บทที่ 6

บทที่ 6 แผนภาพปฏิสัมพันธ์
Lec06 : Interaction Diagram

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

สาขาวิชา วิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์และเทคโนโลยี
มหาวิทยาลัยราชภัฏพระนคร

Agenda

บทที่ 6 แผนภาพปฏิสัมพันธ์
Lec06 : Interaction Diagram

6.1 ภาพรวม UML Diagrams
อธิบาย UML 1.x = 9 diagrams และ UML 2.x = 14 diagrams ให้ชัด

6.2 Static View และ Dynamic View
อธิบาย Structural Diagram และ Behavioral Diagram

6.3 Behavioral Diagrams คืออะไร
แยก Use Case, Activity, State Machine และ Interaction Diagram

6.4 Interaction Diagrams คืออะไร
อธิบายว่าใช้แสดงการโต้ตอบระหว่าง Actor/Object ผ่าน Message

6.5 ประเภทของ Interaction Diagrams
Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Di...

6.6 Sequence Diagram
ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

Agenda

แผนภาพปฏิสัมพันธ์
Lec06 : Interaction Diagrams

6.7 **Message ใน Sequence Diagram**
Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.8 **Combined Fragment**
alt, opt, loop, par พร้อมตัวอย่าง

6.9 **ขั้นตอนการสร้าง Sequence Diagram จาก Use Case**
เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

6.10 **ตัวอย่าง Sequence Diagram: Login / Register / Order Product**

6.11 **Communication Diagram**
ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

6.12 **เปรียบเทียบ Sequence Diagram กับ Communication Diagram**

6.1 ภาพรวม UML Diagrams

อธิบาย UML 1.x = 9 diagrams และ UML 2.x = 14 diagrams ให้ชัด

what

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

แผนภาพเชิงโครงสร้าง

Structural Diagrams

สำหรับบทที่ 6 นี้

how

5. Use Case Diagram
6. Sequence Diagram
7. Communication Diagram (formerly Collaboration Diagram)
8. State Machine Diagram / State Transition Diagram
9. Activity Diagram

Behavioral Diagrams

แผนภาพเชิงพฤติกรรม

ในระบบใดๆ จะมี 2 แบบ หรือ 2 ลักษณะ

- หยุดนิ่ง static
- ไม่หยุดนิ่ง เคลื่อนไหว Dynamic

6.1 ภาพรวม UML Diagrams

อธิบาย UML 1.x = 9 diagrams และ UML 2.x = 14 diagrams ให้ชัด

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

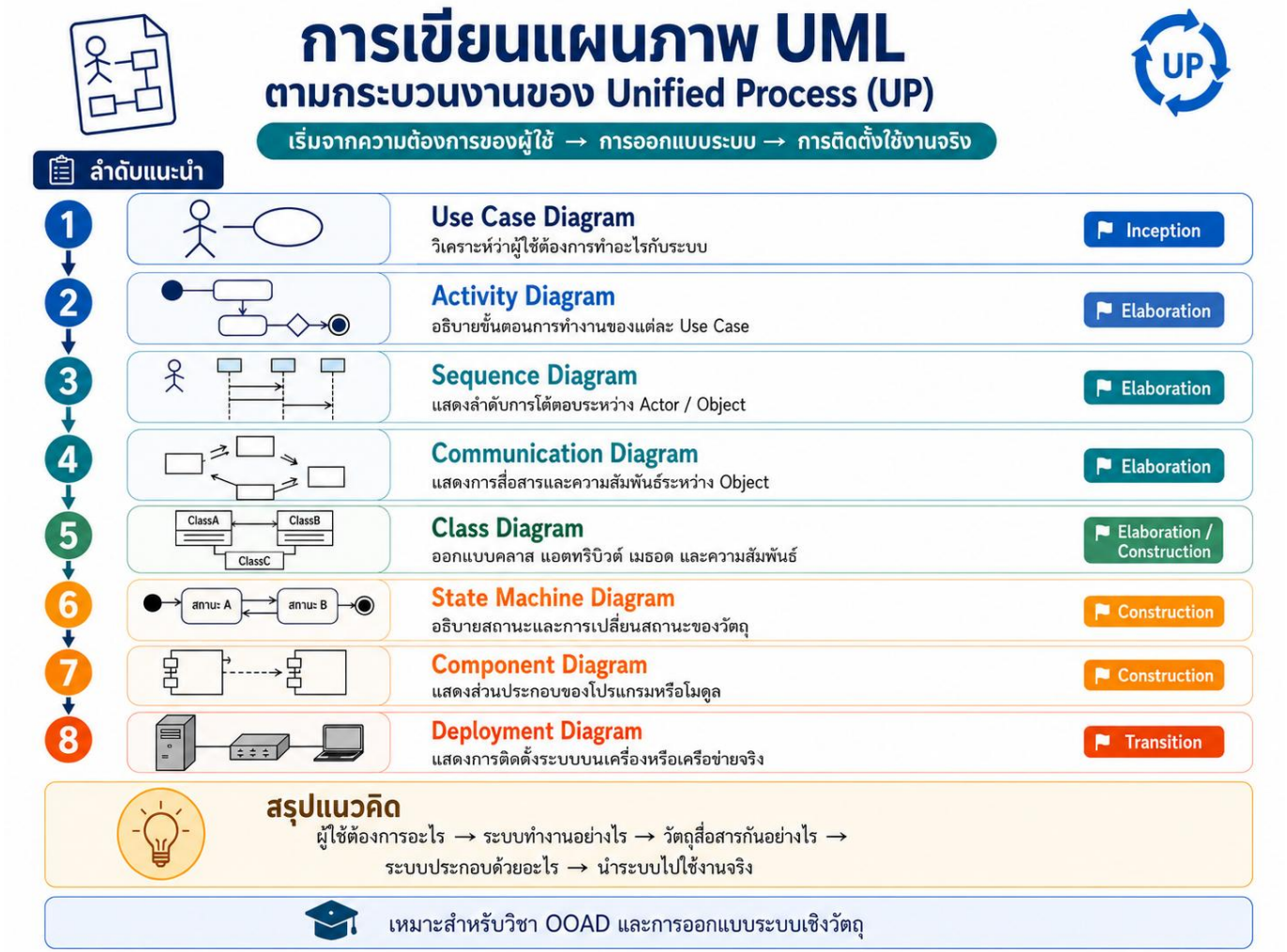
6.1 ภาพรวม UML Diagrams

อธิบาย UML 1.x = 9 diagrams และ UML 2.x = 14 diagrams ให้ชัด

ลำดับ	แผนภาพ UML	ใช้ในช่วงของ UP	จุดประสงค์
1	Use Case Diagram	Inception / Elaboration	วิเคราะห์ว่าผู้ใช้ต้องการทำอะไรกับระบบ
2	Activity Diagram	Elaboration	อธิบายขั้นตอนการทำงานของแต่ละ Use Case
3	Sequence Diagram	Elaboration	แสดงลำดับการโต้ตอบระหว่าง Actor/Object
4	Communication Diagram	Elaboration	แสดงการสื่อสารและความสัมพันธ์ระหว่าง Object
5	Class Diagram	Elaboration / Construction	ออกแบบ Class, Attribute, Method และความสัมพันธ์
6	State Machine Diagram	Elaboration / Construction	แสดงสถานะของ Object ที่มีการเปลี่ยนแปลงหลายสถานะ
7	Component Diagram	Construction	แสดงส่วนประกอบของโปรแกรมหรือโมดูล
8	Deployment Diagram	Transition	แสดงการติดตั้งระบบบน Server, Client, Database

6.1 ภาพรวม UML Diagrams

อธิบาย UML 1.x = 9 diagrams และ UML 2.x = 14 diagrams ให้ชัด



6.1 ภาพรวม UML Diagrams

อธิบาย UML 1.x = 9 diagrams และ UML 2.x = 14 diagrams ให้ชัด

5 มุมมองหลักของ UML (Unified Modeling Language) เป็นวิธีการที่ใช้ในการแสดงแบบจำลองของระบบเชิงวัตถุ (Object-Oriented System) โดย UML ครอบคลุมมุมมองหลักที่สำคัญ 5 มุมมอง ดังนี้

1. มุมมองการออกแบบ (Design View)

- เน้นโครงสร้างภายในของระบบและองค์ประกอบต่าง ๆ เช่น คลาส, วัตถุ, ความสัมพันธ์ระหว่างคลาส (Classes and Relationships) รวมถึงพฤติกรรมของวัตถุและระบบในช่วงเวลาต่าง ๆ โดยใช้ไดอะแกรม เช่น Class Diagram และ Object Diagram

2. มุมมองเชิงพฤติกรรม (Behavioral View)

- เน้นพฤติกรรมและกระบวนการที่เกิดขึ้นในระบบ รวมถึงลำดับของการกระทำหรือการตอบสนองของระบบที่เกิดจากการโต้ตอบของผู้ใช้งาน ใช้ไดอะแกรม เช่น Use Case Diagram, Sequence Diagram, และ Activity Diagram

3. มุมมองการทำงานร่วมกัน (Collaborative View)

- เน้นการทำงานร่วมกันระหว่างวัตถุต่าง ๆ ในระบบ โดยแสดงให้เห็นถึงการส่งข้อความระหว่างวัตถุเพื่อให้บรรลุเป้าหมายของระบบ ใช้ Communication Diagram และ Interaction Overview Diagram

6.1 ภาพรวม UML Diagrams

อธิบาย UML 1.x = 9 diagrams และ UML 2.x = 14 diagrams ให้ชัด

4. มุมมองการใช้งาน (Use Case View)

- เน้นความสัมพันธ์ระหว่างผู้ใช้งาน (Actor) และระบบว่าผู้ใช้งานต้องการทำอะไร และระบบจะตอบสนองอย่างไร ใช้ Use Case Diagram เพื่อแสดงฟังก์ชันที่ระบบต้องมีเพื่อสนับสนุนการทำงานของผู้ใช้

5. มุมมองการติดตั้งใช้งาน (Implementation View)

- เน้นการแสดงโครงสร้างทางกายภาพของระบบ เช่น โครงสร้างการติดตั้งซอฟต์แวร์หรือฮาร์ดแวร์ และการจัดเก็บคอมโพเนนต์ต่าง ๆ ในระบบ ใช้ไดอะแกรม เช่น Deployment Diagram และ Component Diagram

มุมมองทั้ง 5 นี้ช่วยให้ผู้ออกแบบและพัฒนาสามารถเข้าใจและสร้างระบบเชิงวัตถุได้ครบถ้วนทุกมิติ

อ้างอิง

- Booch, G., Rumbaugh, J., Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley.
- Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley.

6.2 Static View และ Dynamic View

อธิบาย Structural Diagram และ Behavioral Diagram

โดยปกติแล้วในระบบใด ๆ มักประกอบด้วยโครงสร้าง 2 แบบ ได้แก่

- ***Static*(นิ่งๆ ไม่เคลื่อนไหว) และ *dynamic*(เคลื่อนไหว กิจกรรม)**

- **โครงสร้างของ Structural Diagram เป็นแบบ *static***
  - **แสดงองค์ประกอบของระบบ คลาส แอททริบิวต์ เมทธอด และ ความสัมพันธ์ระหว่างคลาส**
  - **ไม่ระบุขั้นตอนการดำเนินงาน ลำดับการทำงานก่อนหลัง**

- **โครงสร้างของ Behavioral Diagrams เป็นแบบ *dynamic***
  - **โครงสร้างที่แสดงถึงการทำงาน หรือ กิจกรรม ที่เกิดขึ้นในระบบที่กำลังพัฒนา**

6.2 Static View และ Dynamic View
อธิบาย Structural Diagram และ Behavioral Diagram

UML Structural Diagrams เป็นกลุ่มของไดอะแกรมใน Unified Modeling Language (UML) ที่ใช้ในการแสดงโครงสร้างภายในของระบบ ซอฟต์แวร์ หรือโมเดลต่าง ๆ โดยเน้นที่องค์ประกอบต่าง ๆ เช่น คลาส วัตถุ โมดูล และความสัมพันธ์ระหว่างองค์ประกอบเหล่านั้น ไดอะแกรมในกลุ่มนี้แสดงให้เห็นถึงความสัมพันธ์เชิงโครงสร้างและการจัดวางขององค์ประกอบในระบบ

ไดอะแกรมในกลุ่ม Structural Diagrams ประกอบด้วย:

1. Class Diagram
2. Object Diagram
3. Component Diagram
4. Deployment Diagram
5. Package Diagram
6. Composite Structure Diagram
7. Profile Diagram

6.2 Static View และ Dynamic View

อธิบาย Structural Diagram และ Behavioral Diagram

1. Static View คืออะไร

Static View คือมุมมองที่แสดงว่า ระบบประกอบด้วยอะไรบ้าง เช่น คลาส วัตถุ ส่วนประกอบ โมดูล ฐานข้อมูล หรืออุปกรณ์ที่เกี่ยวข้องกับระบบ
มุมมองนี้เน้นคำถามว่า
ระบบมีองค์ประกอบอะไร และองค์ประกอบเหล่านั้นสัมพันธ์กันอย่างไร
แผนภาพที่อยู่ในกลุ่มนี้เรียกว่า Structural Diagram หรือ แผนภาพเชิงโครงสร้าง

ตัวอย่าง Structural Diagram ได้แก่

แผนภาพ	ใช้แสดง
Class Diagram	คลาส แอตทริบิวต์ เมธอด และความสัมพันธ์
Object Diagram	ตัวอย่างวัตถุที่เกิดจากคลาส
Component Diagram	ส่วนประกอบหรือโมดูลของโปรแกรม
Deployment Diagram	การติดตั้งระบบบนเครื่อง Server, Client หรือ Network
Package Diagram	การจัดกลุ่มคลาสหรือโมดูล

ตัวอย่างเช่น ระบบขายสินค้าออนไลน์ อาจมีคลาส เช่น

- Customer
- Product
- Order
- Payment
- ShoppingCart

สิ่งเหล่านี้เป็นการอธิบายว่า ระบบประกอบด้วยอะไร จึงอยู่ใน Static View

6.2 Static View และ Dynamic View

อธิบาย Structural Diagram และ Behavioral Diagram

2. Dynamic View คืออะไร

Dynamic View คือมุมมองที่แสดงว่า ระบบทำงานอย่างไร มีขั้นตอนการทำงาน การโต้ตอบการส่งข้อความ หรือการเปลี่ยนสถานะอย่างไร

มุมมองนี้เน้นคำถามว่า

ระบบมีพฤติกรรมอย่างไร และทำงานตามลำดับอย่างไร

แผนภาพที่อยู่ในกลุ่มนี้เรียกว่า Behavioral Diagram หรือ แผนภาพเชิงพฤติกรรม

ตัวอย่าง Behavioral Diagram ได้แก่

แผนภาพ	ใช้แสดง
Use Case Diagram	ผู้ใช้ทำอะไรกับระบบ
Activity Diagram	ขั้นตอนหรือกระบวนการทำงาน
Sequence Diagram	ลำดับการส่งข้อความระหว่าง Actor/Object
Communication Diagram	การสื่อสารระหว่างวัตถุ
State Machine Diagram	สถานะและการเปลี่ยนสถานะของวัตถุ

ตัวอย่างเช่น ระบบขายสินค้าออนไลน์ มีขั้นตอนว่า

- ลูกค้าเลือกสินค้า
- เพิ่มสินค้าลงตะกร้า
- ยืนยันคำสั่งซื้อ
- ชำระเงิน
- ระบบบันทึกคำสั่งซื้อ
- แจ้งผลการสั่งซื้อ

ส่วนนี้เป็นการอธิบายว่า ระบบทำงานอย่างไร จึงอยู่ใน Dynamic View

6.2 Static View และ Dynamic View

อธิบาย Structural Diagram และ Behavioral Diagram

ตัวอย่างอธิบายให้นักศึกษาเข้าใจง่าย
สมมติว่าเราจะออกแบบ ระบบ Login

Static View จะถามว่า ระบบต้องมีอะไรบ้าง เช่น
- **User**
- **LoginForm**
- **AuthController**
- **Database**

จึงอาจเขียนเป็น Class Diagram

Dynamic View จะถามว่า ระบบ Login ทำงานอย่างไร เช่น
- **User กรอก Username และ Password**
- **LoginForm ส่งข้อมูลไปตรวจสอบ**
- **AuthController ตรวจสอบกับ Database**
- **Database ส่งผลลัพธ์กลับ**
- **ระบบแจ้งว่า Login สำเร็จหรือไม่สำเร็จ**

จึงอาจเขียนเป็น Sequence Diagram หรือ Activity Diagram

6.3 Behavioral Diagrams คืออะไร

แยก Use Case, Activity, State Machine และ Interaction Diagram

Behavioral Diagrams คืออะไร

Behavioral Diagrams หรือ แผนภาพเชิงพฤติกรรม คือกลุ่มแผนภาพ UML ที่ใช้แสดงว่าระบบทำงานอย่างไร มีขั้นตอน เหตุการณ์ การเปลี่ยนสถานะ และการโต้ตอบระหว่างผู้ใช้กับระบบหรือระหว่างวัตถุในระบบ

พูดง่าย ๆ คือ

Behavioral Diagrams = แผนภาพที่อธิบายพฤติกรรมและการทำงานของระบบ

แผนภาพที่อยู่ในกลุ่ม Behavioral Diagrams

แผนภาพ	ใช้อธิบาย
Use Case Diagram	ผู้ใช้หรือ Actor ต้องการทำอะไรกับระบบ
Activity Diagram	ขั้นตอนการทำงานหรือ Workflow ของระบบ
State Machine Diagram	สถานะของวัตถุและการเปลี่ยนสถานะ
Interaction Diagram	การโต้ตอบหรือการส่งข้อความระหว่างวัตถุ

Behavioral Diagrams คือแผนภาพ UML ที่ใช้แสดงพฤติกรรมของระบบ โดยอธิบายว่า ผู้ใช้ทำอะไรกับระบบ ระบบทำงานเป็นขั้นตอนอย่างไร วัตถุเปลี่ยนสถานะอย่างไร และวัตถุต่าง ๆ โต้ตอบกันอย่างไร ผ่านแผนภาพ Use Case, Activity, State Machine และ Interaction Diagram.

ตัวอย่างให้เข้าใจง่าย

ถ้าเป็น ระบบสั่งซื้อสินค้าออนไลน์

- Use Case Diagram แสดงว่า ลูกค้าสามารถสมัครสมาชิก เข้าสู่ระบบ สั่งซื้อสินค้า และชำระเงินได้
- Activity Diagram แสดงขั้นตอนตั้งแต่เลือกสินค้า → ใส่ตะกร้า → ชำระเงิน → ยืนยันคำสั่งซื้อ
- State Machine Diagram แสดงสถานะคำสั่งซื้อ เช่น รอชำระเงิน → ชำระเงินแล้ว → จัดส่งแล้ว
- Interaction Diagram แสดงการส่งข้อมูลระหว่าง User, ระบบสินค้า, ระบบชำระเงิน และฐานข้อมูล

6.3 Behavioral Diagrams คืออะไร

แยก Use Case, Activity, State Machine และ Interaction Diagram

ใน UML (Unified Modeling Language) Behavioral Diagrams คือ *Behavioral Diagrams* เป็นโครงสร้างแบบ dynamic เป็นกลุ่มของไดอะแกรมที่ใช้แสดงพฤติกรรมของระบบ ซึ่งรวมถึงการโต้ตอบระหว่างผู้ใช้งานหรือวัตถุต่าง ๆ และการดำเนินงานภายในระบบ ไดอะแกรมในกลุ่มนี้จะแสดงกระบวนการ การส่งข้อความ หรือเหตุการณ์ต่าง ๆ ที่เกิดขึ้นในระบบ

Behavioral Diagrams ประกอบด้วยไดอะแกรมหลักดังนี้:

1. Use Case Diagram
2. Activity Diagram
3. Sequence Diagram
4. Communication Diagram (หรือที่เรียกว่า Communication Diagram (formerly Collaboration Diagram))

แผนภาพปฏิสัมพันธ์ (Interaction Diagrams)

5. State Machine Diagram (หรือ Statechart Diagram)
6. Interaction Overview Diagram
7. Timing Diagram

6.4 Interaction Diagrams คืออะไร

อธิบายว่าใช้แสดงการโต้ตอบระหว่าง Actor/Object ผ่าน Message

Interaction Diagrams คืออะไร

Interaction Diagrams หรือ แผนภาพปฏิสัมพันธ์ คือกลุ่มแผนภาพ UML ที่ใช้แสดงว่า Actor และ Object ต่าง ๆ ติดต่อสื่อสารกันอย่างไร ภายในระบบ โดยการสื่อสารนั้นจะแสดงผ่านสิ่งที่เรียกว่า Message

พูดง่าย ๆ คือ

Interaction Diagrams ใช้อธิบายการโต้ตอบระหว่าง Actor / Object ผ่าน Message

1. Actor คืออะไร

Actor คือ ผู้ใช้หรือสิ่งภายนอกที่เข้ามาโต้ตอบกับระบบ เช่น คน ระบบอื่น หรืออุปกรณ์ภายนอก

Actor	ตัวอย่าง
ผู้ใช้ระบบ	ลูกค้า, นักศึกษา, ผู้ดูแลระบบ
ระบบภายนอก	Payment Gateway, Email Server
อุปกรณ์ภายนอก	เครื่องสแกนบัตร, Barcode Scanner

ตัวอย่างในระบบ Login

User คือ Actor ที่กรอก Username และ Password เพื่อเข้าสู่ระบบ

6.4 Interaction Diagrams คืออะไร

อธิบายว่าใช้แสดงการโต้ตอบระหว่าง Actor/Object ผ่าน Message

2. Object คืออะไร

Object คือ วัตถุหรือส่วนประกอบภายในระบบที่ทำหน้าที่รับ ส่ง หรือประมวลผลข้อมูล

ตัวอย่าง Object ในระบบ Login

Object	หน้าที่
LoginForm	รับข้อมูล Username/Password
AuthController	ตรวจสอบการเข้าสู่ระบบ
UserDatabase	ค้นหาข้อมูลผู้ใช้
Session	จัดเก็บสถานะการเข้าสู่ระบบ

Object เหล่านี้จะทำงานร่วมกันเพื่อให้ Use Case หนึ่งสำเร็จ

3. Message คืออะไร

Message คือ ข้อความ คำสั่ง หรือการเรียกใช้เมธอดที่ส่งจาก Actor/Object หนึ่งไปยัง Actor/Object อื่น

ตัวอย่าง Message

Message	ความหมาย
login()	เรียกใช้การเข้าสู่ระบบ
checkPassword()	ตรวจสอบรหัสผ่าน
getUserData()	ดึงข้อมูลผู้ใช้
return result	ส่งผลลัพธ์กลับ

ตัวอย่างเช่น
User ส่ง Message ไปยัง LoginForm ว่า submitLogin()
LoginForm ส่งต่อไปยัง AuthController ว่า validateUser()
AuthController ส่งไปยัง Database ว่า checkPassword()

6.4 Interaction Diagrams คืออะไร

อธิบายว่าใช้แสดงการโต้ตอบระหว่าง Actor/Object ผ่าน Message

4. Interaction Diagrams ใช้ทำอะไร

Interaction Diagrams ใช้เพื่อแสดงว่าใน Use Case หนึ่ง ๆ มีการติดต่อกันระหว่างวัตถุอย่างไร เช่น Use Case “เข้าสู่ระบบ” หรือ “สั่งซื้อสินค้า”
ช่วยตอบคำถามสำคัญ เช่น

คำถาม	คำตอบจาก Interaction Diagram
ใครเป็นผู้เริ่มต้นการทำงาน	Actor
มี Object ใดเกี่ยวข้องบ้าง	LoginForm, Controller, Database
Object ใดส่ง Message ไปหาใคร	LoginForm → AuthController
ลำดับการทำงานเป็นอย่างไร	ส่ง Message ตามลำดับเวลา
ระบบตอบกลับอย่างไร	Return Message

5. ประเภทของ Interaction Diagrams

Interaction Diagrams ที่นิยมใช้มี 2 แบบหลัก

แผนภาพ	จุดเน้น
Sequence Diagram	เน้นลำดับเวลาก่อน-หลังของ Message
Communication Diagram หรือ Collaboration Diagram เดิม	เน้นความสัมพันธ์และการสื่อสารระหว่าง Object

6.4 Interaction Diagrams คืออะไร

อธิบายว่าใช้แสดงการโต้ตอบระหว่าง Actor/Object ผ่าน Message

ตัวอย่าง Interaction Diagram: ระบบ Login

ลำดับการโต้ตอบของระบบ Login สามารถอธิบายได้ดังนี้

1. User กรอก Username และ Password
2. LoginForm รับข้อมูลจาก User
3. LoginForm ส่ง Message validateUser() ไปยัง AuthController
4. AuthController ส่ง Message checkUser() ไปยัง UserDatabase
5. UserDatabase ส่งผลลัพธ์กลับ
6. AuthController ตรวจสอบผลลัพธ์
7. ระบบแจ้งว่า Login สำเร็จหรือไม่สำเร็จ

เขียนแบบย่อได้ว่า

User → LoginForm → AuthController → UserDatabase → AuthController → LoginForm → User

Interaction Diagrams คือแผนภาพ UML ที่ใช้แสดงการโต้ตอบระหว่าง Actor และ Object ภายในระบบ ผ่านการส่ง Message เพื่ออธิบายว่าระบบทำงานร่วมกันอย่างไรในแต่ละ Use Case

สรุป

Interaction Diagrams = แผนภาพที่แสดงว่า Actor/Object ส่ง Message ติดต่อกันอย่างไร

6.5 ประเภทของ Interaction Diagrams

Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram

รูปแบบของ Interaction Diagrams

Interaction Diagrams ใน UML 2.x มี 4 ประเภท ได้แก่ Sequence Diagram, Communication Diagram, Timing Diagram และ Interaction Overview Diagram อย่างไรก็ตาม ในรายวิชา OOAD ระดับพื้นฐานมักเน้น Sequence Diagram และ Communication Diagram เพราะใช้บ่อยในการวิเคราะห์ Use Case

รูปแบบ	จุดเน้น	แผนภาพ
Time-based	เน้นลำดับเหตุการณ์ก่อน-หลังตามเวลา	Sequence Diagram
Organization-based	เน้นโครงสร้างการสื่อสารและความสัมพันธ์ระหว่างวัตถุ	Communication Diagram หรือ Collaboration Diagram เดิม

Interaction Diagrams คือแผนภาพที่ใช้แสดงการปฏิสัมพันธ์ระหว่างวัตถุหลายตัวภายในระบบ โดยมักใช้เพื่ออธิบายการทำงานของ Use Case ที่สำคัญ แสดงให้เห็นว่าวัตถุต่าง ๆ ติดต่อสื่อสารกันอย่างไรผ่าน Message โดยแบ่งเป็น 2 รูปแบบหลัก คือ Sequence Diagram ที่เน้นลำดับเวลา และ Communication Diagram ที่เน้นโครงสร้างการสื่อสารระหว่างวัตถุ

6.5 ประเภทของ Interaction Diagrams

Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram

- โดยปกติ ระบบหนึ่ง ๆ มักจะประกอบไปด้วย หลายยูสเคส(use case) ดังนั้น
- Interaction diagrams จะแสดงเฉพาะบาง Use Case เท่านั้นเนื่องจาก ไม่สามารถแสดงแทนทั้งหมดได้
- จะนำเฉพาะยูสเคส ที่สำคัญ หรือที่เราสนใจมาเขียนเป็นแผนภาพปฏิสัมพันธ์ (interaction diagrams)
  - Sequence diagram
  - Communication Diagram (formerly Collaboration Diagram)

จาก *Interaction diagrams*

- เราจะสรุปได้ว่า ทั้งสองไดอะแกรม เป็นไดอะแกรมปฏิสัมพันธ์ (มีการโต้ตอบหรือมีการสื่อสารกันระหว่างวัตถุ) ระหว่างวัตถุ แต่เน้นคนละอย่าง
  - Sequence เน้นที่ลำดับ เหตุการณ์ ก่อนหลัง
  - Collaboration เน้นที่การจัดการ โครงสร้างการเชื่อมต่อ ประสาน
- เราอาจจะใช้ทั้งสองแผนภาพ หรือเลือกนำเสนอเพียงอย่างใดอย่างหนึ่ง
- ขึ้นอยู่กับระบบที่เราพัฒนาว่ามีความซับซ้อนเพียงใด หรือ
- การพิจารณาร่วมกันของทีมพัฒนา

6.5

ประเภทของ Interaction Diagrams

Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram

ประเภทของ Interaction Diagrams

Interaction Diagrams ใน UML 2.x ประกอบด้วย 4 ประเภท ได้แก่ Sequence Diagram, Communication Diagram, Timing Diagram และ Interaction Overview Diagram โดยในรายวิชานี้จะเน้น Sequence Diagram และ Communication Diagram เพราะเป็นแผนภาพที่ใช้บ่อยที่สุดในการวิเคราะห์และออกแบบระบบเชิงวัตถุ

ทั้งสอง จะเรียก ว่า Interaction Diagrams

6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

Sequence & Communication Diagram

Sequence Diagram และ Communication Diagram เป็นแผนภาพในกลุ่ม Interaction Diagrams ที่ใช้จำลองการปฏิสัมพันธ์ระหว่างวัตถุในระบบ ทั้งสองแผนภาพสามารถใช้เพื่ออธิบายการทำงานของระบบได้ทั้งในระดับ

- การปฏิสัมพันธ์ของวัตถุในระบบโดยรวม
- การปฏิสัมพันธ์เฉพาะใน Use Case ใด Use Case หนึ่ง

โดยทั่วไปไม่จำเป็นต้องสร้างทั้งสองแผนภาพเสมอไป อาจเลือกใช้เพียงแผนภาพใดแผนภาพหนึ่งก็ได้ เพราะทั้งสองแผนภาพสื่อความหมายใกล้เคียงกัน แต่มีจุดเน้นต่างกัน

แผนภาพ	จุดเน้น
Sequence Diagram	เน้นลำดับเวลาการทำงานก่อน-หลังของ Message
Communication Diagram	เน้นการส่ง Message และความสัมพันธ์ระหว่าง Object

ในระบบงานหนึ่ง อาจเลือกใช้เพียง Sequence Diagram หรือ Communication Diagram หรือใช้ทั้งสองแผนภาพร่วมกันก็ได้ ขึ้นอยู่กับความซับซ้อนของระบบและความเหมาะสมในการนำเสนอ

6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.1 ความหมายของ Sequence Diagram

Sequence Diagram หรือ แผนภาพลำดับ คือแผนภาพ UML ที่ใช้แสดง ลำดับการโต้ตอบระหว่าง Actor และ Object ภายในระบบ โดยแสดงการส่ง Message จากวัตถุหนึ่งไปยังอีกวัตถุหนึ่งตามลำดับเวลา

แผนภาพนี้เหมาะสำหรับอธิบายการทำงานของ Use Case ใด Use Case หนึ่ง เช่น การเข้าสู่ระบบ การสมัครสมาชิก การสั่งซื้อสินค้า หรือการชำระเงิน

Sequence Diagram ใช้อธิบายว่า ใครติดต่อกับใคร ส่ง Message อะไร และเกิดขึ้นก่อน-หลังอย่างไร จุดเด่นของ Sequence Diagram คือการแสดง ลำดับเวลา โดยอ่านจาก บนลงล่าง ข้อความหรือ Message ที่อยู่ด้านบนจะเกิดขึ้นก่อน และ Message ที่อยู่ถัดลงมาจะเกิดขึ้นภายหลัง

ตัวอย่างเช่น Use Case: เข้าสู่ระบบ

1. User กรอก Username และ Password
2. LoginForm ส่งข้อมูลไปยัง AuthController
3. AuthController ตรวจสอบข้อมูลกับ Database
4. Database ส่งผลลัพธ์กลับ
5. ระบบแจ้งผลการเข้าสู่ระบบ

6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.2 องค์ประกอบของ Sequence Diagram

ในยูเอ็มแอลส่วนประกอบของแผนภาพลำดับ (Sequence diagram) มีดังนี้

องค์ประกอบ	ความหมาย
Actor	ผู้ใช้งานหรือระบบภายนอกที่เริ่มต้นหรือมีส่วนร่วมในการทำงาน
Object / Participant	วัตถุหรือส่วนประกอบของระบบที่เข้าร่วมการโต้ตอบ เช่น Form, Controller, Service หรือ Database
Lifeline	เส้นประแนวตั้งที่แสดงช่วงเวลาการมีอยู่ของ Participant เช่น Actor, Object หรือ Component ในการโต้ตอบหนึ่ง ๆ
Message	ข้อความ คำสั่ง หรือการเรียกใช้เมธอดที่ส่งระหว่าง Participant
Activation Bar	แถบแสดงช่วงเวลาที่ Participant กำลังทำงานหรือประมวลผล
Return Message	ข้อความตอบกลับจาก Participant ที่ถูกเรียกใช้งาน
Create Message	Message ที่ใช้สร้าง Object หรือ Participant ใหม่
Destroy Message	Message ที่ใช้ทำลายหรือสิ้นสุดการทำงานของ Object

6.6

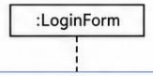
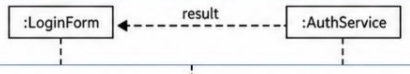
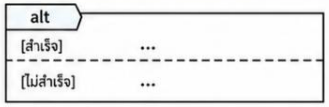
Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.3 สัญลักษณ์สำคัญใน Sequence Diagram

สัญลักษณ์สำคัญใน Sequence Diagram

ตัวอย่างสัญลักษณ์และความหมายในแผนภาพลำดับ

	สัญลักษณ์	ชื่อ	ใช้แทน
1		Actor	ผู้ใช้หรือระบบภายนอก
2		Object	วัตถุในระบบ
3		Lifeline	เส้นแสดงช่วงชีวิตของวัตถุ
4		Call Message	การเรียกใช้เมธอดหรือส่งคำสั่ง
5		Return Message	ค่าหรือผลลัพธ์ที่ส่งกลับ
6		Activation Bar	ช่วงเวลาที่วัตถุกำลังทำงาน
7		Destroy Message	การทำลายวัตถุ
8		Combined Fragment	กรอบเงื่อนไข เช่น alt, loop, opt

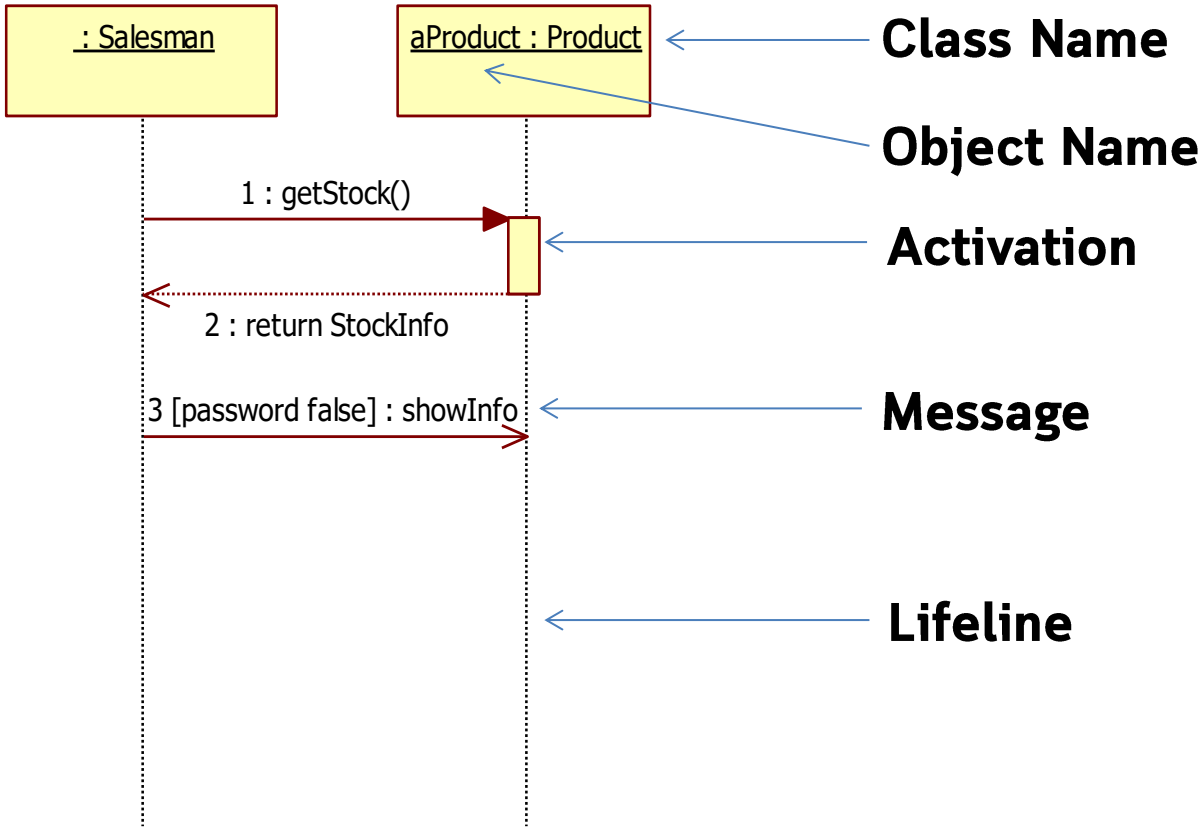
สรุป Sequence Diagram ใช้แสดงลำดับการโต้ตอบระหว่าง Actor และ Object ผ่าน Message โดยอ่านจากบนลงล่าง

6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.3 สัญลักษณ์สำคัญใน Sequence Diagram

ตัวอย่างสัญลักษณ์ใน Sequence Diagram

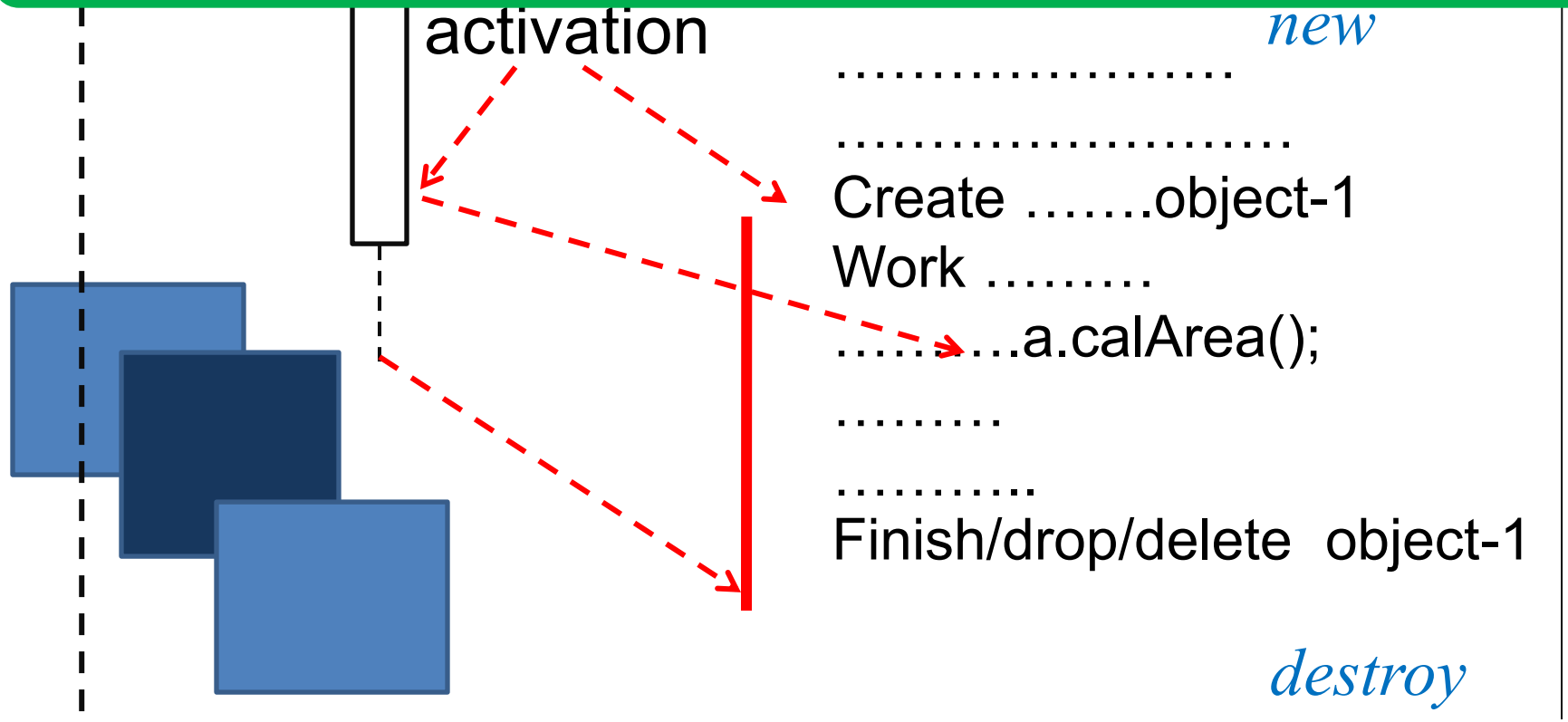


6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.3 สัญลักษณ์สำคัญใน Sequence Diagram

