物件導向系統分析與設計
Object Oriented Analysis and Design

Sequence Diagram

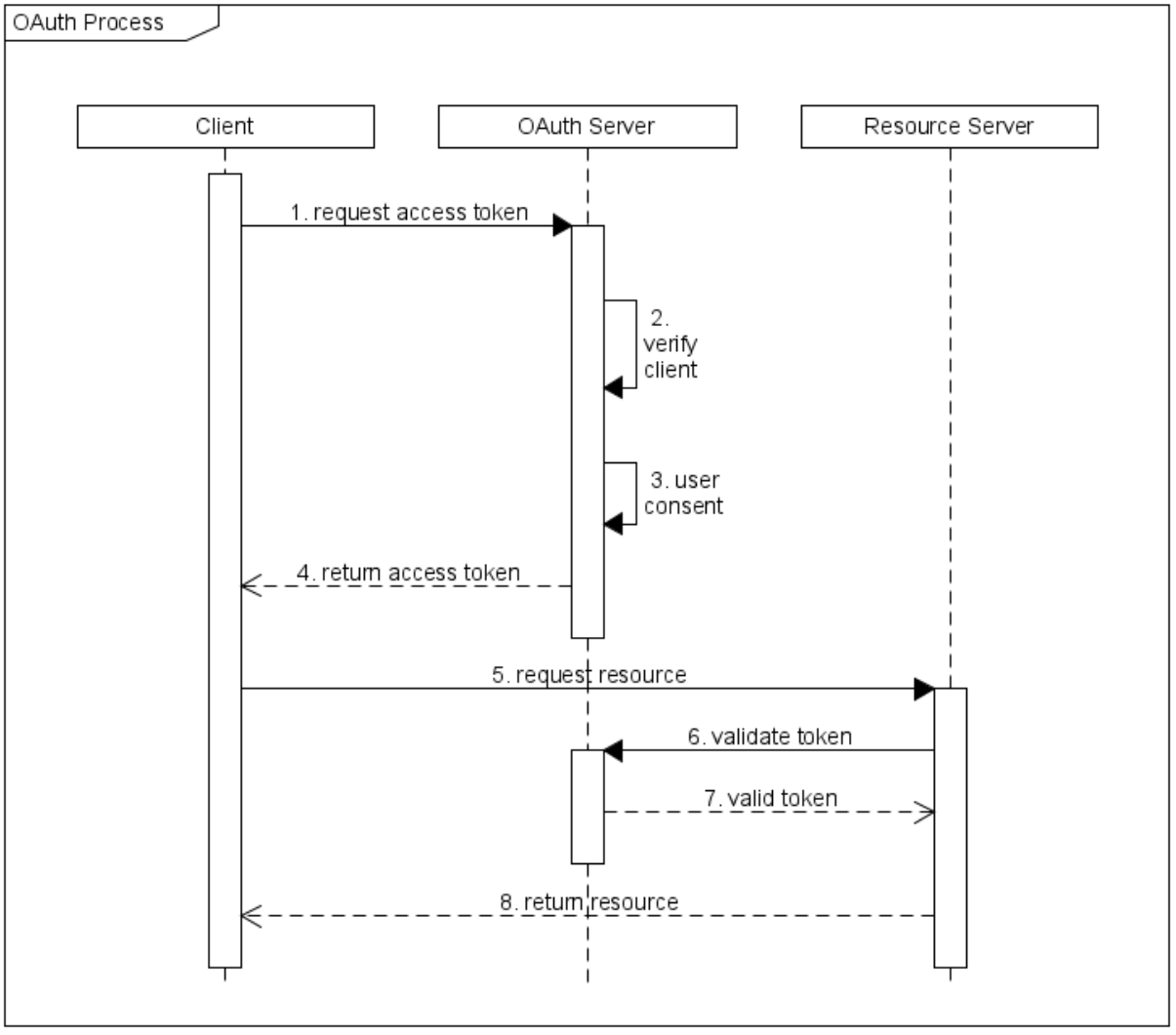
劉儒斌 Paladin R. Liu
paladin@ntub.edu.tw

AGENDA

- 循序圖簡介
- 循序圖圖示
- 互動概觀圖
- 綜合應用

循序圖簡介

- 描問題領域中物件之間互動的情形
- 描述物件與物件之間傳遞訊息的先後順序
- 強調訊息傳遞的時間性
- UML 中常用且重要的一種工具
  - 可以與通訊圖相互轉換


