



01076105, 01075106

Object Oriented Programming

Object Oriented Programming Project

UML #2



Case Study : ห้องสมุด

- ระบบห้องสมุด ประกอบด้วยคลาสดังนี้
 - Library: เป็นตัวแทนของห้องสมุด เก็บข้อมูล คือ ชื่อ และ ที่อยู่
 - Book: หมายถึงหนังสือแต่ละรายการ ประกอบด้วย ISBN, Title, Subject, Publishers, etc.
 - BookItem: หนังสือแต่ละรายการอาจมีหลายเล่ม ดังนั้นจึงต้องกำหนดหมายเลขเพื่อให้รู้ว่าเล่มไหน
 - Account: มี 2 ประเภท คือ สมาชิก และบรรณารักษ์
 - LibraryCard: บัตรห้องสมุด ใช้ในการระบุถึงสมาชิกแต่ละคน และใช้ในการยืมคืนหนังสือ
 - BookReservation: รับผิดชอบในการจองหนังสือแต่ละเล่ม



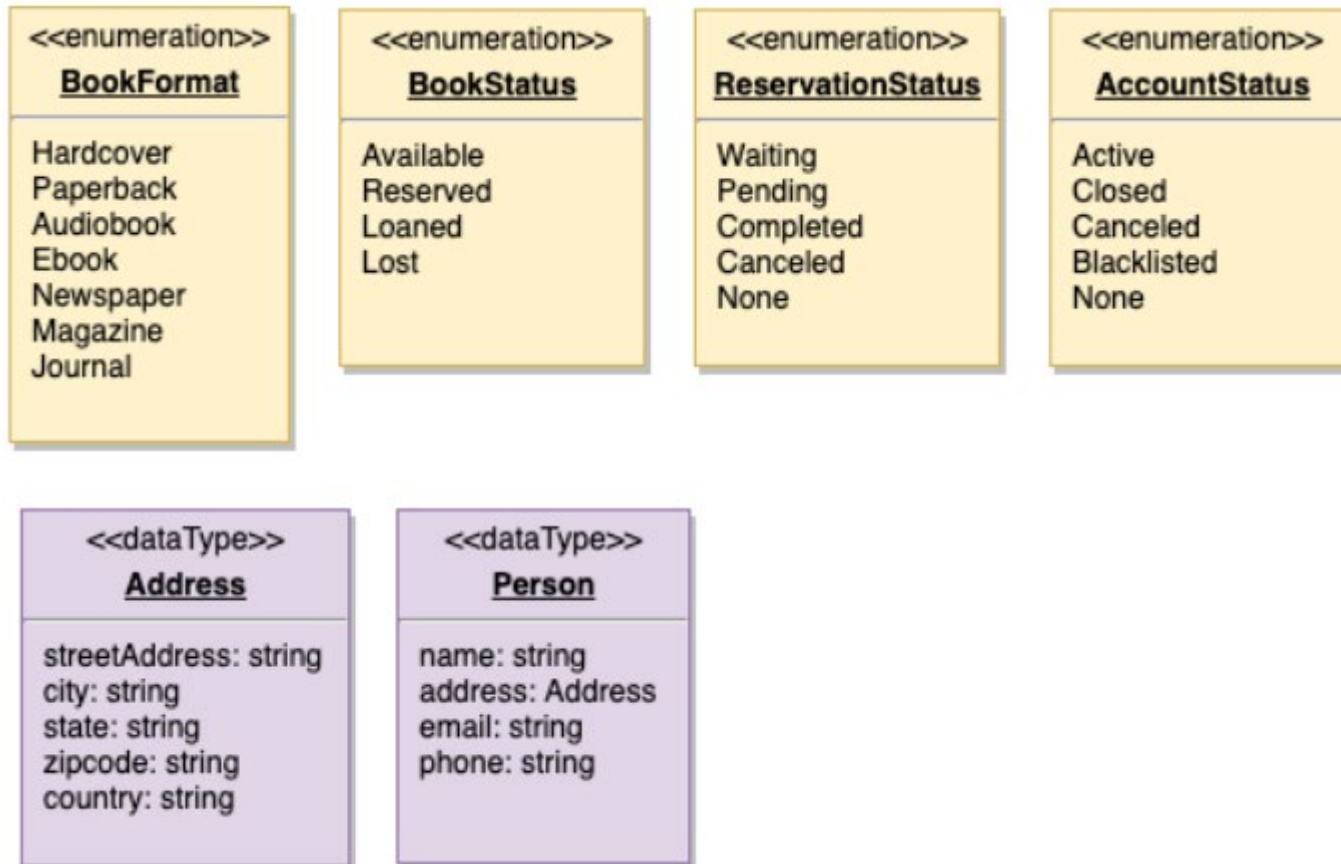
Case Study : ห้องสมุด

- ระบบห้องสมุด ประกอบด้วยคลาสดังนี้
 - BookLending: การให้ยืมหนังสือ
 - Catalog: เป็นรายการของหนังสือที่อยู่ในห้องสมุด ซึ่งสามารถค้นหาได้ Title, Author, Subject
 - Fine: รับผิดชอบในการคำนวณและจัดเก็บค่าปรับ
 - Author: ชื่อผู้เขียน
 - Rack: หนังสือจะจัดเก็บบนชั้นหนังสือ แต่ละ rack จะระบุโดย rack No
 - Notification: รับผิดชอบเรื่องการแจ้งเตือนต่างๆ ในระบบ



Case Study : ห้องสมุด

- Class ประเภท enumerate และ Data Type ประกอบด้วย





Case Study : ห้องสมุด

- Class ประเภท Enumerate และ Data Type มีหน้าที่ในการรวบรวมค่าคงที่ให้อยู่เป็นที่เป็นทาง ทำให้ code เป็นระเบียบ และ ใช้งานง่าย อ่านเข้าใจง่าย
- โปรแกรมในส่วน Enumerate ใช้ในการแทน status ต่างๆ

```
class BookFormat(Enum):  
    HARDCOVER, PAPERBACK, AUDIO_BOOK, EBOOK, NEWSPAPER, MAGAZINE, JOURNAL = 1, 2, 3, 4, 5, 6, 7  
  
class BookStatus(Enum):  
    AVAILABLE, RESERVED, LOANED, LOST = 1, 2, 3, 4  
  
class ReservationStatus(Enum):  
    WAITING, PENDING, CANCELED, NONE = 1, 2, 3, 4  
  
class AccountStatus(Enum):  
    ACTIVE, CLOSED, CANCELED, BLACKLISTED, NONE = 1, 2, 3, 4, 5
```



Case Study : ห้องสมุด

- โปรแกรมในส่วน Data Type เพื่อทำหน้าที่เก็บข้อมูลกลุ่มเดียวกันให้เป็นก้อนเดียว

```
class Address:
    def __init__(self, street, city, state, zip_code, country):
        self.__street_address = street
        self.__city = city
        self.__state = state
        self.__zip_code = zip_code
        self.__country = country

class Person(ABC):
    def __init__(self, name, address, email, phone):
        self.__name = name
        self.__address = address
        self.__email = email
        self.__phone = phone
```