2. เส้นชีวิต (Life line) คือ เส้นประแนวตั้งที่แสดงช่วงเวลาการมีอยู่ของ Participant เช่น Actor, Object หรือ Component ในการโต้ตอบหนึ่ง ๆ



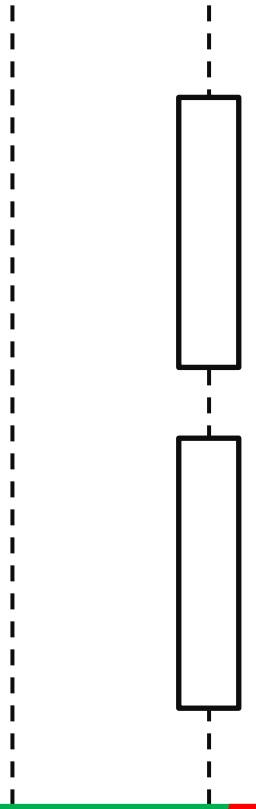
6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.3 สัญลักษณ์สำคัญใน Sequence Diagram

2. เส้นชีวิต (Life line) คือเส้นประแนวตั้งที่แสดงช่วงเวลาการมีอยู่ของ Participant เช่น Actor, Object หรือ Component ในการโต้ตอบหนึ่ง ๆ

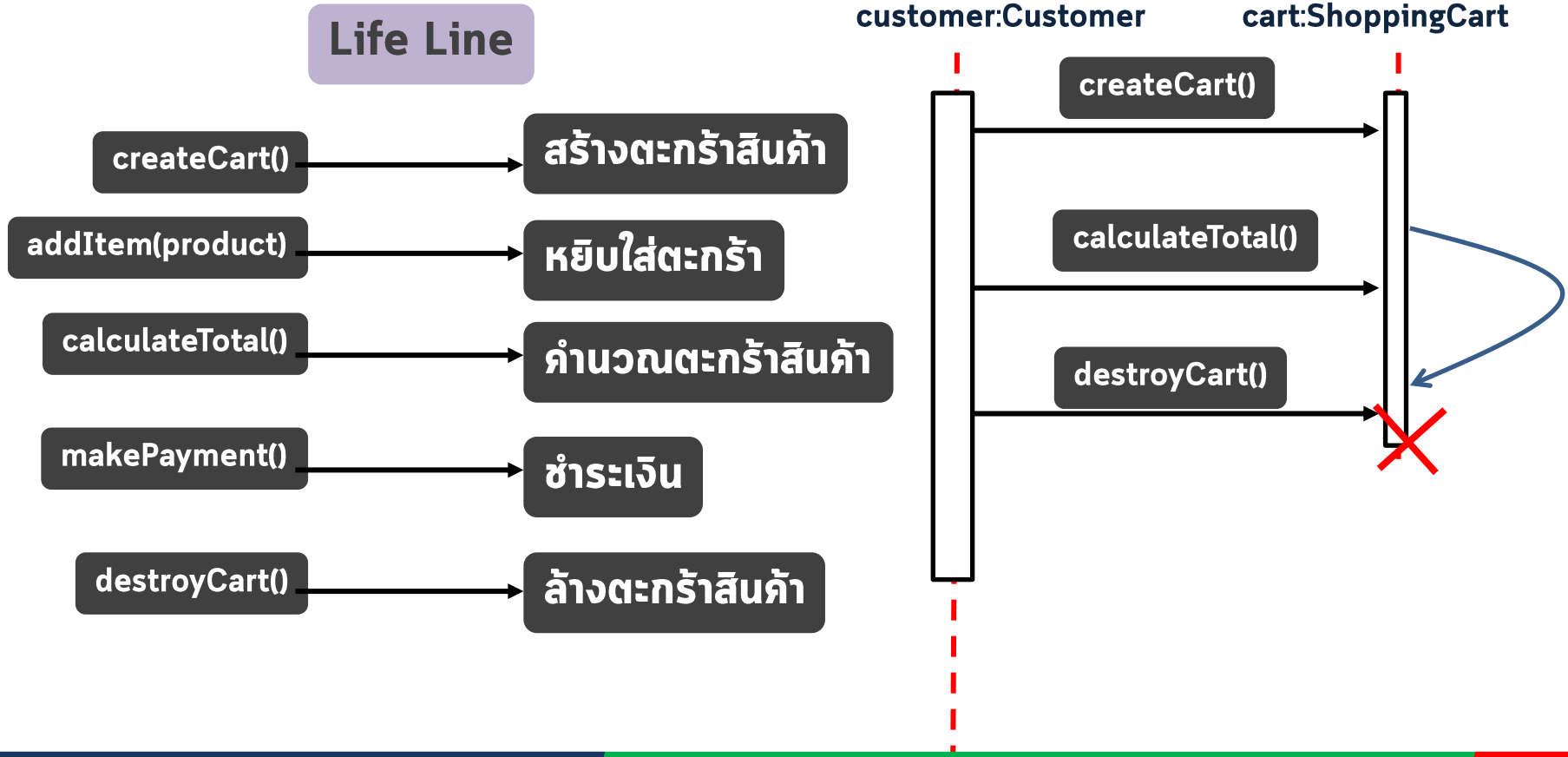
- Create .....
- ...work()
- ....
- ....
- ...call method
- Finalize ... delete destroy



6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

2. เส้นชีวิต (Life line) คือเส้นประแนวตั้งที่แสดงช่วงเวลาการมีอยู่ของ Participant เช่น Actor, Object หรือ Component ในการโต้ตอบหนึ่ง ๆ

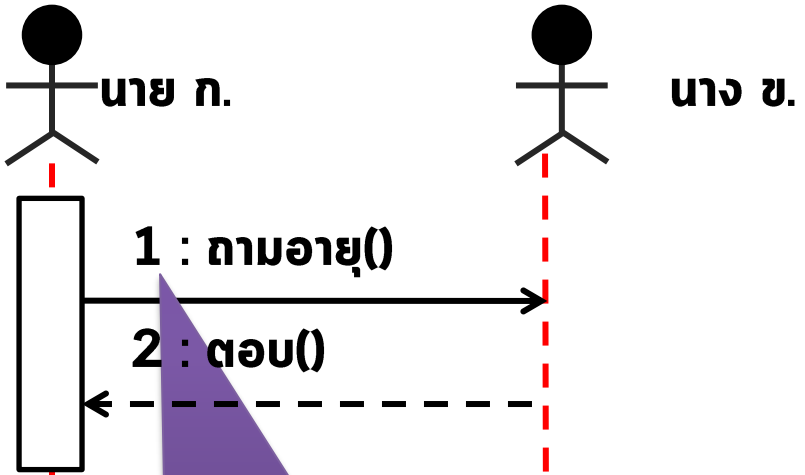


6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.3 สัญลักษณ์สำคัญใน Sequence Diagram

2. เส้นชีวิต (Life line) คือเส้นประแนวตั้งที่แสดงช่วงเวลาการมีอยู่ของ Participant เช่น Actor, Object หรือ Component ในการโต้ตอบหนึ่ง ๆ



เส้นแนวตั้ง เป็นเส้นชีวิต (lifeline)

Lifeline

หมายเลขบอกลำดับ

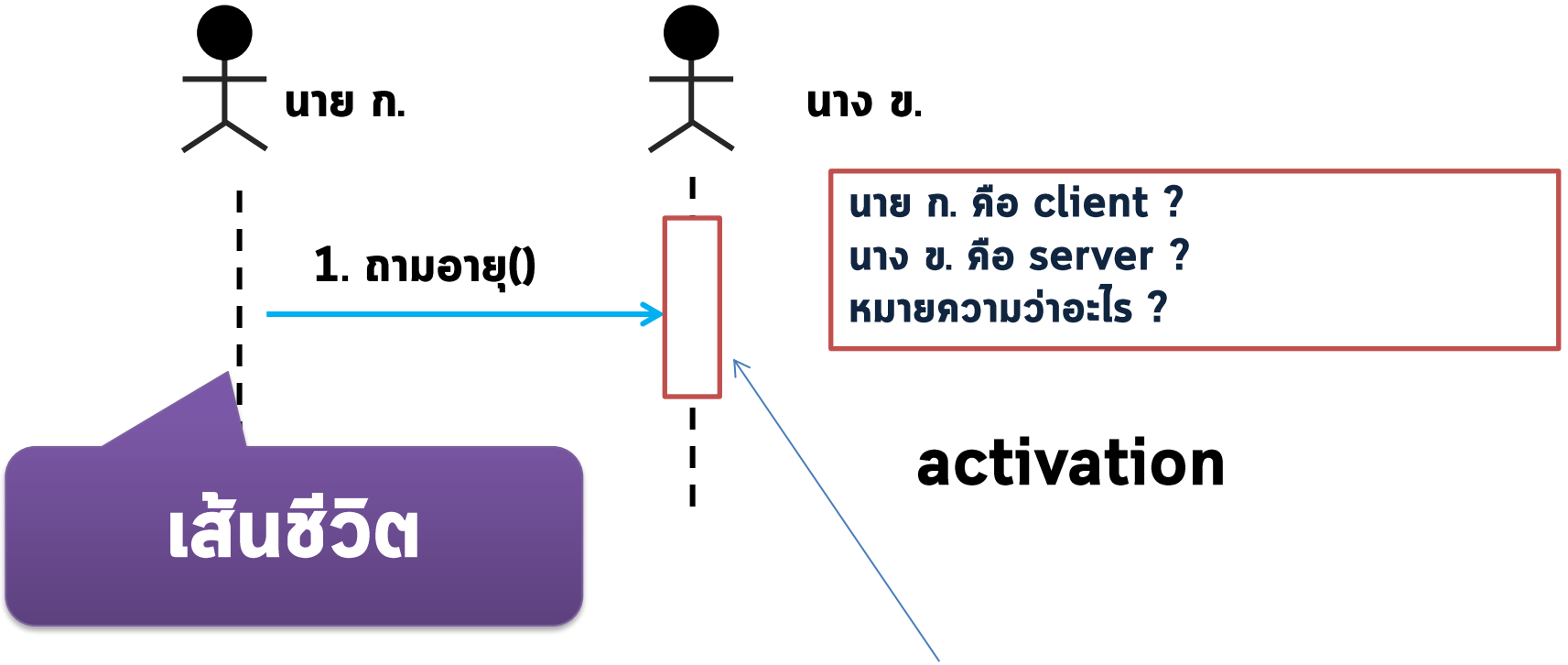
สัญลักษณ์ใน UML ไม่ได้เน้นสี ไม่ต้องใส่สี

6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.3 สัญลักษณ์สำคัญใน Sequence Diagram

2. เส้นชีวิต (Life line) คือเส้นประแนวตั้งที่แสดงช่วงเวลาการมีอยู่ของ Participant เช่น Actor, Object หรือ Component ในการโต้ตอบหนึ่ง ๆ



6.6 Sequence Diagram

ความหมาย องค์ประกอบ สัญลักษณ์ และหลักการอ่าน

6.6.4 หลักการอ่าน Sequence Diagram

การอ่าน Sequence Diagram ควรอ่านตามลำดับดังนี้

1. ดู Actor ทางซ้ายสุด
   เพื่อดูว่าใครเป็นผู้เริ่มต้นการทำงาน
2. ดู Object ที่เกี่ยวข้อง
   เช่น Form, Controller, Database หรือ Service
3. อ่าน Message จากบนลงล่าง
   Message ด้านบนเกิดก่อน Message ด้านล่างเกิดทีหลัง
4. ดูทิศทางของลูกศร
   ลูกศรชี้ไปที่วัตถุใด แสดงว่าวัตถุนั้นเป็นผู้รับ Message
5. ดู Activation Bar
   เพื่อดูว่าวัตถุใดกำลังทำงานหรือประมวลผล
6. ดู Return Message
   เพื่อดูว่ามีการส่งผลลัพธ์กลับไปยังผู้เรียกหรือไม่
7. ดูเงื่อนไขพิเศษ
   เช่น alt หมายถึงทางเลือก, loop หมายถึงการทำซ้ำ, opt หมายถึงกรณีที่อาจเกิดขึ้นหรือไม่เกิดขึ้นก็ได้

6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.1 ความหมายของ Message

- ข่าวสาร Message คือ ข้อความ คำสั่ง หรือการเรียกใช้เมธอดที่ส่งจาก Participant หนึ่งไปยังอีก Participant หนึ่ง เพื่อให้เกิดการทำงาน การตอบกลับ การสร้างวัตถุ หรือการทำลายวัตถุ โดยเกิดจาก
  - การสร้างวัตถุ (create)
  - การเรียกใช้งานเมธอดใด ๆ ของวัตถุ (call)
  - การส่งสัญญาณ (Signal) คือ Message แบบส่งสัญญาณหรือแจ้งเหตุการณ์ โดยผู้ส่งไม่จำเป็นต้องรอผลลัพธ์ทันที เช่น sendNotification(), timeout(), alert()
  - การทำลายวัตถุ (Destroy)

Button → CalculatorController : power(2,3)
CalculatorController → Textbox : displayResult(8)

การเรียกใช้งานเมธอดใด ๆ ของวัตถุ (call)

6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

ข่าวสาร (Message) เป็นการติดต่อที่ส่งจากวัตถุหนึ่งไปยังอีกวัตถุหนึ่งหรืออาจส่งกลับมาหาตัวเองก็ได้โดยจะแบ่งการติดต่อออกเป็น 5 แบบตามลักณะของการส่งได้ดังนี้คือ

1. Call Message คือ Message ที่ Sender หรือ Caller ส่งไปยัง Receiver หรือ Callee เพื่อเรียกใช้เมธอดหรือฟังก์ชัน เช่น **x.draw()**
2. Return – Message ที่ใช้ส่งข้อมูลที่ถูกร้องขอกลับไปยัง Sender
3. Send –Message การส่งสัญญาณเพื่อบอกหรือกระตุ้นวัตถุอื่น แต่ไม่ใช่การเรียกใช้
4. Create – Message ที่ส่งออกไปโดยมีจุดประสงค์เพื่อให้เกิดการสร้าง วัตถุของคลาสขึ้น
5. Destroy – Message ที่ส่งออกไปโดยมีจุดประสงค์เพื่อให้วัตถุนั้นทำลายตัวเอง

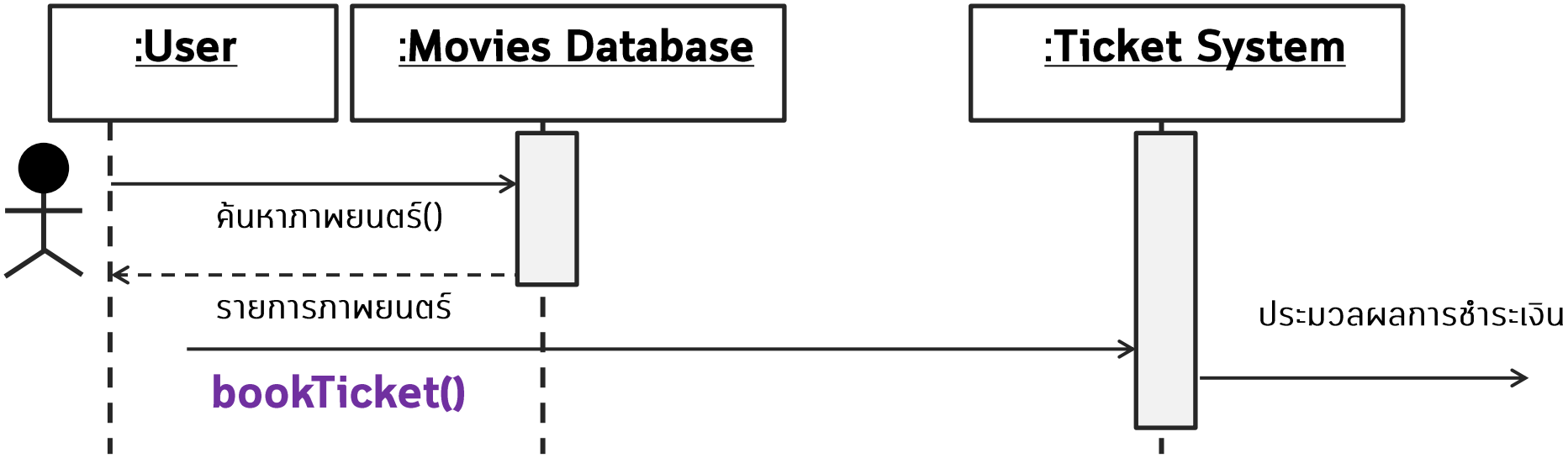
6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

1. Call Message

- คำอธิบาย: เป็นการเรียกใช้เมธอดหรือฟังก์ชันจากวัตถุหนึ่งไปยังอีกวัตถุหนึ่ง
- ตัวอย่าง: ในระบบการจองตั๋วหนัง เมื่อผู้ใช้ (User) คลิกที่ปุ่ม "จองตั๋ว"
  - User → Ticket System: bookTicket()
  - การเรียกใช้เมธอด bookTicket() ใน Ticket System เพื่อเริ่มกระบวนการจอง



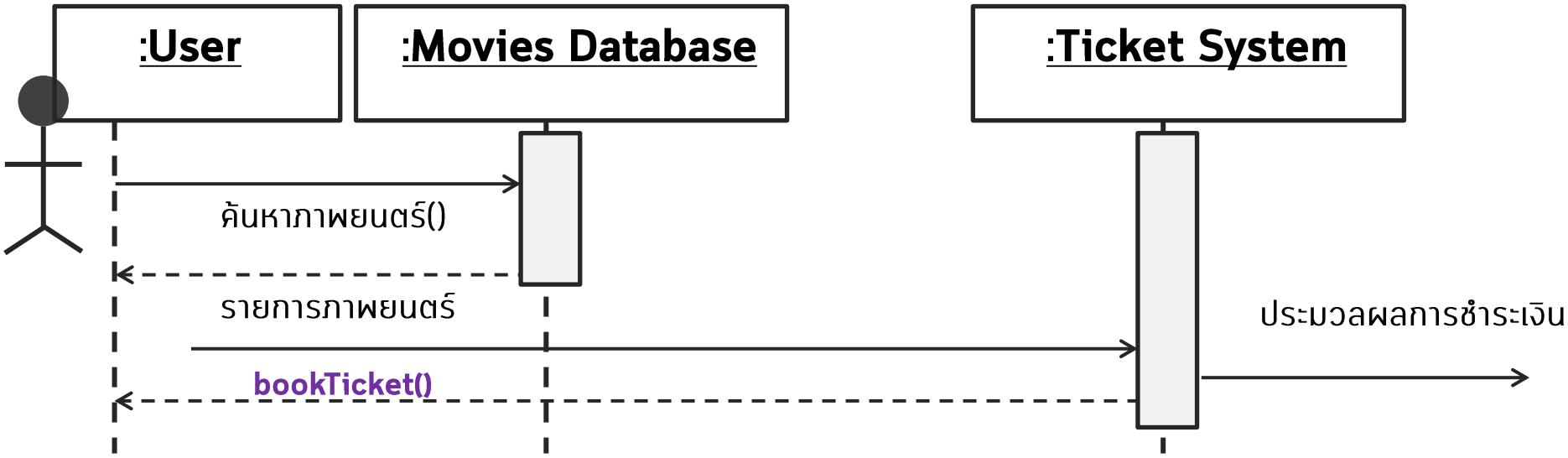
6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

2. Return Message

- คำอธิบาย: เป็นการส่งข้อมูลที่ถูกขอจากวัตถุกลับไปยังวัตถุที่ร้องขอ
- ตัวอย่าง: หลังจาก Ticket System ทำการบันทึกตั๋วแล้ว
  - Ticket System → User: returnConfirmation()
  - ส่งการยืนยันการจองกลับไปยังผู้ใช้



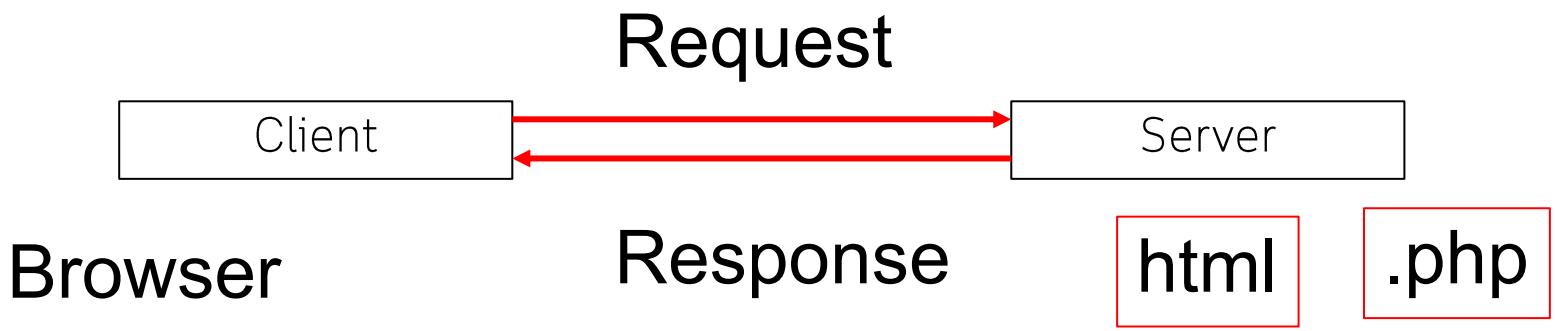
6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

2. Return Message

Return – Message ที่ใช้ส่งข้อมูลที่ถูกร้องขอกลับไปยัง Sender



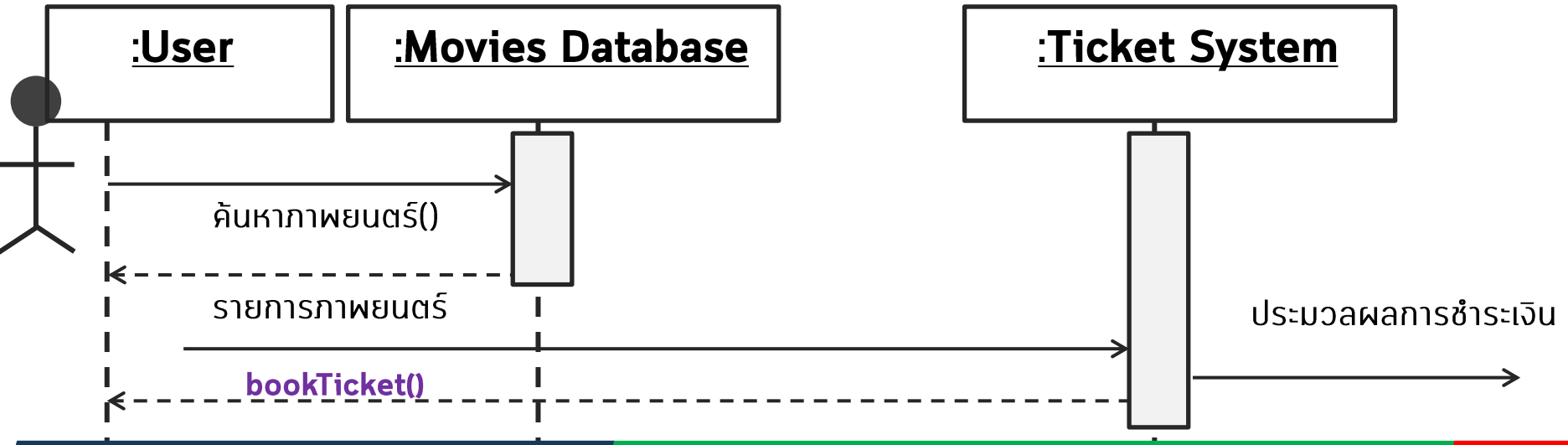
6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

3. Send Message

- คำอธิบาย: เป็นการส่งสัญญาณหรือกระตุ้นวัตถุอื่นโดยไม่ใช่การเรียกใช้เมธอด
- ตัวอย่าง: ในระบบการแจ้งเตือนให้ผู้ใช้ทราบ
  - Notification System → User: alertUser()
  - การแจ้งเตือนผู้ใช้เกี่ยวกับข้อมูลใหม่ เช่น คะแนนใหม่ในระบบการจัดการโรงเรียน



6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

4. Create – Message

Create – Message ที่ส่งออกไปโดยมีจุดประสงค์เพื่อให้เกิดการสร้าง วัตถุของคลาสขึ้น

```
public class test_person {
  Person p1 = new Person();   //  Create message
  p1.setName("Dr. Nattapong"); // call message
}
```

```
class person {
        String name;
        String id;
        String address;
        public setName(String n) {
                Name = n;
        }
}
```

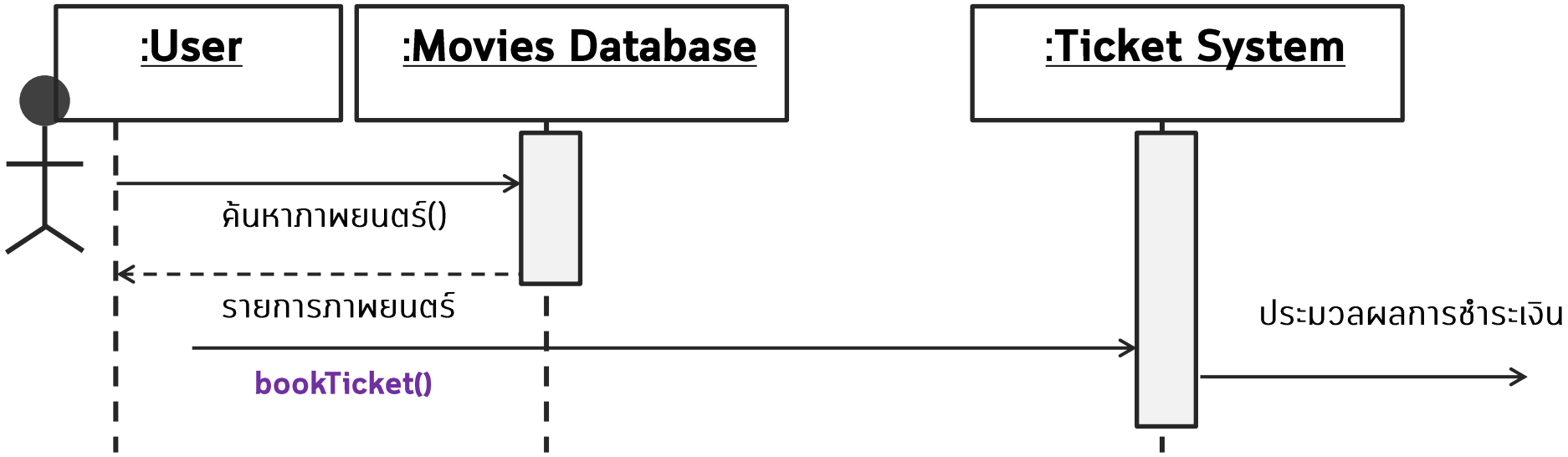
6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

4. Create Message

- คำอธิบาย: ส่งข้อความที่มีจุดประสงค์เพื่อสร้างวัตถุใหม่ของคลาส
- ตัวอย่าง: เมื่อระบบต้องการสร้างบัตรรายงานผลใหม่
  - Grade System → Report Card: createReportCard()
  - ส่งข้อความเพื่อสร้างวัตถุใหม่ชื่อ Report Card



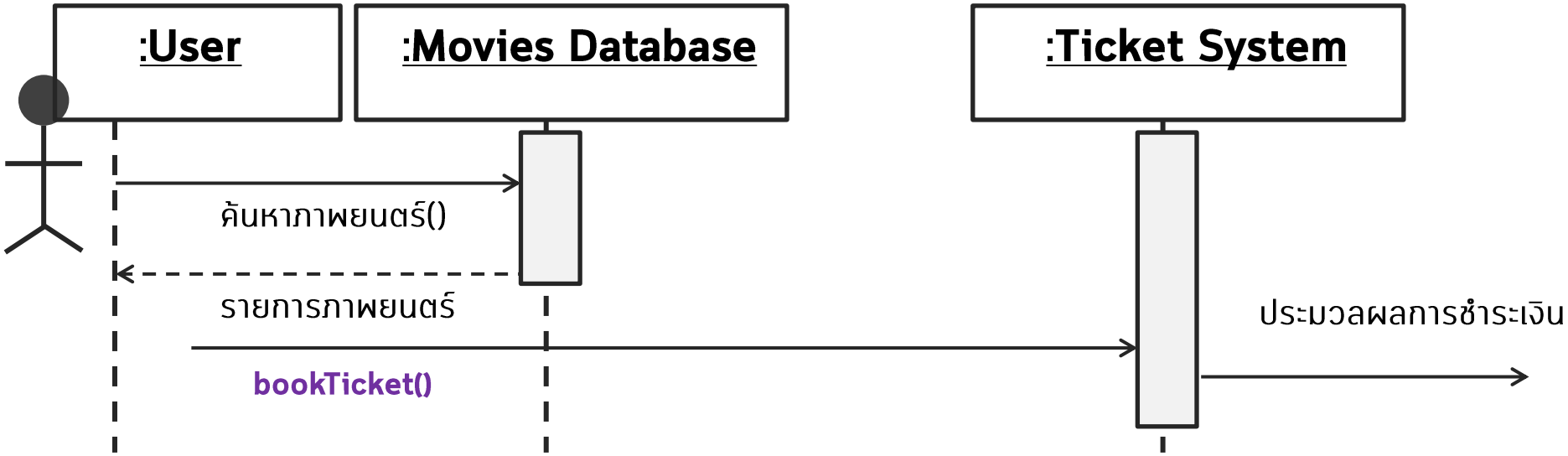
6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

6.7.2 ประเภทของข่าวสาร

5. Destroy Message

- คำอธิบาย: ส่งข้อความเพื่อให้วัตถุทำลายตัวเอง
- ตัวอย่าง: หลังจากที่ผู้ใช้เสร็จสิ้นการดูข้อมูล
  - User → Session: destroySession()
  - การส่งข้อความเพื่อทำลายวัตถุ Session เมื่อผู้ใช้ล็อกเอาท์จากระบบ



6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

นอกจากนี้ข่าวสารนั้นยังสามารถแบ่งออกเป็นประเภทต่าง ๆ ตามวิธีการส่งได้ดังนี้

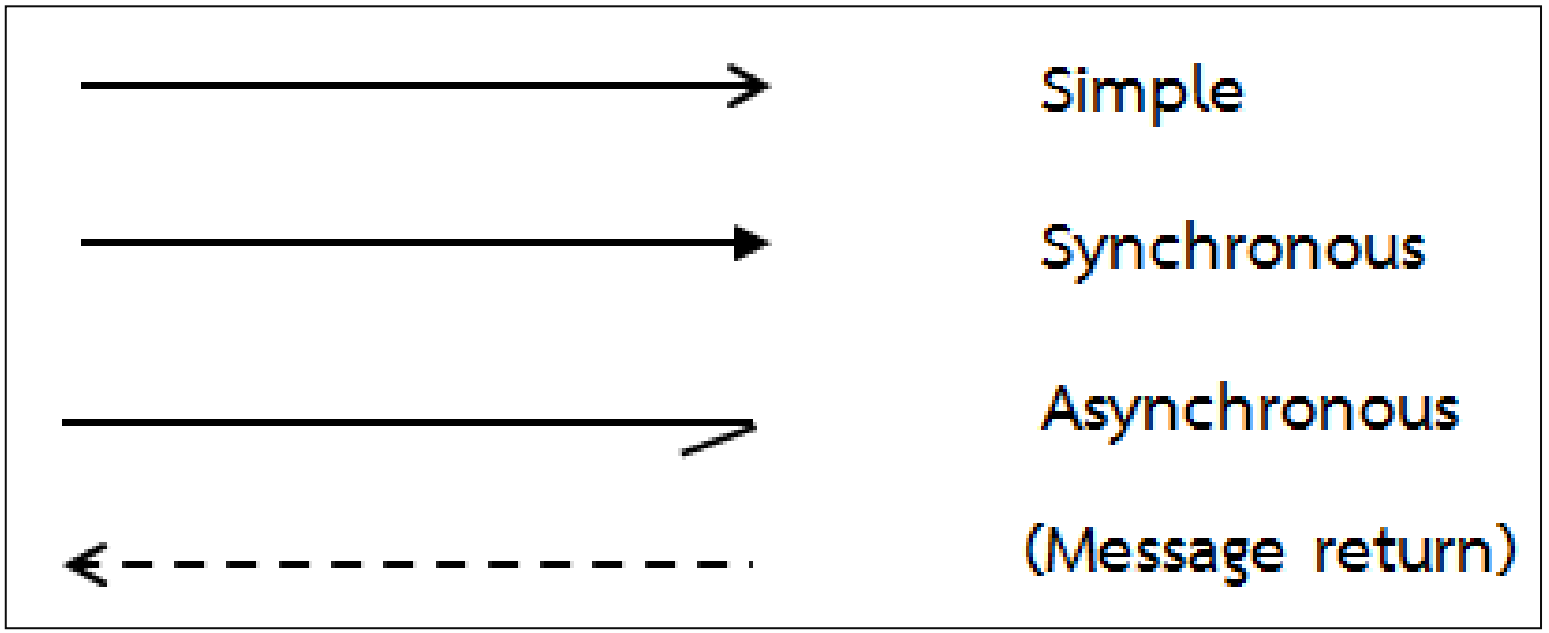
1. Simple เป็นข่าวสารที่เกิดจาก Sender หรือ Receiver โดยไม่ระบุรายละเอียดของวิธีการติดต่อสื่อสารระหว่างวัตถุ
2. Synchronous คือการเรียกใช้ฟังก์ชันของวัตถุโดย Sender หรือ Caller รอจนสิ้นสุดฟังก์ชันโดย Receiver จัดเป็น Passive Object
3. Asynchronous ข้อความที่ Sender ส่งไปยัง Receiver โดยไม่ต้องรอให้ Receiver ทำงานเสร็จทันที Sender สามารถทำงานต่อได้ และ Receiver อาจตอบกลับภายหลังหรือไม่ตอบกลับก็ได้
4. Return Message เป็นข่าวสารที่ส่งกลับยังวัตถุที่มีการเรียก

ข่าวสารจะต้องอธิบายถึงสิ่งที่ส่งไปในเครื่องหมาย () เช่น แสดงข้อความเตือน() หากมีการส่งข้อมูลไปด้วยให้ใส่ค่าลงในวงเล็บ เช่น แสดงข้อความเตือน (Error Message) ถ้าหากเป็นข่าวสารเงื่อนไขจะเขียนเงื่อนไขไว้ในวงเล็บก้ามปู [ ] โดยข่าวสารจะถูกส่งก็ต่อเมื่อเงื่อนไขนั้นเป็นจริง เช่น [กรอกรหัสผิด] แสดงข้อความเตือน()

6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

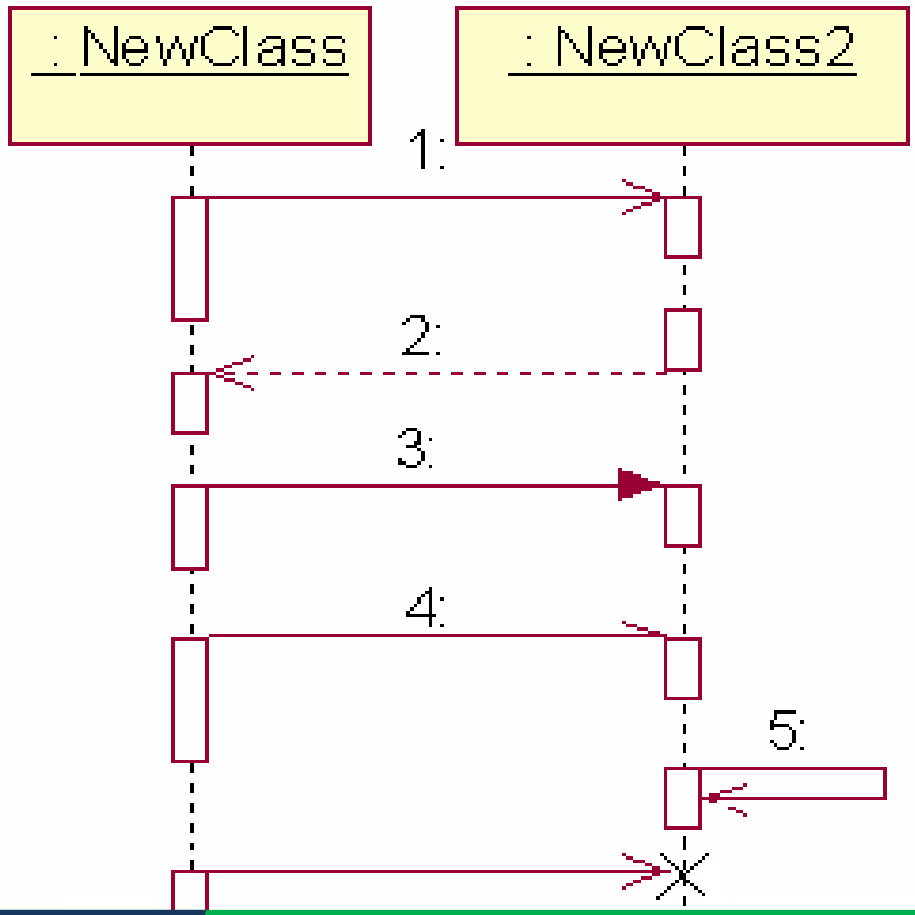
สัญลักษณ์ของข่าวสาร



6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

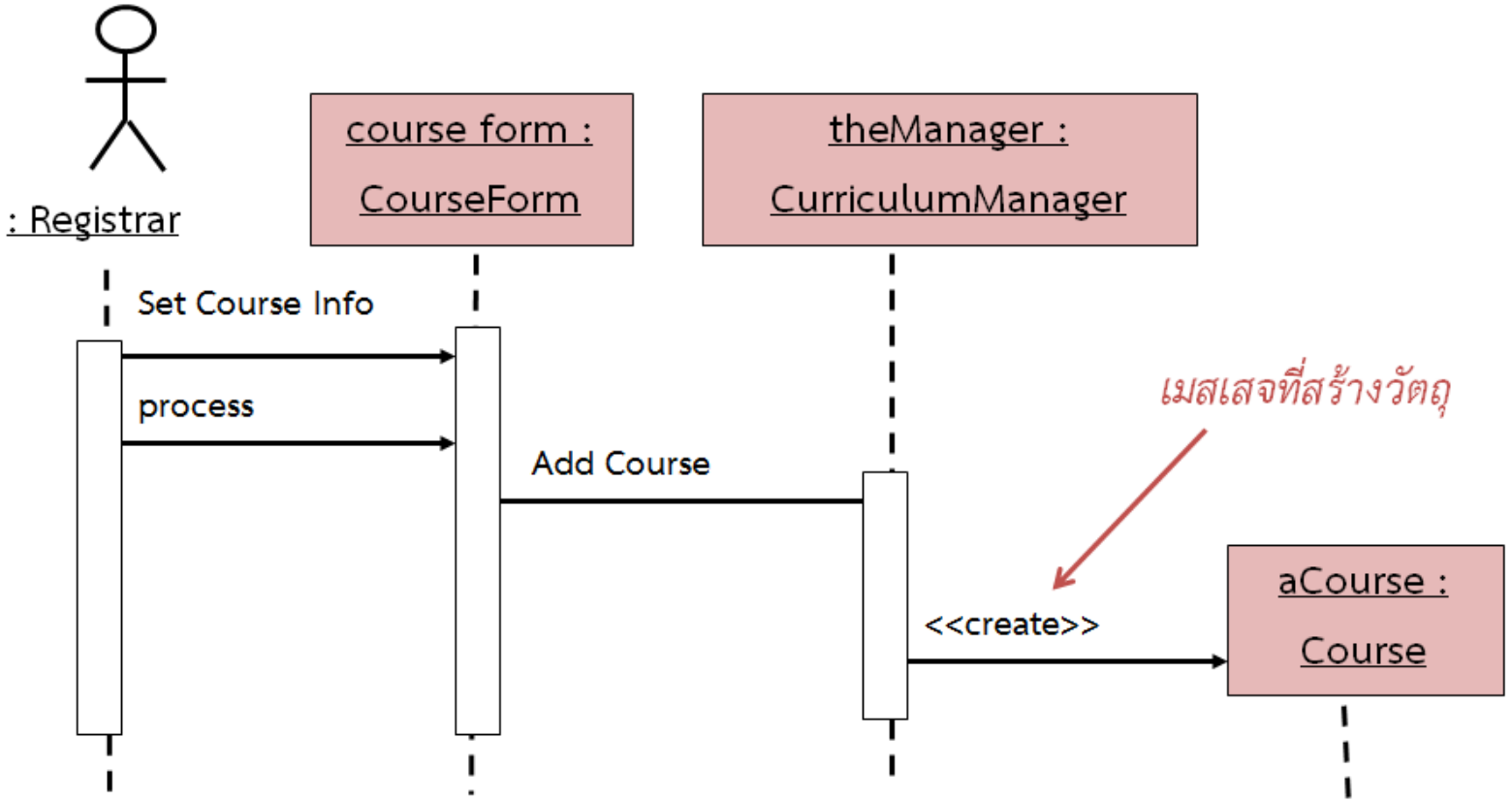
ตัวอย่าง สัญลักษณ์ของข่าวสารชนิดต่าง ๆ ที่ใช้ในการสื่อสารกันระหว่าง 2 วัตถุ