OAuth Process
Client
OAuth Server
Resource Server
1. request access token
2. verify client
3. user consent
4. return access token
5. request resource
6. validate token
7. valid token
8. return resource

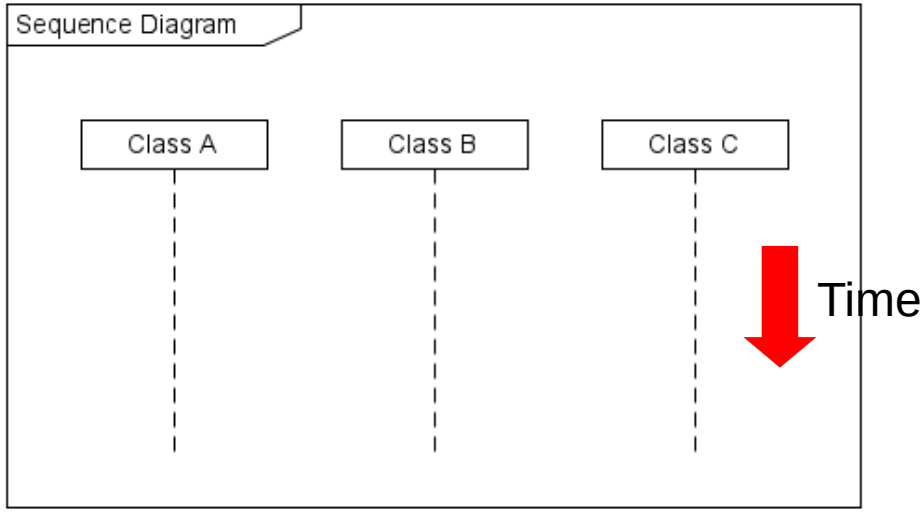
循序圖圖示

循序圖圖示

- **標題 Title**
  - 說明這張圖的用途，非必要
- **參與者 Participant**
  - 使用類別 Class 或物件 Object 圖示
  - 位於循序圖的最上方
    - 不可以互相重疊
  - 代表這些類別 / 物件會參與互動的進行