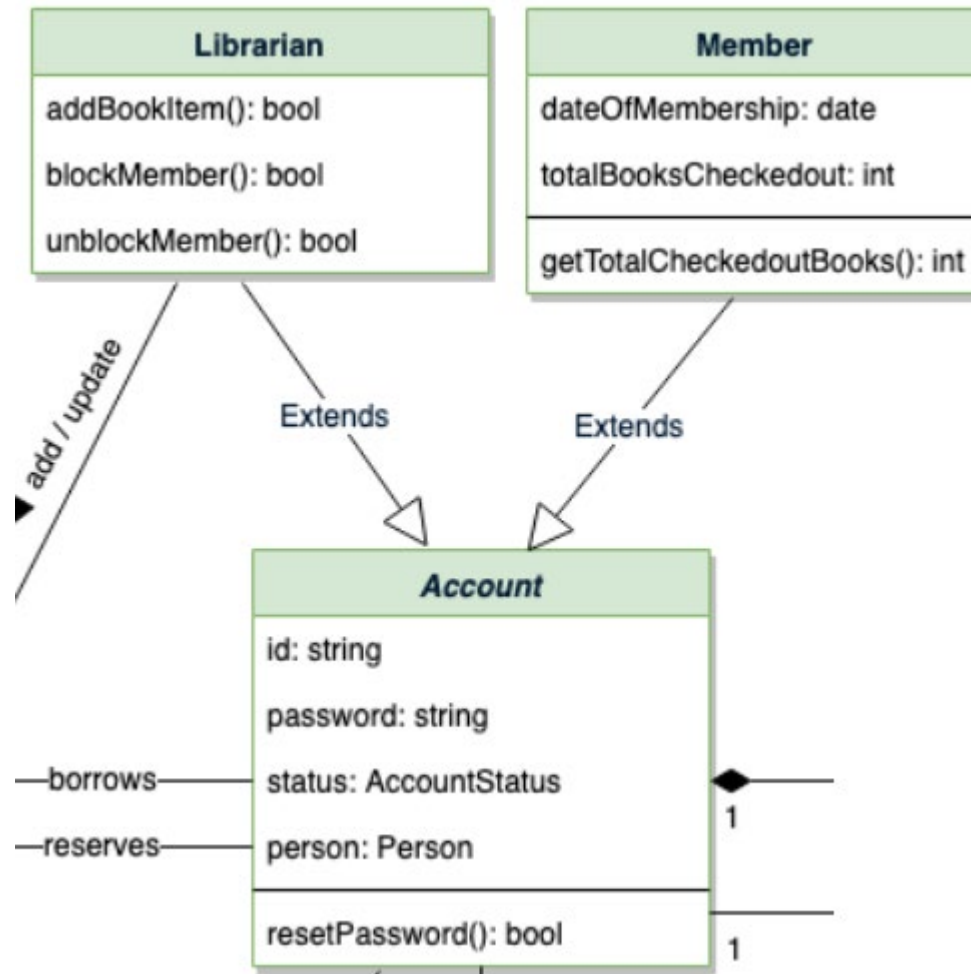
Case Study : ห้องสมุด

- การใช้งาน Class ประเภท Enumerate จะใช้ลักษณะของค่าคงที่ เช่น
 - BookStatus. AVAILABLE จะมีค่า = 1
 - BookFormat. AUDIO_BOOK จะมีค่า = 3
- จะเห็นได้ว่าทำให้ code อ่านเข้าใจง่ายขึ้น
- การใช้งาน Class ลักษณะที่เป็น Data Type มักจะมีความสัมพันธ์ลักษณะที่เป็นองค์ประกอบ เช่น ที่อยู่ของบุคคล ดังนั้นจะมีความสัมพันธ์เป็น Association หรือ Aggregation ก็ได้
- การใช้งาน Class ลักษณะที่เป็น Data Type นอกจากจะทำให้ Code อ่านง่ายขึ้นแล้วยังช่วยลด Attribute ของคลาสหลักด้วย เช่น Class Library



Case Study : ห้องสมุด

- Class กลุ่ม Actor





Case Study : ห้องสมุด

- คลาส Account เป็น Abstract Base Class คือเป็นต้นแบบให้ Inherit ไปยัง Member และ Librarian
- จะเห็นว่า attribute เป็น Private ทั้งหมด, มี method reset_password

```
class Account(ABC):  
    def __init__(self, id, password, person, status=AccountStatus.ACTIVE):  
        self.__id = id  
        self.__password = password  
        self.__status = status  
        self.__person = person  
  
    def reset_password(self):  
        pass
```



Case Study : ห้องสมุด

- คลาสบรรณารักษ์ จะสืบทอดมาจากคลาส Account

```
class Librarian(Account):  
    def __init__(self, id, password, person, status=AccountStatus.ACTIVE):  
        super().__init__(id, password, person, status)  
  
    def add_book_item(self, book_item):  
        pass  
  
    def block_member(self, member):  
        pass  
  
    def un_block_member(self, member):  
        pass
```

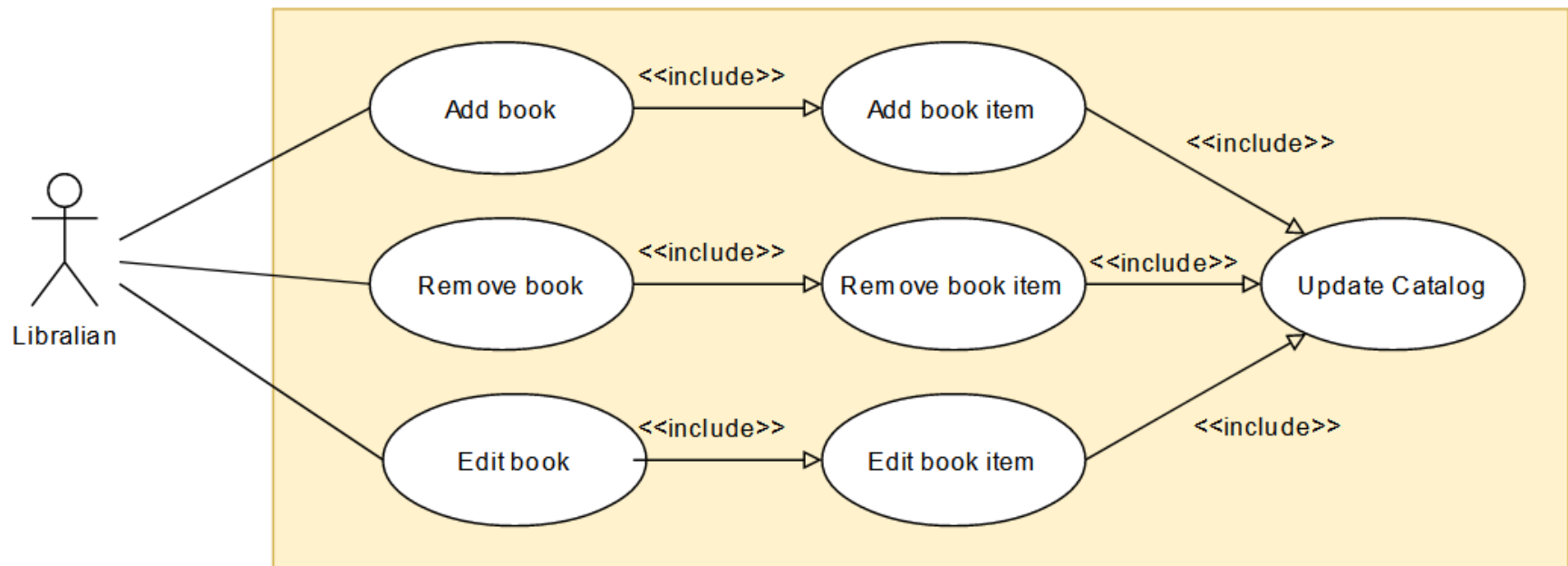
Librarian
addBookItem(): bool
blockMember(): bool
unblockMember(): bool



Case Study : ห้องสมุด

- ข้อสังเกตของคลาส Librarian คือ ยังขาดฟังก์ชันตาม Use Cased Diagram ไปหลายฟังก์ชัน หากเขียนโปรแกรมจริงต้องทำให้ครบ

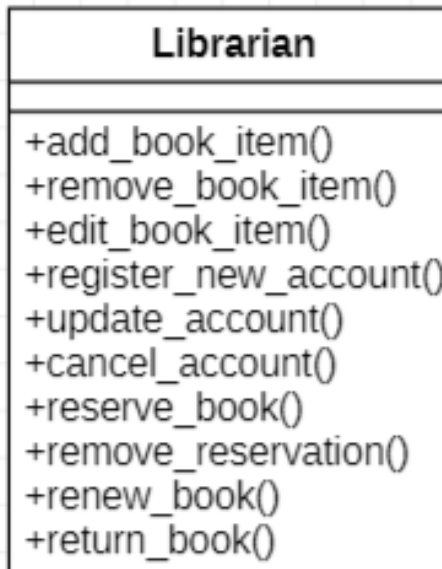
Librarian
addBookItem(): bool
blockMember(): bool
unblockMember(): bool