6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

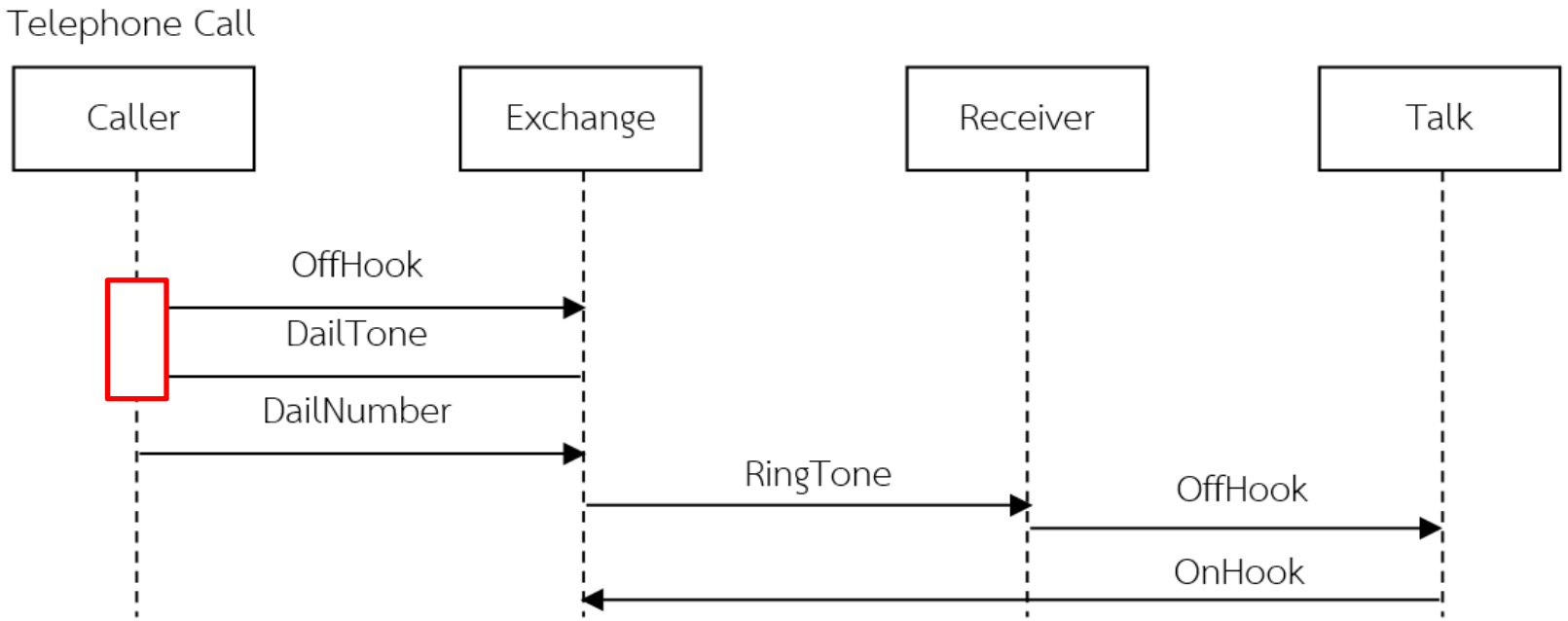
ตัวอย่างแผนภาพลำดับ (Sequence diagram) ของการขอเปิดรายวิชาเรียน



6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

ตัวอย่างแผนภาพลำดับ (Sequence diagram)ของการใช้โทรศัพท์



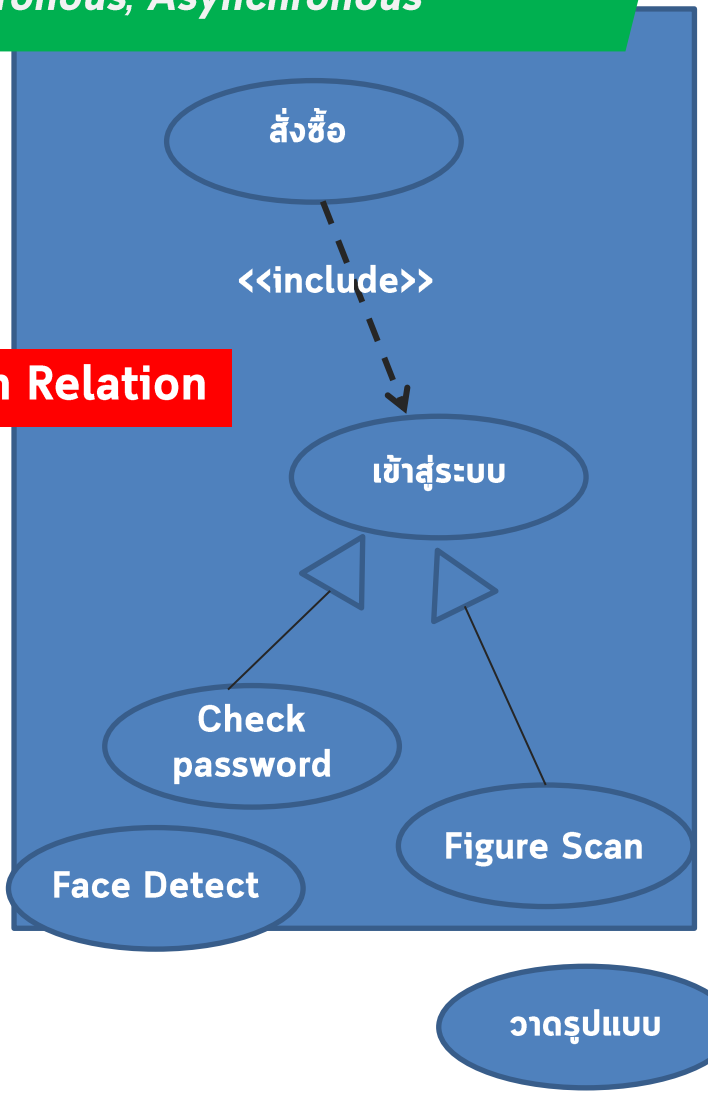
6.7 Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

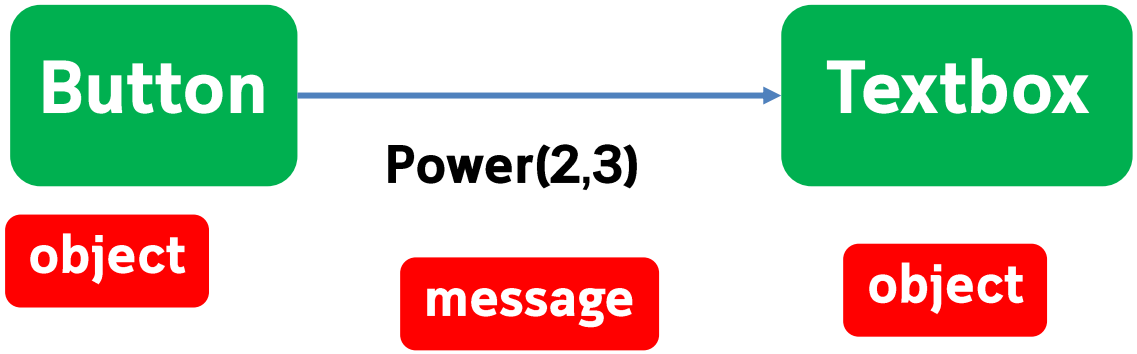
ข่าวสาร คือสิ่งส่งหากันระหว่างวัตถุ โดยเกิดจาก

- การขาย()
- การสั่งซื้อ()
- การสมัครสมาชิก()
- การ Login()

Generalization Relation



การเรียกใช้งานเมธอดใด ๆ ของวัตถุ (call)

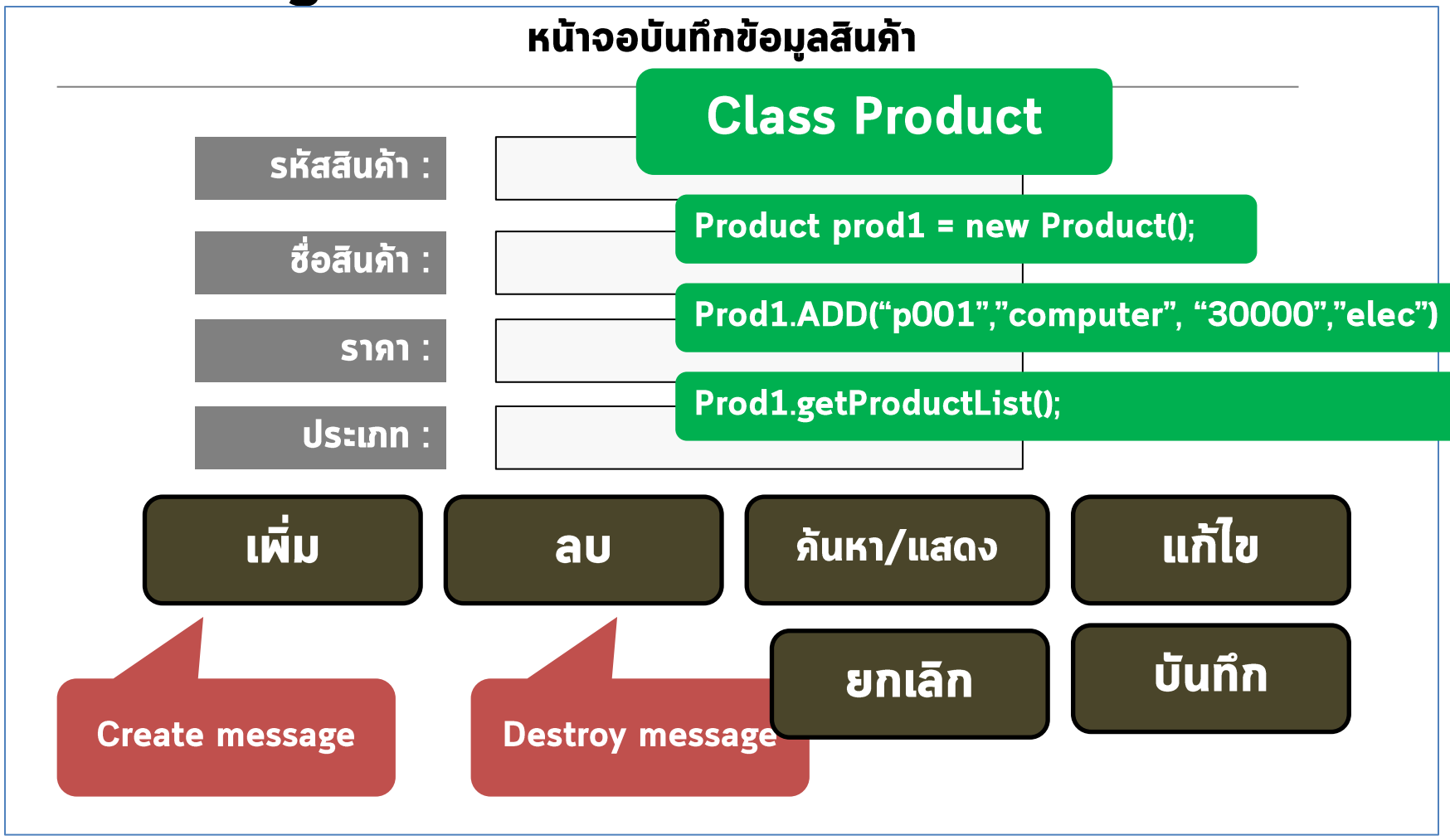


6.7

Message ใน Sequence Diagram

Call, Return, Create, Destroy, Self, Synchronous, Asynchronous

Message



6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

6.8 Combined Fragment

Combined Fragment คือกรอบสี่เหลี่ยมที่ใช้ใน Sequence Diagram เพื่อแสดงเงื่อนไขพิเศษของลำดับการทำงาน เช่น การเลือกทางเลือก การทำงานบางกรณี การทำซ้ำ หรือการทำงานพร้อมกัน โดยปกติ Combined Fragment จะเขียนเป็นกรอบครอบ Message ที่เกี่ยวข้อง และมีคำกำกับมุมซ้ายบน เช่น alt,

1. alt : Alternative

alt ใช้แสดงกรณีที่ระบบมี หลายทางเลือก คล้ายคำสั่ง if...else

ตัวอย่าง: ระบบ Login

เงื่อนไข	การทำงาน
[username/password ถูกต้อง]	แสดงหน้า Dashboard
[username/password ไม่ถูกต้อง]	แสดงข้อความ Login ไม่สำเร็จ

ตัวอย่างข้อความใน Sequence Diagram:

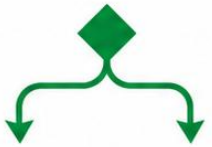
```
alt
[ข้อมูลถูกต้อง]
    AuthController → LoginForm : showDashboard()

[ข้อมูลไม่ถูกต้อง]
    AuthController → LoginForm : showError()
```

สรุป:
alt ใช้เมื่อมีทางเลือกมากกว่า 1 ทาง เช่น
สำเร็จ/ไม่สำเร็จ ใช่/ไม่ใช่ มีสินค้า/ไม่มีสินค้า

6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

1. alt : Alternative

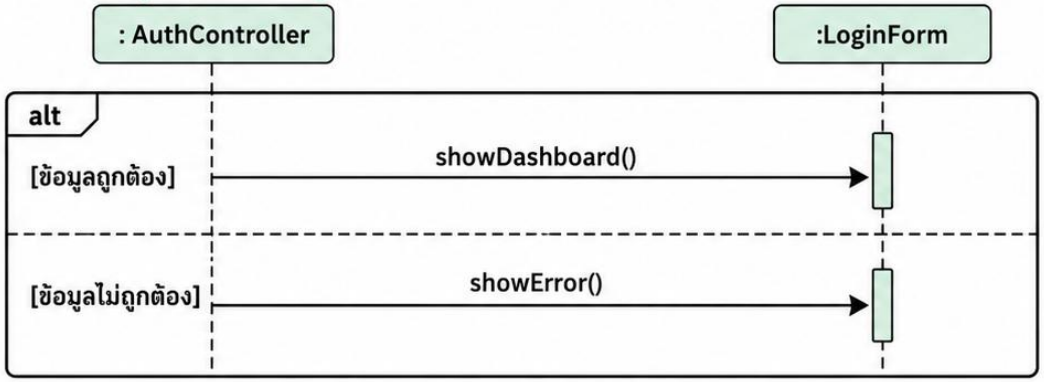


alt ใช้แสดงกรณีที่ระบบมีหลายทางเลือก คล้ายคำสั่ง if...else

ตัวอย่าง: ระบบ Login

เงื่อนไข	การทำงาน
[username/password ถูกต้อง]	แสดงหน้า Dashboard
[username/password ไม่ถูกต้อง]	แสดงข้อความ Login ไม่สำเร็จ

ตัวอย่างข้อความใน Sequence Diagram



สรุป: alt ใช้เมื่อมีทางเลือกมากกว่า 1 ทาง เช่น สำเร็จ/ไม่สำเร็จ ใช่/ไม่ใช่ มีสินค้า/ไม่มีสินค้า

6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

2. opt : Optional

opt ใช้แสดงกรณีที่ระบบอาจทำหรือไม่ทำก็ได้ ขึ้นอยู่กับเงื่อนไข คล้ายคำสั่ง if แบบไม่มี else

ตัวอย่าง: ระบบสั่งซื้อสินค้า

เงื่อนไข	การทำงาน
[ลูกค้ามีคูปองส่วนลด]	ระบบคำนวณส่วนลด
ถ้าไม่มีคูปอง	ข้ามขั้นตอนนี้

ตัวอย่างข้อความใน Sequence Diagram:

```
opt
[มีคูปองส่วนลด]
    OrderController → DiscountService : applyCoupon()
```

สรุป:

opt ใช้กับขั้นตอนที่เป็น “ทางเลือกเสริม” อาจเกิดขึ้นหรือไม่เกิดขึ้นก็ได้

6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

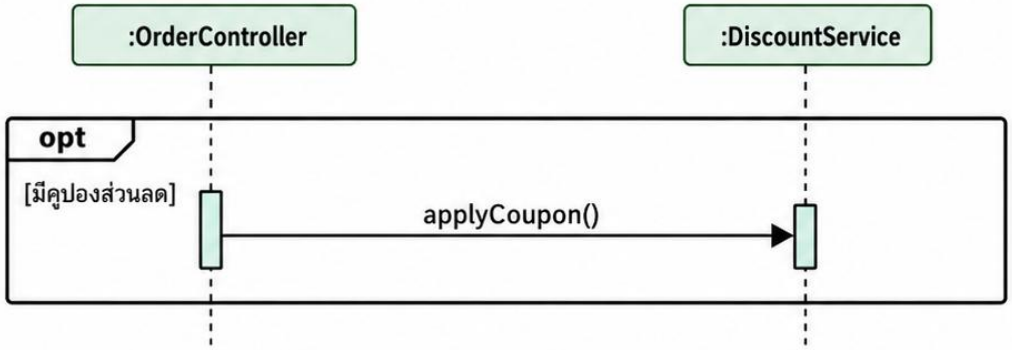
2. opt : Optional

opt ใช้แสดงกรณีที่ระบบอาจทำหรือไม่ทำก็ได้ ขึ้นอยู่กับเงื่อนไข คล้ายคำสั่ง if แบบไม่มี else

ตัวอย่าง: ระบบสั่งซื้อสินค้า

เงื่อนไข	การทำงาน
[ลูกค้ามีคูปองส่วนลด]	ระบบคำนวณส่วนลด
ถ้าไม่มีคูปอง	ข้ามขั้นตอนนี้

ตัวอย่างข้อความใน Sequence Diagram



สรุป: opt ใช้กับขั้นตอนที่เป็น “ทางเลือกเสริม” อาจเกิดขึ้นหรือไม่เกิดขึ้นก็ได้

6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

3. loop : Loop

loop ใช้แสดงการทำงานที่เกิดซ้ำหลายครั้ง คล้ายคำสั่ง for, while

ตัวอย่าง: ระบบตะกร้าสินค้า

เงื่อนไข	การทำงาน
[สินค้าทุกชิ้นในตะกร้า]	ตรวจสอบราคาและจำนวนสินค้าแต่ละรายการ

ตัวอย่างข้อความใน Sequence Diagram:

```
loop
[for each item in cart]
    Cart → ProductService : checkStock(item)
    ProductService → Cart : stockResult
```

สรุป:

loop ใช้เมื่อมีการทำงานซ้ำ เช่น ตรวจสอบสินค้าหลายรายการ แสดงข้อมูลหลายแถว หรือประมวลผลรายการหลายชุด

6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

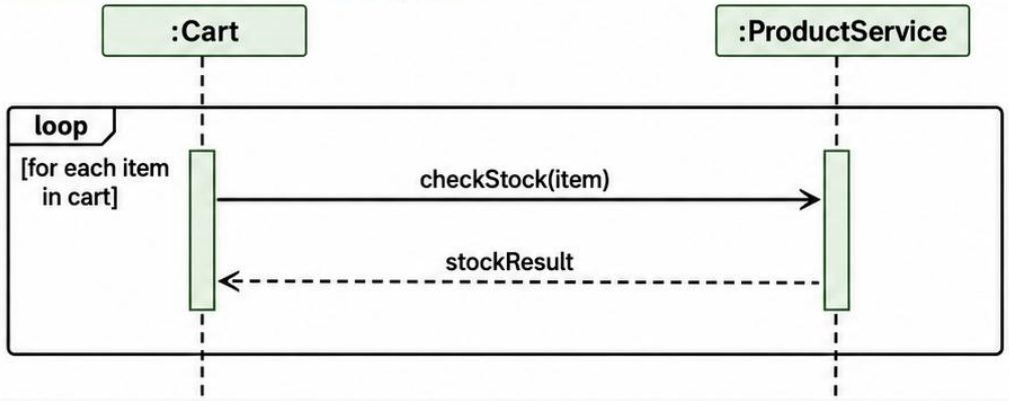
3. loop : Loop

loop ใช้แสดงการทำงานที่เกิดซ้ำหลายครั้ง คล้ายคำสั่ง for, while

ตัวอย่าง: ระบบตะกร้าสินค้า

เงื่อนไข	การทำงาน
[สินค้าทุกชิ้นในตะกร้า]	ตรวจสอบราคาและจำนวนสินค้าแต่ละรายการ

ตัวอย่างข้อความใน Sequence Diagram



สรุป: loop ใช้เมื่อมีการทำงานซ้ำ เช่น ตรวจสอบสินค้าหลายรายการ แสดงข้อมูลหลายแถว หรือประมวลผลรายการหลายชุด

6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

4. par : Parallel

par ใช้แสดงการทำงานที่เกิดขึ้น พร้อมกันหรือขนานกัน ในเวลาใกล้เคียงกัน

ตัวอย่าง: ระบบยืนยันคำสั่งซื้อ

งานที่ทำพร้อมกัน	การทำงาน
งานที่ 1	ส่งอีเมลยืนยันคำสั่งซื้อ
งานที่ 2	ส่งแจ้งเตือนไปยังแอป
งานที่ 3	บันทึก Log การทำรายการ

ตัวอย่างข้อความใน Sequence Diagram:

```
par
[ส่งอีเมล]
    OrderService → EmailService : sendEmail()
[ส่งแจ้งเตือน]
    OrderService → NotificationService : sendNotification()
[บันทึก Log]
    OrderService → LogService : saveLog()
```

สรุป:

par ใช้เมื่อระบบต้องทำหลายงานพร้อมกัน เช่น ส่งอีเมล แจ้งเตือน และบันทึกข้อมูล

6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง

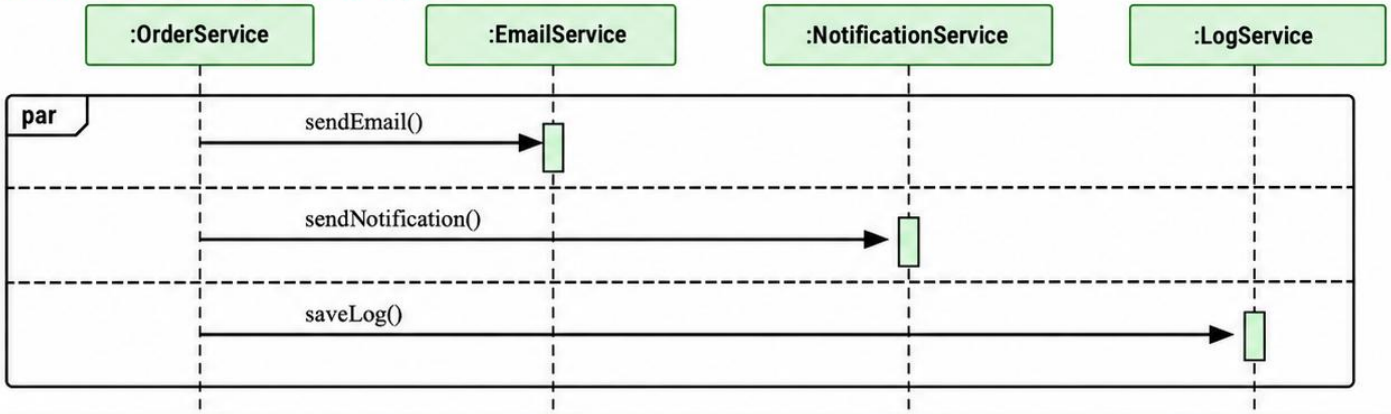
4. par : Parallel

par ใช้แสดงการทำงานที่เกิดขึ้นพร้อมกันหรือขนานกัน ในเวลาใกล้เคียงกัน

ตัวอย่าง: ระบบยืนยันคำสั่งซื้อ

งานที่ทำพร้อมกัน	การทำงาน
งานที่ 1	ส่งอีเมลยืนยันคำสั่งซื้อ
งานที่ 2	ส่งแจ้งเตือนไปยังแอป
งานที่ 3	บันทึก Log การทำรายการ

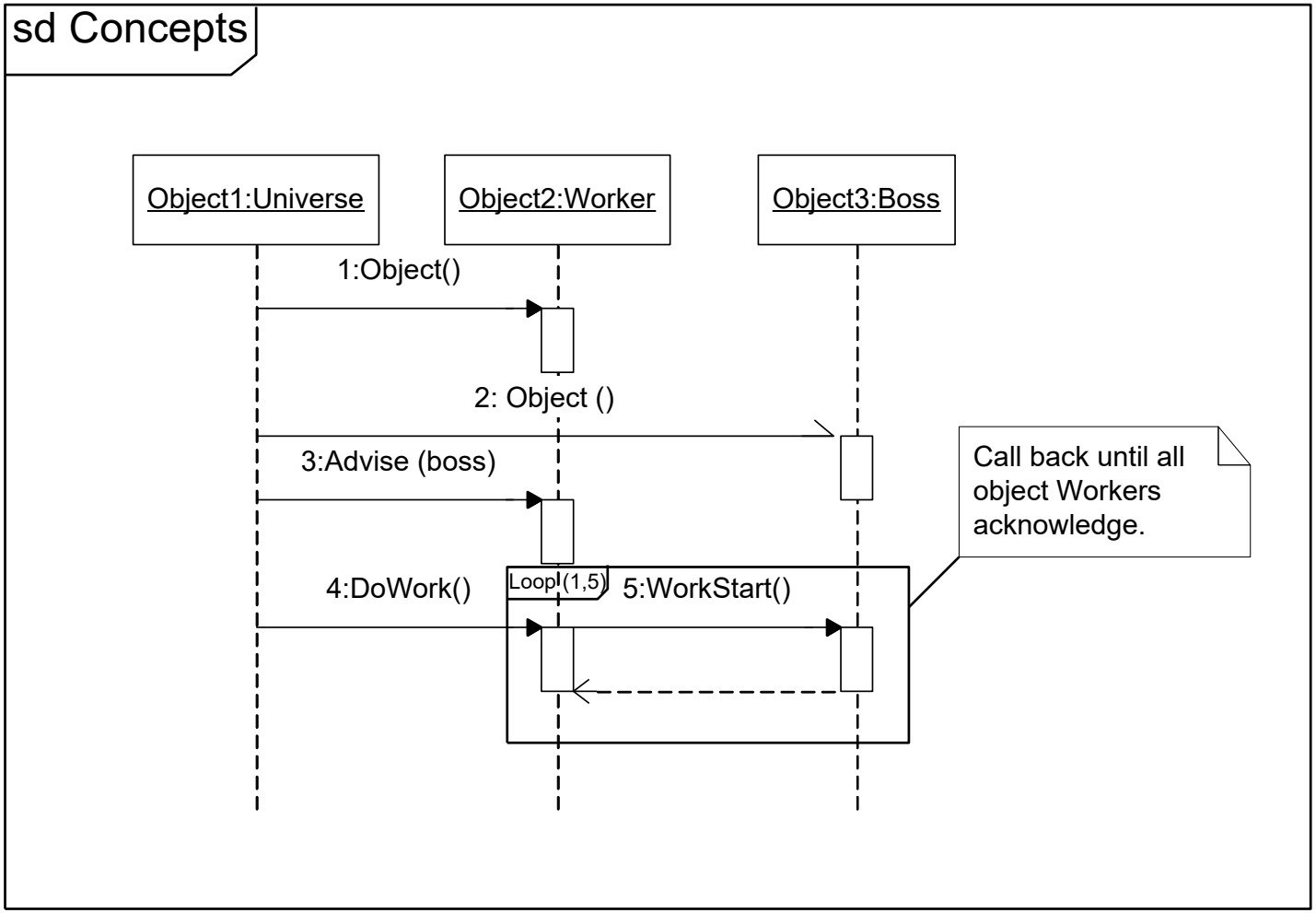
ตัวอย่างข้อความใน Sequence Diagram



สรุป:
par ใช้เมื่อระบบต้องทำหลายงานพร้อมกัน เช่น ส่งอีเมล แจ้งเตือน และบันทึกข้อมูล

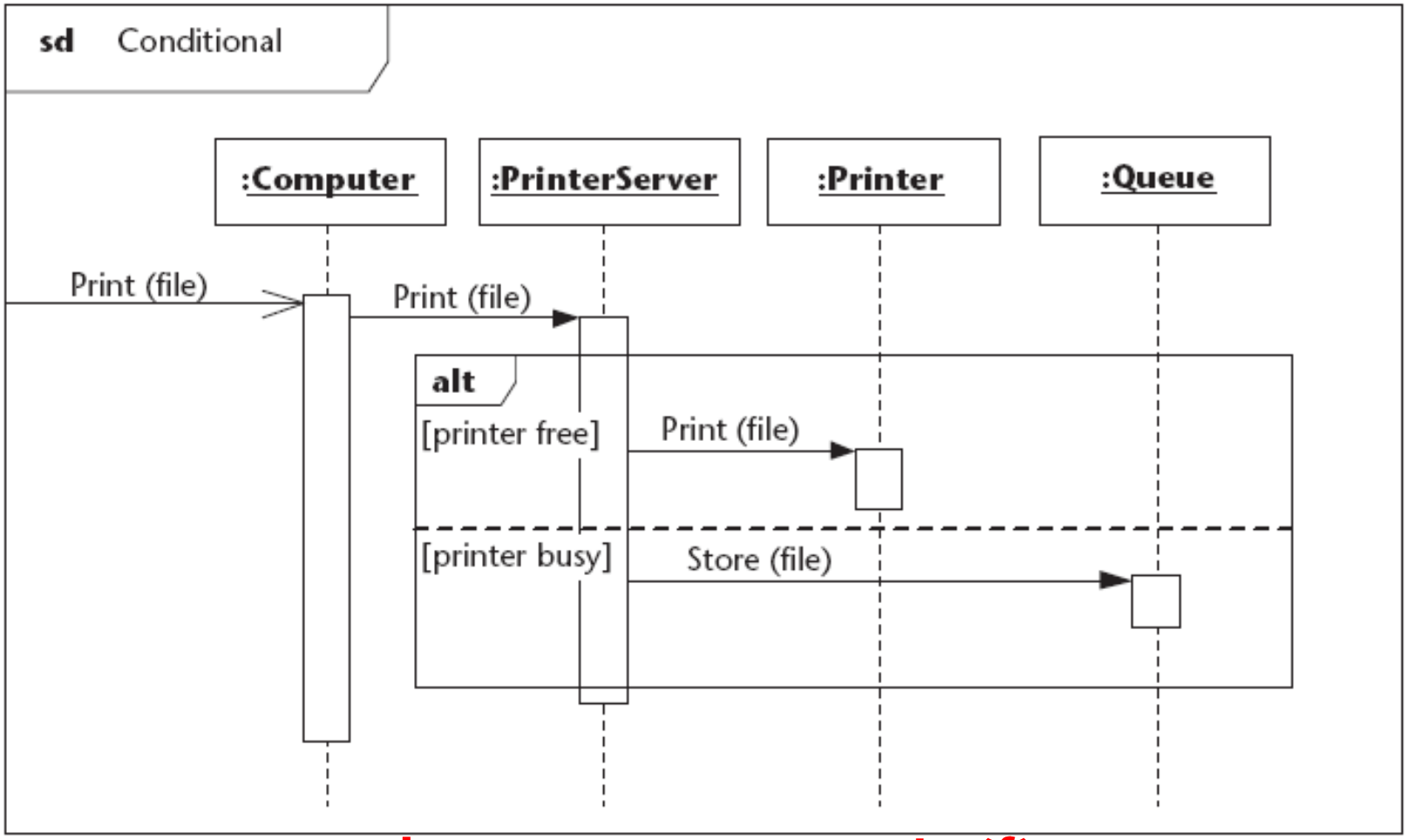
6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง



6.8 Combined Fragment

alt, opt, loop, par พร้อมตัวอย่าง



Actor

alt

p.print(f);

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ในการสร้างแผนภาพลำดับ (Sequence diagram)มีขั้นตอนดังนี้

ขั้นตอนที่ 1. กำหนด External หรือ Internal Entity เริ่มจากการกำหนดผู้เกี่ยวข้องกับระบบ โดยแยกเป็น Actor หรือระบบภายนอกที่เริ่มต้นการทำงาน และ Object ภายในระบบที่รับคำขอ ประมวลผล หรือจัดเก็บข้อมูล จากนั้นนำ Actor และ Object เหล่านี้ไปจัดวางเป็น Lifeline ใน Sequence Diagram

ขั้นตอนที่ 2. การระบุวัตถุและคลาสที่จะมีการทำงาน

ขั้นตอนที่ 3. การรับ – ส่ง ข่าวสารระหว่างวัตถุโดยเริ่มต้นจากวัตถุหนึ่งไปยังวัตถุอื่น ๆ ลงไปตามเส้นเวลาทุกครั้งที่มีการเรียกใช้ข่าวสาร

ขั้นตอนที่ 4. Activation บริเวณที่มี Activation bar ครอบอยู่แสดงให้เห็นว่าวัตถุที่สั่งงานไปยังคงรอคอยผลลัพธ์กลับจากวัตถุที่ทำงานให้อยู่

ขั้นตอนที่ 5. ข่าวสารในลำดับถัดมาจะเป็นผลของการสร้างวัตถุของ Activation ใหม่

ขั้นตอนที่ 6. ที่ตำแหน่งสุดท้ายของ Activation อาจจะใช้สำหรับการคืนค่ากลับไปยัง Caller ซึ่งจะแทนด้วยเส้นประที่เริ่มจากผู้รับไปยังผู้ส่ง

ขั้นตอนที่ 7. ข่าวสารที่ปรากฏในส่วนบนจะเกิดขึ้นก่อนเมสเจที่อยู่ถัดลงมาซึ่งเป็นไปตาม Lifetime

ขั้นตอนที่ 8. ข่าวสารที่อยู่ท้ายสุดจะเป็นการทำงานลำดับท้ายของยูสเคส

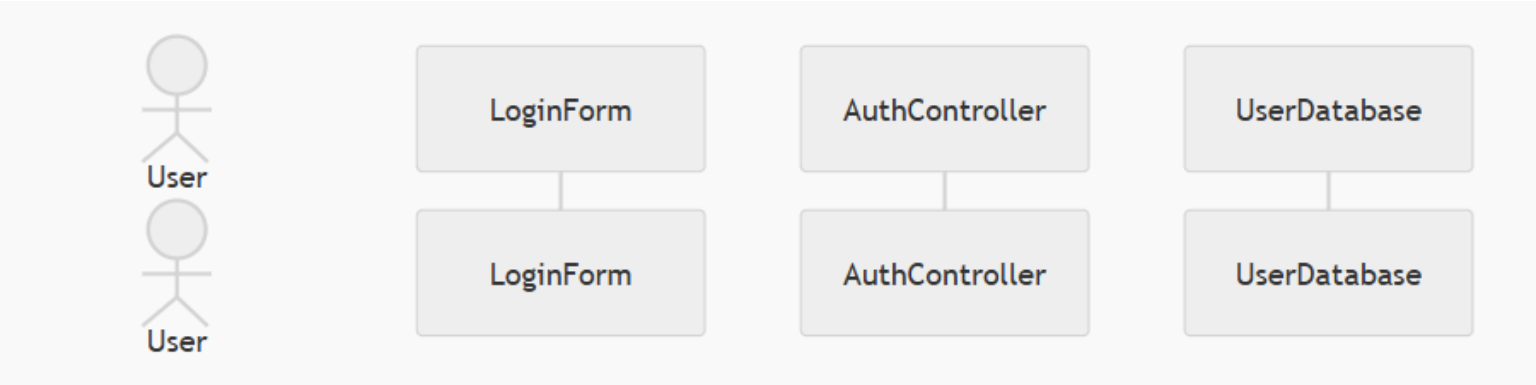
6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 1: กำหนด External หรือ Internal Entity

- **External Entity: เป็นหน่วยงานหรือระบบภายนอกที่โต้ตอบกับระบบของเรา อาจจะเป็นระบบอื่น ๆ หรือผู้ใช้งาน (Actor) ที่อยู่ภายนอก โดย External Entity จะเป็นตัวเริ่มต้นในการส่งข้อความเข้าไปยังระบบที่เรากำลังออกแบบ**
- **Internal Entity: คือส่วนประกอบภายในของระบบที่ทำหน้าที่ตอบสนองต่อ External Entity หรือการส่งข้อความภายในระบบ เช่น การโต้ตอบระหว่างคลาสหรือวัตถุต่าง ๆ ภายในระบบ**
- **ถ้าเป็น Internal Entity กระบวนการจะเริ่มจาก Actor ซึ่งถูกกำหนดให้เป็นตัวเริ่มต้นในการโต้ตอบภายในระบบ**

ขั้นตอนนี้ยังไม่ต้องใส่ Message แต่ต้องรู้ก่อนว่าในระบบมีใครหรือ Object ใดเกี่ยวข้องบ้าง



6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 1: กำหนด External หรือ Internal Entity

ตัวอย่างขั้นตอนการกำหนด External หรือ Internal Entity ในการสร้าง Sequence Diagram:

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

ในระบบนี้มีทั้ง External Entity (ผู้ใช้งาน) และ Internal Entity (ระบบภายใน)

External Entity:

- **ผู้ใช้งาน (User): ผู้ใช้เป็นบุคคลภายนอกที่โต้ตอบกับระบบผ่านอินเทอร์เฟซ ผู้ใช้งานจะเป็นจุดเริ่มต้นในการโต้ตอบ เช่น ผู้ใช้งานเริ่มการค้นหาภาพยนตร์ หรือเลือกซื้อตั๋ว**

Internal Entity:

- **ระบบฐานข้อมูล (Database): ระบบฐานข้อมูลภายในจะตอบสนองต่อคำขอจากผู้ใช้งาน เช่น การส่งข้อมูลภาพยนตร์หรือที่นั่งว่างในโรงหนัง**
- **ระบบการชำระเงิน (Payment System): เป็น Internal Entity ที่รับผิดชอบในการจัดการธุรกรรมการเงิน เช่น การตรวจสอบบัตรเครดิตหรือการยืนยันการชำระเงิน**

Sequence Diagram จะเริ่มจาก External Entity (ผู้ใช้งาน) ทำการเลือกภาพยนตร์ จากนั้นระบบภายใน (Internal Entity) จะส่งข้อมูลให้ผู้ใช้งาน และมีการสื่อสารกับระบบการชำระเงินเพื่อยืนยันการซื้อ

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 1: กำหนด External หรือ Internal Entity

ตัวอย่างที่ 2: ระบบการจัดการโรงเรียน (School Management System)

External Entity:

- ครู (Teacher): ครูเป็น Actor ภายนอกที่สามารถเข้าสู่ระบบเพื่อบันทึกคะแนนของนักเรียน

Internal Entity:

- ระบบการจัดเก็บคะแนน (Grade Storage System): ระบบภายในที่รับคำขอจากครูในการบันทึกคะแนนและจัดเก็บข้อมูลคะแนนลงในฐานข้อมูลของนักเรียน
- ระบบแจ้งเตือน (Notification System): ระบบภายในที่แจ้งเตือนผู้ปกครองเมื่อคะแนนของนักเรียนถูกบันทึกลงไป

ใน Sequence Diagram จะเริ่มจากครู (External Entity) ที่เข้าสู่ระบบและบันทึกคะแนน นักเรียนจะถูกบันทึกลงในฐานข้อมูลผ่านระบบการจัดเก็บคะแนน (Internal Entity)

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

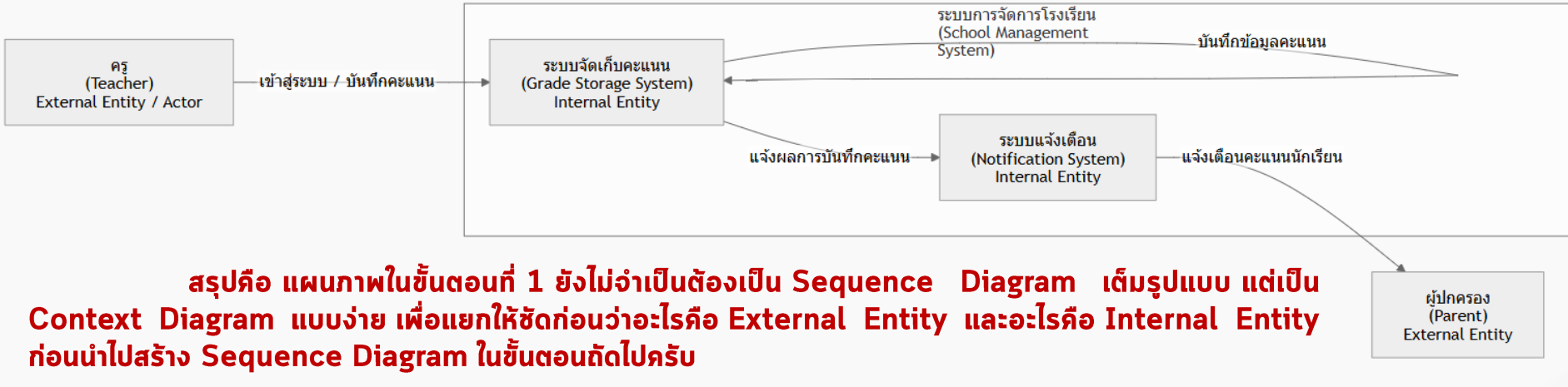
เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 1: กำหนด External หรือ Internal Entity

คำอธิบายแผนภาพ

แผนภาพนี้แสดงการกำหนด External Entity และ Internal Entity ก่อนเริ่มสร้าง Sequence Diagram

ประเภท	ตัวอย่าง	คำอธิบาย
External Entity	ครู	Actor ภายนอกที่เข้าสู่ระบบเพื่อบันทึกคะแนนนักเรียน
Internal Entity	ระบบจัดเก็บคะแนน	ระบบภายในที่รับคำขอจากครูและบันทึกคะแนนลงฐานข้อมูล
Internal Entity	ระบบแจ้งเตือน	ระบบภายในที่แจ้งเตือนผู้ปกครองเมื่อมีการบันทึกคะแนน
External Entity	ผู้ปกครอง	ผู้รับการแจ้งเตือนจากระบบ



สรุปคือ แผนภาพในขั้นตอนที่ 1 ยังไม่จำเป็นต้องเป็น Sequence Diagram เต็มรูปแบบ แต่เป็น Context Diagram แบบง่าย เพื่อแยกให้ชัดก่อนว่าอะไรคือ External Entity และอะไรคือ Internal Entity ก่อนนำไปสร้าง Sequence Diagram ในขั้นตอนถัดไปครับ

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

วัตถุ (Objects) และคลาส (Classes) ที่เกี่ยวข้อง:

1. User (ผู้ใช้): คลาสนี้แสดงถึงผู้ใช้งานที่ทำการเลือกภาพยนตร์และจองตั๋ว
2. Movie (ภาพยนตร์): คลาสนี้จัดเก็บข้อมูลภาพยนตร์ เช่น ชื่อภาพยนตร์, เวลา, โรงภาพยนตร์ และที่นั่งที่ว่าง
3. Ticket System (ระบบจัดการตั๋ว): คลาสนี้เป็นระบบภายในที่ทำหน้าที่จัดการการจองตั๋ว ตรวจสอบที่นั่งว่าง และเก็บข้อมูลการจอง
4. Payment Gateway (เกตเวย์การชำระเงิน): คลาสนี้เป็นระบบที่เชื่อมต่อกับบริการภายนอก เช่น บัตรเครดิตหรือธนาคาร เพื่อทำธุรกรรมการเงิน

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

การทำงานระหว่างวัตถุและคลาส:

1. User ทำการเลือกภาพยนตร์ → ส่งคำขอไปที่ Movie
2. Movie ส่งข้อมูลที่นั่งที่ว่างกลับไปยัง User
3. User เลือกที่นั่งและยืนยันการจอง → ส่งข้อมูลไปที่ Ticket System
4. Ticket System ตรวจสอบการจอง → หากการจองสำเร็จ จะส่งต่อไปที่ Payment Gateway เพื่อยืนยันการชำระเงิน
5. Payment Gateway ประมวลผลการชำระเงิน และส่งผลลัพธ์กลับไปยัง Ticket System
6. Ticket System ส่งข้อมูลการยืนยันการจองตั๋วกลับไปที่ User

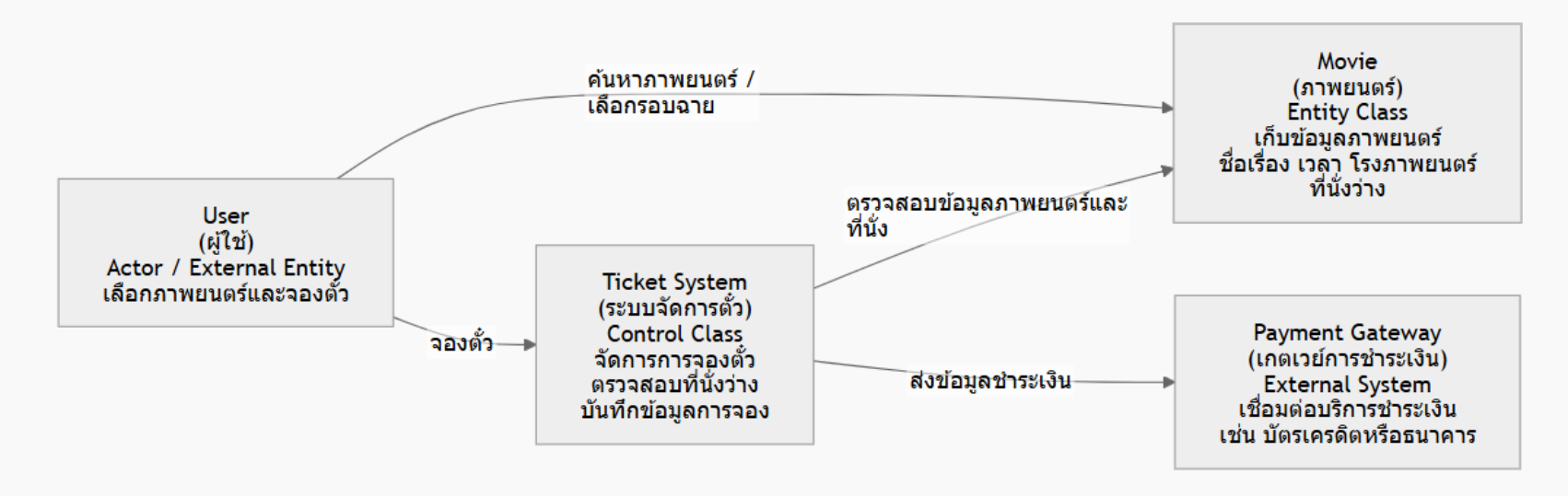
6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

คำอธิบายแผนภาพ

แผนภาพนี้ยังไม่ใช่ Sequence Diagram เต็มรูปแบบ แต่เป็นการเตรียมข้อมูลก่อนวาด Sequence Diagram โดยระบุว่า Object หรือ Class ใดจะถูกนำไปใช้เป็น Participant / Lifeline ในแผนภาพลำดับ



6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

ขั้นตอนที่ 2: ระบุวัตถุและคลาสที่เกี่ยวข้อง

Actor:
- User

Entity Class:
- Movie

Control Class:
- Ticket System

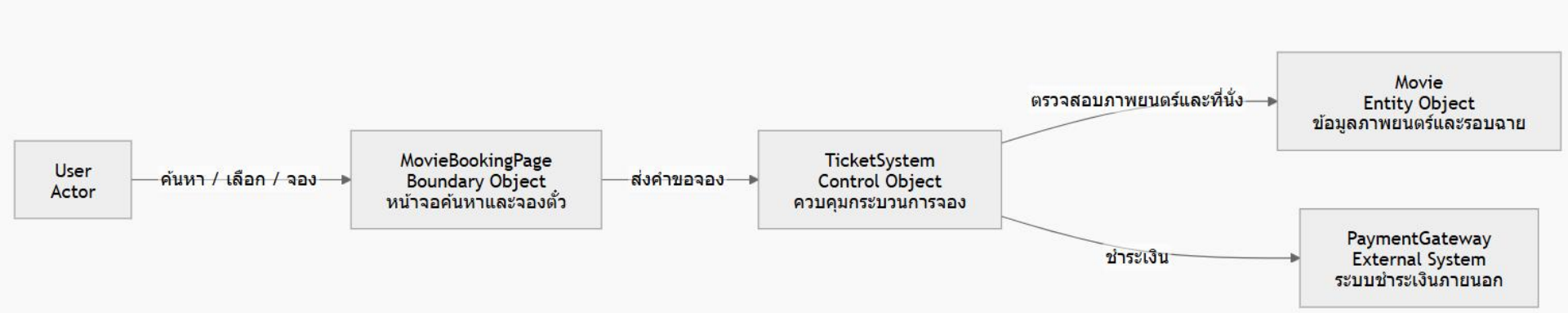
External System:
- Payment Gateway

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

แผนภาพนี้แสดงการระบุ Object/Class ที่เกี่ยวข้องกับระบบจองตั๋วหนังออนไลน์ตามแนวคิด Boundary → Control → Entity โดยผู้ใช้ (User) ติดต่อกับระบบผ่าน MovieBookingPage ซึ่งเป็น Boundary Object จากนั้นคำขอจองจะถูกส่งไปยัง TicketSystem ซึ่งเป็น Control Object เพื่อควบคุมกระบวนการจอง ตรวจสอบข้อมูลภาพยนตร์และที่นั่งกับ Movie ซึ่งเป็น Entity Object และติดต่อ PaymentGateway ซึ่งเป็น External System สำหรับการชำระเงิน



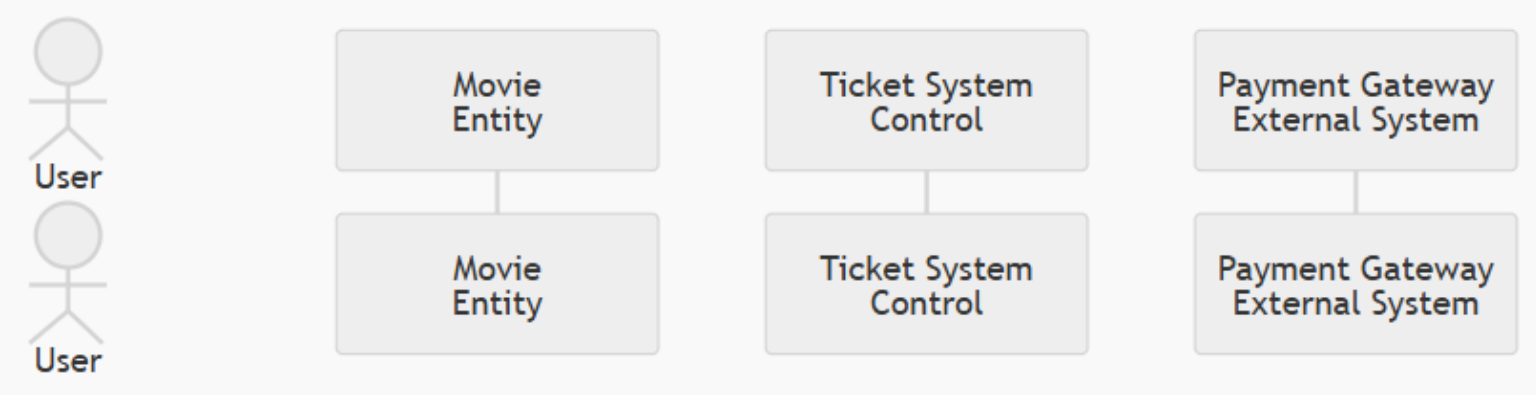
6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

ลำดับ	Object / Class	ประเภท	หน้าที่
1	User	Actor	ผู้ใช้ระบบ เลือกภาพยนตร์ เลือกรอบฉาย และจองตั๋ว
2	Movie	Entity Class	เก็บข้อมูลภาพยนตร์ เช่น ชื่อเรื่อง เวลา โรงภาพยนตร์ และที่นั่งว่าง
3	Ticket System	Control Class	ควบคุมกระบวนการจอง ตรวจสอบที่นั่ง และบันทึกการจอง
4	Payment Gateway	External System	ระบบภายนอกที่ใช้ประมวลผลการชำระเงิน

เมื่อนำไปวางเป็น Lifeline ใน Sequence Diagram
หลังจากระบุ Object/Class ได้แล้ว จะนำมาวางเรียงเป็น Lifeline ดังนี้



6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

ตัวอย่างที่ 2: ระบบการจัดการโรงเรียน (School Management System)

วัตถุและคลาสที่เกี่ยวข้อง:

1. Teacher (ครู): คลาสที่แสดงถึงครูที่เข้าสู่ระบบเพื่อบันทึกคะแนนของนักเรียน
2. Student (นักเรียน): คลาสที่เก็บข้อมูลเกี่ยวกับนักเรียน เช่น ชื่อ, ชั้นเรียน, และคะแนน
3. Grade System (ระบบบันทึกคะแนน): คลาสที่ทำหน้าที่บันทึกคะแนนของนักเรียนลงในฐานข้อมูล
4. Notification System (ระบบแจ้งเตือน): คลาสที่ส่งการแจ้งเตือนไปยังผู้ปกครองเมื่อคะแนนถูกบันทึก

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 2: การระบุวัตถุและคลาสที่จะมีการทำงาน ในการสร้าง Sequence Diagram

ตัวอย่างที่ 2: ระบบการจัดการโรงเรียน (School Management System)

การทำงานระหว่างวัตถุและคลาส:

1. Teacher ทำการเข้าสู่ระบบ → ส่งข้อมูลการบันทึกคะแนนไปที่ Grade System
2. Grade System บันทึกคะแนนของนักเรียนในคลาส Student
3. เมื่อคะแนนถูกบันทึกเรียบร้อย Grade System จะส่งข้อมูลไปยัง Notification System
4. Notification System ส่งการแจ้งเตือนไปยังผู้ปกครองนักเรียน

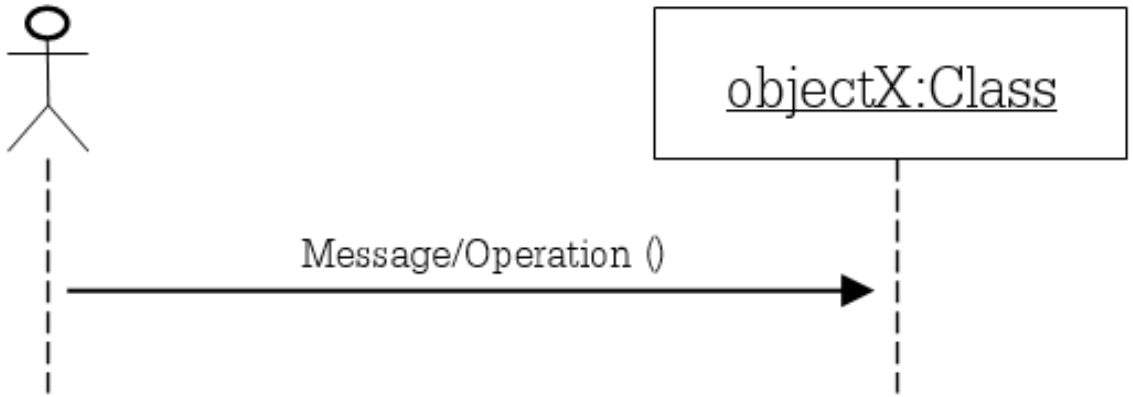
ในขั้นตอนนี้ เราจะระบุวัตถุและคลาสทั้งหมดที่มีการโต้ตอบกันใน Sequence Diagram เพื่อแสดงกระบวนการทำงานอย่างเป็นขั้นเป็นตอน

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนการสร้างแผนภาพลำดับ (Sequence diagram)

ขั้นตอนที่ 2. การระบุวัตถุและคลาสที่จะมีการทำงาน



แสดงการส่งข่าวสารระหว่างแอกเตอร์กับวัตถุ

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 3: การรับ – ส่ง ข่าวสารระหว่างวัตถุ โดยเริ่มต้นจากวัตถุหนึ่งไปยังวัตถุอื่น ๆ ใน Sequence Diagram

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

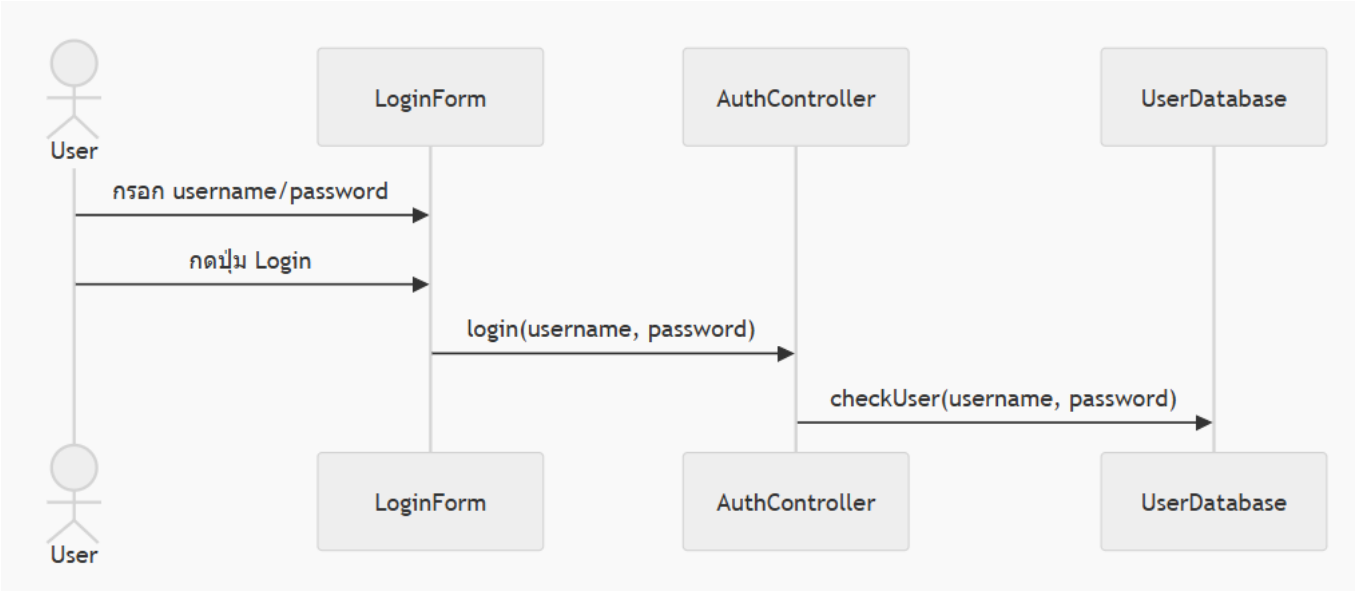
วัตถุที่เกี่ยวข้อง:

- User (ผู้ใช้)
- Movie Database (ฐานข้อมูลภาพยนตร์)
- Ticket System (ระบบจัดการตั๋ว)
- Payment Gateway (เกตเวย์การชำระเงิน)

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 3: การรับ – ส่ง ข่าวสารระหว่างวัตถุ โดยเริ่มต้นจากวัตถุหนึ่งไปยังวัตถุอื่น ๆ ใน Sequence Diagram



Message จะถูกอ่านจากบนลงล่าง โดย Message ด้านบนเกิดก่อน Message ด้านล่าง

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

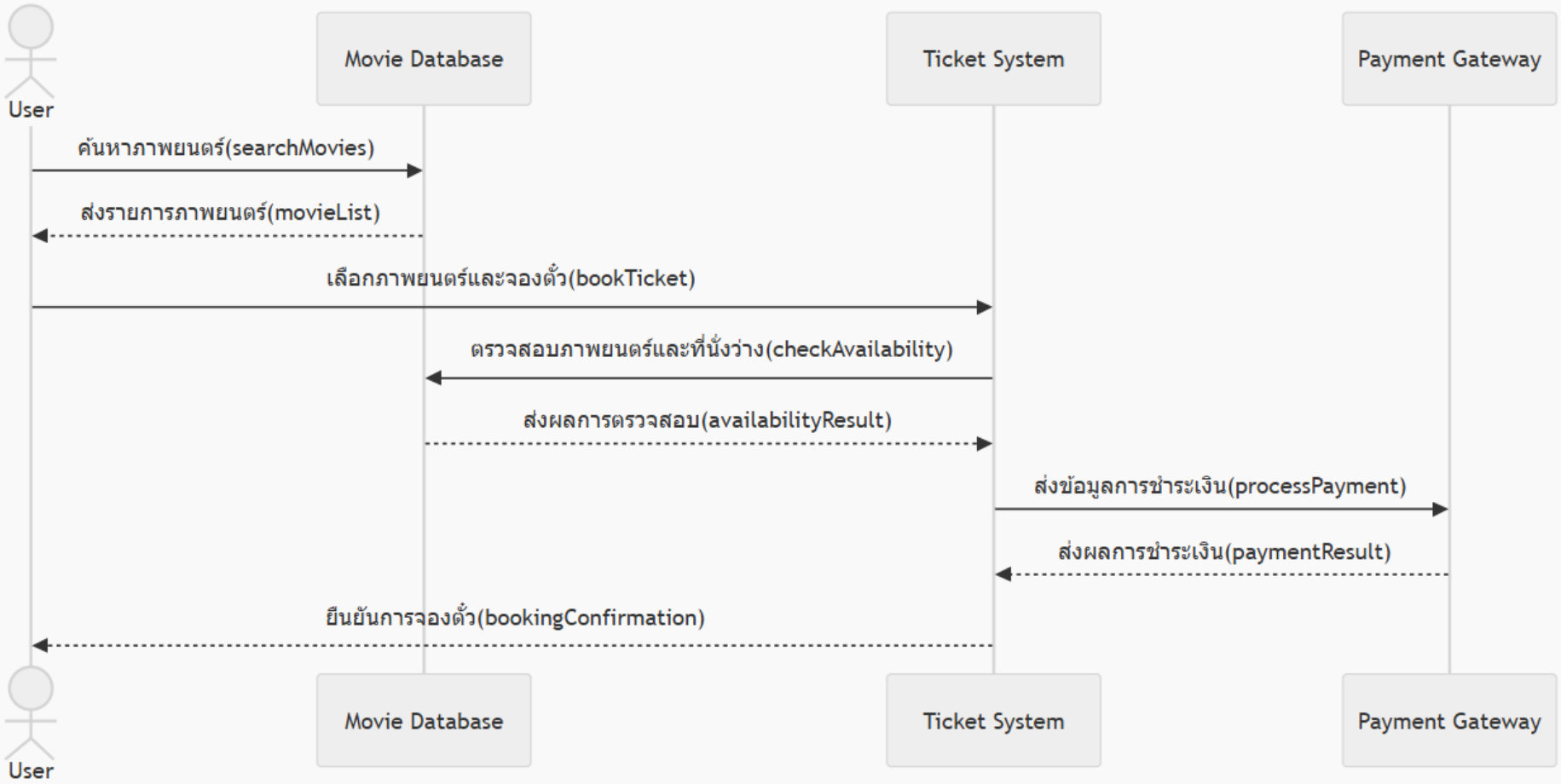
ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

กระบวนการรับ – ส่ง ข่าวสาร:

1. User ส่งคำขอ "ค้นหาภาพยนตร์" → ไปยัง Movie Database
   ข้อความ: ค้นหาภาพยนตร์ที่มีการฉายในวันที่ต้องการ
2. Movie Database ส่งรายการภาพยนตร์ที่ค้นพบ → กลับไปยัง User
   ข้อความ: แสดงรายการภาพยนตร์ที่ฉายในวันที่เลือก
3. User เลือกภาพยนตร์และทำการจอง → ส่งคำขอไปที่ Ticket System
   ข้อความ: จองที่นั่ง
4. Ticket System ตรวจสอบที่นั่งที่ว่างและยืนยันการจอง → ส่งข้อมูลการจองกลับไปยัง User
   ข้อความ: การจองสำเร็จ
5. Ticket System ส่งข้อมูลการชำระเงินไปยัง Payment Gateway เพื่อทำธุรกรรม
   ข้อความ: ประมวลผลการชำระเงิน
6. Payment Gateway ส่งสถานะการชำระเงินกลับไปที่ Ticket System
   ข้อความ: การชำระเงินสำเร็จ
7. Ticket System ส่งการยืนยันการจองและชำระเงินสำเร็จกลับไปยัง User
   ข้อความ: ยืนยันการจองตั๋วและชำระเงินสำเร็จ

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram



6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

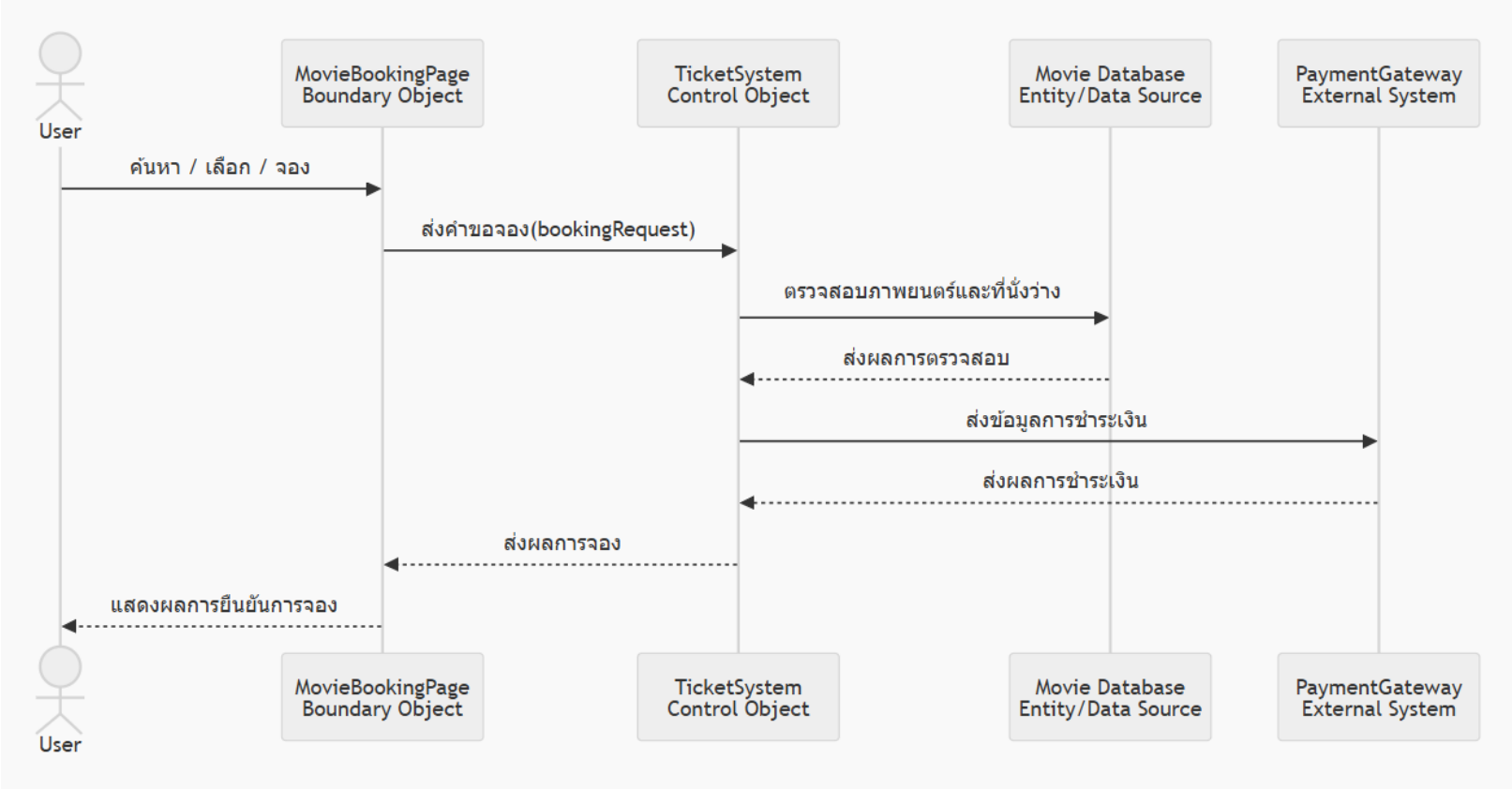
เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ลำดับ Message ในแผนภาพ

ลำดับ	ผู้ส่ง	ผู้รับ	Message
1	User	Movie Database	ค้นหาภาพยนตร์
2	Movie Database	User	ส่งรายการภาพยนตร์
3	User	Ticket System	เลือกภาพยนตร์และจองตั๋ว
4	Ticket System	Movie Database	ตรวจสอบภาพยนตร์และที่นั่งว่าง
5	Movie Database	Ticket System	ส่งผลการตรวจสอบ
6	Ticket System	Payment Gateway	ส่งข้อมูลการชำระเงิน
7	Payment Gateway	Ticket System	ส่งผลการชำระเงิน
8	Ticket System	User	ยืนยันการจองตั๋ว

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram



ขั้นตอนที่ 3 เป็นการกำหนด Message หรือข่าวสารที่รับ-ส่งระหว่าง Actor/Object ตามลำดับเวลา โดยอ่านจากบนลงล่าง Message ที่อยู่ด้านบนจะเกิดขึ้นก่อน และ Message ที่อยู่ถัดลงมาจะเกิดขึ้นภายหลัง ในตัวอย่างระบบจองตั๋วหนังออนไลน์ User ส่งคำขอค้นหาและจองตั๋ว ระบบจึงตรวจสอบข้อมูลภาพยนตร์ ที่นั่งว่าง และดำเนินการชำระเงิน ก่อนส่งผลการยืนยันการจองกลับไปยังผู้ใช้

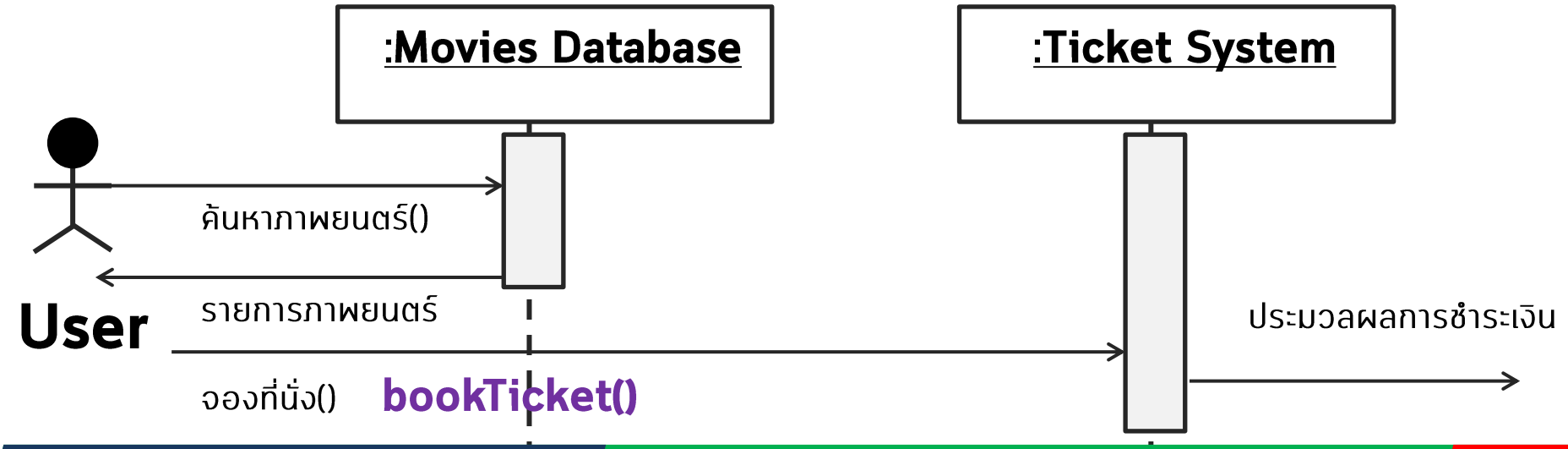
6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

กระบวนการรับ – ส่ง ข่าวสาร:

1. User ส่งคำขอ "ค้นหาภาพยนตร์" → ไปยัง Movie Database
   ข้อความ: ค้นหาภาพยนตร์ที่มีการฉายในวันที่ต้องการ
2. Movie Database ส่งรายการภาพยนตร์ที่ค้นพบ → กลับไปยัง User
   ข้อความ: แสดงรายการภาพยนตร์ที่ฉายในวันที่เลือก



6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

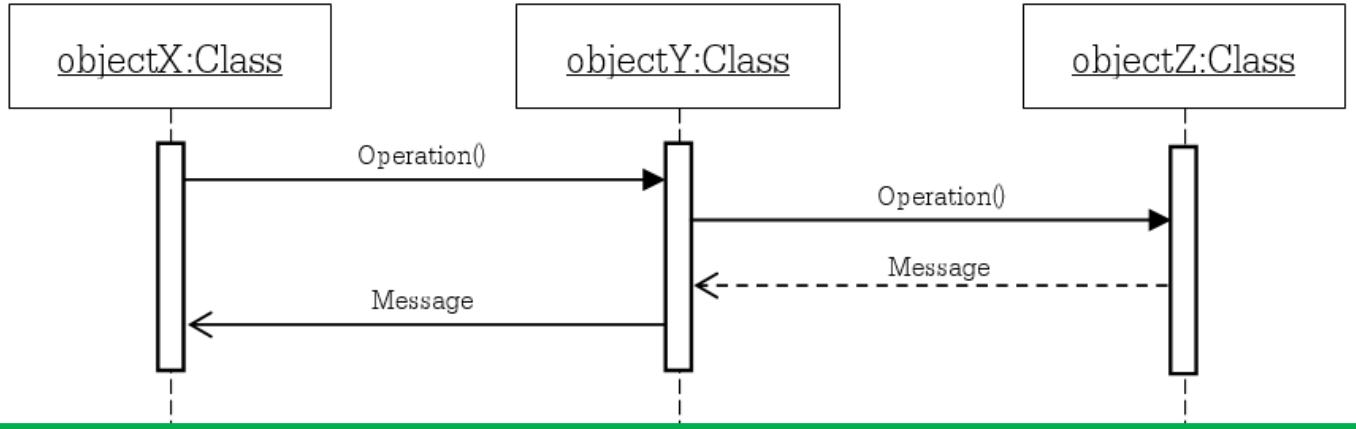
เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 3: การรับ – ส่ง ข่าวสารระหว่างวัตถุ โดยเริ่มต้นจากวัตถุหนึ่งไปยังวัตถุอื่น ๆ ใน Sequence Diagram

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

การทำงานตามเส้นเวลา:

- ลำดับของการส่งข้อความ (Messages) เริ่มจากการค้นหาภาพยนตร์ → การจอง → การชำระเงิน → การยืนยัน ผลลัพธ์ที่ได้รับจะส่งกลับไปยังผู้ใช้ในลำดับเวลา



แผนภาพลำดับ (Sequence diagram)ของการรับ-ส่งข่าวสาร

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 4: Activation (การรอผลลัพธ์) ใน Sequence Diagram

Activation Bar คือแถบที่แสดงถึงช่วงเวลาที่วัตถุทำงานหรือตอบสนองต่อคำขอ โดยจะปรากฏเมื่อวัตถุหนึ่งส่งคำขอไปยังวัตถุอื่น และรอผลลัพธ์จากการทำงานนั้น ๆ

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

วัตถุที่เกี่ยวข้อง:

1. User (ผู้ใช้)
2. Movie Database (ฐานข้อมูลภาพยนตร์)
3. Ticket System (ระบบจัดการตั๋ว)
4. Payment Gateway (เกตเวย์การชำระเงิน)

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 4: Activation (การรอผลลัพธ์) ใน Sequence Diagram

Activation Bar คือแถบที่แสดงถึงช่วงเวลาที่วัตถุทำงานหรือตอบสนองต่อคำขอ โดยจะปรากฏเมื่อวัตถุหนึ่งส่งคำขอไปยังวัตถุอื่น และรอผลลัพธ์จากการทำงานนั้น ๆ

กระบวนการรับ – ส่ง ข่าวสารและการ Activation:

1. User ส่งคำขอ "ค้นหาภาพยนตร์" ไปที่ Movie Database
   - หลังจากส่งคำขอแล้ว วัตถุ User จะมี Activation Bar เพื่อแสดงว่ารอคำตอบจาก Movie Database
   - Movie Database เริ่มทำงานและมี Activation Bar เพื่อแสดงการประมวลผล
   - เมื่อ Movie Database ส่งผลลัพธ์กลับมา Activation Bar ของทั้งสองวัตถุจะสิ้นสุด
2. User เลือกภาพยนตร์และทำการจองตั๋ว → ส่งคำขอไปที่ Ticket System
   - User มี Activation Bar ขณะรอการยืนยันการจองจาก Ticket System
   - Ticket System เริ่มทำงานและมี Activation Bar เพื่อประมวลผลการจอง
   - หลังจากตรวจสอบการจองเสร็จสิ้นและส่งผลลัพธ์กลับไปที่ User Activation Bar ของทั้งสองฝ่ายจะสิ้นสุด
3. Ticket System ส่งข้อมูลการชำระเงินไปยัง Payment Gateway
   - ขณะที่รอผลการชำระเงิน Ticket System จะมี Activation Bar
   - Payment Gateway เริ่มประมวลผลการชำระเงินและมี Activation Bar
   - หลังจากการชำระเงินสำเร็จ Payment Gateway ส่งผลลัพธ์กลับไปที่ Ticket System และ Activation Bar จะสิ้นสุด

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 4: Activation (การรอผลลัพธ์) ใน Sequence Diagram