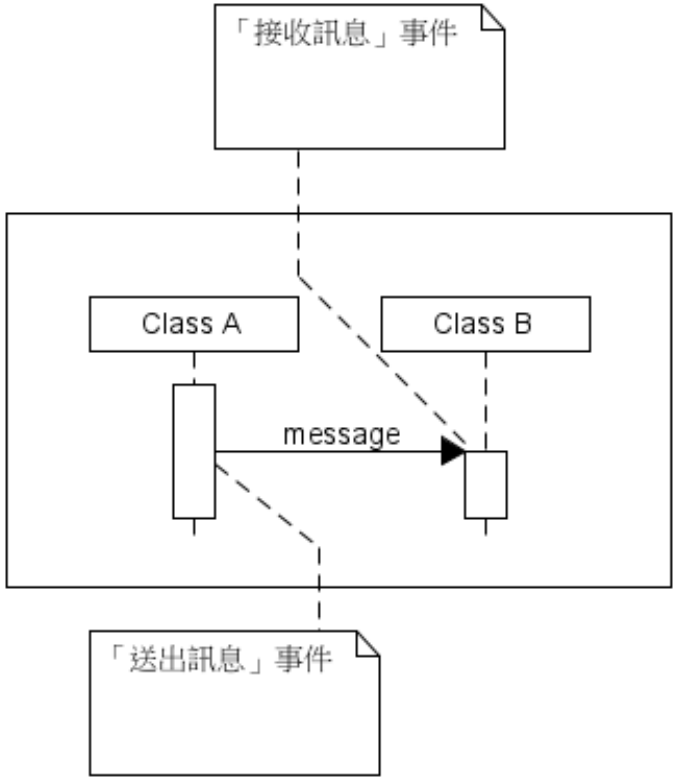
循序圖圖示

- **生命線 Life line**
  - 使用垂直的虛線表示
  - 代表物件在系統中的存續
- **時間軸**
  - 由上往下，代表時間的消逝
  - 也代表了互動執行的順序
  - 但與實際執行了多久，沒有直接關係



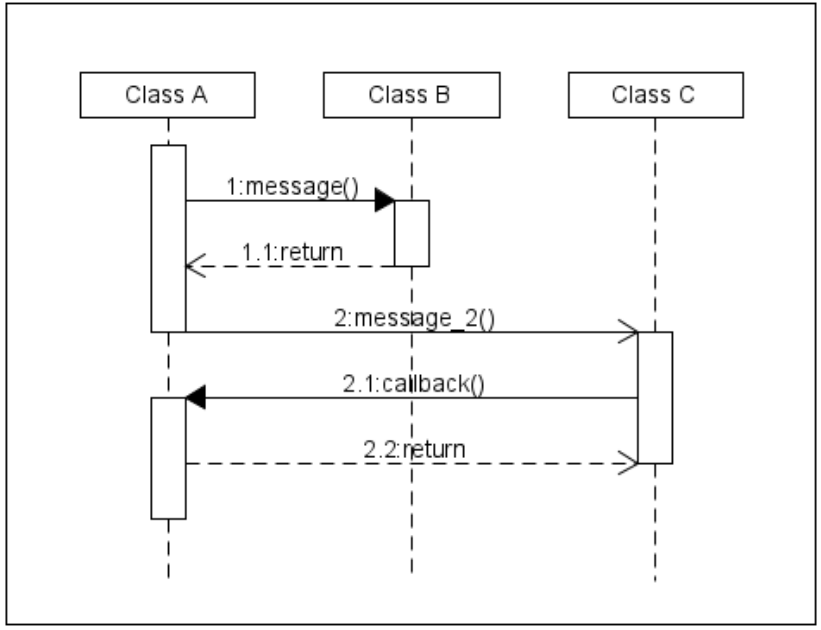
循序圖圖示

- **事件 Event**
  - 代表有一個活動在某一個時間點發生了
- **訊息 Message**
  - 物件、資料，或是一段簡短的描述
- **啓動 Activation Bar**
  - 代表物件取得了控制權
  - 正在執行 / 運作中



循序圖圖示

- **訊息的種類**
  - 同步訊息
    - 呼叫並等待結果的回傳
    - 回傳訊息 (不一定要畫)
  - 非同步訊息
    - 呼叫後就結束
    - 被呼叫端在執行結束後，再發起一個 Callback
    - Javascript 中，使用 AJAX 呼叫



循序圖圖示

同步訊息

```
class ClassA {
  process(...) {

    ClassB.message();

  }
}
```

```
class ClassB {
  static message() {

    // do something…

    return;
  }
}
```