Case Study : ห้องสมุด

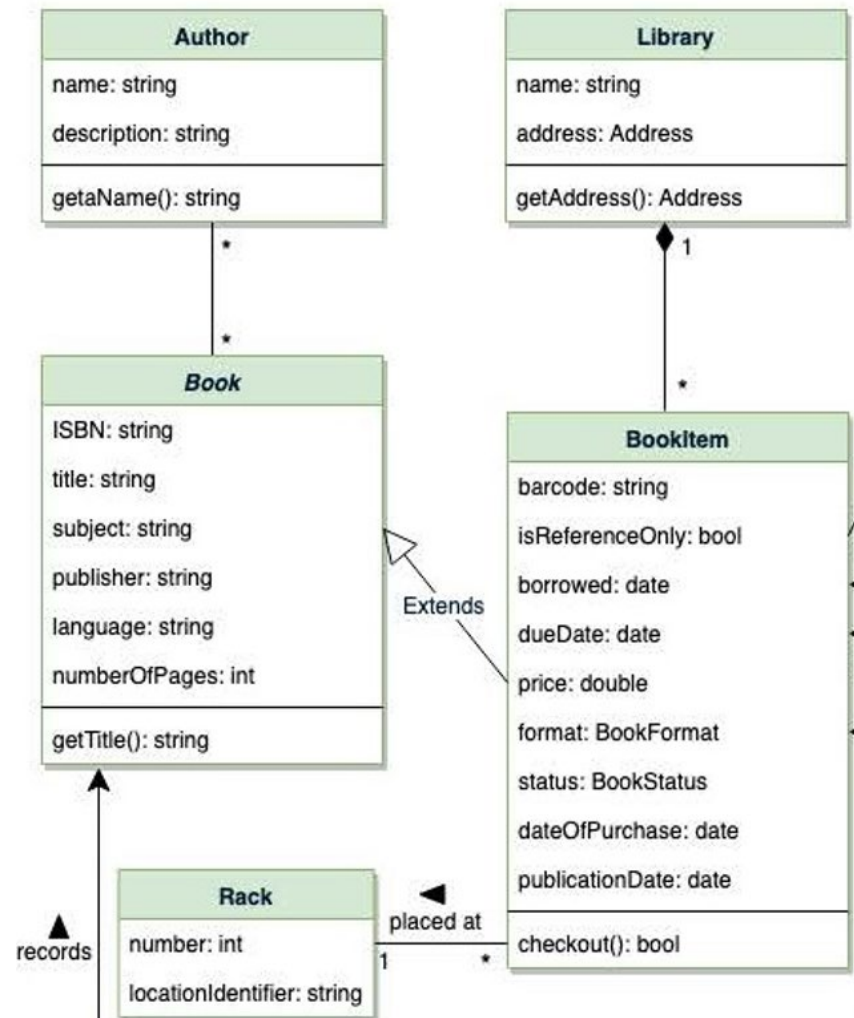
- Exercise :
 - ให้เขียน Class Librarian ที่ครบถ้วนให้ครบตาม Use Case Diagram
 - ให้เขียน Class Member ที่ครบถ้วนให้ครบตาม Use Case Diagram ว่ามี method อะไรบ้าง



Case Study : ห้องสมุด



- ตัวอย่างแนวทางการสร้าง Class จากตัวอย่างเป็นความสัมพันธ์แบบ Composition คือ BookItem เป็นส่วนหนึ่งของห้องสมุด (ซึ่ง BookItem extend มาจาก Book อีกที)





Case Study : ห้องสมุด

- เราจะเริ่มจากสร้าง Class Book ขึ้นมาก่อน เพื่อให้ BookItem ทำการ Inherit จึงใช้เป็น Abstract Base Class
- คลาสนี้มีความสัมพันธ์กับ Author เป็นแบบ Association ในแบบ 1..m คือ หนังสือ 1 ชื่อ สามารถมีผู้แต่งได้หลายคน จึงต้องเพิ่ม Attribute เข้าไปใน Class เป็น List ด้วย

```
class Book(ABC):  
    def __init__(self, ISBN, title, subject, publisher, language, number_of_pages):  
        self.__ISBN = ISBN  
        self.__title = title  
        self.__subject = subject  
        self.__publisher = publisher  
        self.__language = language  
        self.__number_of_pages = number_of_pages  
        self.__authors = []
```



Case Study : ห้องสมุด

- จากนั้น ก็สร้าง BookItem และ Rack

```
class BookItem(Book):
    def __init__(self, ISBN, title, subject, publisher, language, number_of_pages,
                  barcode, price, book_format, status,
                  date_of_purchase, publication_date, placed_at):
        super().__init__(ISBN, title, subject, publisher, language, number_of_pages)
        self.__barcode = barcode
        self.__price = price
        self.__format = book_format
        self.__status = status
        self.__due_date = None
        self.__date_of_purchase = date_of_purchase
        self.__publication_date = publication_date
        self.__placed_at = placed_at

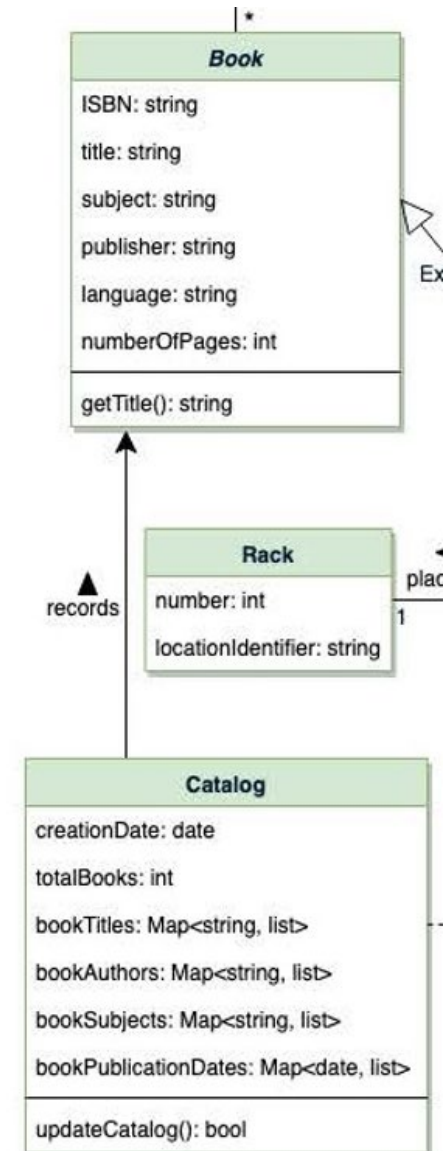
class Rack:
    def __init__(self, number, location_identifier):
        self.__number = number
        self.__location_identifier = location_identifier

rack1 = Rack('001', 'Second Floor')
book1 = BookItem('9781449355739', 'Learning Python', 'Programming', 'o'reilly', 'En', 1580,
                 '12345', 1792, BookFormat.HARDCOVER, BookStatus.AVAILABLE,
                 '01/01/2021', 'June 2013', rack1)
```