Activation Bar คือแถบที่แสดงถึงช่วงเวลาที่วัตถุทำงานหรือตอบสนองต่อคำขอ โดยจะปรากฏเมื่อวัตถุหนึ่งส่งคำขอไปยังวัตถุอื่น และรอผลลัพธ์จากการทำงานนั้น ๆ

ตัวอย่างที่ 2: ระบบการจัดการโรงเรียน (School Management System)

วัตถุที่เกี่ยวข้อง:

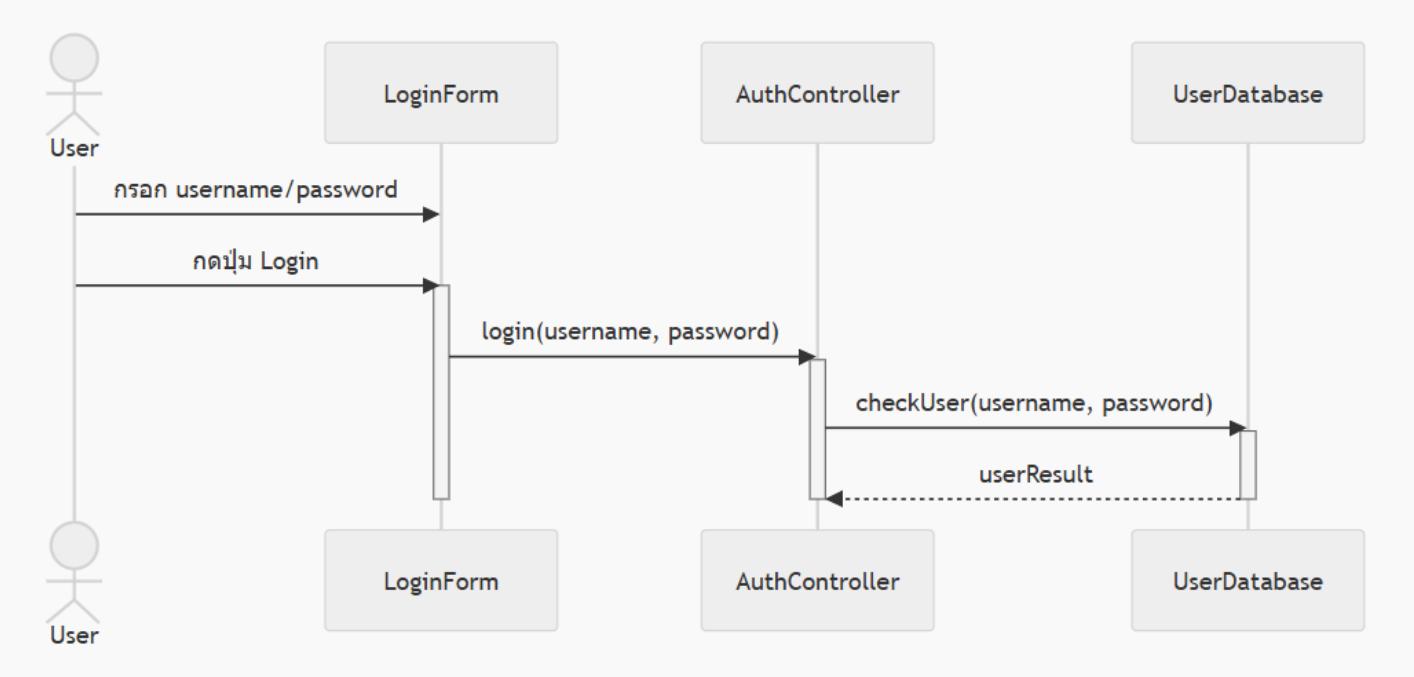
1. Teacher (ครู)
2. Grade System (ระบบบันทึกคะแนน)
3. Student Database (ฐานข้อมูลนักเรียน)
4. Notification System (ระบบแจ้งเตือน)

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 4: Activation (การรอผลลัพธ์) ใน Sequence Diagram

Activation Bar คือแถบที่แสดงถึงช่วงเวลาที่วัตถุทำงานหรือตอบสนองต่อคำขอ โดยจะปรากฏเมื่อวัตถุหนึ่งส่งคำขอไปยังวัตถุอื่น และรอผลลัพธ์จากการทำงานนั้น ๆ



6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 4: Activation (การรอผลลัพธ์) ใน Sequence Diagram

Activation Bar คือแถบที่แสดงถึงช่วงเวลาที่วัตถุทำงานหรือตอบสนองต่อคำขอ โดยจะปรากฏเมื่อวัตถุหนึ่งส่งคำขอไปยังวัตถุอื่น และรอผลลัพธ์จากการทำงานนั้น ๆ

ตัวอย่างที่ 2: ระบบการจัดการโรงเรียน (School Management System)
กระบวนการรับ – ส่ง ข่าวสารและการ Activation:

1. Teacher ส่งคำขอ "บันทึกคะแนน" ไปที่ Grade System
   - Teacher มี Activation Bar ขณะที่รอการตอบสนองจาก Grade System
   - Grade System เริ่มบันทึกคะแนนและมี Activation Bar แสดงการทำงาน
2. Grade System ส่งคำขอ "ตรวจสอบข้อมูลนักเรียน" ไปที่ Student Database
   - Grade System มี Activation Bar ขณะรอการดึงข้อมูลจาก Student Database
   - Student Database เริ่มดึงข้อมูลและมี Activation Bar
   - หลังจากดึงข้อมูลเสร็จ Student Database ส่งข้อมูลกลับไปที่ Grade System Activation Bar สิ้นสุด

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 4: Activation (การรอผลลัพธ์) ใน Sequence Diagram

Activation Bar คือแถบที่แสดงถึงช่วงเวลาที่วัตถุทำงานหรือตอบสนองต่อคำขอ โดยจะปรากฏเมื่อวัตถุหนึ่งส่งคำขอไปยังวัตถุอื่น และรอผลลัพธ์จากการทำงานนั้น ๆ

ตัวอย่างที่ 2: ระบบการจัดการโรงเรียน (School Management System)
กระบวนการรับ – ส่ง ข่าวสารและการ Activation:

3. Grade System ส่งการแจ้งเตือนไปยัง Notification System
   - Grade System มี Activation Bar ขณะรอการยืนยันจาก Notification System
   - Notification System เริ่มแจ้งเตือนและมี Activation Bar
   - เมื่อการแจ้งเตือนเสร็จสิ้น Notification System ส่งผลลัพธ์กลับไปที่ Grade System และ Activation Bar ของทั้งสองวัตถุจะสิ้นสุด

หมายเหตุ:
•Activation Bar จะอยู่บน Lifeline ของวัตถุที่ทำงานอยู่ และจะสิ้นสุดเมื่อวัตถุนั้นส่งผลลัพธ์กลับไปยังวัตถุที่ส่งคำขอ

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 5: ข่าวสารในลำดับถัดมาจะเป็นผลของการสร้างวัตถุของ Activation ใหม่ ใน Sequence Diagram

อธิบายขั้นตอนที่ 5

ในขั้นตอนนี้ หากมีการสร้างวัตถุใหม่ในระหว่างกระบวนการทำงาน ผลลัพธ์หรือการตอบสนองจากวัตถุเดิมอาจนำไปสู่การสร้างวัตถุใหม่ ซึ่งจะมี Activation ของตัวเอง และสามารถโต้ตอบกับวัตถุอื่น ๆ ได้

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)

วัตถุที่เกี่ยวข้อง:

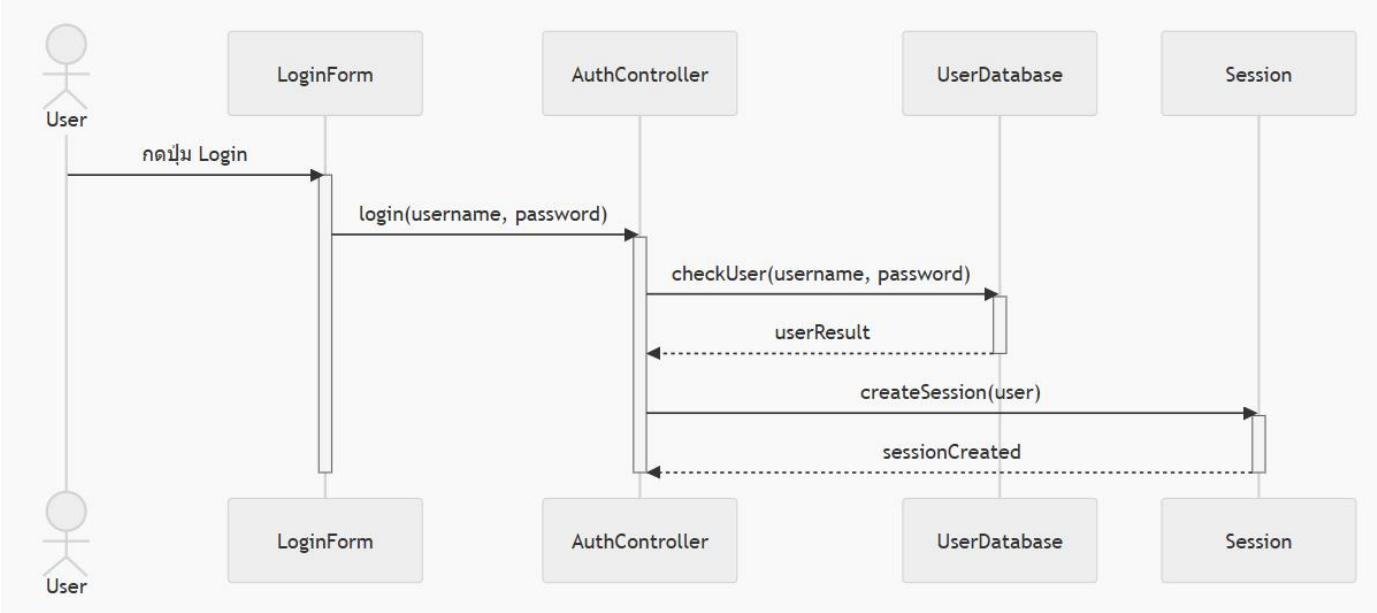
1. User (ผู้ใช้)
2. Movie Database (ฐานข้อมูลภาพยนตร์)
3. Ticket System (ระบบจัดการตั๋ว)
4. Payment Gateway (เกตเวย์การชำระเงิน)
5. Confirmation Email (อีเมลยืนยันการจอง)

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 5: ข่าวสารในลำดับถัดมาจะเป็นผลของการสร้างวัตถุของ Activation ใหม่ ใน Sequence Diagram

ขั้นตอนที่ 5: เพิ่ม Message ที่เกิดจากการสร้าง Activation ใหม่
ในตัวอย่าง Login ถ้าตรวจสอบสำเร็จ ระบบอาจสร้าง Session ใหม่ให้ผู้ใช้



อธิบาย: เมื่อ Login สำเร็จ AuthController อาจสร้าง Object ใหม่หรือเรียกใช้ Object ใหม่ เช่น Session เพื่อเก็บสถานะการเข้าสู่ระบบ

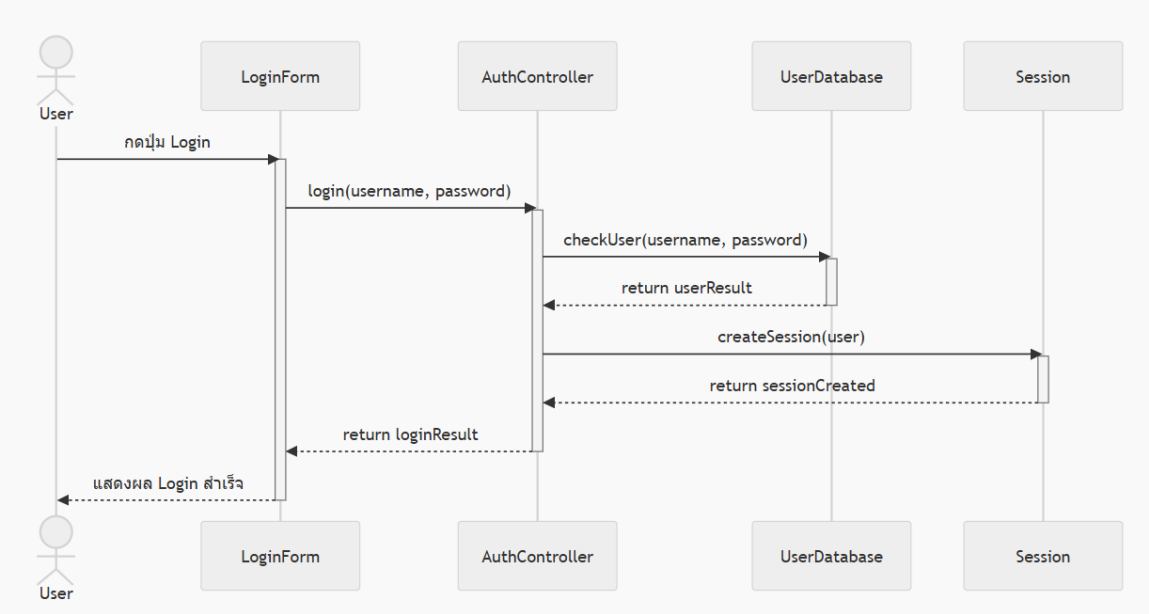
6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 6. ที่ตำแหน่งสุดท้ายของ Activation อาจจะใช้สำหรับการคืนค่ากลับไปยัง Caller ซึ่งจะแทนด้วยเส้นประที่เริ่มจากผู้รับไปยังผู้ส่ง

ขั้นตอนที่ 6: เพิ่ม Return Message

Return Message มักใช้เส้นประเพื่อแสดงการส่งผลลัพธ์กลับไปยังผู้เรียก



อธิบาย:

เส้นประ -->> แสดงการส่งค่ากลับ เช่น ผลการตรวจสอบผู้ใช้ หรือผลการสร้าง Session

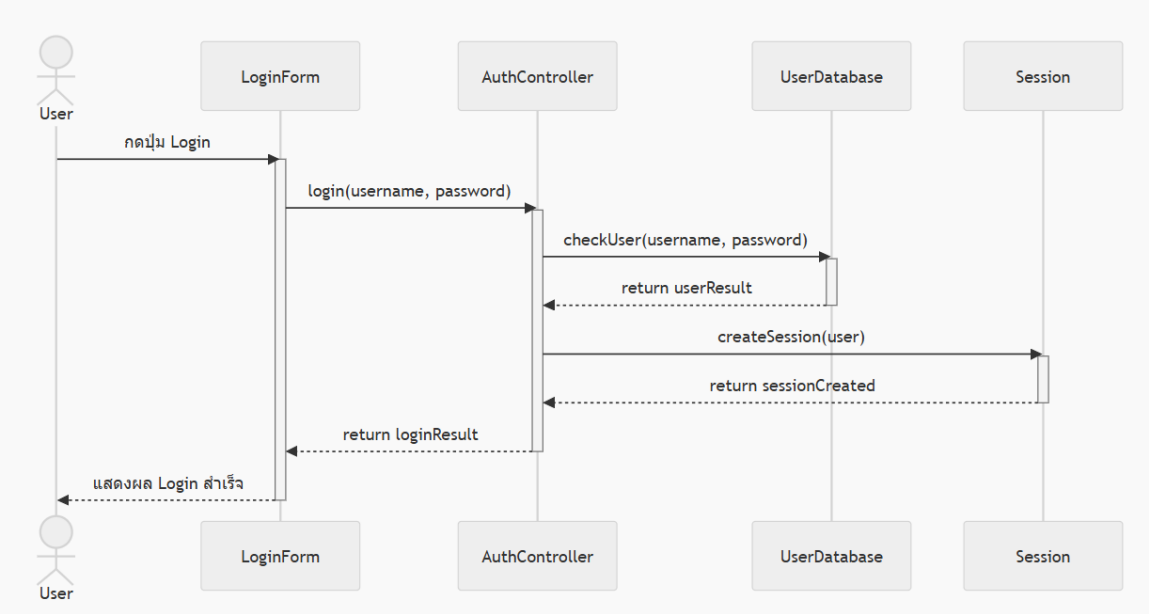
6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 6. ที่ตำแหน่งสุดท้ายของ Activation อาจจะใช้สำหรับการคืนค่ากลับไปยัง Caller ซึ่งจะแทนด้วยเส้นประที่เริ่มจากผู้รับไปยังผู้ส่ง

ขั้นตอนที่ 6: เพิ่ม Return Message

Return Message มักใช้เส้นประเพื่อแสดงการส่งผลลัพธ์กลับไปยังผู้เรียก



อธิบาย:

เส้นประ -->> แสดงการส่งค่ากลับ เช่น ผลการตรวจสอบผู้ใช้ หรือผลการสร้าง Session

6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 7. ข่าวสารที่ปรากฏในส่วนบนจะเกิดขึ้นก่อนเมสเจที่อยู่ถัดลงมาซึ่งเป็นไปตาม Lifetime

ขั้นตอนที่ 7: ตรวจสอบลำดับจากบนลงล่าง

เมื่อนำ Message มาวางครบแล้ว ต้องตรวจสอบว่า Message ที่อยู่ด้านบนเกิดก่อน และ Message ด้านล่างเกิดทีหลัง

ลำดับควรเป็นดังนี้

ลำดับ	Message
1	User กดปุ่ม Login
2	LoginForm ส่ง username/password ไปยัง AuthController
3	AuthController ตรวจสอบข้อมูลกับ UserDatabase
4	UserDatabase ส่งผลลัพธ์กลับ
5	AuthController สร้าง Session
6	Session ส่งผลกลับ
7	AuthController ส่งผล Login กลับไปยัง LoginForm
8	LoginForm แสดงผลให้ User

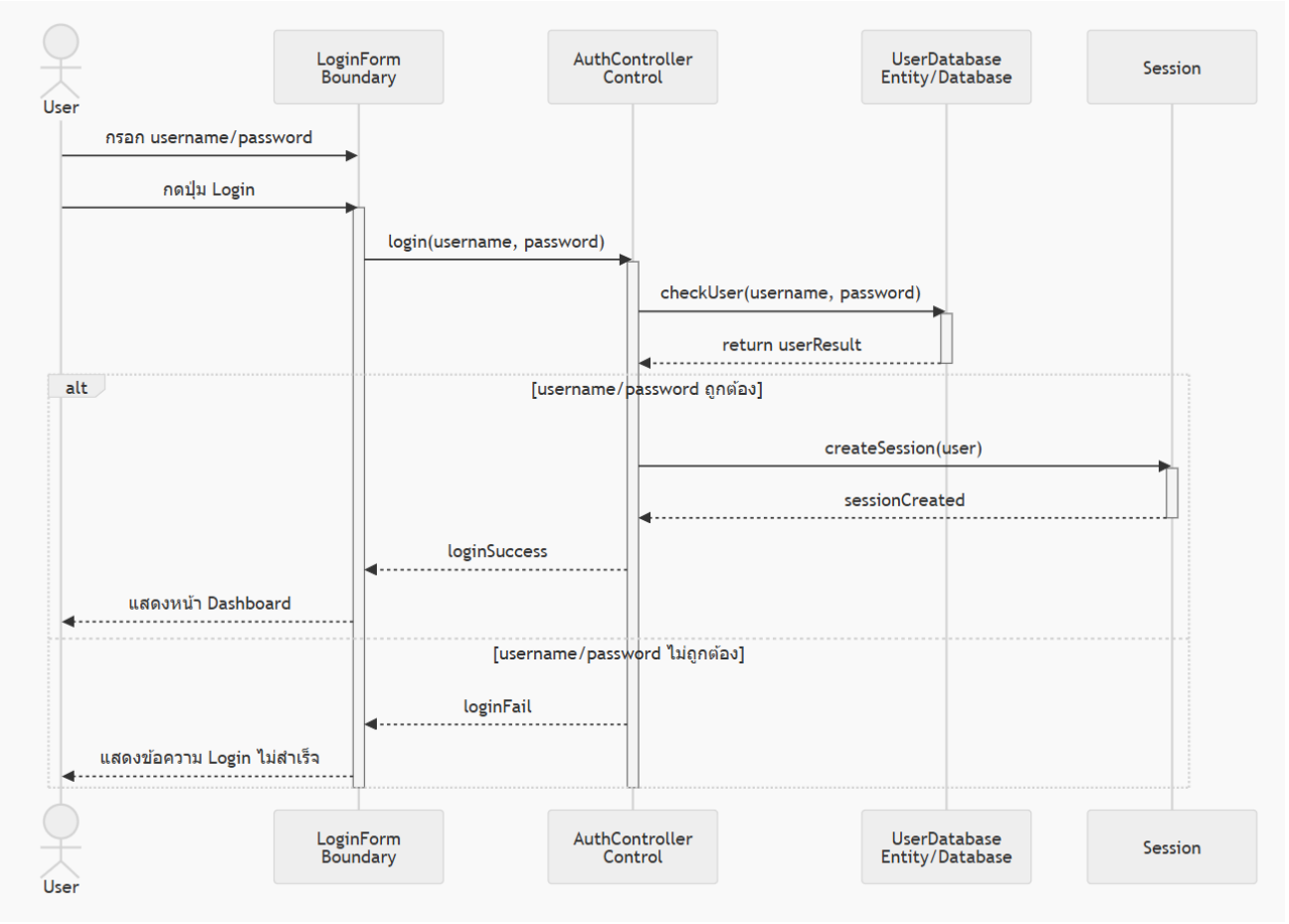
6.9 ขั้นตอนการสร้าง Sequence Diagram จาก Use Case

เลือก Use Case → อ่าน Main Flow → ระบุ Actor/Object → จัดลำดับ Message → เพิ่มเงื่อนไข → ตรวจสอบกับ Class Diagram

ขั้นตอนที่ 8. ข่าวสารที่อยู่ท้ายสุดจะเป็นการทำงานลำดับท้ายของยูสเคส

ขั้นตอนที่ 8: แผนภาพ Sequence Diagram ฉบับสมบูรณ์

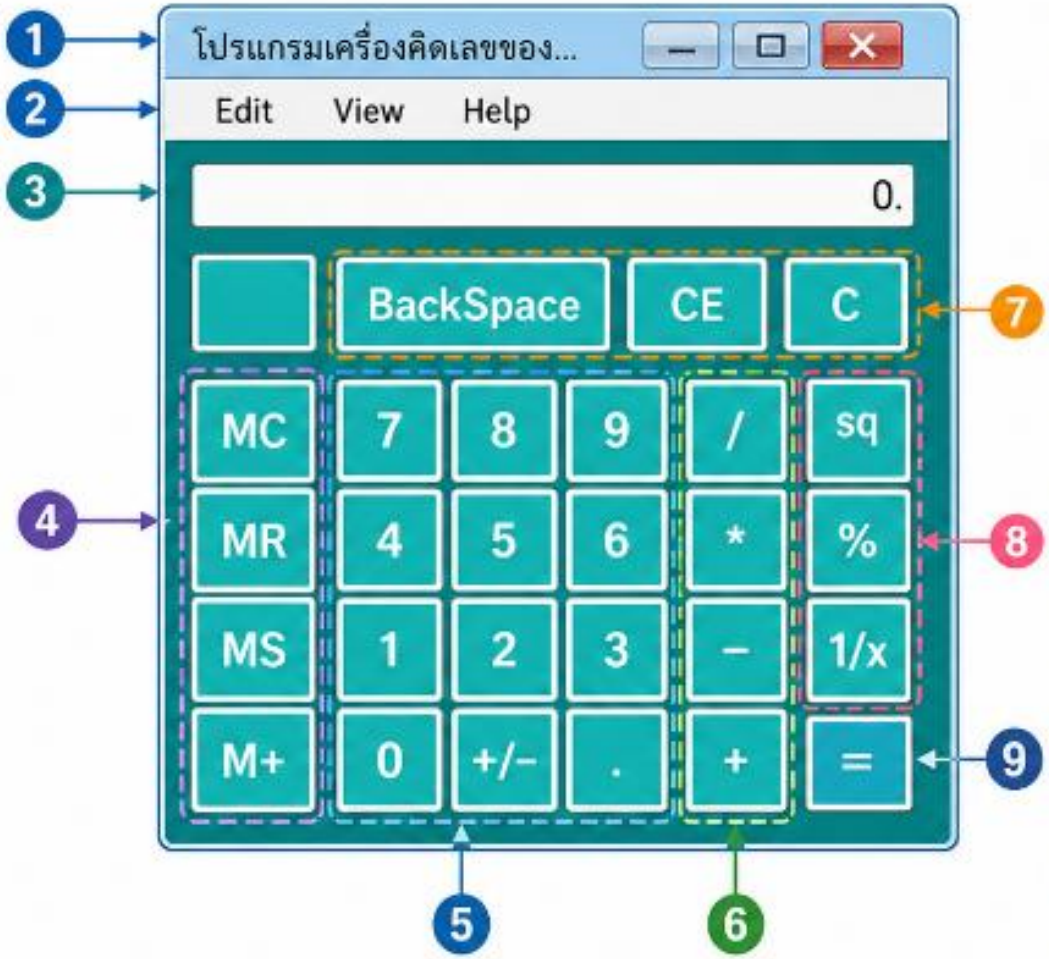
เวอร์ชันนี้เพิ่มเงื่อนไข Login สำเร็จ / ไม่สำเร็จ ด้วย alt



6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

ตัวอย่างโปรแกรมเครื่องคิดเลข

อธิบายส่วนประกอบของหน้าจอโปรแกรม

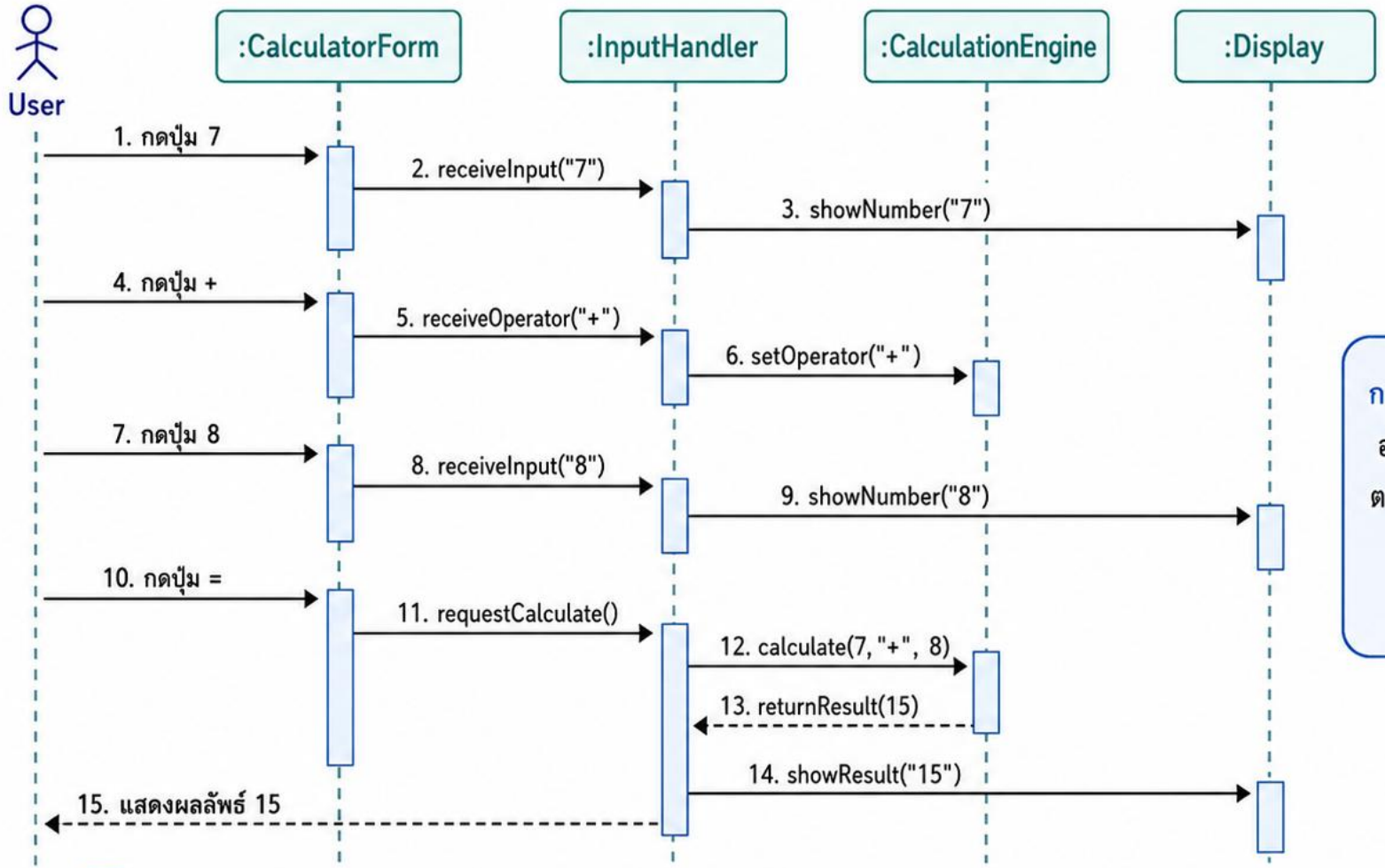


1. **Title Bar** — แถบชื่อโปรแกรม
2. **Menu Bar** — เมนูคำสั่ง เช่น Edit, View, Help
3. **Display** — ช่องแสดงผลการคำนวณ
4. **Memory Buttons** — ปุ่มหน่วยความจำ เช่น MC, MR, MS, M+
5. **Numeric Buttons** — ปุ่มตัวเลข 0–9
6. **Operator Buttons** — ปุ่มเครื่องหมายคำนวณ เช่น +, -, *, /
7. **Control Buttons** — ปุ่มควบคุม เช่น BackSpace, CE, C
8. **Special Function Buttons** — ปุ่มพิเศษ เช่น sq, %, 1/x, +/-, จุดทศนิยม
9. **Equal Button** — ปุ่ม = สำหรับแสดงผลลัพธ์

สรุป: โปรแกรมนี้รับค่าจากผู้ใช้ผ่านปุ่มคำสั่ง ประมวลผลการคำนวณ และแสดงผลผ่านช่อง Display

ตัวอย่าง Sequence Diagram : โปรแกรมเครื่องคิดเลข

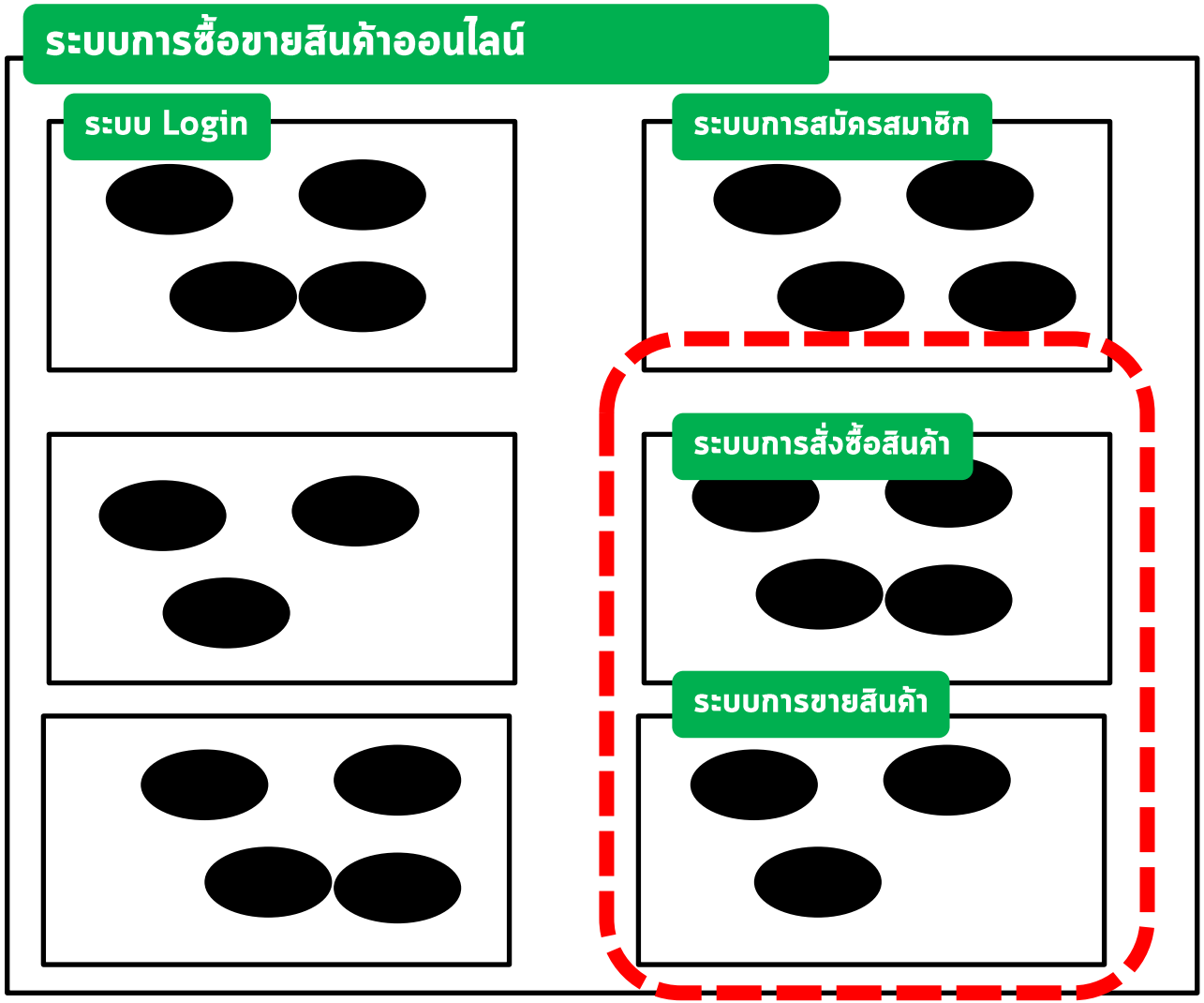
แสดงลำดับการโต้ตอบระหว่าง User, CalculatorForm, InputHandler, CalculationEngine และ Display



การอ่านไดอะแกรม:
อ่านจากบนลงล่าง
ตามลำดับเหตุการณ์

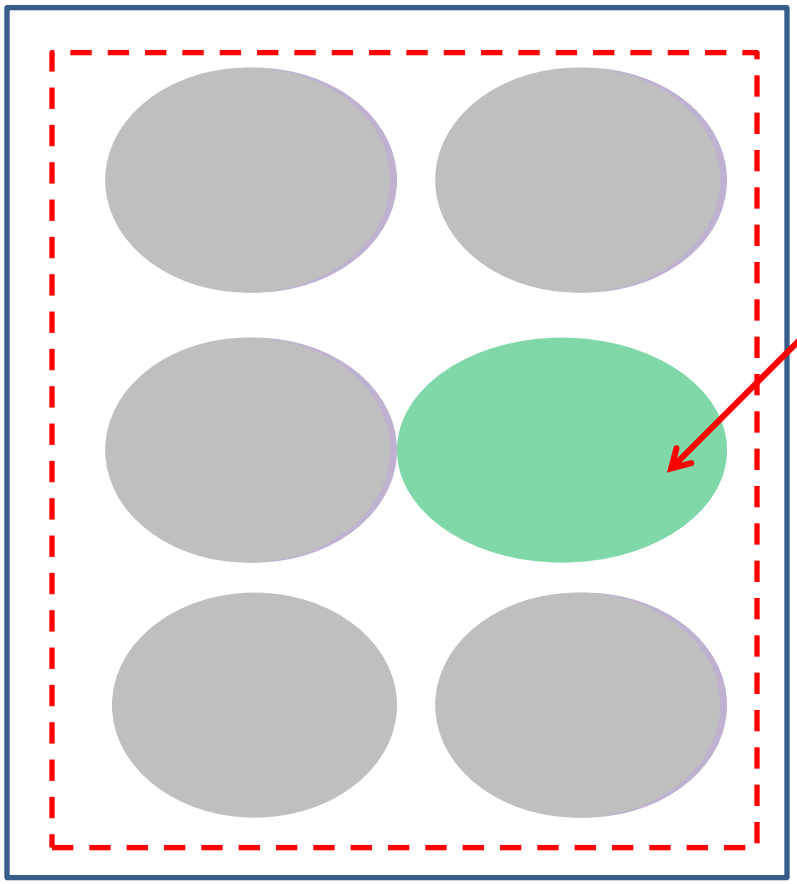
สรุป: ผู้ใช้กดตัวเลขและเครื่องหมายผ่านหน้าจอ CalculatorForm
จากนั้นระบบรับข้อมูล ประมวลผลใน CalculationEngine และแสดงผลลัพธ์ผ่าน Display

6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product



ในการพิจารณาว่าจะเลือกระบบใดไปเขียนเป็น Interaction Diagram / แผนภาพปฏิสัมพันธ์ ไม่มีกฎเกณฑ์ที่ตายตัว ทั้งนี้ขึ้นอยู่กับดุลยพินิจของ SA และทีมพัฒนาระบบตกลงร่วมกัน

6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product



ยกมา แล้วแต่ระบบ ที่ต้องจะพิจารณาว่า มีการปฏิสัมพันธ์ ระหว่างวัตถุอย่างไร

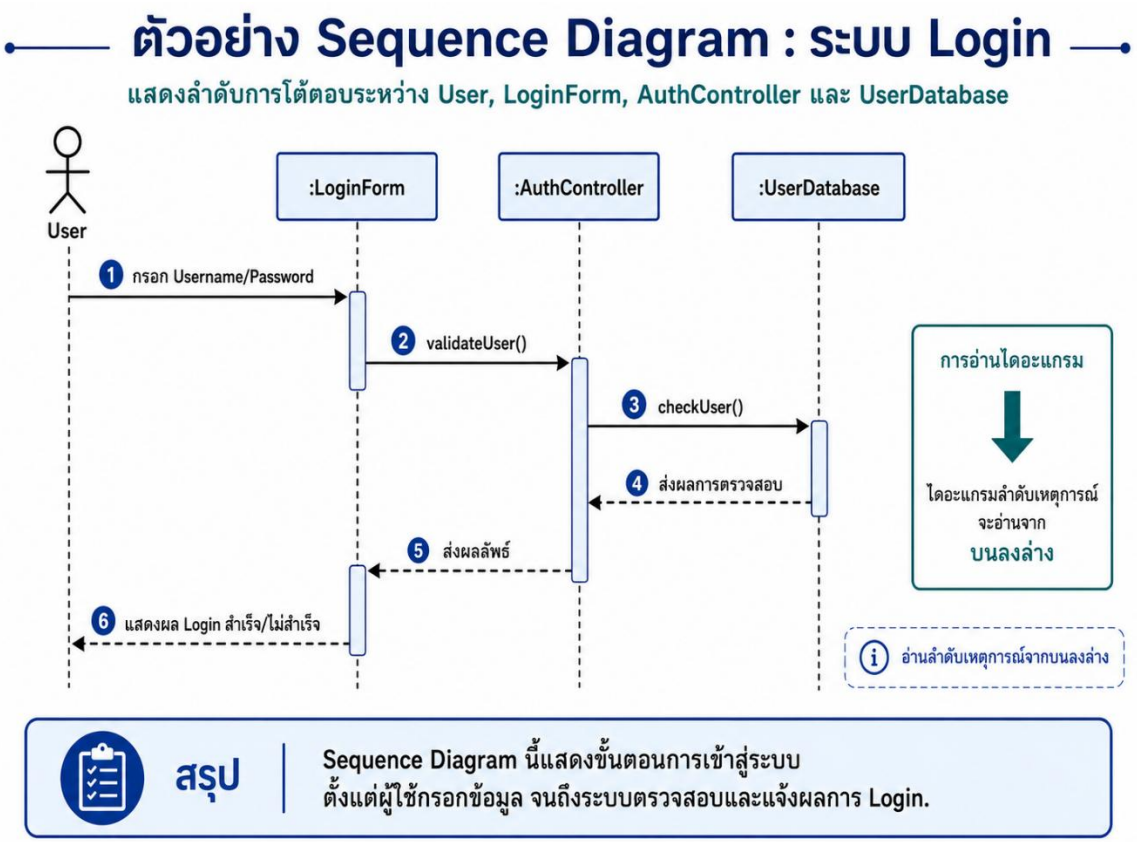
จะพิจารณาบางส่วนของยูสเคสหรือบางยูสเคส หรือทั้งระบบ