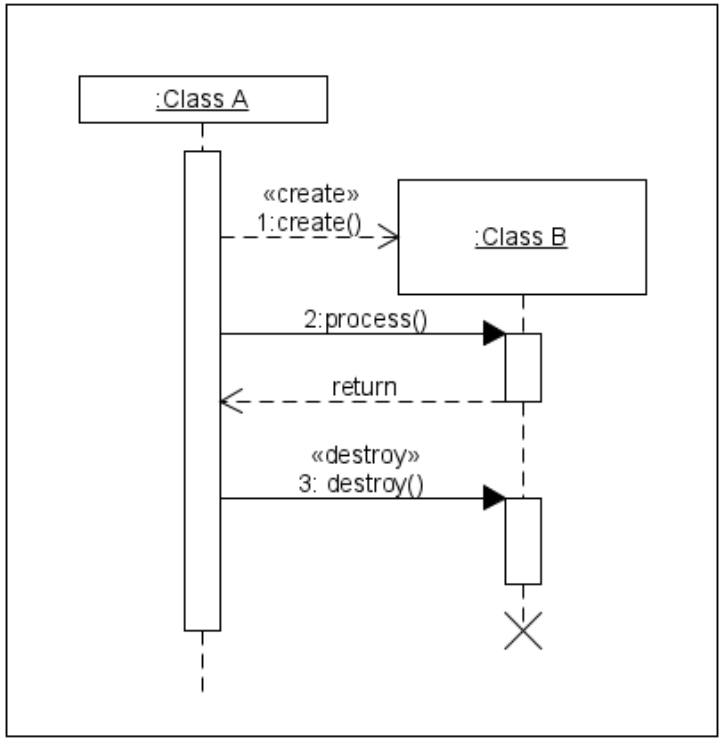
循序圖圖示

非同步訊息

```
fetch('http://xxx.xxx.xxx/xxx.json')
  .then((resp) => resp.json())
  .then((obj) => console.log(obj));
```

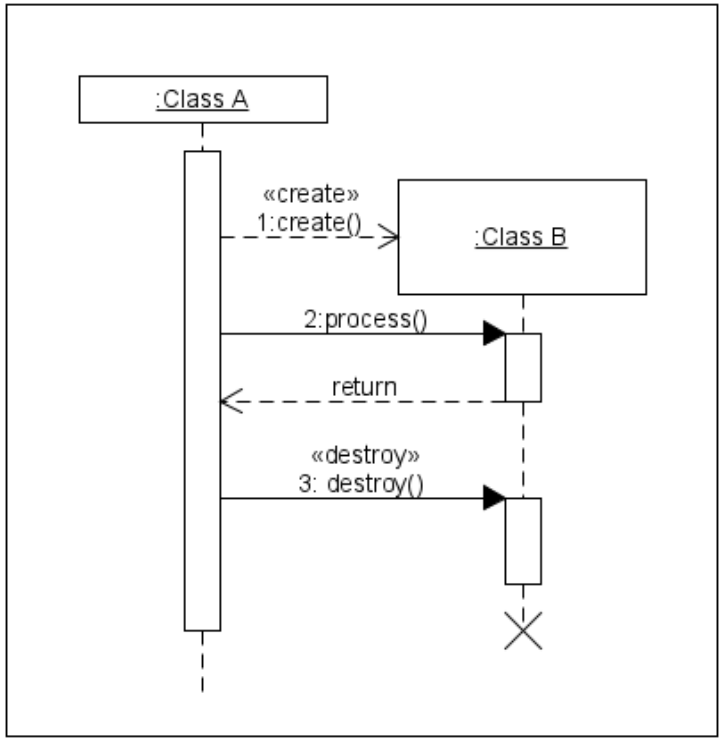
循序圖圖示

- **訊息的種類**
  - 建構訊息 Creation
    - 使用 «create»
    - 建立新的物件
  - 消滅訊息 Destruction
    - 使用 «destroy»
    - 釋放一個物件