Case Study : ห้องสมุด



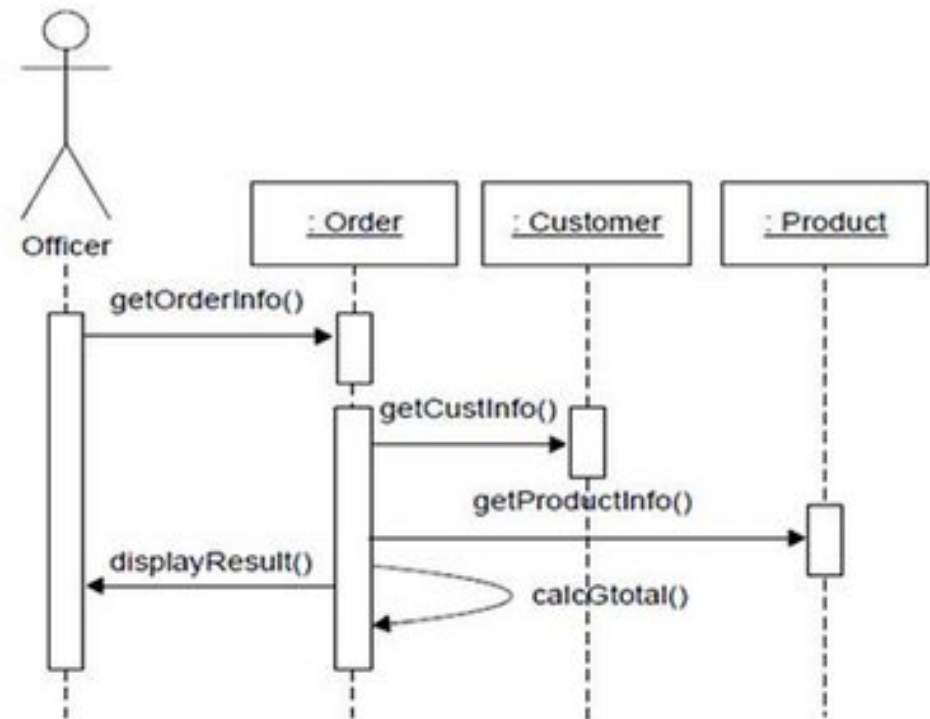
- ประเด็นต่อไปที่ต้องพิจารณา คือ ในคลาส Book ควรมี method อะไรบ้าง
- คลาส Book มีความสัมพันธ์แบบ Association กับ Catalog โดยรายชื่อหนังสือ จะอยู่ใน Catalog
- ดังนั้น Book จะถูกเรียกใช้โดย Catalog ในการ Search
- **คำถาม :** จากรูป Class Catalog ควรมี Attribute ตามรูปหรือไม่
- **คำถาม :** คลาส Book ควรมี method อะไรบ้าง





Sequence Diagram


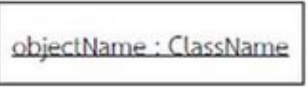


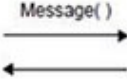

- แสดงลำดับการทำงานของระบบ
แสดงปฏิสัมพันธ์(Interaction)
ระหว่าง Object ตามลำดับของ
เหตุการณ์ที่เกิดขึ้น ณ เวลาที่กำหนด
- message ที่เกิดขึ้นระหว่าง class จะ
สามารถนำไปสู่การสร้าง method
ใน class ที่เกี่ยวข้องได้





Sequence Diagram

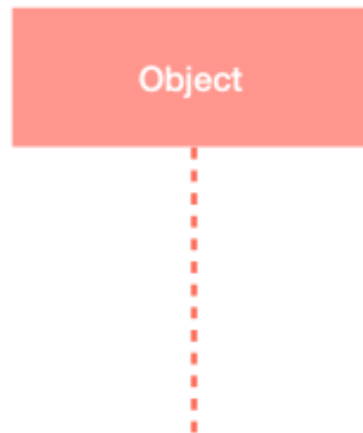
- Actor เป็น actor เดียวกับใน use case diagram
- Object ต้องเป็น Class เดียวกับ Class Diagram
- Lifeline แทนเส้นเวลาเหตุการณ์
- Activation Bar แสดงขอบเขตหรืออายุการทำงานของ event นั้นๆ
- Message แสดงชื่อ method ที่เรียกใช้ และผลลัพธ์ (ถ้ามี)
- Call back สำหรับการคืนมาใน Object เดียวกัน

สัญลักษณ์	ชื่อ	ความหมาย
	Actor	ผู้ที่เกี่ยวข้องกับระบบ
	Object	อ็อบเจกต์ที่ต้องทำหน้าที่ตอบสนองต่อ Actor
	Lifeline	เส้นแสดงชีวิตของอ็อบเจกต์หรือคลาส
	Focus of Control / Activation	จุดเริ่มต้นและจุดสิ้นสุดของแต่ละกิจกรรมในระหว่างที่มีชีวิตอยู่
	Message	คำสั่งหรือฟังก์ชันที่อ็อบเจกต์หนึ่งส่งให้อ็อบเจกต์หนึ่ง ซึ่งสามารถส่งกลับได้ด้วย
	Callback / Self Delegation	การประมวลผลและคืนค่าที่ได้ภายในอ็อบเจกต์เดียวกัน



Sequence Diagram

- Lifeline Notation (เส้นชีวิต)
 - คือเส้นชีวิตของวัตถุหรือ class เป็นตัวแทนของวัตถุหรือส่วนประกอบต่างๆที่มีปฏิสัมพันธ์ซึ่งกันและกันในระบบในลำดับต่างๆ
 - format การเขียนชื่อเส้นชีวิตคือ Instance Name:Class Name
 - เส้นชีวิตกับสัญลักษณ์ Actor จะใช้เมื่อลำดับใดลำดับหนึ่งโดยเฉพาะนั้นเป็นส่วนหนึ่งของ Use Case





Sequence Diagram

- Activation Bars
 - Activation Bars จะวางอยู่บนเส้นชีวิตเพื่อแสดงการโต้ตอบระหว่าง object, function หรือ module ความยาวของสี่เหลี่ยมจะแสดงระยะเวลาการโต้ตอบของ object หรือ จุดเริ่มต้นและจุดสิ้นสุดของแต่ละกิจกรรมของ object นั้น
 - การโต้ตอบระหว่าง 2 object เกิดเมื่อวัตถุหนึ่งส่ง message ไปให้อีก object โดย object ที่ส่งข้อความเรียกว่า Message Caller และ object ที่รับข้อความเรียกว่า Message Receiver เมื่อมีแถบ Activation บนเส้นชีวิตของวัตถุ นั้นหมายความว่า วัตถุนั้นมีการทำงานในขณะที่ส่งข้อความโต้ตอบกัน





Sequence Diagram

- Activation Bars
 - **Message Arrows** ลูกศรจาก Message Caller จะชี้ไปที่ Message Receiver และระบุทิศทางของ message ว่าไหลไปในทางใด โดยสามารถไหลไปในทิศทางใดก็ได้ จากซ้ายไปขวา ขวาไปซ้าย หรือส่ง message กลับไปที่ตัวมันเองก็ได้
 - รูปแบบของ Message มี 2 แบบ ได้แก่
 - Synchronous message จะถูกใช้เมื่อ object ที่ส่งข้อความรอให้ object ที่รับ message ประมวลผลและส่ง return กลับมา ก่อนที่จะส่ง message อันต่อไป หัวลูกศรที่ใช้จะเป็นลูกศรแบบทึบ
 - Asynchronous message จะถูกใช้เมื่อ object ที่ส่ง message ไม่รอให้ object ที่รับ message ประมวลผลข้อความและส่งค่า return กลับมา แต่จะส่งข้อความต่อไปให้แก่วัตถุอื่นในระบบเลย หัวลูกศรที่แสดงในข้อความประเภทนี้เป็นหัวลูกศรเส้น

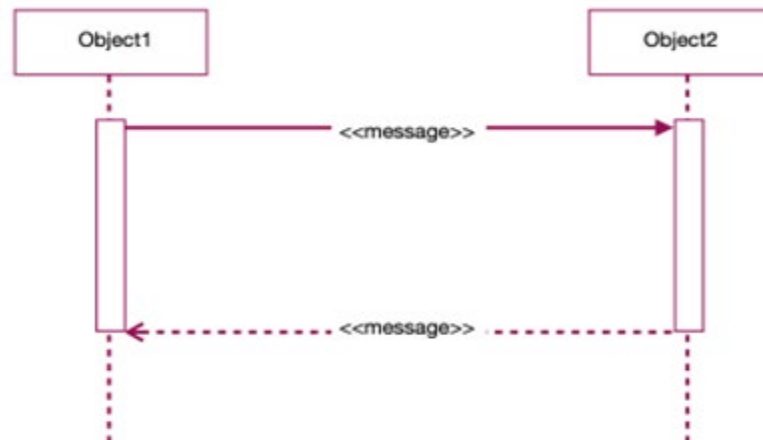




Sequence Diagram

- Activation Bars

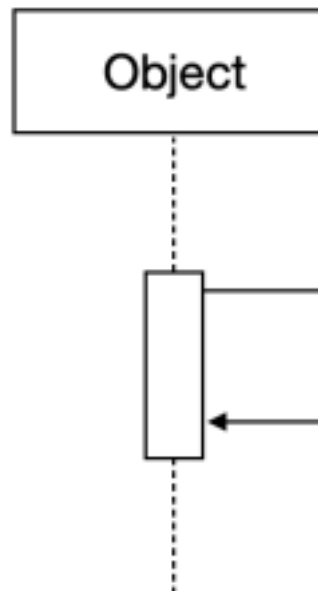
- return message ใช้เพื่อระบุว่า object รับ message และประมวลผล message เสร็จสิ้นแล้ว และกำลังส่งคืนการควบคุมไปยังวัตถุที่ทำหน้าที่ส่ง message
- return message เป็นตัวเลือก ที่จะเลือกให้มีหรือไม่มีก็ได้ สำหรับการส่ง message บนแถบ Activation ด้วย Synchronous message จะให้ความหมายโดยนัยว่ามี return message ด้วยแม้จะไม่ได้มีเส้น return message แสดงก็ตาม
- เราสามารถหลีกเลี่ยงการทำให้แผนภาพดูยุ่งเหยิงโดยการไม่ใช้ return message เมื่อไม่จำเป็น