ขึ้นอยู่กับระบบที่พัฒนา หรือทีมงานพัฒนาระบบตกลงกัน

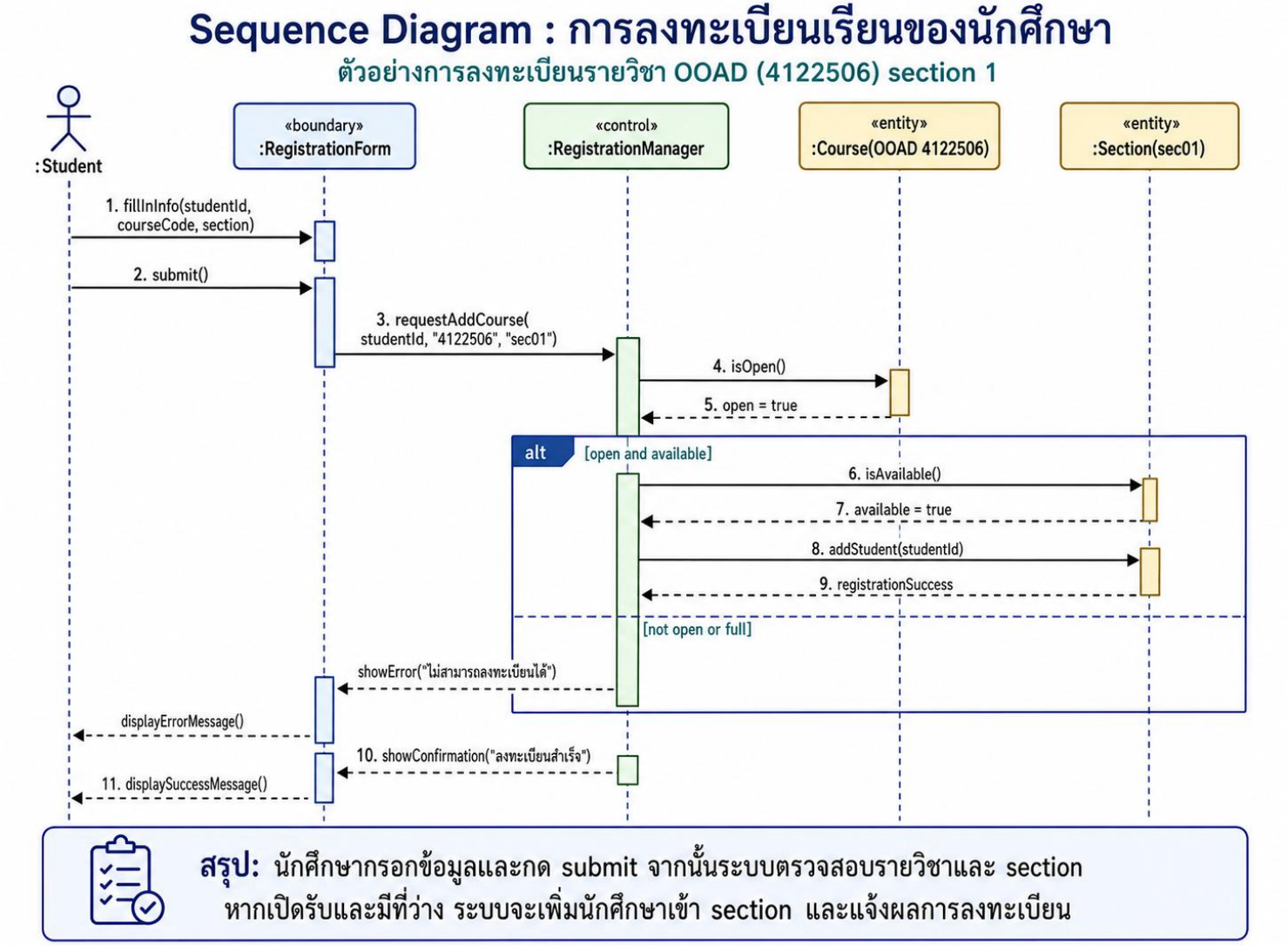
6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

ตัวอย่างแบบย่อ: ระบบ Login

ลำดับ	การโต้ตอบ
1	User → LoginForm: กรอก Username/Password
2	LoginForm → AuthController: validateUser()
3	AuthController → UserDatabase: checkUser()
4	UserDatabase → AuthController: ส่งผลการตรวจสอบ
5	AuthController → LoginForm: ส่งผลลัพธ์
6	LoginForm → User: แสดงผล Login สำเร็จ/ไม่สำเร็จ



6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

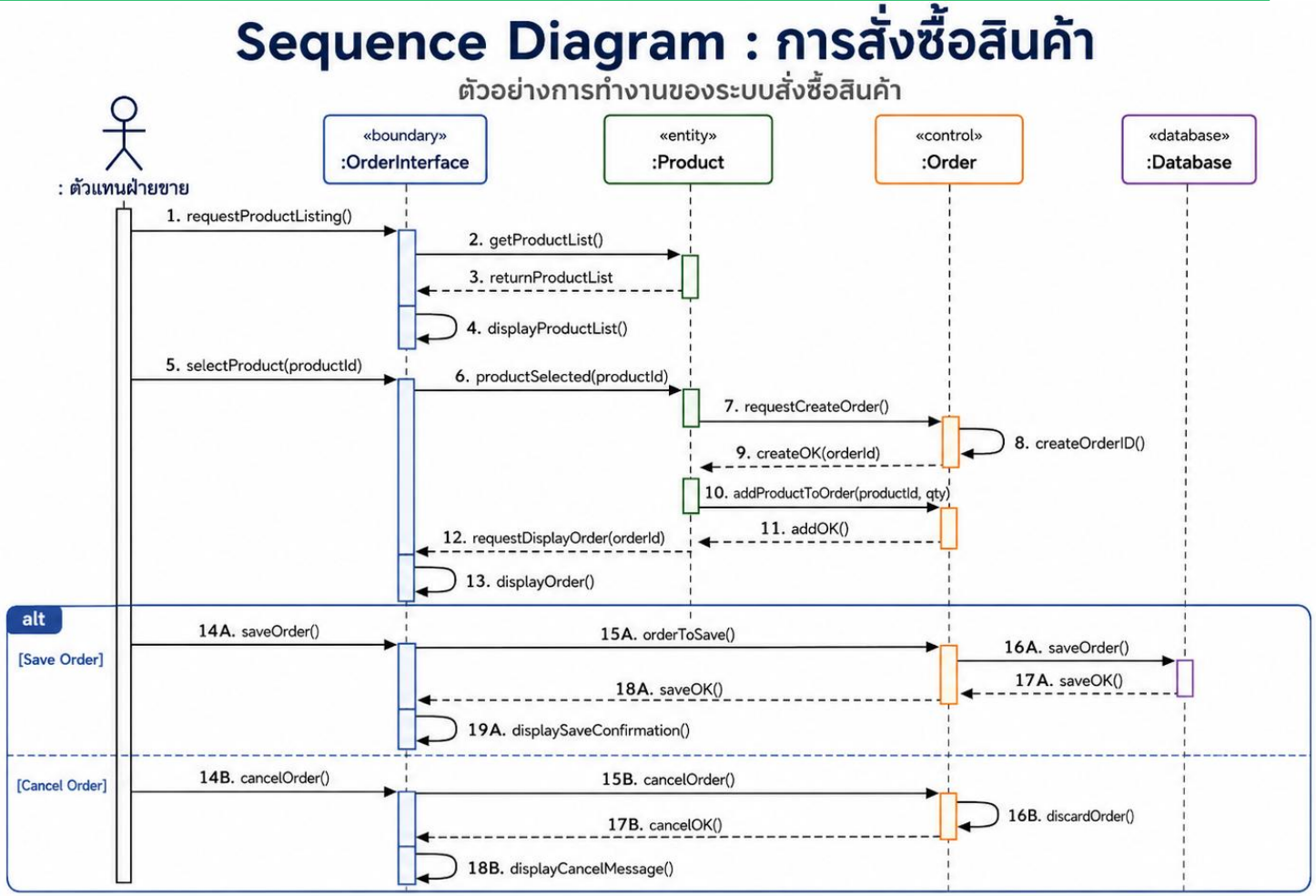


6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

Sequence Diagram สำหรับการสั่งซื้อสินค้า มีการทำงานเรียงตามลำดับเวลาดังนี้

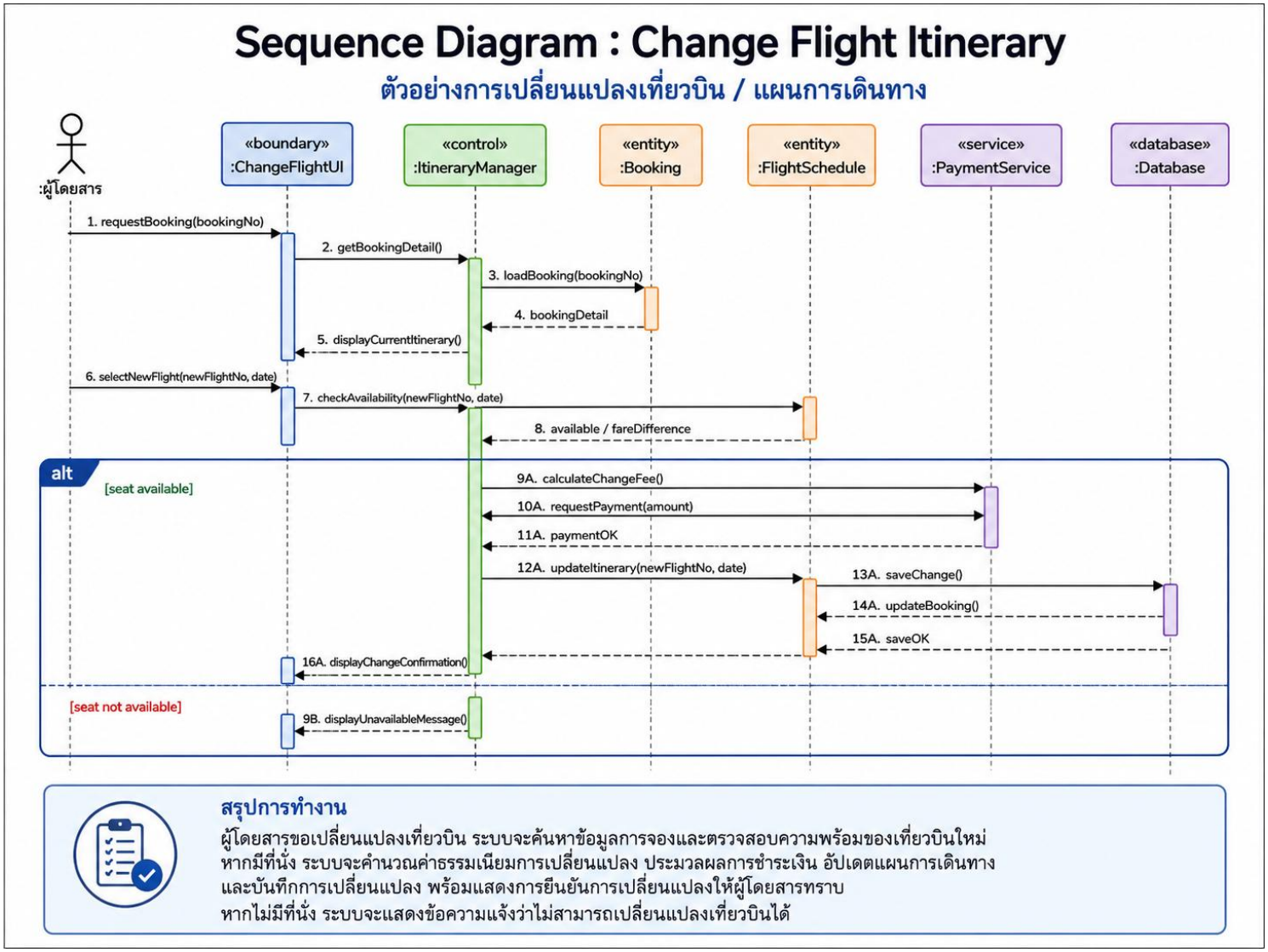
1. Request Product Listing: จาก User Interface ที่แสดงบนหน้าจอ Actor จะทำการขอให้ระบบแสดงรายการของสินค้าทั้งหมดออกมาทางหน้าจอ
2. Get Product List: Object User Interface จะทำการส่งตอคำขอนั้นไปให้เจ้าของเรื่อง (Product)
3. Return Product List: เจ้าของเรื่อง (Object Product) ก็จะทำการส่งรายการสินค้าทั้งหมดที่มีอยู่กลับคืนมาให้ Interface
4. Display Product List: Object Interface เมื่อได้รับรายการสินค้าแล้ว ก็จะทำการแสดงรายการของสินค้าเหล่านั้นออกมาทางหน้าจอให้ Actor ดู ซึ่งจะเห็นได้ว่าเป็นการส่งข้อความเข้าหาตัวเอง
5. Select Product: Actor จะทำการคลิกที่สินค้าตัวหนึ่ง (จากรายการสินค้าทั้งหมดที่แสดงบนหน้าจอ) ซึ่งลูกค้าต้องการซื้อ
6. Product Selected: Interface ก็จะทำการส่งต่อข้อมูลสินค้าที่ Actor ทำการเลือกแล้วนั้นไปยัง Product
7. Request to Creatr Order ID: เมื่อ Product ได้รับข้อมูลสินค้านั้นแล้ว ก็จะส่งคำขอไปยัง Prder เพื่อขอให้สร้าง ID ใหม่ ให้ตัวหนึ่งสำหรับใบสั่งซื้อสินค้าใบนี้
8. Create Order ID: เจ้าของเรื่องคือ Order ก็จะทำการสร้าง ID ใหม่ขึ้นมา ID หนึ่ง สำหรับใบสั่งซื้อนี้ ซึ่งจะเห็นได้ว่าเป็นการส่งข้อความเข้าหาตัวเองเช่นเดียวกับ Display Product List (เป็น Reflexive Message)
9. Create OK: หลังจากนั้น Object Order ก็ส่งข้อความไปบอก Product ว่าคำขอเพื่อสร้าง ID นั้นได้ทำสำเร็จแล้ว
10. Add Product to Order: หลังจากนั้น Product จึงทำการส่งข้อมูลของสินค้า (ที่ลูกค้าต้องการซื้อ) ไปให้ Order เพื่อทำการลงบันทึกในใบสั่งซื้อสินค้านั้น
11. Add OK: เมื่อเสร็จสิ้นการบันทึกแล้ว Order จึงบอกให้ Product รู้ว่าได้ดำเนินการเสร็จแล้ว
12. Request to Display Order: พร้อมกันนั้น Order ก็ส่งคำขอไปยัง Interface เพื่อขอให้ Interface ทำการแสดงข้อมูลรายละเอียดทั้งหมดของใบสั่งซื้อนั้นออกมาทางหน้าจอ เพื่อให้ Acter ดู
13. Display Order: Interface ทำการแสดงข้อมูลรายละเอียดของใบสั่งซื้อนี้ออกทางหน้าจอ (ตามคำขอจาก Order)
14. Save Order: ต่อจากนั้น โดย Interface Actor ก็จะทำการคลิกปุ่ม Save เพื่อทำการบันทึกข้อมูลใบสั่งซื้อสินค้านั้น
15. Cancel Order: ให้สังเกต Diagram ที่ข้อความ Save Order ที่ส่งจากตัวแทนฝ่ายขาย จะมีข้อความอีกอันหนึ่งคือข้อความ Cancel Order แยกออกมาจากจุดร่วมเดียวกัน จุดที่มีการแยกออกดังกล่าวเป็นสัญลักษณ์ของการตัดสินใจที่จะเลือกทำอันใดอัน หนึ่งเท่านั้น

6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product



สรุป: ตัวแทนฝ่ายขายขอรายการสินค้า เลือกสินค้า จากนั้นระบบสร้างใบสั่งซื้อ เพิ่มสินค้าในคำสั่งซื้อ และแสดงรายละเอียดคำสั่งซื้อ โดยผู้ใช้สามารถเลือกบันทึกคำสั่งซื้อหรือยกเลิกรายการได้

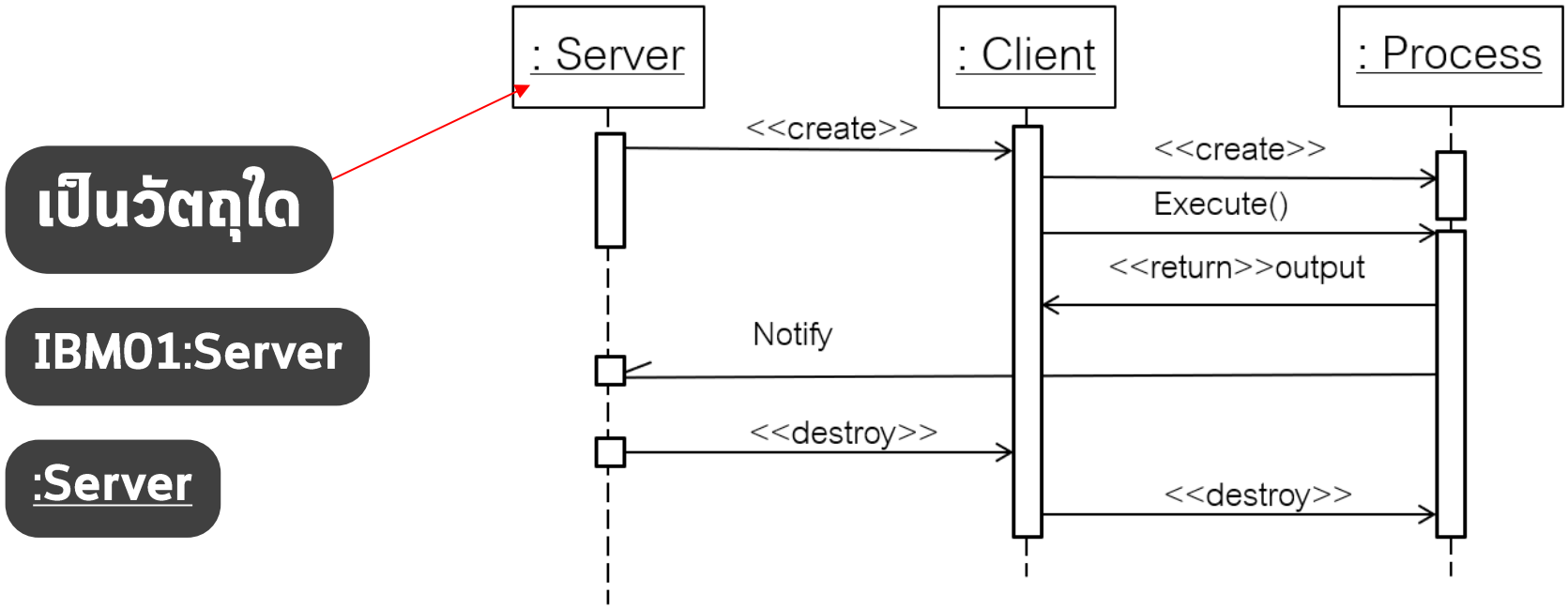
6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product



6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

แผนภาพลำดับ (Sequence diagram) หรือซีเควนไดอะแกรม (Sequence Diagram) เป็นแผนภาพที่ใช้อธิบายการทำงานของยูสเคส(Use Case) เพื่อแสดงถึงขั้นตอนการทำงานและแสดงลำดับของข่าวสารที่ส่งผ่านระหว่างคลาสที่โต้ตอบกัน นอกจากนี้แล้วแผนภาพลำดับ (Sequence diagram)ยังรวมถึงเงื่อนไขเวลาที่ใช้ในการทำงานด้วย โดยแผนภาพลำดับ (Sequence diagram)เป็นแผนภาพที่ใช้อธิบายถึงการติดต่อกันระหว่างวัตถุ ณ เวลาต่าง ๆ (นัฐพงศ์ ส่งเนียม, 2563) ที่แสดงถึงลำดับการส่งข่าวสารโดยมีส่วนประกอบที่สำคัญดังนี้

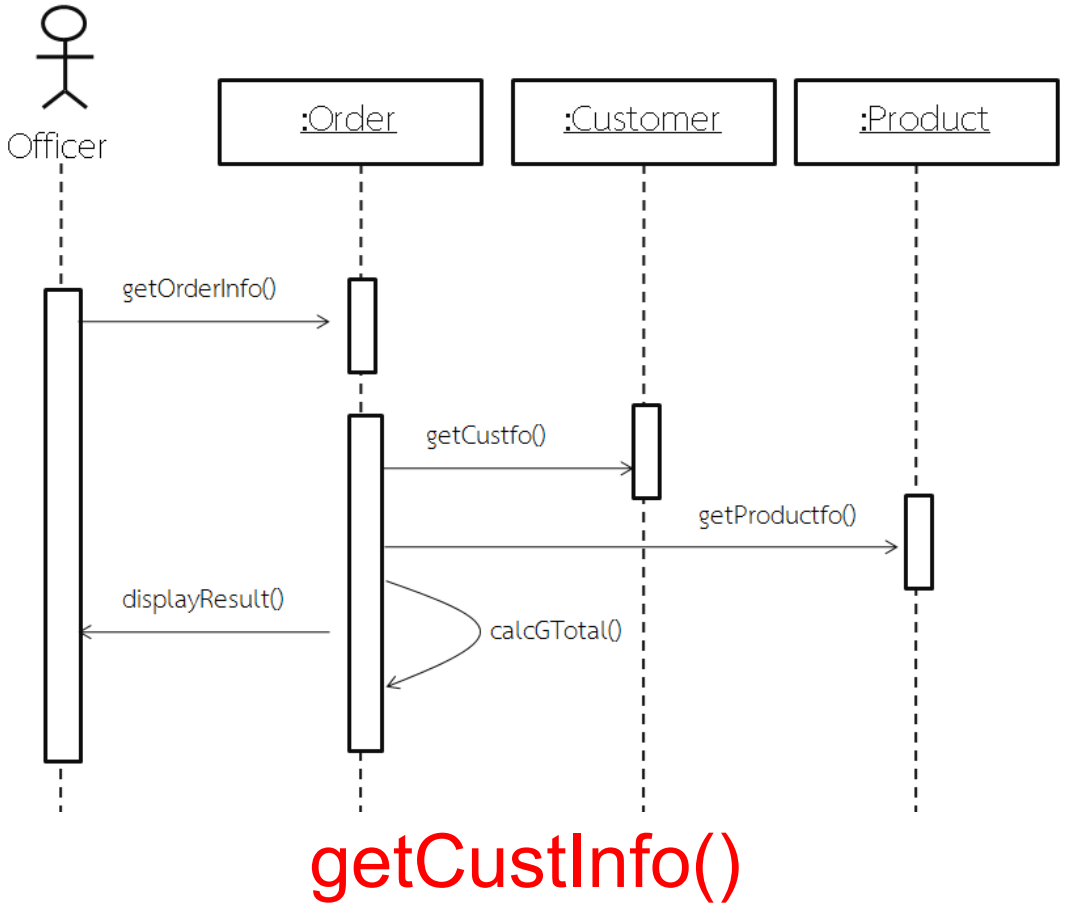
1. แกนตั้งแสดงเวลา
2. แกนนอนแสดงกลุ่มของวัตถุ



6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

ตัวอย่างแผนภาพลำดับ (Sequence diagram) ของการคำนวณการสั่งซื้อสินค้า

จากตัวอย่างแผนภาพลำดับ (Sequence diagram)ในภาพ เป็นการแสดงให้เห็นการส่งข่าวสารระหว่างวัตถุ “Order”, “Customer” และ “Product” โดยแอกเตอร์ “Officer” ต้องการทราบรายละเอียดรายการซื้อสินค้าจึงส่งข่าวสาร (Message) “getOrderInfo()” เข้าสู่ระบบมายังวัตถุ “Order” จากนั้นวัตถุ “Order”ส่ง ข่าวสาร “getCustInfo()” ไปที่วัตถุ “Customer” เพื่อขอรายละเอียดลูกค้า พร้อมกับส่งข่าวสาร “getProductInfo()” ไปที่วัตถุ “Product” เพื่อขอรายละเอียดสินค้า แล้วนำมาคำนวณหาราคาสินค้าที่ สั่งซื้อทั้งหมด (calcGtotal()) และแสดงผลทางจอภาพต่อแอกเตอร์ “Officer” จะเห็นคุณลักษณะของแผนภาพลำดับ (Sequence diagram)ได้อย่างชัดเจนว่าเป็นแผนภาพที่สามารถแสดงให้เห็นถึงการปฏิสัมพันธ์ระหว่างวัตถุของคลาสตามลำดับของเวลาเป็นสำคัญ



6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

ตัวอย่างที่ 1: ระบบจองตั๋วหนังออนไลน์ (Online Movie Ticket Booking System)
กระบวนการรับ – ส่ง ข่าวสารและการสร้างวัตถุใหม่:

1. User ส่งคำขอ "ค้นหาภาพยนตร์" ไปที่ Movie Database
   - Movie Database ส่งรายการภาพยนตร์กลับไปยัง User
2. User เลือกภาพยนตร์และทำการจองตั๋ว → ส่งคำขอไปที่ Ticket System
   - Ticket System ตรวจสอบและยืนยันการจอง
3. Ticket System ส่งข้อมูลการชำระเงินไปที่ Payment Gateway
   - เมื่อการชำระเงินสำเร็จ, Payment Gateway ส่งผลลัพธ์กลับไปที่ Ticket System
4. Ticket System ส่งการยืนยันการจองตั๋วกลับไปยัง User
   - พร้อมกับคำขอสร้าง Confirmation Email
   - Ticket System จะสร้างวัตถุใหม่ชื่อ Confirmation Email และส่งข้อมูลไปที่ Confirmation Email
   - Confirmation Email มี Activation Bar ที่แสดงว่ากำลังสร้างอีเมลยืนยัน
5. Confirmation Email ส่งอีเมลยืนยันไปยัง User
   - เมื่อส่งอีเมลเรียบร้อยแล้ว Activation Bar ของ Confirmation Email จะสิ้นสุด

6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

สรุป Object ที่อยู่ในแผนภาพ

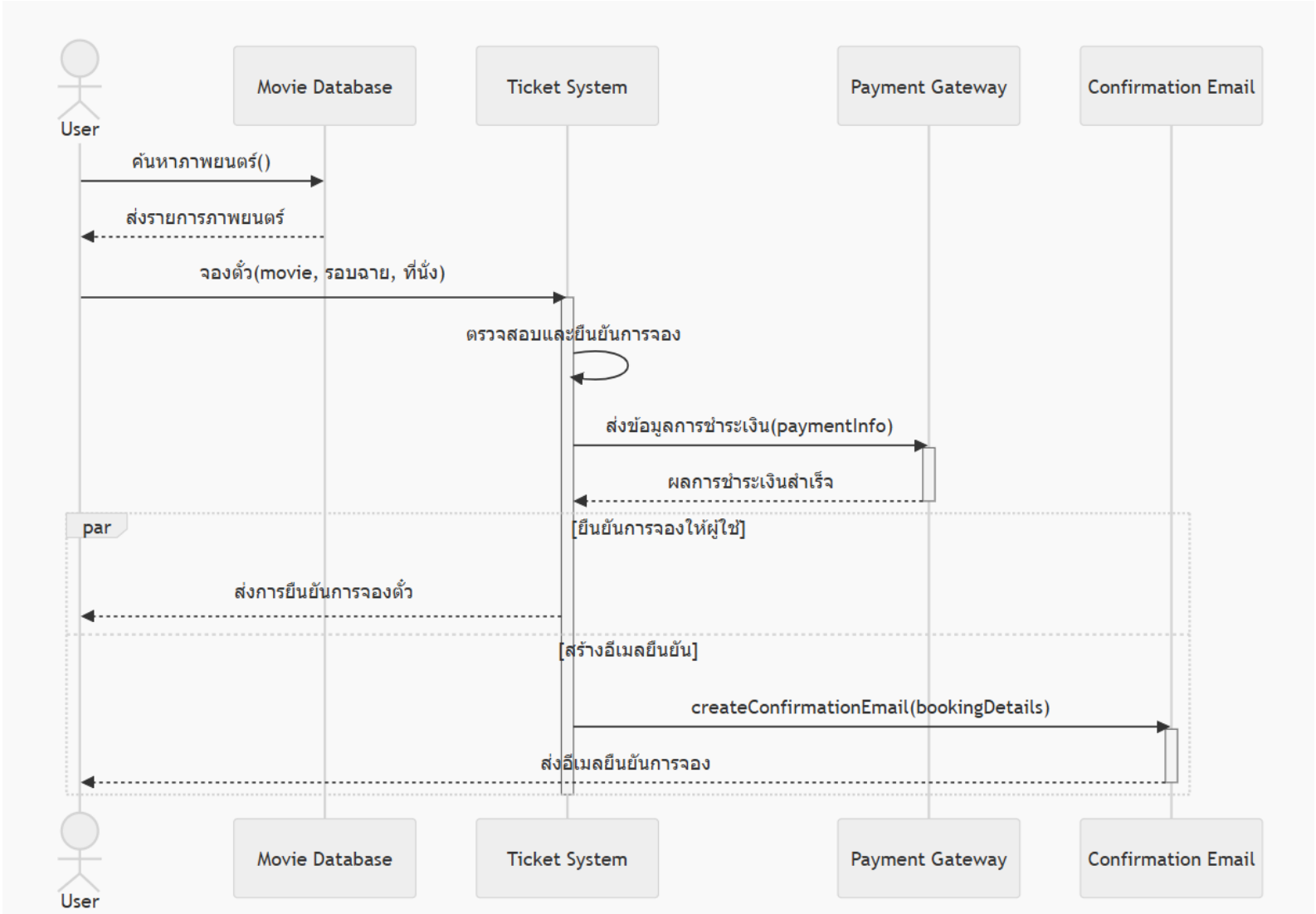
ประเภท	Object
Actor	User
Participant / Object	Movie Database
Participant / Object	Ticket System
Participant / Object	Payment Gateway
Created Object	Confirmation Email

สามารถจัดกลุ่มได้แบบนี้:

Actor: User

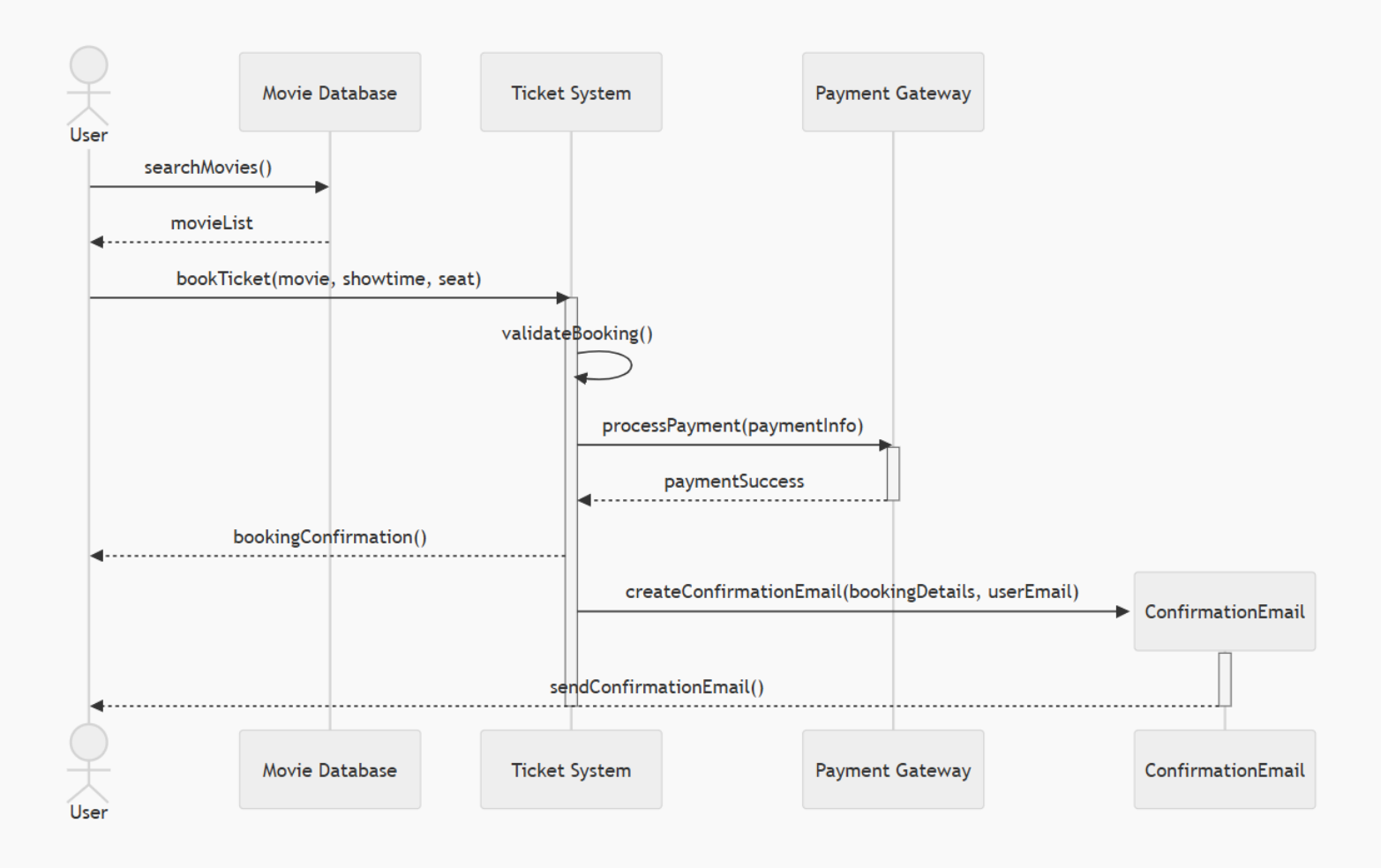
- **Boundary Object: Ticket System**
- **Entity / Data Source: Movie Database**
- **External System: Payment Gateway**
- **Created Object: Confirmation Email**

6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product



6.10 ตัวอย่าง Sequence Diagram: Login / Register / Order Product

- ถ้าต้องการให้เห็น “การสร้างวัตถุใหม่” ชัดขึ้น
- เวอร์ชันนี้จะเน้นว่า Confirmation Email ถูกสร้างขึ้นใหม่ หลังจากชำระเงินสำเร็จ



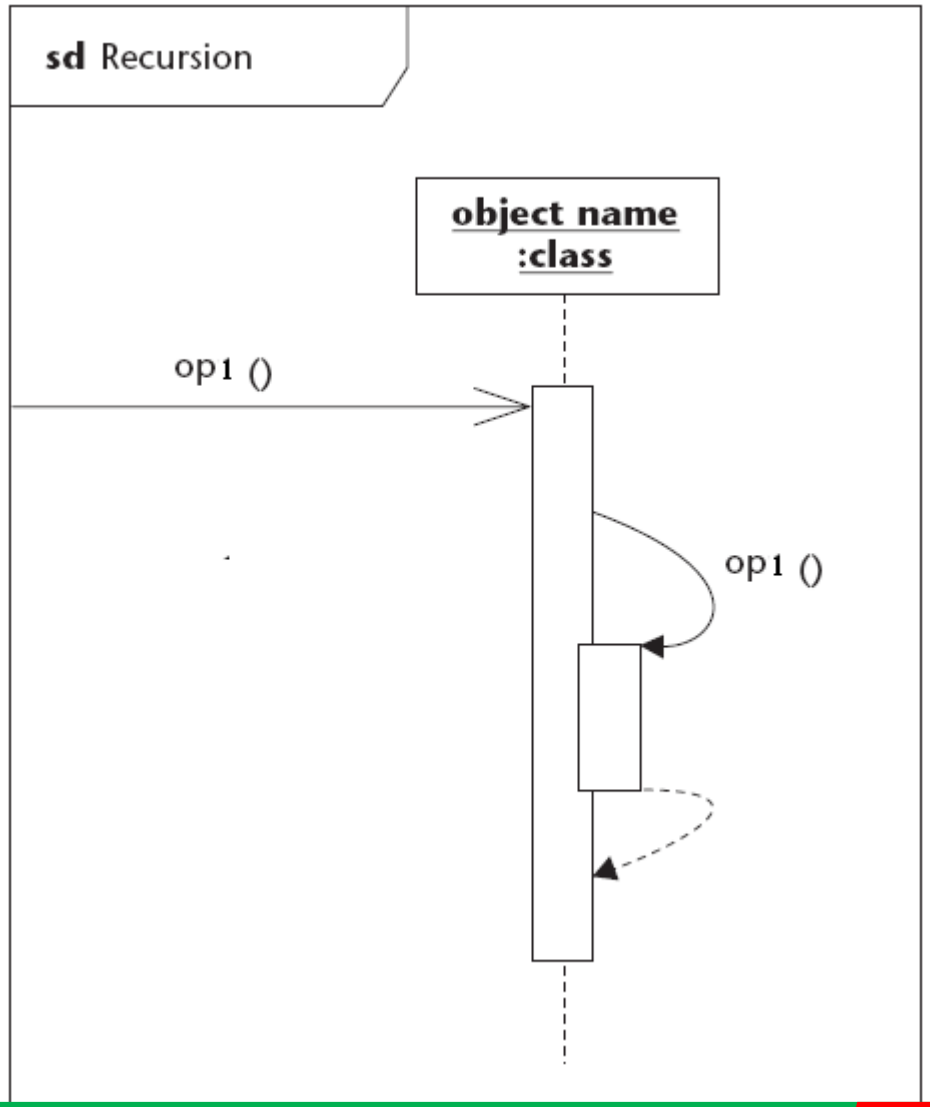
หลักที่สำคัญ / ขั้นตอนที่สำคัญ

1. เริ่มต้นจากการทำ Requirement specifications {หาข้อกำหนดของความต้องการ จากผู้ใช้}
2. แล้วมาทำการวิเคราะห์ และออกแบบระบบเพื่อให้ได้มาซึ่ง Use Case diagrams และ use case description
3. จากนั้นก็นำเอา Use case ที่ได้จากข้อ 2. มา ออกแบบ หรือมาทำ Sequence และ Communication Diagram (formerly called Collaboration Diagram) { ถ้าเป็น class diagram ก็ให้ทำแบบเดียวกัน}