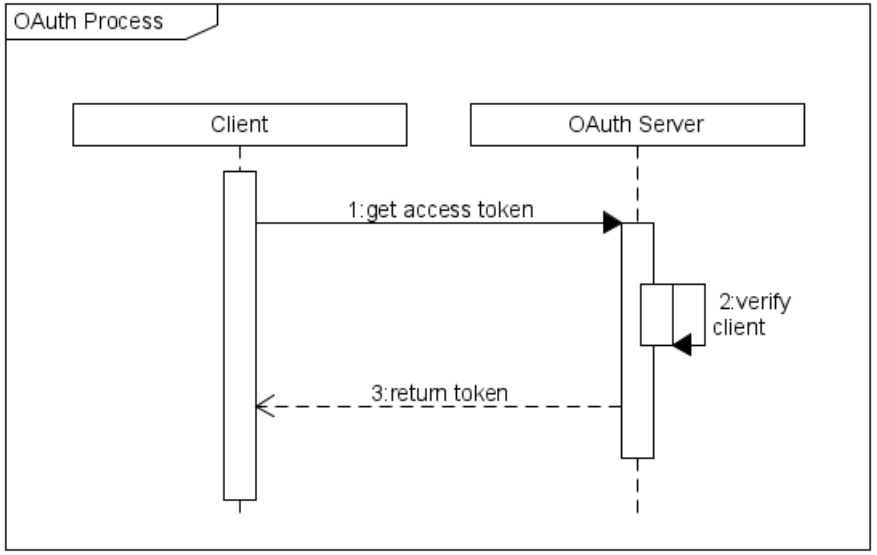
循序圖圖示

```
void main(…) {

  ClassA obj = new ClassA();

  obj.process();

  delete obj;
}
```



循序圖圖示

- **Self Message**
  - 把訊息傳送給自己
  - E.g. 執行某些物件內部的重要功能執行
    - 驗證服務裡的帳號驗證功能

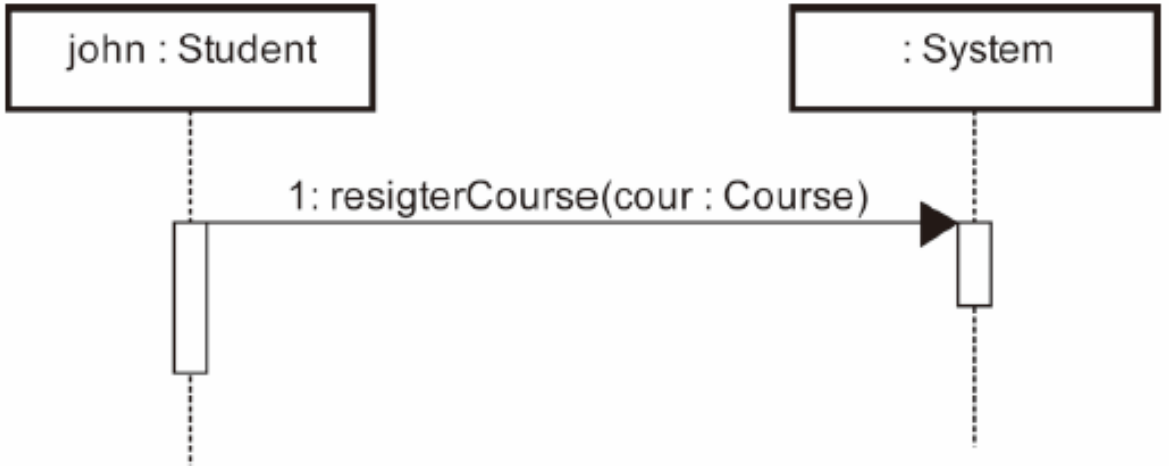


循序圖圖示

- **訊息參數 Message Signature**
  - 傳遞訊息給另一個物件時，訊息也可以一起傳送參數
  - 就像呼叫函數所提供的參數
  - 格式
    - 參數名稱：型態

