และการปรับปรุงระบบ โดยที่ไม่จำเป็นต้องลงมือพัฒนาซอสโค้ดก่อน