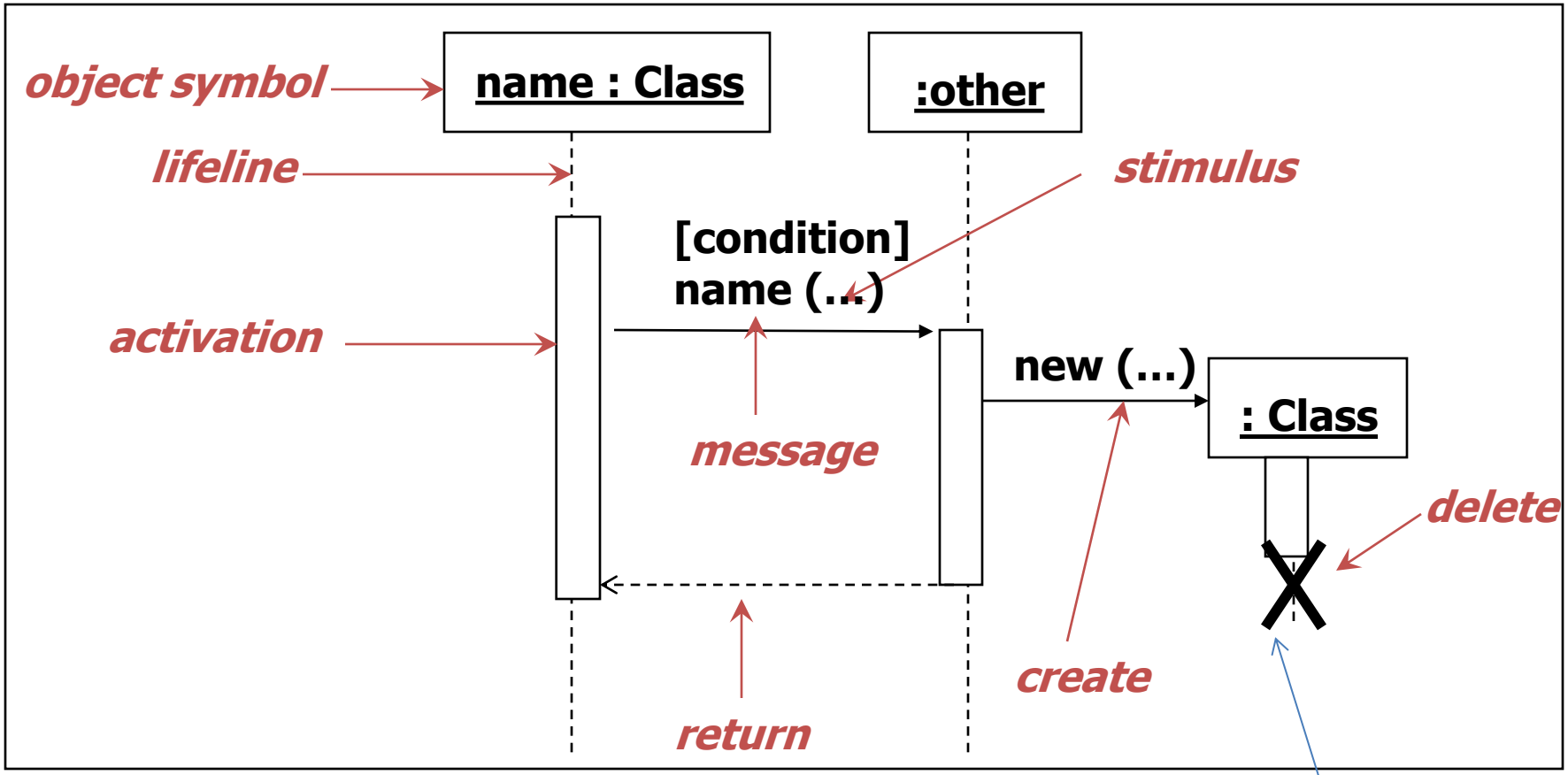
Sequence Diagrams ใน UML 2.0 (Recursion)

Loop back recursion

ในบางครั้งอาจมีการเรียกใช้งานฟังก์ชันตัวเอง หรือเรียกว่าลูปแบค (Loop back)

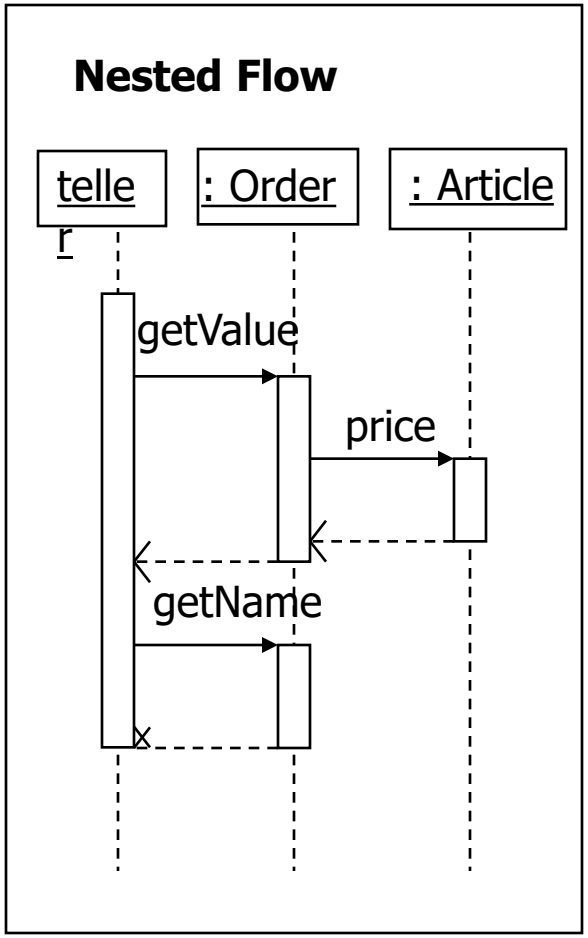


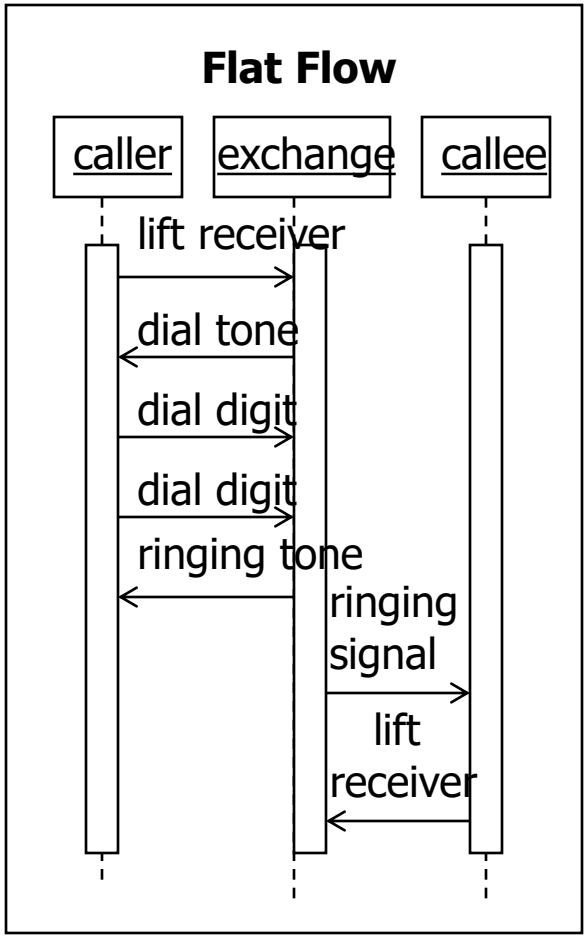
Notation : Sequence Diagram

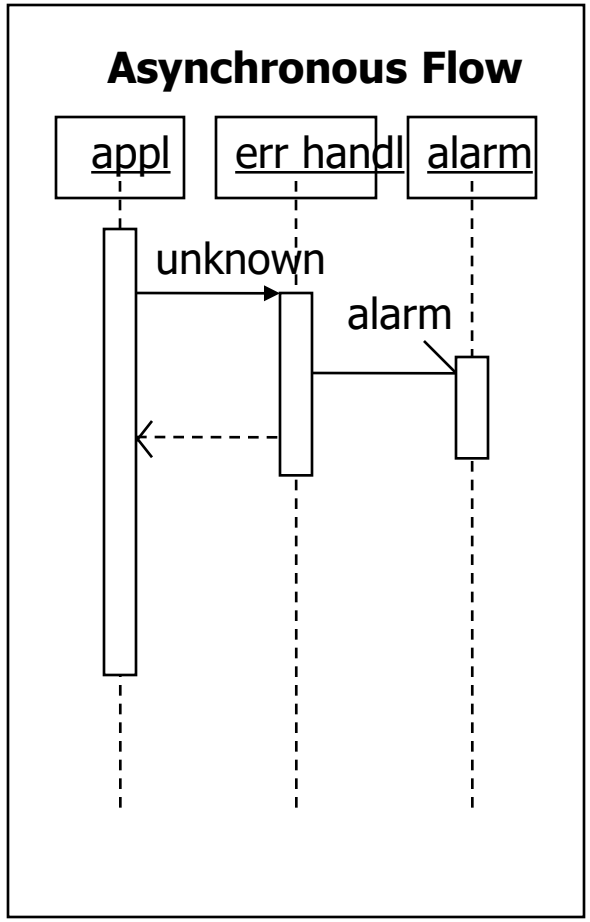


Return Function

Example: Different Arrows

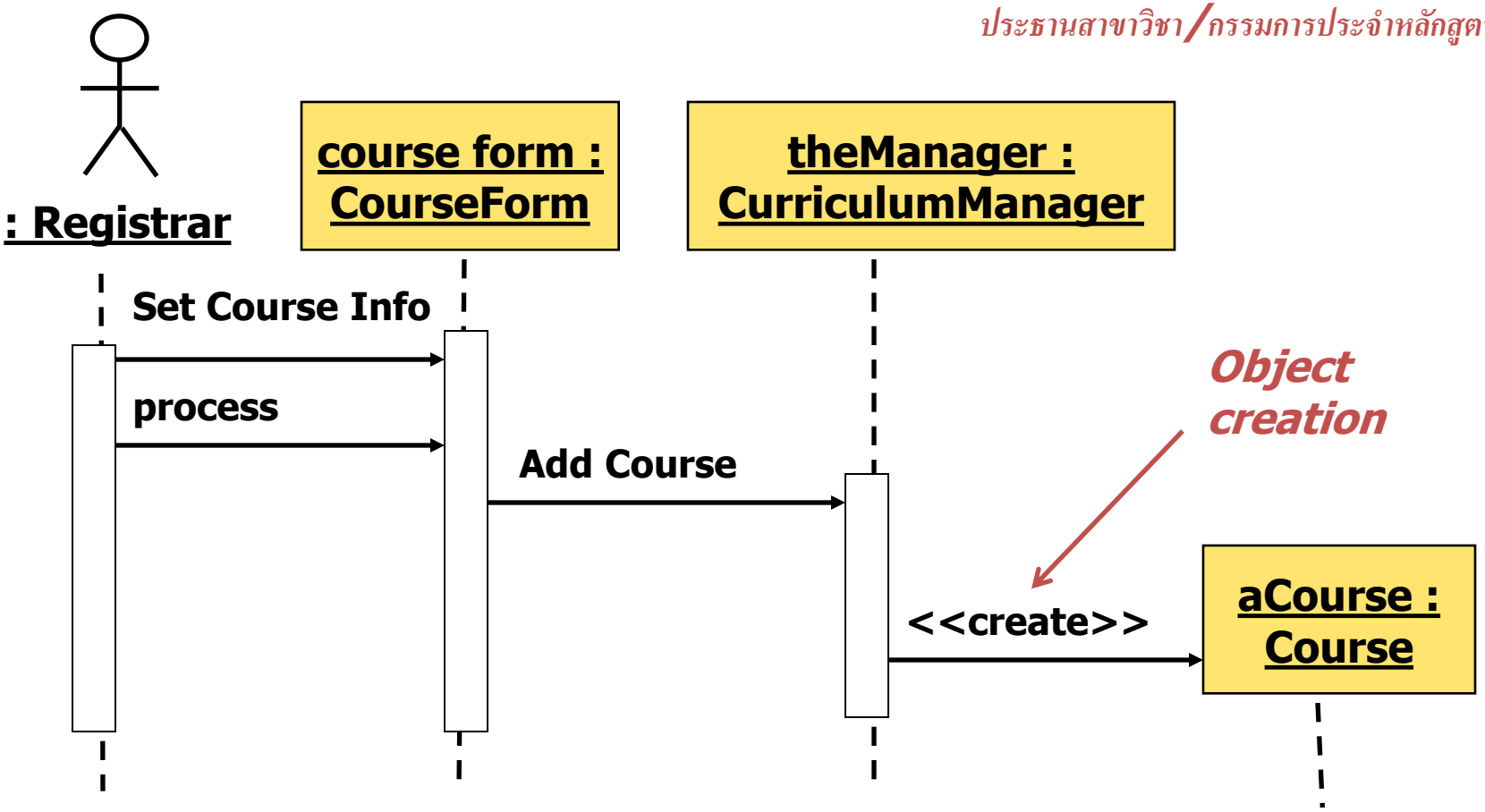






Example: Sequence diagram

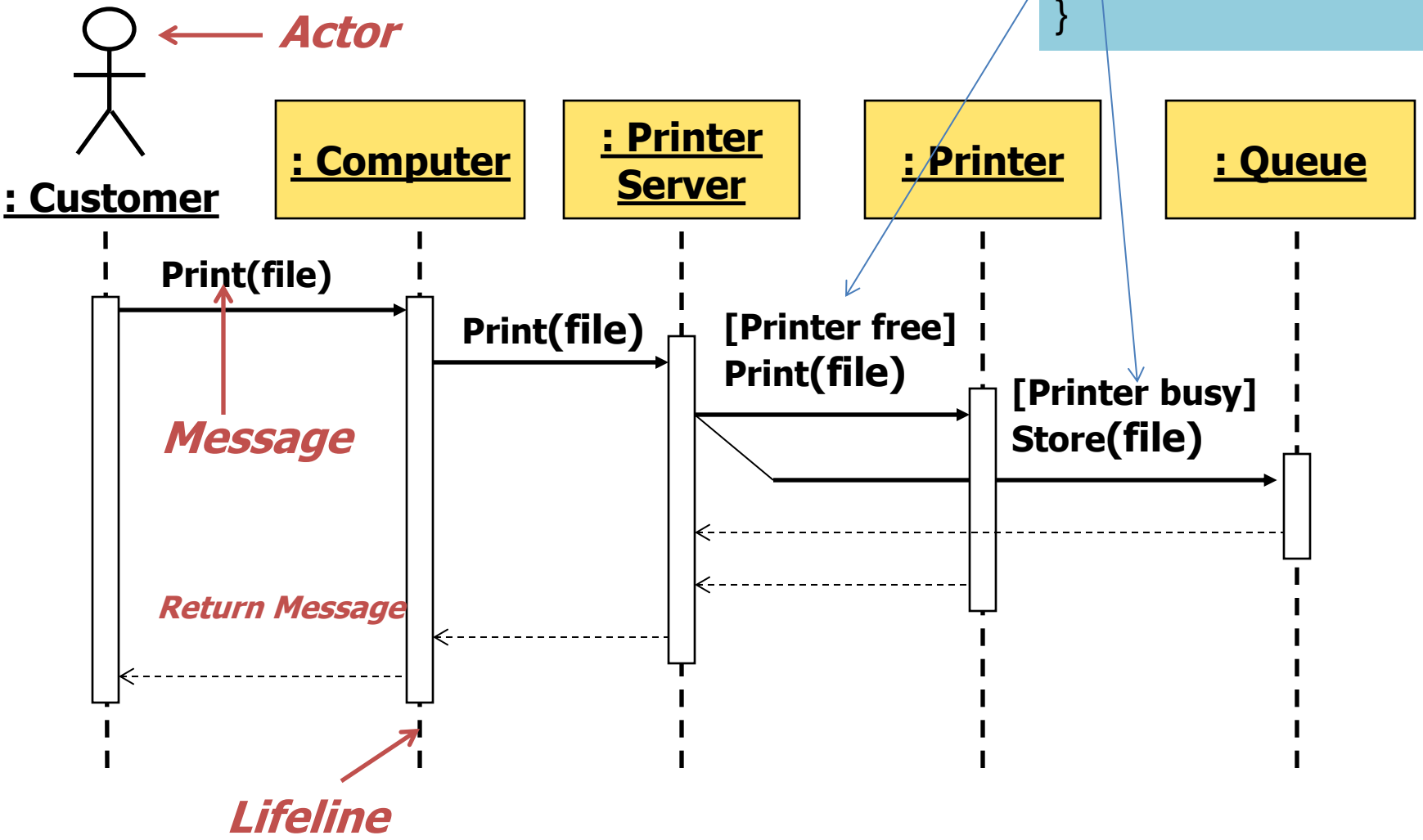
ประธานสาขาวิชา/กรรมการประจำหลักสูตร



เช่นวิชา OOP , OOAD , Java Programinng ,IoT

Example: Sequence diag

```
If (printer == free) {
…
} else {
…
}
```



ทำไมต้อง Sequence Diagram?

- เมื่อมีการออกแบบ Class และ Object แล้วจะต้องแสดงว่า แต่ละ Object มีการส่ง *message* ให้กันอย่างไร
- Message ที่ส่งให้กัน คือการกระทำต่าง ๆ ที่เกิดขึ้นระหว่าง Object หรือ function / Method(OOP) นั่นเอง
- Message แสดงให้เห็นว่า Object ต่าง ๆ มีการติดต่อสื่อสารกันอย่างไร ในช่วงระยะเวลาหนึ่ง
- เพื่อใช้ในการทดสอบการทำงานของโปรแกรมว่าเป็นไปตามข้อกำหนดของความต้องการหรือไม่

เทคนิคการสร้าง Sequence Diagram จาก Use Case และ Class Diagram

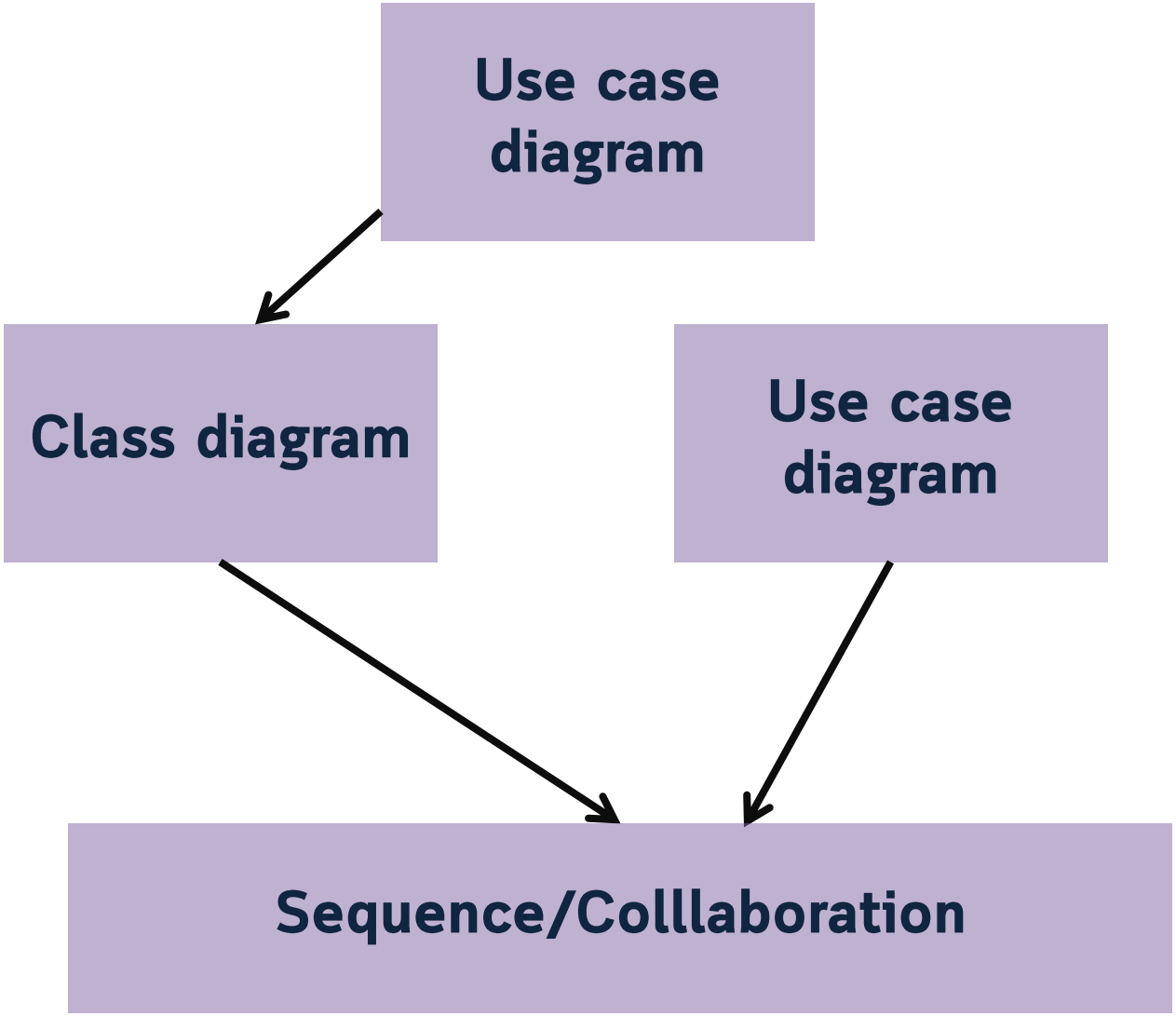
1) พิจารณาทีละ Use Case โดยยังไม่ต้องคำนึงถึงความสัมพันธ์ที่แต่ละ Use Case มีต่อกัน
2) พิจารณาแต่ละ Use Case ว่ามี Class/Object ใดร่วมทำให้เกิดกิจกรรมใน Use Case นั้น ๆ บ้าง
3) นำเอา Class/Object ต่าง ๆ มาเรียงต่อกันในแนวนอน โดยนำ Actor (ในกรณีที่ Use Case นั้นมี Actor ด้วย) ไว้ที่ด้านซ้ายมือสุดเสมอ แล้วนำ Class/Object ต่าง ๆ เรียงต่อกันจากซ้ายไปขวาตามความเหมาะสม
4) หาก Use Case นั้นมี Actor โดยปกติแล้วกิจกรรมแรกที่ถูกเรียกมักจะเกิดจาก Actor ก่อนเสมอ ดังนั้นเมื่อเกิดกิจกรรมที่ไปยัง Class/Object ใด ให้ย้าย Class/Object นั้นมาทางซ้าย ทำเช่นนี้ไปเรื่อย ๆ จนกระทั่งกิจกรรมทั้งหมดครบถ้วน
5) กรณีที่มีกิจกรรมเกิดขึ้นใหม่ แต่ Method ที่เกิดขึ้นไม่มีใน Class/Object ที่ลูกศรชี้ไป ให้เข้าไปเพิ่ม Method นั้น ๆ ลงไปที่ Class นั้นใน Class Diagram

เทคนิคการสร้าง Sequence Diagram จาก Use Case และ Class Diagram

6) หากต้องมีการเพิ่ม Class ใหม่เข้าไปใน Sequence Diagram ต้องเข้าไปเพิ่มเติม Class Diagram รวมทั้ง Relationship ที่มีทั้งหมดใน Class Diagram ด้วย

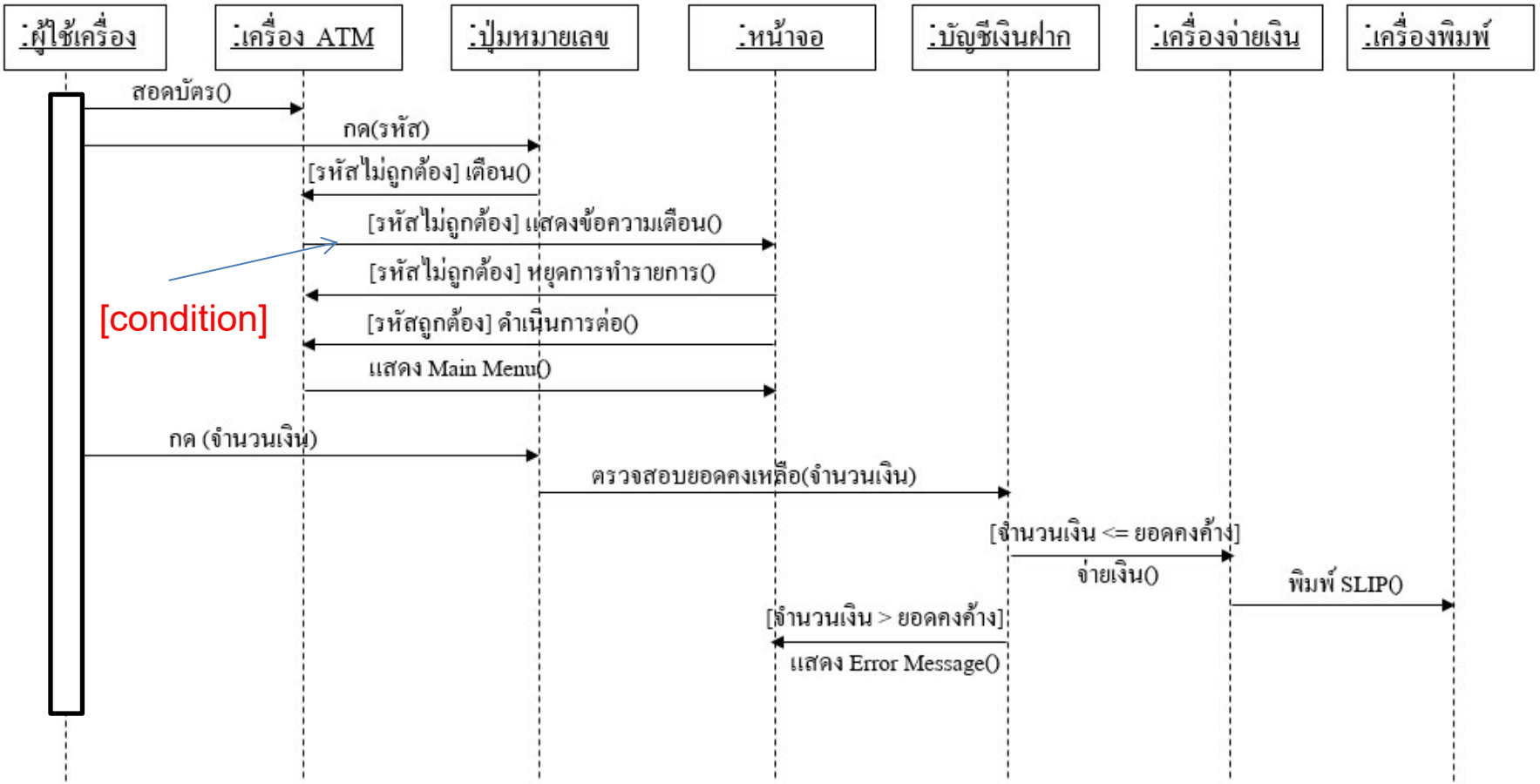
7) ทำขั้นตอน 1-6 จนครบทุก Use Case

8) การสร้างความสัมพันธ์ของ Sequence Diagram จาก Use Case ที่มีการ Include/Extends ทำได้โดยการนำ Class และกิจกรรมที่เกิดขึ้นใน Use Case ที่ถูก Include/Extends มาแทรกเข้าไปใน Use Case ที่เรียกใช้ และใช้กิจกรรมเพื่อเชื่อมโยง Sequence Diagram ทั้งสอง


Use case diagram
Class diagram
Use case diagram
Sequence/Colllaboration

ตัวอย่างที่ 1 Sequence Diagram ของระบบ ATM

- ระบบ ATM ประกอบด้วย Use Case ต่าง ๆ ดังนี้
  - การถอนเงิน
  - การฝากเงิน
  - การโอนเงิน
  - การเข้าระบบ (กรอกรหัส)
- ระบบประกอบด้วย Class/object ต่าง ๆ ดังนี้
  - เครื่อง ATM
  - เงินสด
  - บัญชีเงินฝาก
  - บัตร ATM
  - ธนาคาร (Actor)
  - ผู้ใช้เครื่อง (Actor)

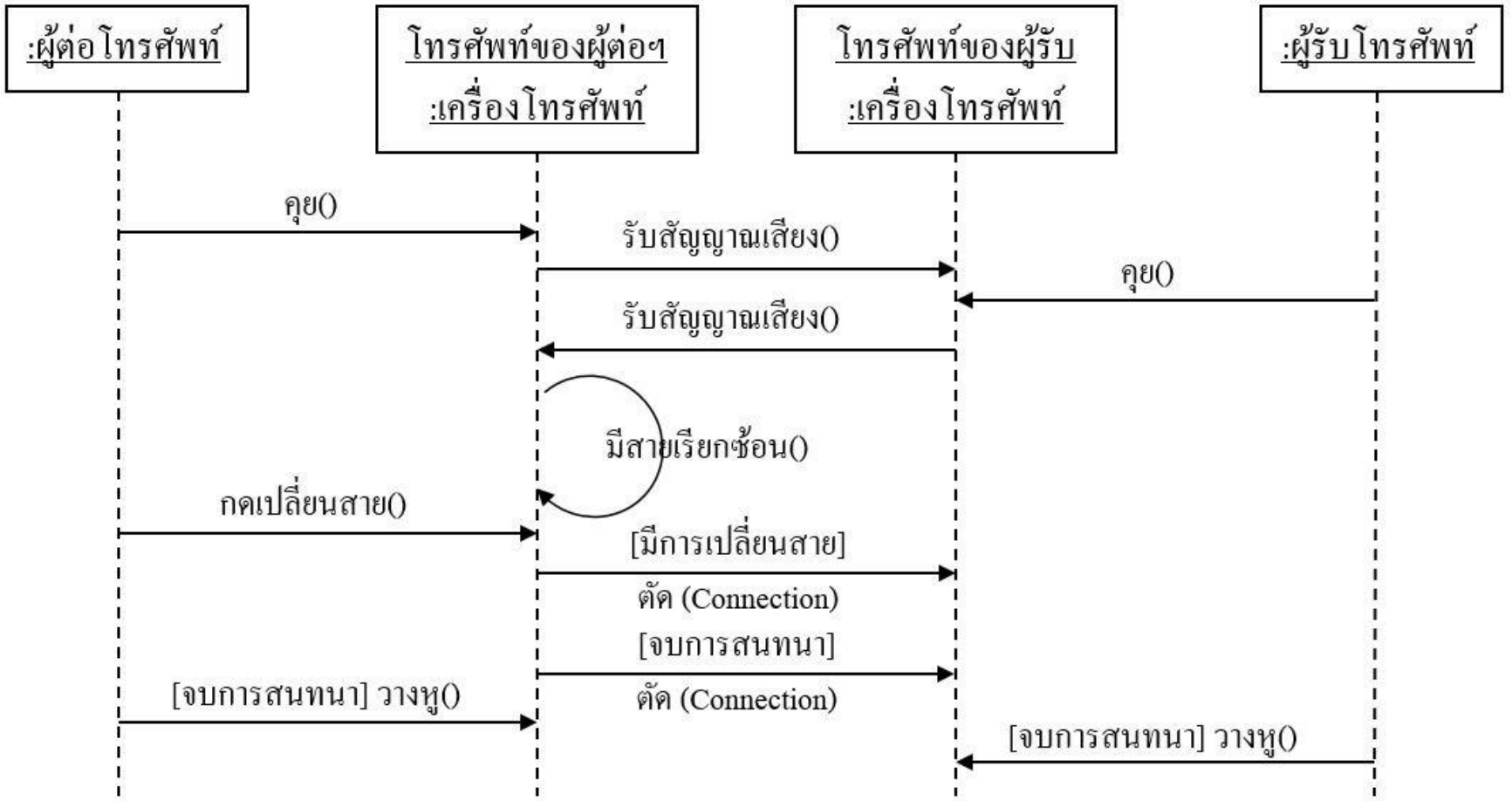


จากรูปเป็นภาพของ Sequence Diagram ที่แสดงภาพของกิจกรรมใน Use Case การถอนเงินจากเครื่อง ATM ซึ่งเมื่อเปรียบเทียบกับรูปที่ผ่านมาก่อนหน้านี้ จะพบว่ามี Class ที่มีส่วนร่วมใน Sequence Diagram นี้เพิ่มขึ้น ซึ่งได้แก่เครื่องจ่ายเงิน (Cash Dispenser) และเครื่องพิมพ์

ถ้าสังเกตให้ดีจะพบว่า จากเส้นกิจกรรมเส้นบนสุดจนกระทั่งถึงเส้นที่ 7 จากด้านบน จะเหมือนกับใน Sequence Diagram ของการขอดูยอดเงินฯ ทุกประการ แต่ความแตกต่างของกิจกรรมจะเกิดขึ้นหลังจากนั้นคือ

หลัง จากที่ Main Menu แสดงให้เห็นผู้ใช้ จะกดจำนวนเงินที่ต้องการถอนที่ปุ่มหมายเลข (ซึ่ง Function กดอยู่ใน Class ปุ่มหมายเลขอยู่แล้ว) จากนั้นบัญชีจะถูกสั่งให้ตรวจสอบว่ามียอดคงเหลือในบัญชีเท่าใด ซึ่งถ้าหากว่าเงินที่มีอยู่ในบัญชีมีจำนวนมากกว่าจำนวนเงินที่ต้องการถอน เครื่องจ่ายเงินจะถูกสั่งให้จ่ายเงิน ต่อจากนั้นเครื่องพิมพ์จะถูกสั่งให้พิมพ์ SLIP แสดงการถอนเงินออกมา

ในทางกลับกัน ถ้าหากเงินในบัญชีมีจำนวนน้อยกว่าจำนวนที่ระบุว่าจะถอน หน้าจอจะถูกสั่งให้แสดงข้อความเตือนความผิดพลาด (Error Message)



จากรูปเป็น Sequence Diagram เพื่อแสดงกิจกรรมของ Use Case การคุยโทรศัพท์ และการมีสายเรียกซ้อน โดย Class และ Objects ที่มีส่วนร่วมใน Sequence Diagram นี้เป็นเช่นเดียวกันกับ Sequence Diagram ของการต่อโทรศัพท์

กิจกรรม เริ่มต้นขึ้นเมื่อ ผู้ต่อสายโทรศัพท์คุยไปยังเครื่องโทรศัพท์ต้นทาง จากนั้นเครื่องโทรศัพท์ต้นทางจึงสั่งให้เครื่องโทรศัพท์ปลายทางรับสัญญาณ เสียงที่ส่งไป โดยสัญญาณเสียงนั้นจะออกไปทางหูโทรศัพท์ ต่อจากนั้นผู้รับโทรศัพท์คุยไปยังโทรศัพท์ปลายทาง หลังจากนั้นโทรศัพท์ปลายทางจะสั่งให้โทรศัพท์ต้นทางรับสายสัญญาณเสียงจากตน ซึ่งเหตุการณ์นี้จะเป็นเช่นนี้ไปจนกว่าจะจบการสนทนา หรือเกิดมีสายเรียกซ้อนขึ้น

กิจกรรม ที่อาจจะเกิดขึ้นใน Use Case มีสายเรียกซ้อน จะเริ่มเกิดขึ้นในเส้นกิจกรรมที่ 4 นับจาก เส้นบนสุด นั่นคือ เมื่อเกิดมีสายเรียกซ้อนขึ้น ถ้าผู้โทรกดเปลี่ยนสาย Connection ที่เกิดขึ้นในตอนแรก สายนั้นจะถูกตัดออกไปเพื่อรับสายใหม่ที่เข้ามา ซึ่งในจุดนี้จะจบกิจกรรมที่อาจจะเกิดขึ้นใน Use Case มีสายเรียกซ้อน

สรุปเหตุผลของการเขียนแผนภาพลำดับ

สรุปเหตุผลของการเขียนแผนภาพลำดับ

เมื่อมีการออกแบบคลาสและวัตถุแล้วแสดงว่าแต่ละวัตถุมีการส่งข่าวสารให้กัน อย่างไรแล้วข่าวสาร ที่ส่งให้กันคือการกระทำต่าง ๆ ที่เกิดขึ้นระหว่างวัตถุหรือเมธอดของวัตถุนั่นเองโดยข่าวสารจะแสดงให้เห็นว่า วัตถุต่าง ๆ มีการติดต่อสื่อสารกันอย่างไร ในช่วงระยะเวลาหนึ่ง

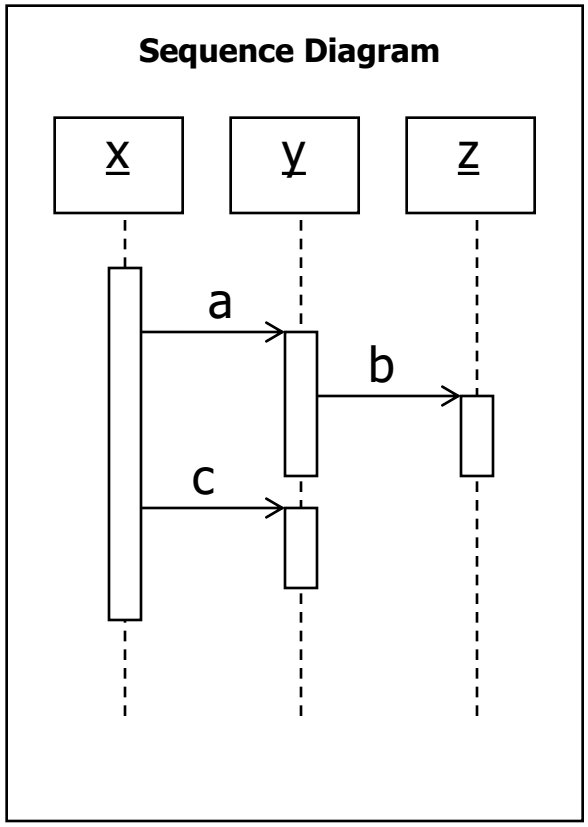
6.11 Communication Diagram

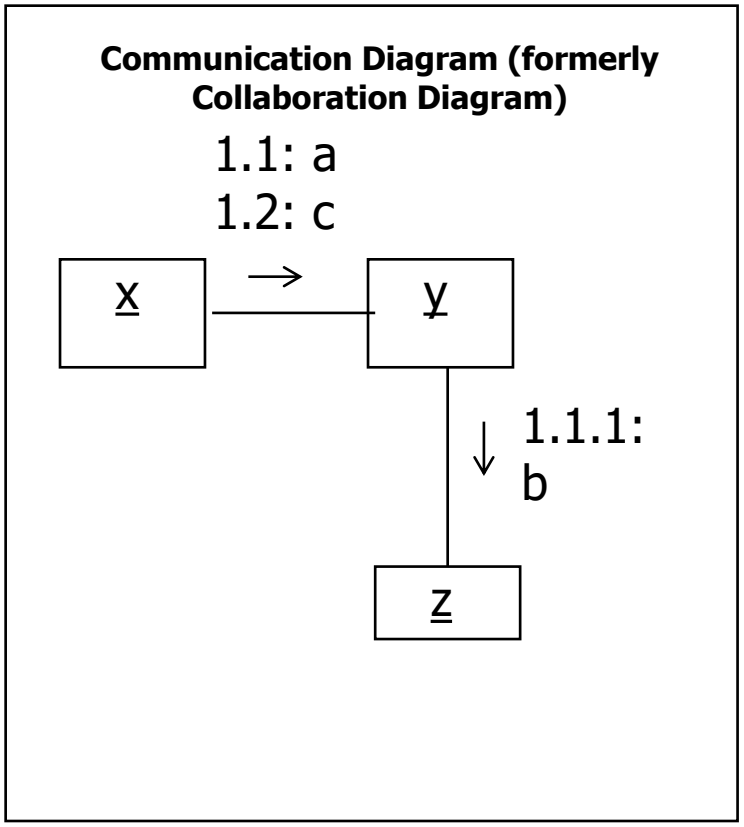
ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

แผนภาพประสาน (Communication Diagram

ความหมายของแผนภาพประสานหรือแผนภาพคอลลาบอเรชัน (Communication Diagram (formerly Collaboration Diagram)) เป็นแผนภาพที่มีลักษณะเช่นเดียวกันกับแผนภาพลำดับโดยที่แผนภาพลำดับจะเป็นแผนภาพที่แสดงถึงการแลกเปลี่ยนข่าวสารตามลำดับเวลาหรือตามเหตุการณ์ก่อนหลัง แต่แผนภาพคอลลาบอเรชันจะนำเสนอแผนภาพการทำงานร่วมกันระหว่างออบเจ็กต์เป็นสำคัญนอกจากนี้ก็ยังแสดงลำดับการทำงานก่อนและหลังด้วยซึ่งจะเห็นได้ว่าคอลลาบอเรชันไดอะแกรมจะแสดงให้เห็นภาพโครงสร้างระบบมากกว่าการเน้นเพียงข่าวสารที่สื่อสารกันหากต้องการแผนภาพที่มุ่งเน้นด้านเวลาและแสดงลำดับก่อนหลังเป็นสำคัญให้เลือกใช้ แผนภาพลำดับแต่หากต้องการแผนภาพที่ให้ความสัมพันธ์ภายในออบเจ็กต์ ก็ให้เลือกใช้คอลลาบอเรชันไดอะแกรม ซึ่ง คอลลาบอเรชันเป็นแผนภาพชนิดเดียวกับ แผนภาพลำดับโดยแผนภาพลำดับจะเป็นแผนภาพที่แสดงถึงการสื่อสาร แต่แผนภาพคอลลาบอเรชัน จะนำเสนอการทำงานร่วมกันระหว่างวัตถุเป็นหลักแต่ก็สามารถแสดงถึงลำดับก่อนหลังได้ด้วย (Object Management Group, 1997)