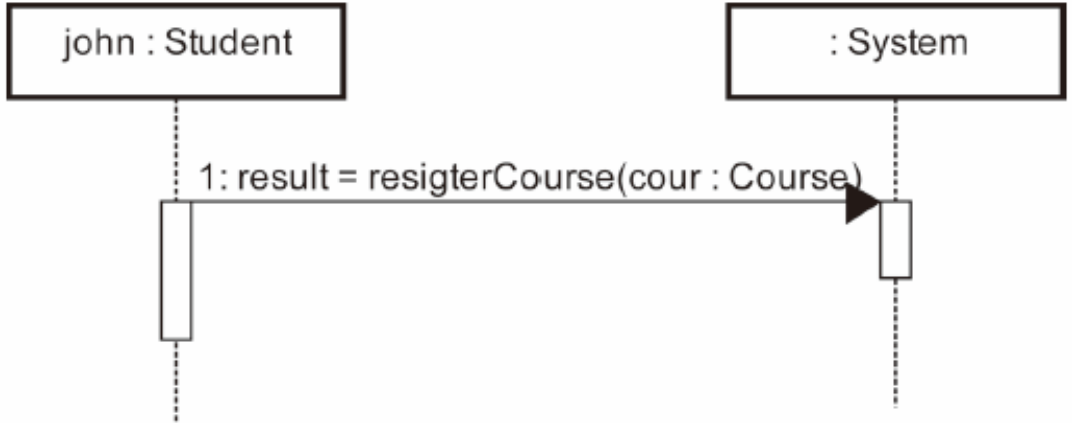
循序圖圖示

- **範例：學生選修 / 註冊課程**
  - 學生 John 向系統要求註冊某一課程「 cour 」



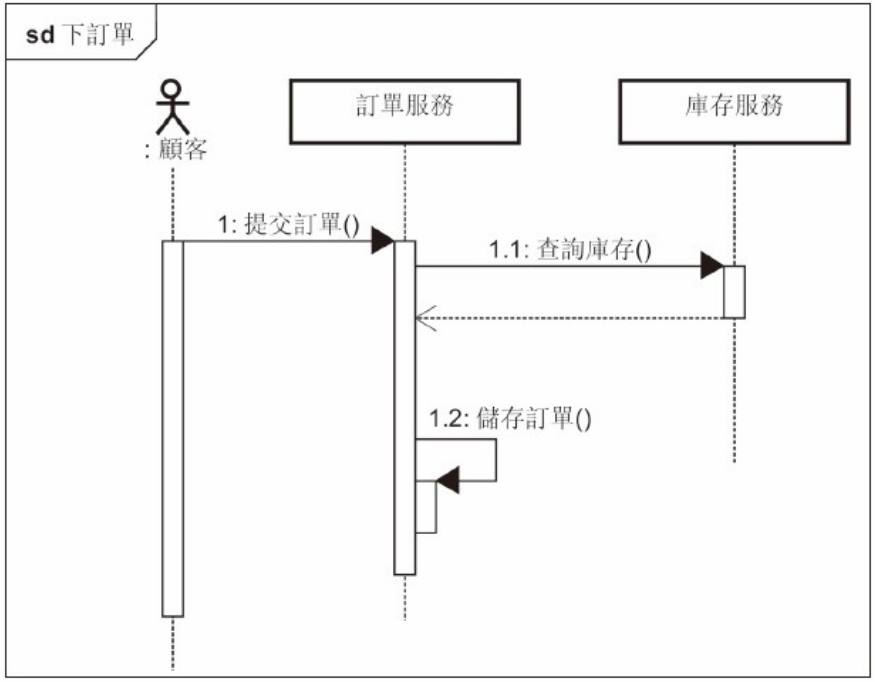
循序圖圖示

- **訊息回傳值 Return Value**
  - 訊息有回傳值時，可以在圖中表示
  - 把結果存放在變數中



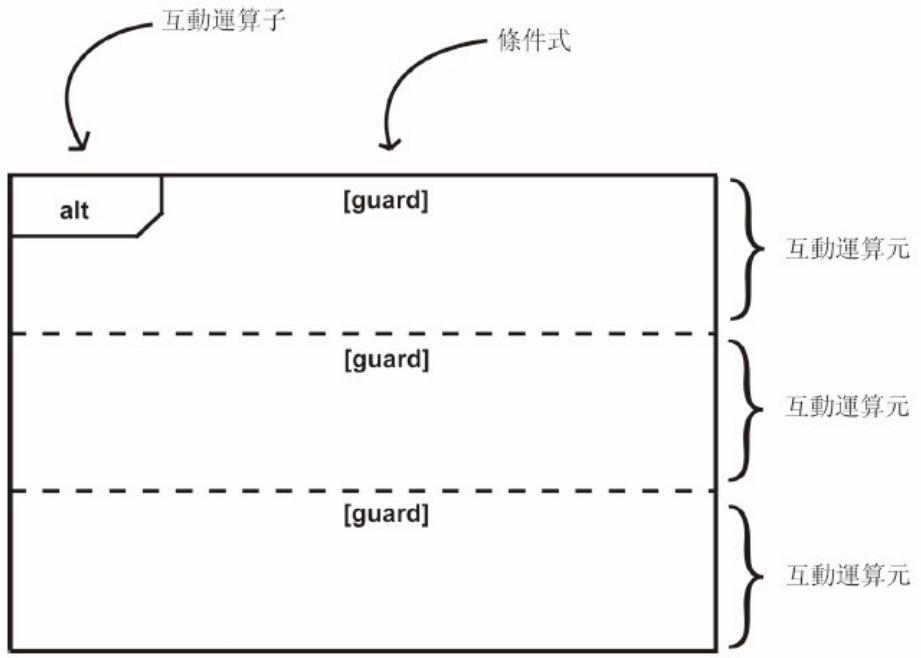