Sequence Diagram

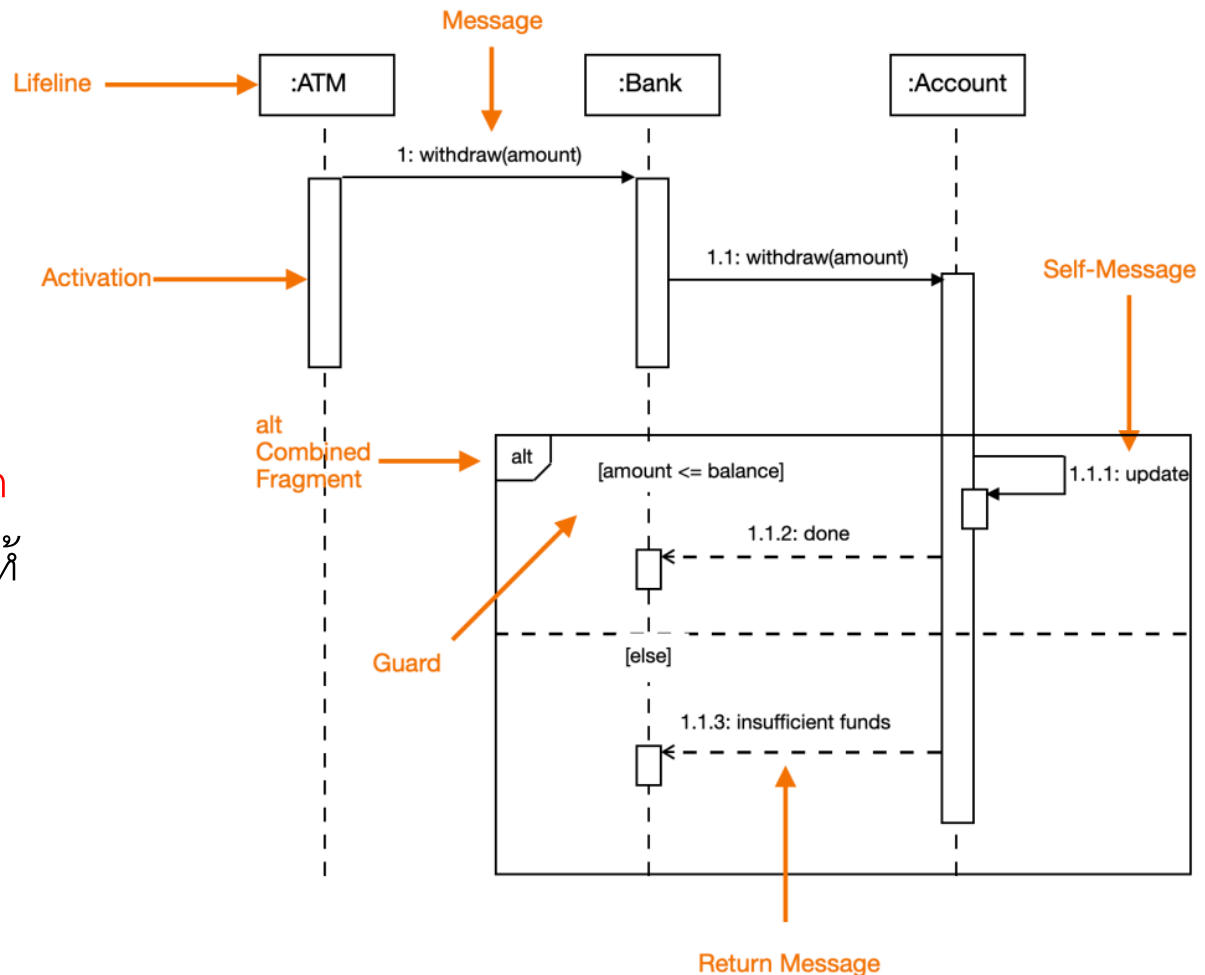
- Activation Bars
 - Reflexive message เมื่อ object ส่งข้อความหาตัวเอง จะเรียกว่า reflexive message แสดงข้อความประเภทนี้โดยใช้ message arrow ที่เริ่มจากจบที่เส้นชีวิตเดียวกัน อย่างตัวอย่างด้านล่างนี้



Sequence Diagram



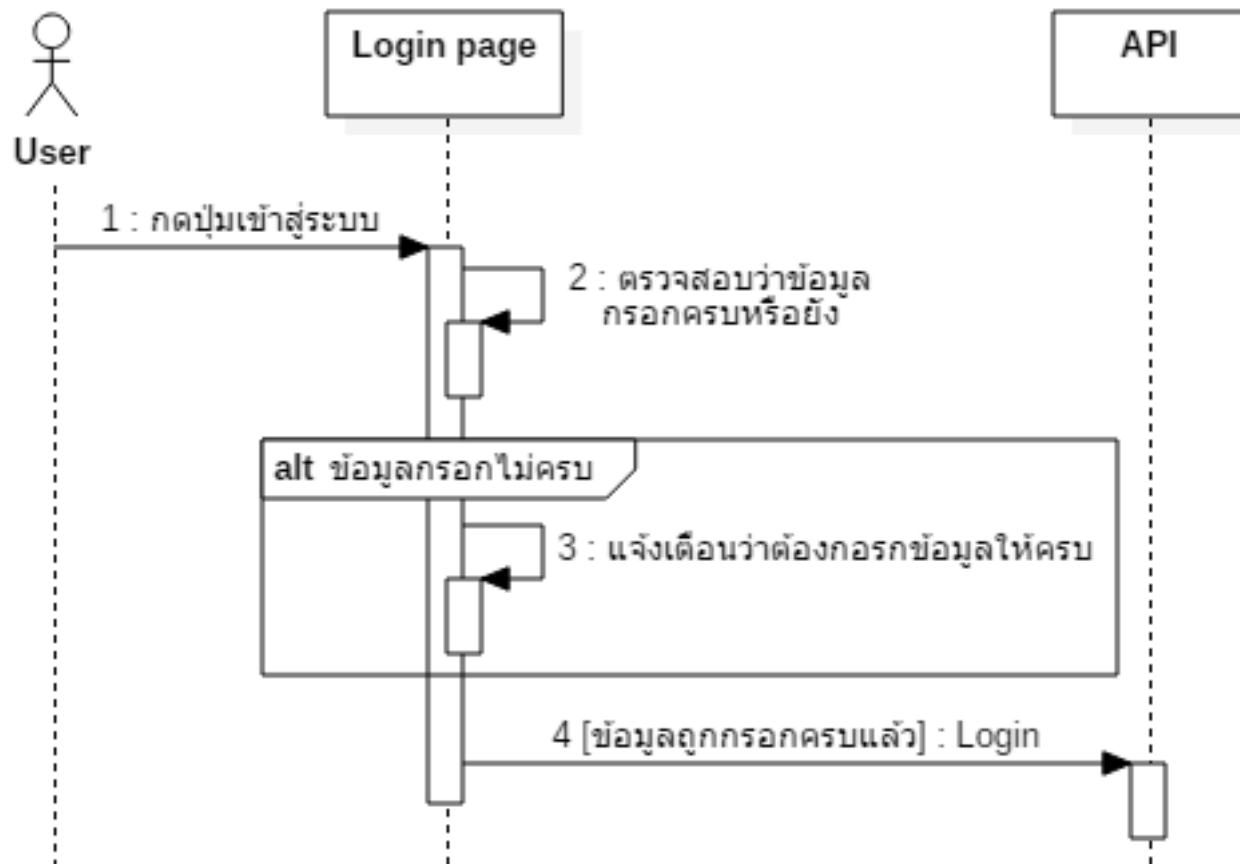
- Sequence Fragment
 - คือกล่องที่มีเครื่องหมายแสดง section การโต้ตอบระหว่างวัตถุใน sequence diagram
- Alternative combination fragment** ใช้เมื่อมีตัวเลือกให้เลือกตั้งแต่ 2 ตัวเลือกขึ้นไป ใช้ตรรกะแบบ “if then else”