Interaction & Communication Diagram





6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

Communication Diagram (formerly Collaboration Diagram) คืออะไร

Communication Diagram (formerly Collaboration Diagram) หรือบางครั้งเรียกว่า Communication Diagram เป็นหนึ่งในแผนภาพของ Unified Modeling Language (UML) ที่แสดงถึงการทำงานร่วมกันของวัตถุในระบบ โดยมุ่งเน้นที่การอธิบายโครงสร้างของการสื่อสารระหว่างวัตถุต่าง ๆ และการส่งข้อความระหว่างวัตถุเหล่านั้น เพื่อให้ระบบทำงานได้ตามที่ออกแบบไว้

Communication Diagram (formerly Collaboration Diagram) ช่วยให้นักพัฒนาและนักวิเคราะห์เข้าใจภาพรวมของการทำงานและการเชื่อมโยงระหว่างวัตถุในระบบได้ชัดเจน รวมถึงสามารถเห็นลำดับและการไหลของข้อมูลได้อย่างเป็นระบบ

6.11 **Communication Diagram**

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

ลักษณะสำคัญของ Communication Diagram

- **การสื่อสารระหว่างวัตถุ: แผนภาพจะแสดงการส่งข้อความ (message) ระหว่างวัตถุต่าง ๆ ที่โต้ตอบกัน**
- **การระบุตัววัตถุ: วัตถุ (object) ในระบบจะถูกแสดงด้วยกล่องหรือไอคอนที่แสดงชื่อของวัตถุ ซึ่งแสดงถึงผู้เล่นหลักในการทำงานร่วมกัน**
- **หมายเลขกำกับการทำงาน: แต่ละข้อความที่ส่งจะมีหมายเลขกำกับเพื่อระบุลำดับการทำงาน ทำให้สามารถเข้าใจลำดับการสื่อสารได้อย่างชัดเจน**
- **การเชื่อมโยงวัตถุ: วัตถุที่เกี่ยวข้องจะถูกเชื่อมโยงกันด้วยเส้น เพื่อแสดงให้เห็นว่าพวกมันทำงานร่วมกันอย่างไร และใครส่งข้อมูลให้ใคร**

6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

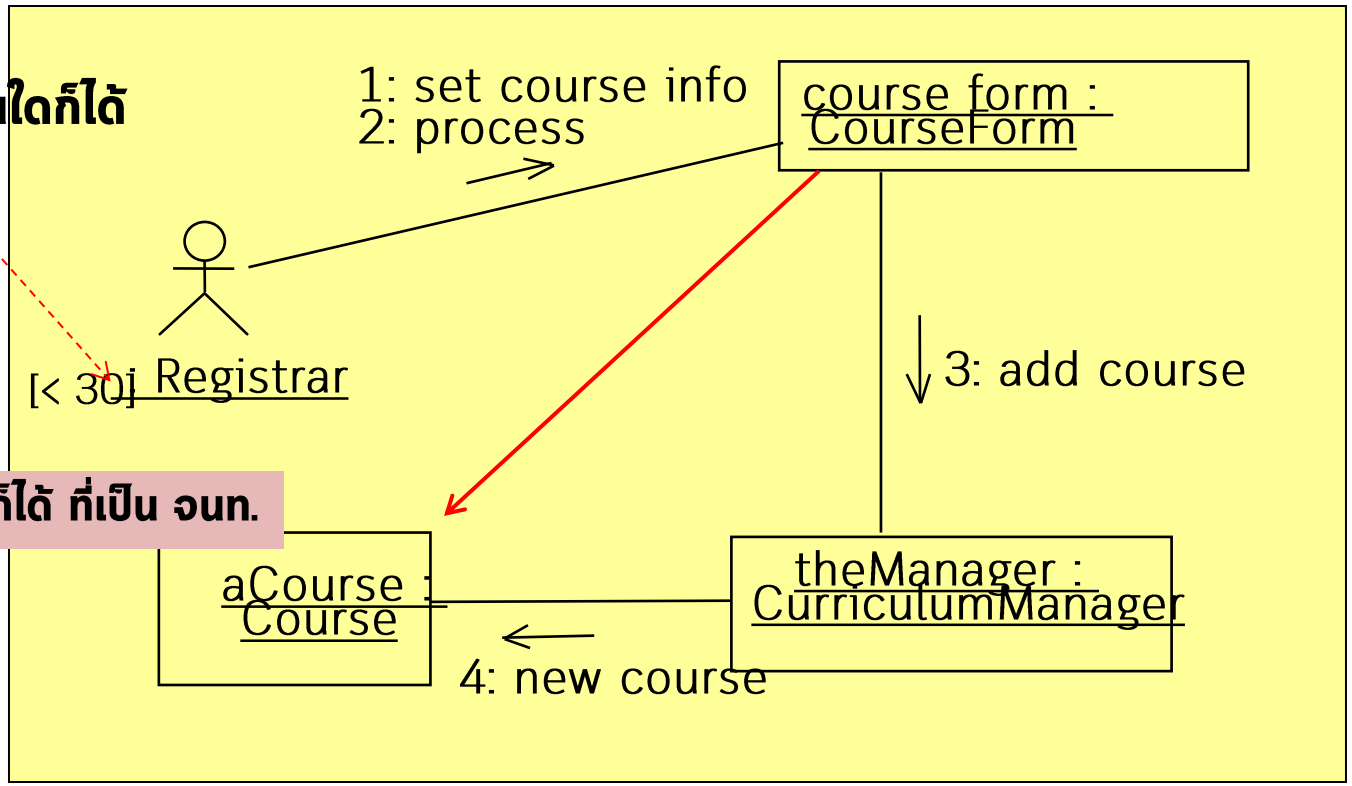
การใช้งาน Communication Diagram

- ใช้เพื่อแสดงถึงการสื่อสารระหว่างวัตถุในมุมมองของโครงสร้างและการเชื่อมโยง ซึ่งเน้นที่ความสัมพันธ์และการทำงานร่วมกัน
- ใช้ในการออกแบบซอฟต์แวร์ โดยเฉพาะการพัฒนาระบบที่ต้องการเห็นโครงสร้างและการส่งผ่านข้อมูลระหว่างวัตถุที่เกี่ยวข้อง
- ใช้ในการวิเคราะห์และตรวจสอบความถูกต้องของการออกแบบ ก่อนจะเข้าสู่ขั้นตอนการเขียนโปรแกรม

ตัวอย่าง Communication Diagram ในระบบจองตั๋วหนัง

- วัตถุที่เกี่ยวข้อง: User, Movie Database, Ticket System
- การสื่อสาร: User ส่งคำขอจองไปที่ Ticket System, Ticket System ติดต่อ Movie Database เพื่อยืนยันข้อมูลภาพยนตร์, และส่งการยืนยันกลับไปที่ User

A Communication Diagram



เปิดวิชาเรียน ให้นึกถึงการเปิดรายวิชา วิชาเลือกเสรี

จากรูปจะทำให้เห็นถึง วัตถุต่าง ๆ และการเชื่อมต่อกันของวัตถุเหล่านั้น
จะเห็นว่ามีวัตถุอะไรบ้าง ในระบบ / หรือใน Use case หนึ่ง อาจจะไม่ใช่ทั้งระบบใหญ่

ตัวอย่าง : การลงทะเบียน เรียนมรภ. พระนคร

1. นักศึกษาเข้าไปที่เว็บ ของพระนคร
2. เข้าสู่ระบบ
3. เลือกระบบ นักศึกษา
4. เลือกลงทะเบียน

ออนไลน์

ออฟไลน์

นักเรียน

1. ไปที่ สนง. ทะเบียน
2. เลือกแบบฟอร์มลงทะเบียน
3. กรอกข้อมูล
4. ยื่นแบบฟอร์ม
5. จนท. ตรวจสอบแล้วไม่มีปัญหา (ตรวจสอบวิชาที่ต้องลงทะเบียนก่อน)
6. ลงทะเบียนเรียบร้อย
7. ถ้ามีปัญหา ก็ยกเลิก

รายวิชาที่เปิด

อาจารย์ที่ปรึกษา

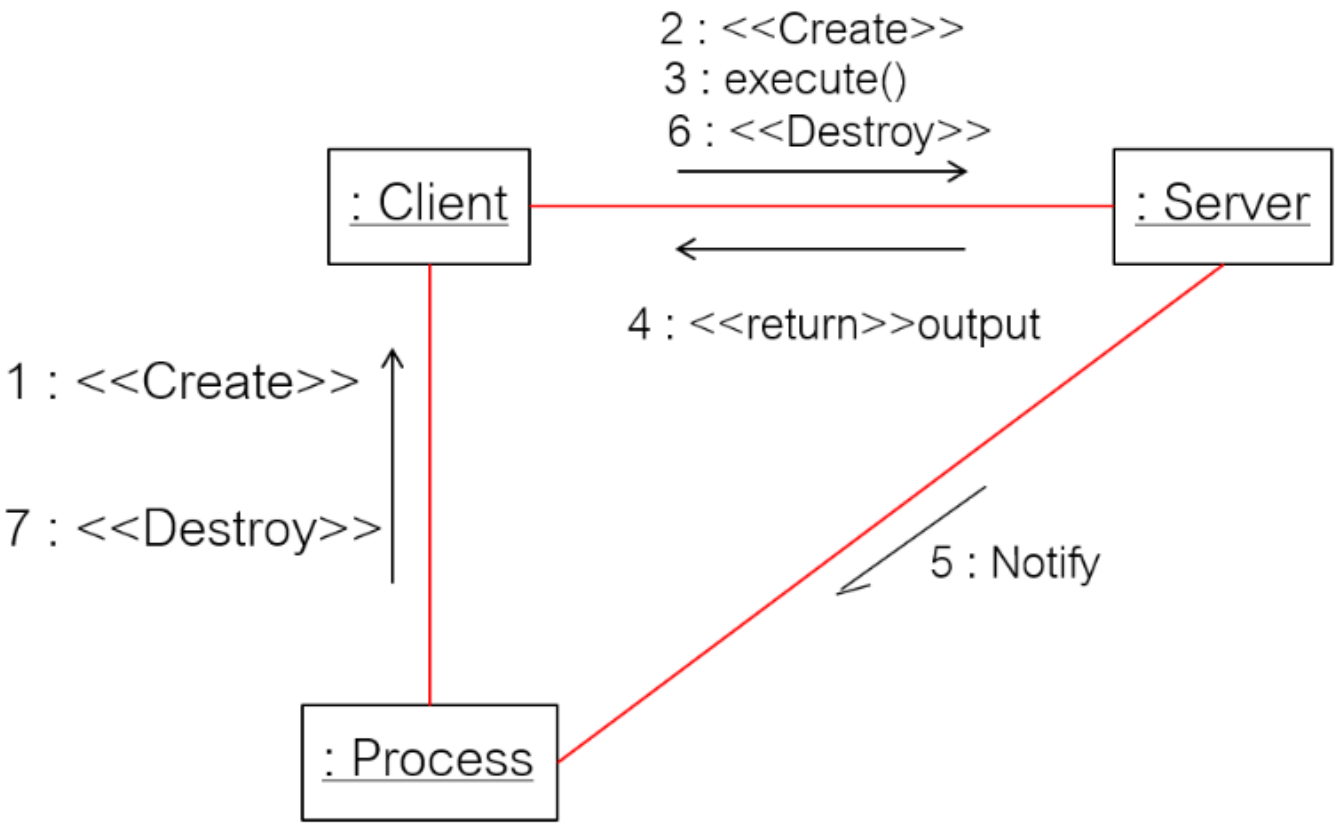
1. ยืนจัดแผนการเรียน 4 ปี
2. ยันแผนแต่ละเทอม
3. ส่งให้ สนง. ทะเบียน

รายวิชาที่เปิด *เลือกเสรี*

6.11 **Communication Diagram**

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

แผนภาพประสาน (Communication Diagram)



6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

แผนภาพประสาน (Communication Diagram)

- **Collaboration**
  - กำหนดบทบาท (role) ของกลุ่มของวัตถุที่กระทำต่องานใดงานหนึ่ง เช่นเดียวกับ operation หรือ use case
- **Interaction**
  - ปฏิสัมพันธ์ที่ระบุรูปแบบการสื่อสาร (communication pattern) ที่กระทำโดยวัตถุที่ กำลังแสดงบทบาทของ collaboration

6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

ส่วนประกอบของแผนภาพประสาน

- วัตถุ (Objects)
  - แลกเปลี่ยน messages ให้แก่กันและกัน
- Messages
  - Synchronous : “call events,” แทนด้วย full arrow
  - Asynchronous: “signals,” แทนด้วย half arrow
  - «create» และ «destroy» messages
- มีการระบุหมายเลข Message ตามลำดับที่เกิดก่อน-หลัง และการอาจมี Loop ของ Message

6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

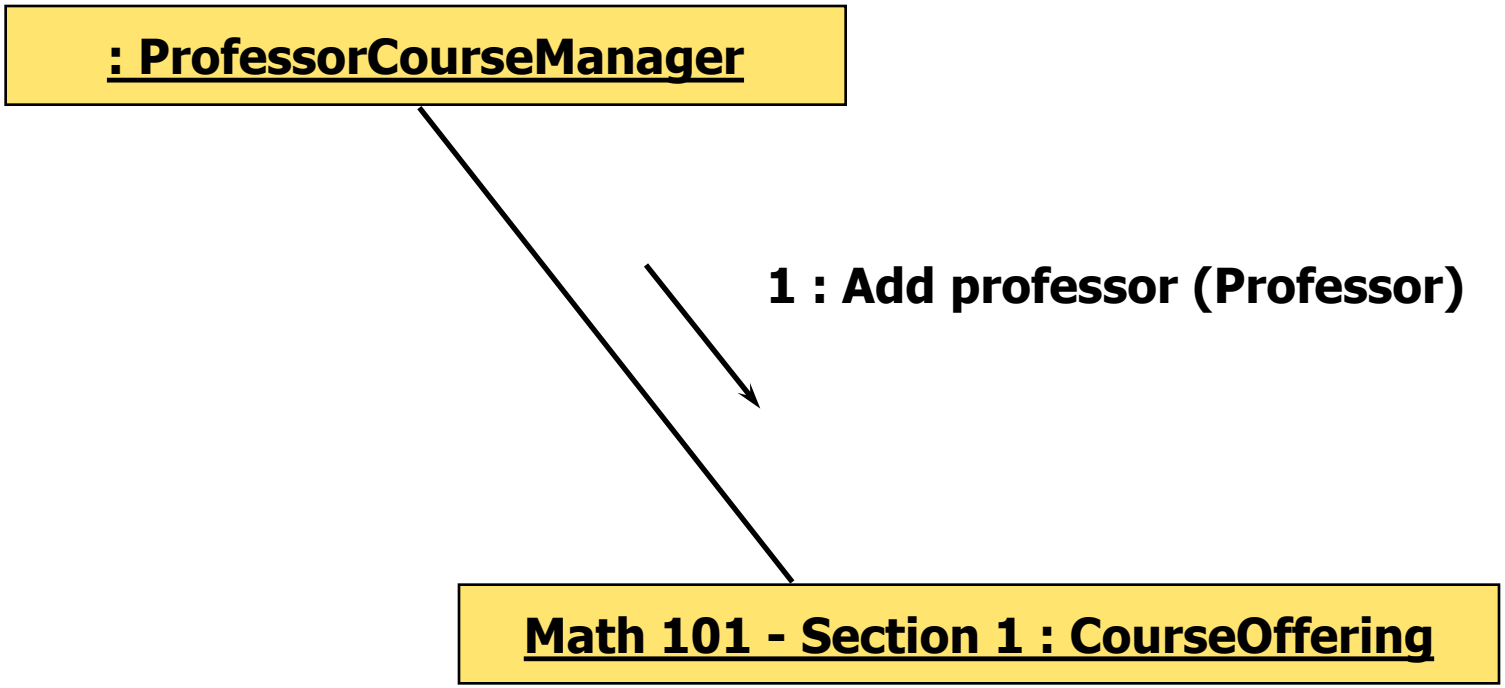
ส่วนประกอบของแผนภาพประสาน

- หมายเลขกำกับ แสดงลำดับของ messages ระบุโดย
  - 1, 2, 3, 4, ...
  - 1, 1.1, 1.2, 1.3, 2, 2.1, 2.1.1, 2.2, 3 (แสดง operation calls ที่เป็นส่วนย่อยของ operation อื่นๆ)

6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

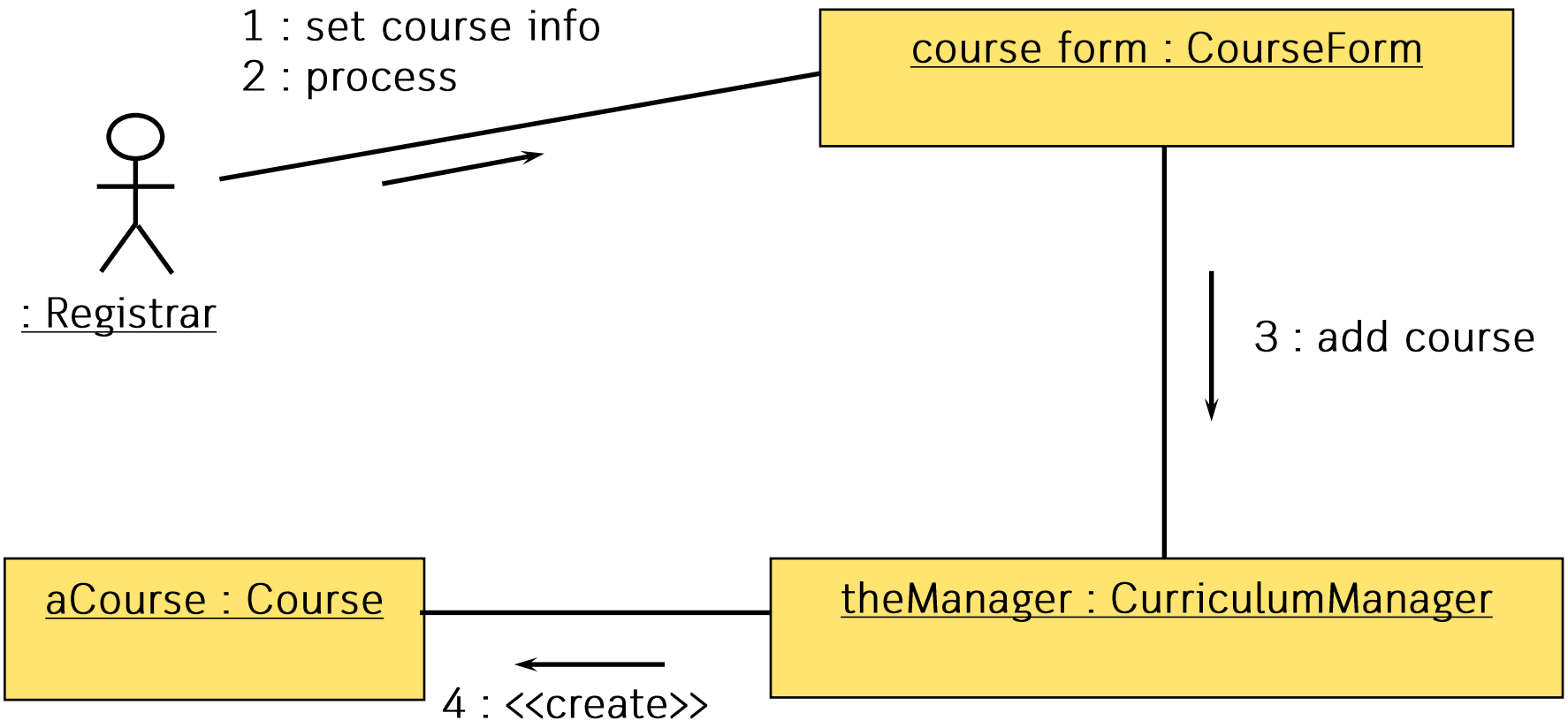
ตัวอย่างแผนภาพประสาน (Communication Diagram)



6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

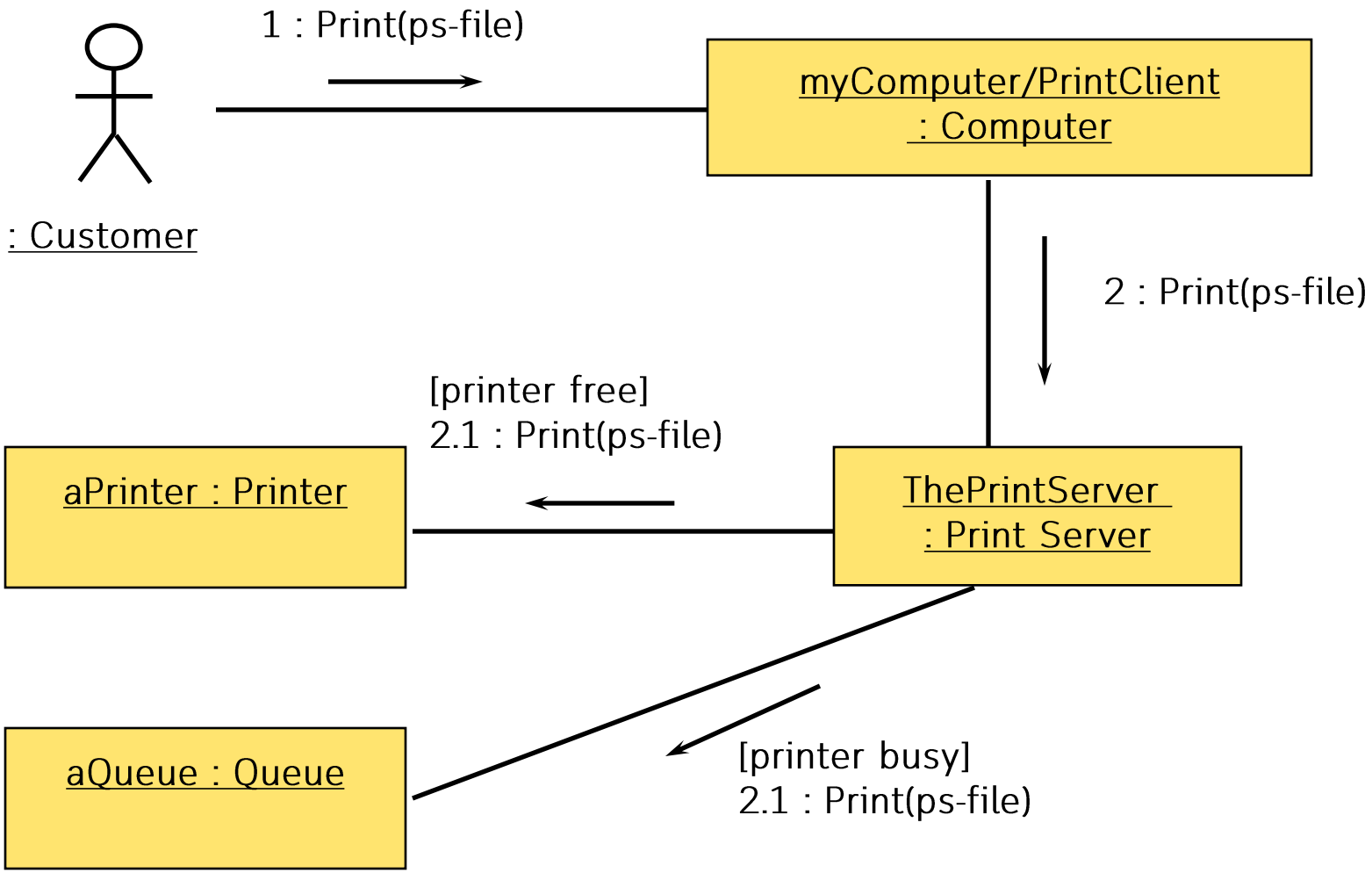
ตัวอย่างแผนภาพประสาน (Communication Diagram)



6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

ตัวอย่างแผนภาพประสาน (Communication Diagram)

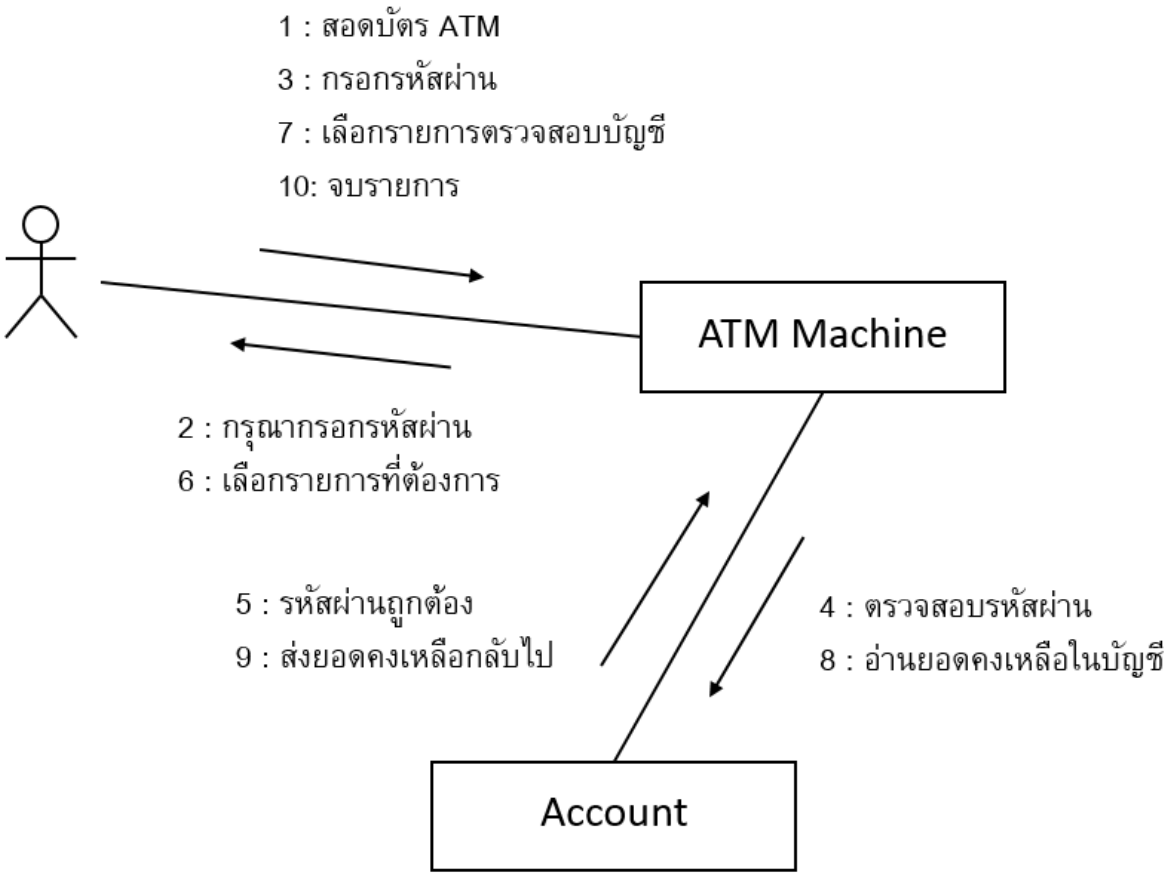


6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

ตัวอย่างแผนภาพประสาน (Communication Diagram)

ตัวอย่างแผนภาพคอลลาบอเรชัน



6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

หลักการเขียน Sequence Diagram และ Communication Diagram

- Sequence Diagram และ Communication Diagram หรือชื่อเดิมคือ Collaboration Diagram เป็นแผนภาพในกลุ่ม Interaction Diagram
- ใน diagram หนึ่ง ๆ เราเขียนเพียงส่วนใดส่วนหนึ่งของระบบ หรือเขียนเฉพาะ Use Case ที่สำคัญ หรือ Scenario ที่ต้องการอธิบาย

ในการเขียน Sequence Diagram หรือ Communication Diagram โดยทั่วไป 1 แผนภาพไม่จำเป็นต้องแสดงการทำงานทั้งหมดของระบบ แต่ควรเลือกแสดงเฉพาะ Use Case หรือ Scenario ที่สำคัญ เนื่องจากระบบหนึ่งอาจมีหลายกระบวนการและมีความซับซ้อนสูง หากนำทุกขั้นตอนมาไว้ในแผนภาพเดียว จะทำให้แผนภาพมีขนาดใหญ่ อ่านยาก และยากต่อการทำความเข้าใจของทีมพัฒนาระบบ

6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

Comparing sequence & Communication Diagrams

- Communication Diagram หรือชื่อเดิมคือ Collaboration Diagram ใช้แสดงการสื่อสารระหว่าง Object โดยเน้นโครงสร้างความสัมพันธ์หรือการเชื่อมโยงของ Object ที่ทำงานร่วมกัน พร้อมระบุลำดับของ Message ด้วยหมายเลขกำกับ
- Sequence Diagram เหมาะสำหรับแสดงลำดับการส่ง Message ตามเวลาโดยอ่านจากบนลงล่าง
- ใน Communication Diagram ลำดับเวลาก่อน-หลังอาจมองเห็นไม่ชัดเท่า Sequence Diagram เพราะต้องพิจารณาจากหมายเลข Message ที่กำกับไว้
- หากปฏิสัมพันธ์ระหว่าง Object มีความซับซ้อนมาก แผนภาพอาจอ่านยาก ไม่ว่าจะใช้ Sequence Diagram หรือ Communication Diagram ดังนั้นควรเลือกเขียนเฉพาะ Use Case หรือ Scenario ที่สำคัญ

6.11 Communication Diagram

ความหมาย สัญลักษณ์ Message Numbering และตัวอย่าง

Interaction Modeling Tips

- ใช้เฉพาะส่วนของวัตถุ Include only those features of the instances that are relevant.
- แสดง flow จากซ้ายไปขวา และจากบนลงล่าง
- ใช้ sequence diagrams
  - เพื่อแสดงลำดับระหว่างสิ่งที่มากระตุ้นให้เกิดปฏิสัมพันธ์ ระหว่างวัตถุ
  - มักใช้ใน real-time modeling
- ใช้ Communication Diagram (formerly called Collaboration Diagram)
  - เมื่อโครงสร้างของระบบ มีความสำคัญ

Example: A Booking System

Use Case Description: Change Flight Itinerary

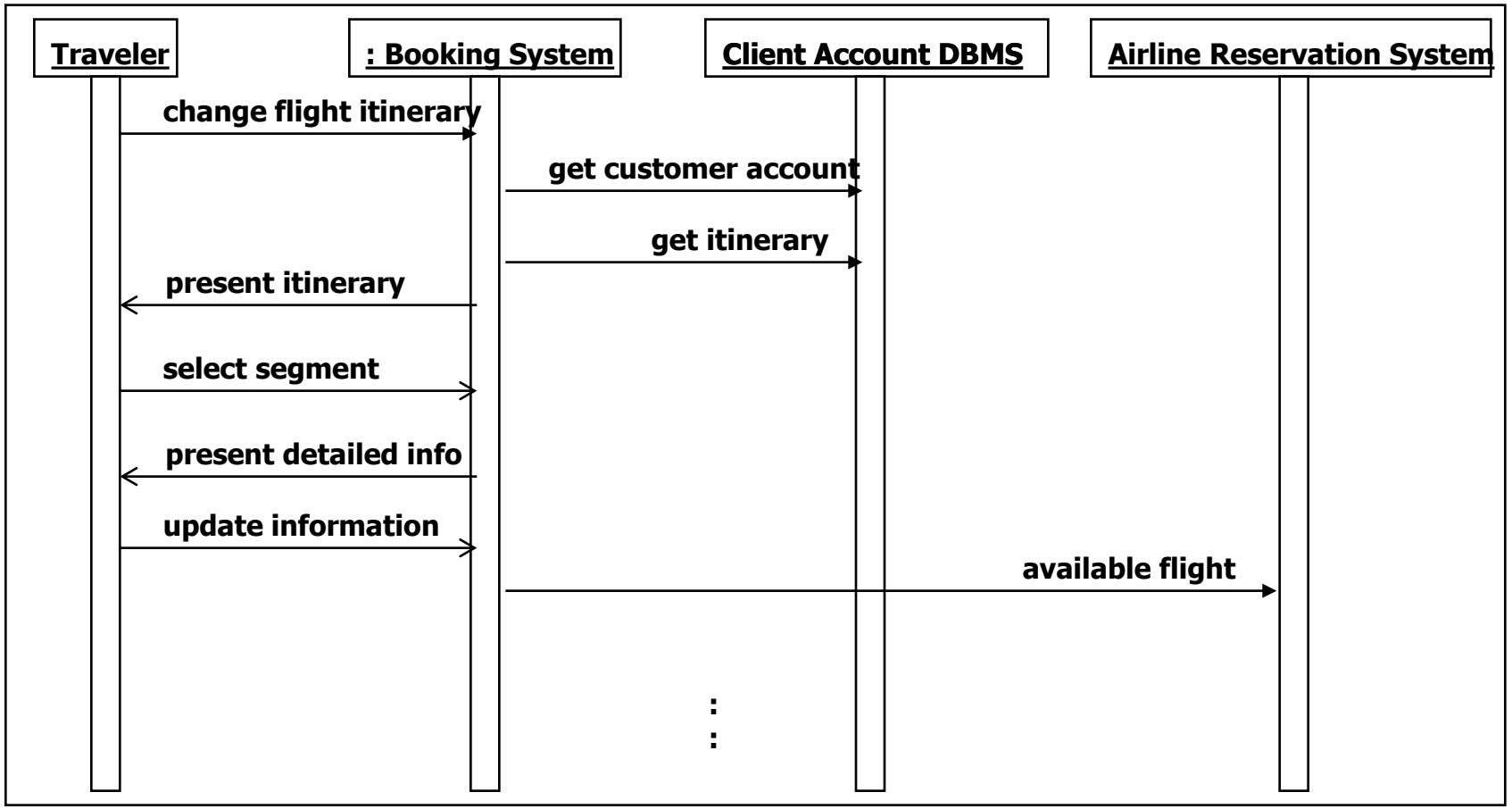
- Actors
  - traveler, client account db, airline reservation system
- Basic course:
  - Traveler เลือก ‘change flight itinerary’ option
  - System ดึงค่า account และ flight itinerary ของ traveler จาก client account database
  - System ถาม traveler ให้ traveler เลือกส่วนของ itinerary segment ที่ต้องการเปลี่ยนแปลง
  - ...

Use Case Description: Change Flt Itinerary

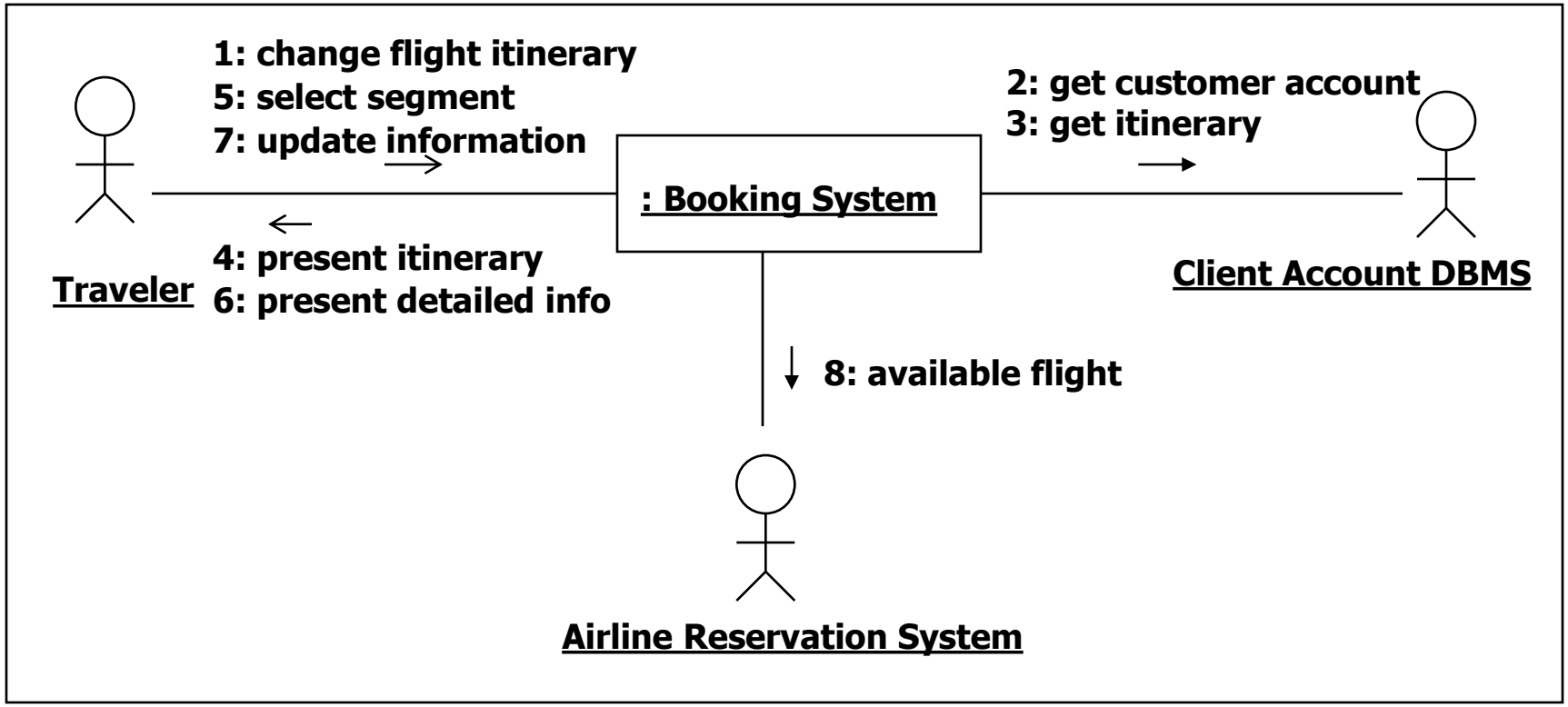
- ...
- System ถาม traveler ข้อมูลเกี่ยวกับเวลา departure และ destination; traveler เป็นคนให้ข้อมูล
- ถ้า flights ที่ traveler ต้องการเปลี่ยน ยังมีที่ว่าง ให้...
- ...
- System แสดงสรุปรายการ transaction

• Alternative course:
- ถ้า flights ที่ traveler ต้องการเปลี่ยน ไม่มีที่ว่าง ให้ ...

Sequence Diagram: Change Flight Itinerary



Communication Diagram (formerly Collaboration Diagram): Change Flt Itinerary



Summary

- UML Behavioral Diagrams
  - Interaction Diagrams
    - Sequence Diagram
    - Communication Diagram (formerly Collaboration Diagram)

แบบฝึกหัดท้ายบทที่ 9

1. จงสร้าง Sequence Diagram จาก Problem Domain ที่กำหนดให้ต่อไปนี้

Problem Domain

ในบริษัทจัดหางานแห่งหนึ่ง จะให้บริการเป็นสื่อกลางในการติดต่อกันระหว่างนายจ้างที่ต้องการลูกจ้างและผู้สมัครงานที่ต้องการหางานทำ โดยจะรับจัดเก็บข้อมูลเกี่ยวกับคุณสมบัติของผู้สมัครและคุณสมบัติของงานแต่ละตำแหน่งที่นายจ้างต้องการ และเมื่อพบว่ามีตำแหน่งงานที่เหมาะแก่ผู้สมัคร บริษัทจะจัดทำจดหมายเพื่อติดต่อให้นายจ้างและผู้สมัครให้มาสัมภาษณ์กันที่บริษัท (เป็นระบบสารสนเทศ)

อ้างอิง

- Booch, G., Rumbaugh, J., Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley.
- Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley.

- http://www.stjohn.ac.th/engineer/information%20technology/file/Download/UML/UML_10.ppt

- http://www2.lpc.rmutl.ac.th/wachira/oop/Sequence%20Diagram.pptx