Sequence Diagram

- ตัวอย่าง Alternative Fragment



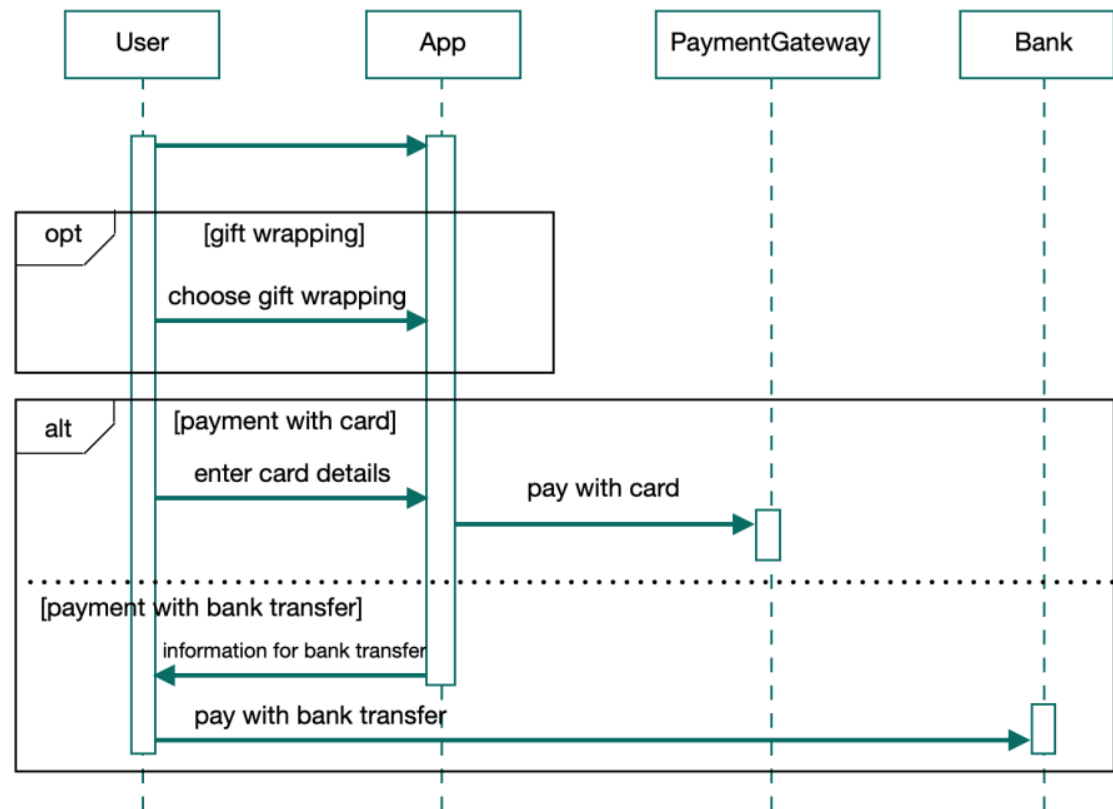
Sequence Diagram



- Options

Combination
fragment

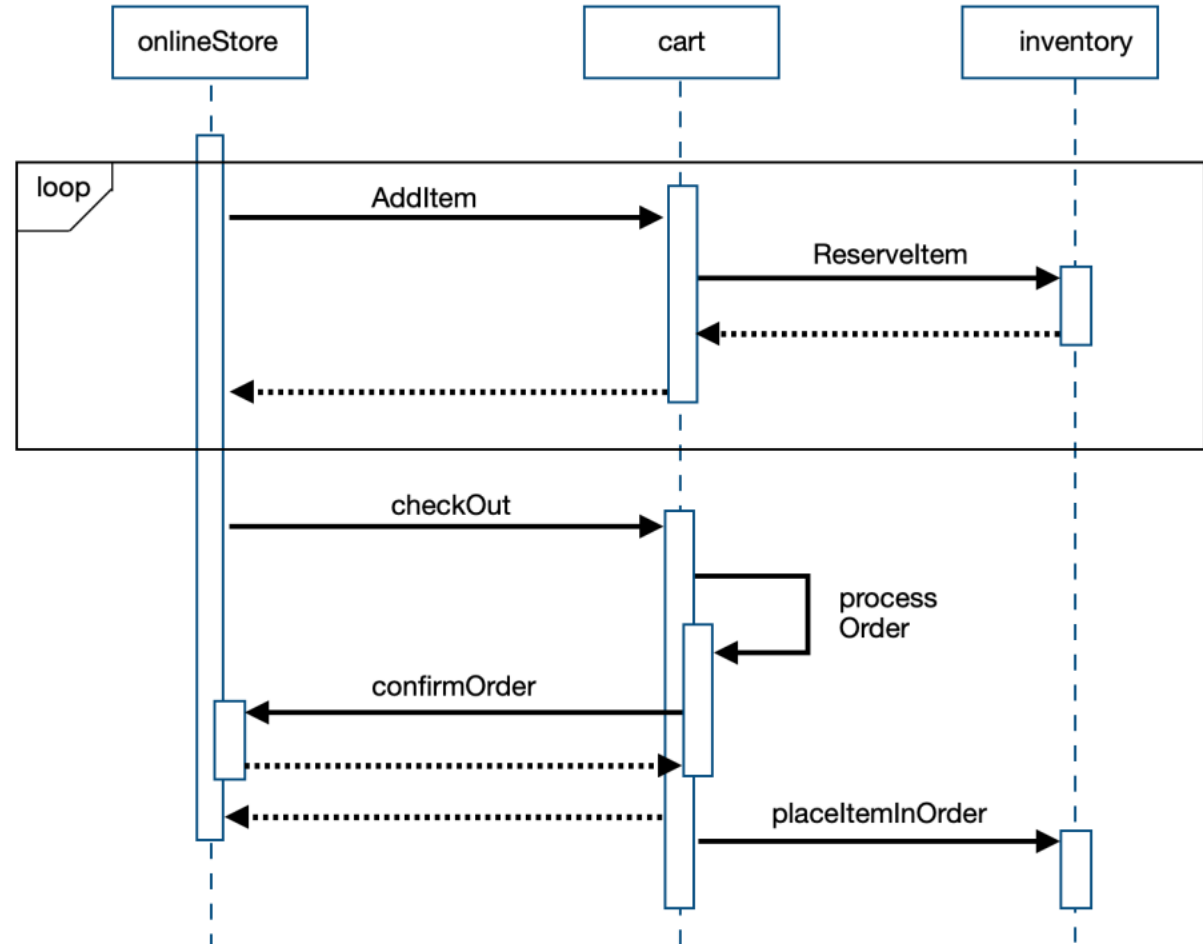
- ใช้เพื่อแสดงถึงลำดับเหตุการณ์ที่เกิดขึ้นภายใต้เงื่อนไขใดเงื่อนไขหนึ่งเท่านั้น ไม่เช่นนั้นเหตุการณ์นั้นจะไม่สามารถเกิดขึ้นได้ ใช้ตรรกะแบบ 'if then'



Sequence Diagram



- **Loop fragment**
- ใช้เพื่อแสดงลำดับเหตุการณ์ที่เกิดขึ้นซ้ำ โดยมี 'loop' เป็น fragment operation และ guard condition ระบุที่มุมด้านซ้ายของกล่อง





Sequence Diagram

- ขอยยความเรื่อง Object โดย Object หรือ Class ที่เราเรียนมาทั้งหมดจะเรียกว่า **Entity Object** โดยจะใช้กับแต่ละส่วนประกอบของเนื่องานจริง ซึ่งจะมองเห็นได้ง่าย เช่น หนังสือ สมาชิก ฯลฯ พุดง่าย ๆ คือ สามารถจับต้องได้
- Object อีกประเภทหนึ่ง ที่ต้องสร้างขึ้นมา ในการพัฒนาซอฟต์แวร์จริง คือ **Boundary Object** โดยความหมายคือ Object ที่รับผิดชอบในการเชื่อมต่อกับระบบของเรา เช่น ระบบของเราต้องไปเชื่อมต่อกับ google maps ก็ต้องสร้าง google map boundary object เพิ่มเข้ามา ซึ่งมักมีลักษณะเป็น Interface Class
- **Boundary Object** อีกประเภทหนึ่ง คือ ส่วนติดต่อกับผู้ใช้ ซึ่งมีหน้าที่ในการรับข้อมูล และแสดงผลการทำงานให้กับผู้ใช้ ถ้าระบบมีส่วนต่อกับผู้ใช่มาก Object กลุ่มนี้ก็จะมากตามไปด้วย จากตัวอย่างที่ผ่านมา Class Menu ใน Notebook เป็นตัวอย่างของ คลาสประเภทนี้



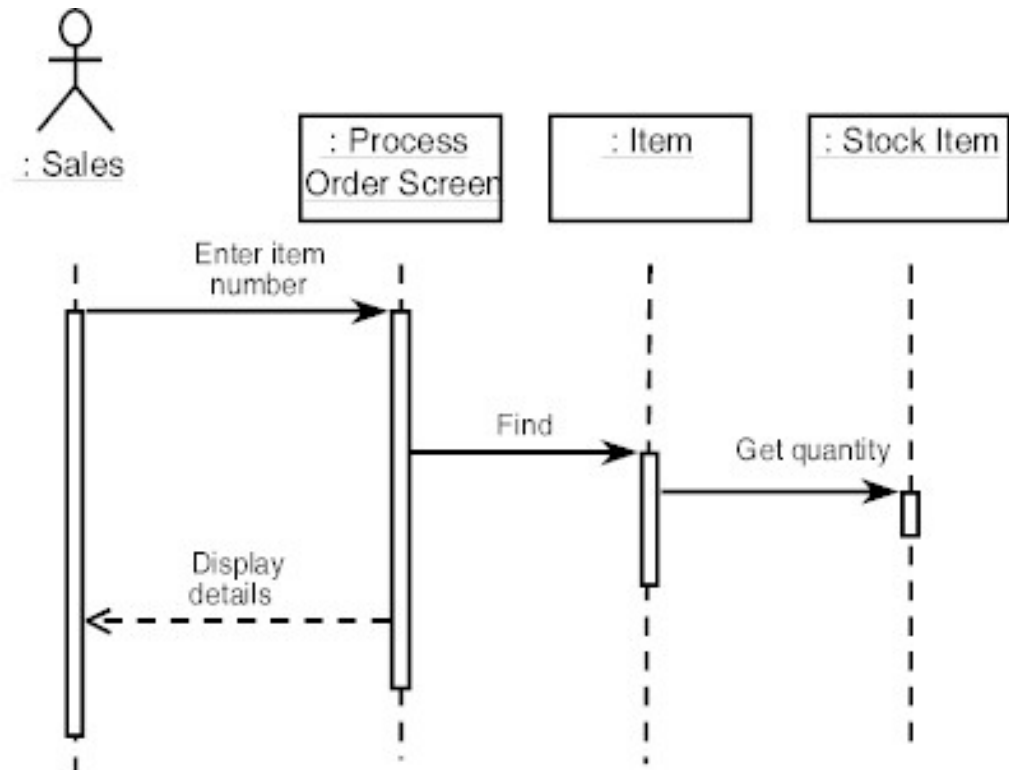
Sequence Diagram

- Object อีกประเภทหนึ่ง คือ **Control Object** ซึ่งรับผิดชอบในการประสานงาน หรือ Object ตัวกลางที่ประสานการทำงานระหว่าง Entity Object มักจะเห็นในระบบที่มีขนาดใหญ่ขึ้นมา และมีการทำงานซับซ้อน ตัวอย่าง เช่น Object การยืมในระบบห้องสมุด
- Control Object มักจะมีชื่อเป็นการทำงาน ไม่ใช่ชื่อของ Object เช่น การยืม การจอง หนังสือ Object ประเภทนี้ มักจะมีความสัมพันธ์กับ Object หรือ Class อื่นๆ หลายคลาส ซึ่งทำให้ Logic การทำงานภายใน Class มีความซับซ้อน ซึ่งบางครั้งก็อาจจะพิจารณาว่าสามารถแยกเป็น Object ย่อยได้อีกหรือไม่



Sequence Diagram

- จากรูปเป็นขั้นตอนการค้นหาข้อมูลสินค้าในคลัง
- เริ่มจากเซลล์ติดต่อกับ Process Order Screen (POS) ซึ่งเป็นคลาสพิเศษที่เรียกว่า **Border Class** คือ ทำหน้าที่ติดต่อกับผู้ใช้ โดยป้อนหมายเลข Item
- จากนั้น POS จะเรียกใช้ method `Find` จาก Object Item
- Object Item ก็เรียกใช้ method `get_quantity` ใน Object Stock Item อีกที





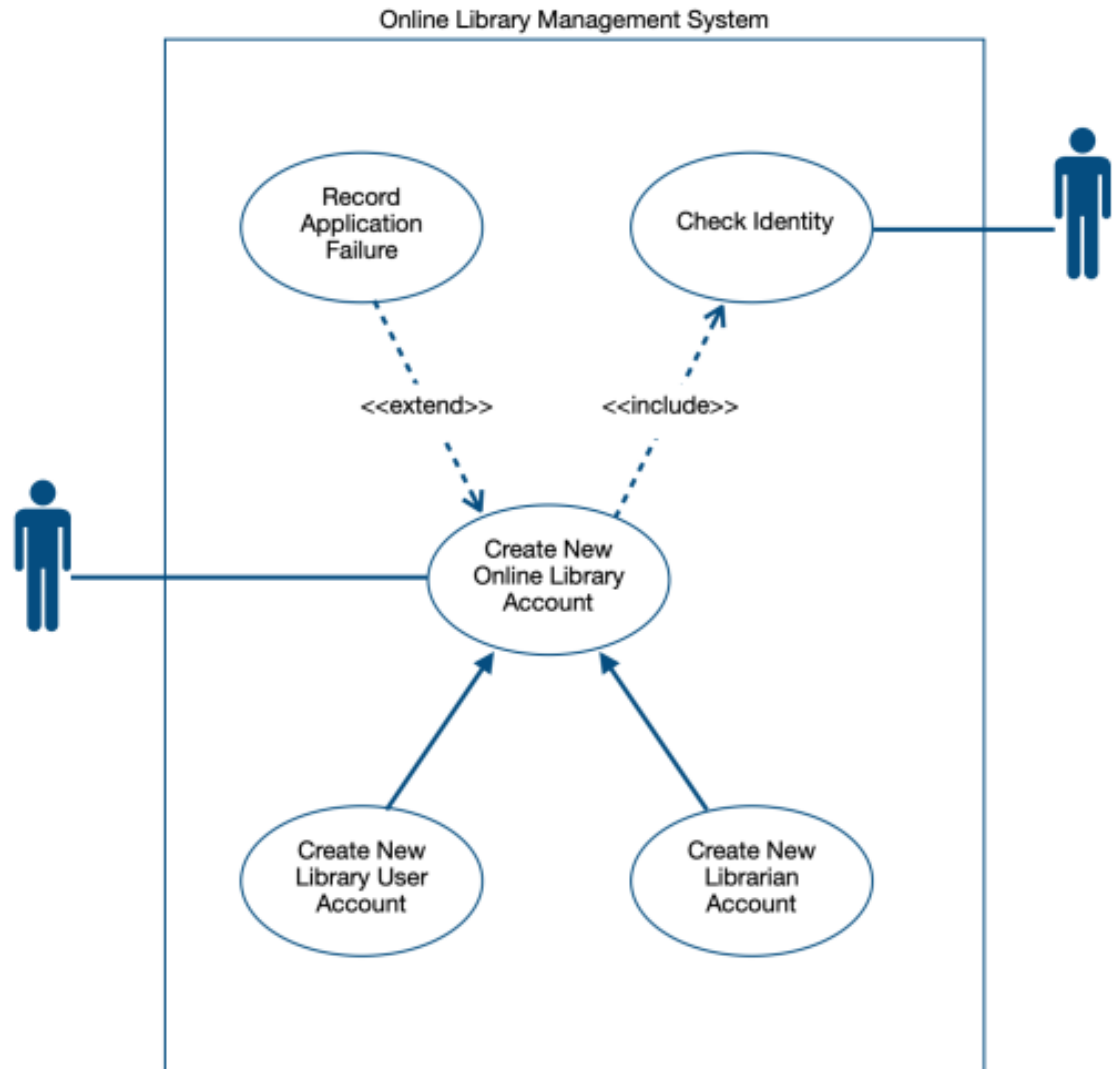
Sequence Diagram

- แนวทางการเขียน Sequence Diagram
- เนื่องจาก sequence diagram แสดงลำดับการไหลของเหตุการณ์ของหนึ่ง Use Case (แปลว่า 1 Use Case จะมี 1 Sequence Diagram) การไหลไปของข้อความหรือ Message ใน Sequence Diagram จะเป็นไปตามทิศทางของ Use Case นั้นๆ
- ดังนั้นก่อนวาด Sequence Diagram เราจะต้องวาด Use Case ก่อน และควรมี Use Case Description หรือ รายละเอียดคร่าวๆ ของการทำงานก่อน เพื่อจะได้พิจารณาว่ามีการโต้ตอบหรือเหตุการณ์ใดบ้างที่จะระบุถึงใน Sequence Diagram



Sequence Diagram

- ตัวอย่าง Use Case ของ Online Library Management System



ตัวอย่าง Use Case Diagram



Sequence Diagram

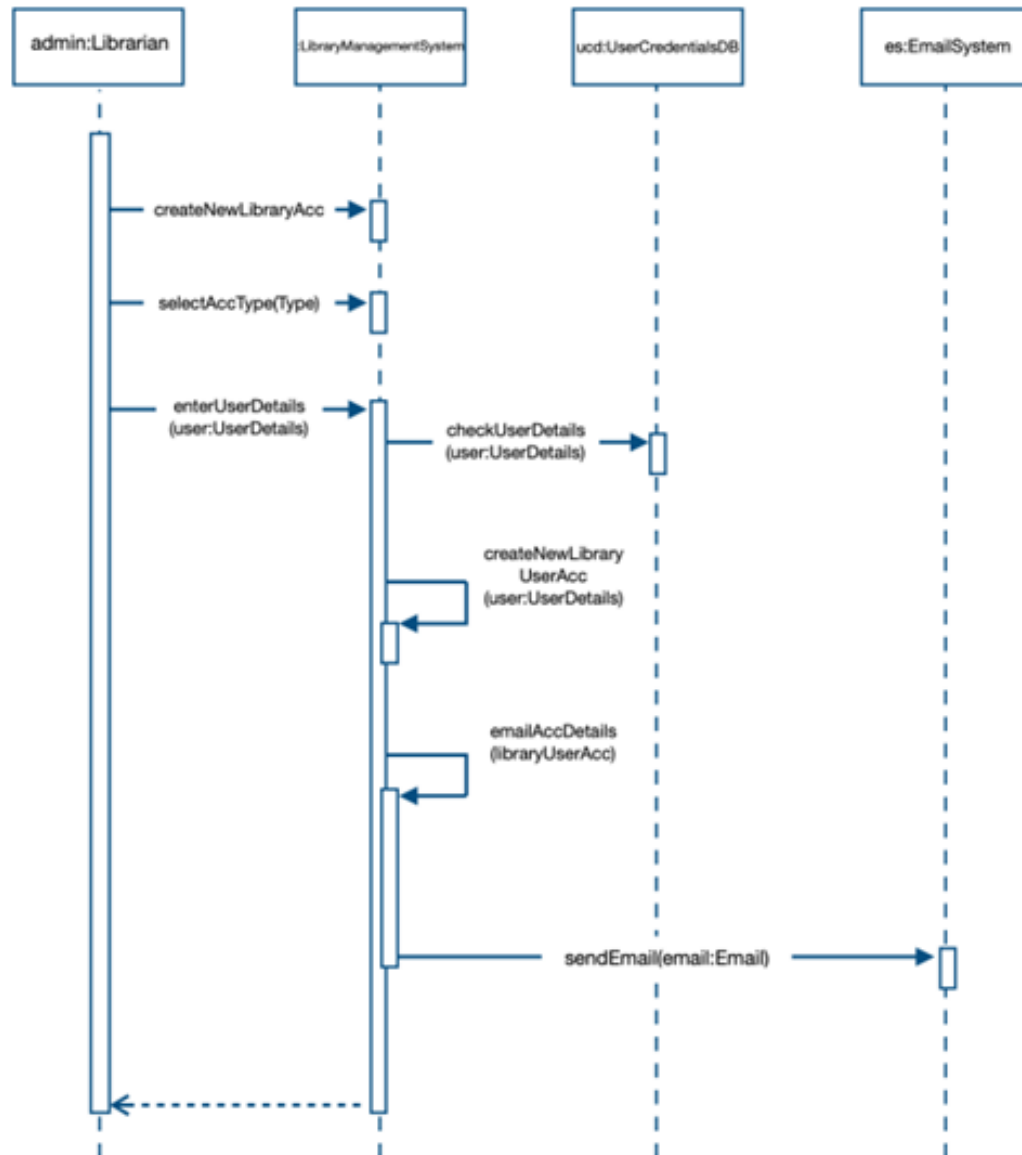
- เลือก ขั้นตอน Create New Online Library Account
- ระบุวัตถุ (Object) หรือ Actor ที่มีเกี่ยวข้องสำหรับการสร้าง new user account ซึ่งมีดังนี้
 - Librarian
 - Online Library Management system
 - User credential database
 - Email system



Sequence Diagram

- ตัวอย่างเหตุการณ์ที่เกิดขึ้นใน Use Case ‘Create New Library User Account’
 - Librarian ร้องขอระบบให้สร้าง online library account ใหม่
 - Librarian เลือกประเภทของ user account
 - Librarian ใส่รายละเอียดของผู้ใช้งาน
 - รายละเอียดของผู้ใช้งานจะถูกตรวจสอบโดย User Credential Database (หรือฐานข้อมูล)
 - บัญชีผู้ใช้ใหม่ถูกสร้าง
 - รวมข้อมูลทั้งหมดสำหรับบัญชีใหม่ และส่งอีเมลให้เจ้าของบัญชี

Sequence Diagram



Lab 5 (ต่อ)



- จากโจทย์เดิม
- จาก Use Case Diagram ให้เขียน Use Case Diagram : Detail Description ตามตัวอย่างใน Slide 9 (ทำคนละ 1 Use Case) ใน Use Case ที่สำคัญ
- ให้เขียน Class Diagram ที่สมบูรณ์ ตาม Use Case และให้บอกว่าเพิ่มตรงไหน หรือ ตัดตรงไหนออก เพราะอะไร (ทำคนละ 1 คลาส) โดยเน้นที่คลาสหลักของระบบ
- ให้เขียนโครงของ Class เป็นภาษา Python ตาม Class Diagram ที่เลือก ให้ครบ โดยดูจากความสัมพันธ์



Lab 5 (ต่อ)

- ให้เขียน Sequence Diagram แสดงการโต้ตอบระหว่าง Class มาคนละ 1 ตัวอย่าง
- ให้อธิบายให้เพื่อนฟัง และ ในกลุ่มช่วยกันตรวจสอบความถูกต้อง
- ให้เตรียมการนำเสนอให้อาจารย์ โดยแต่ละคนต้องอธิบายส่วนของตัวเอง
- จากนั้นนำมาสรุปเป็นรายงาน (โครงงาน #1) คิดเป็น 10 % ของคะแนน



For your attention