循序圖圖示

- **框架 Frame**
  - 一個長方矩形，左上角有一個五邊形的標題
    - 標題可以省略
  - 面對複雜的內容時，可以大略說明圖形的目的



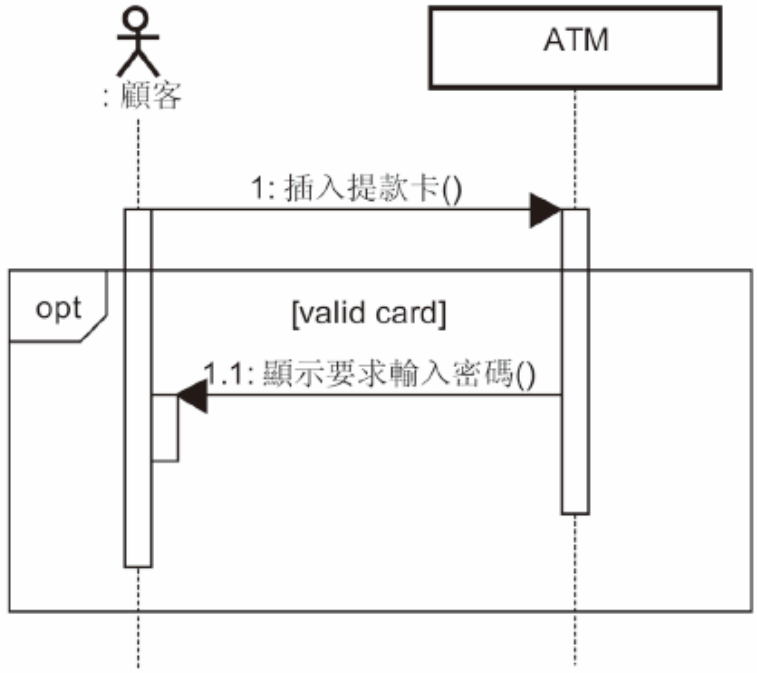
循序圖圖示

- **組合片段 Combined Fragment**
  - UML 2.0 新定義
  - 需要滿足某些條件才會繼續進行
  - 使用迴圈來處理集合中的元素



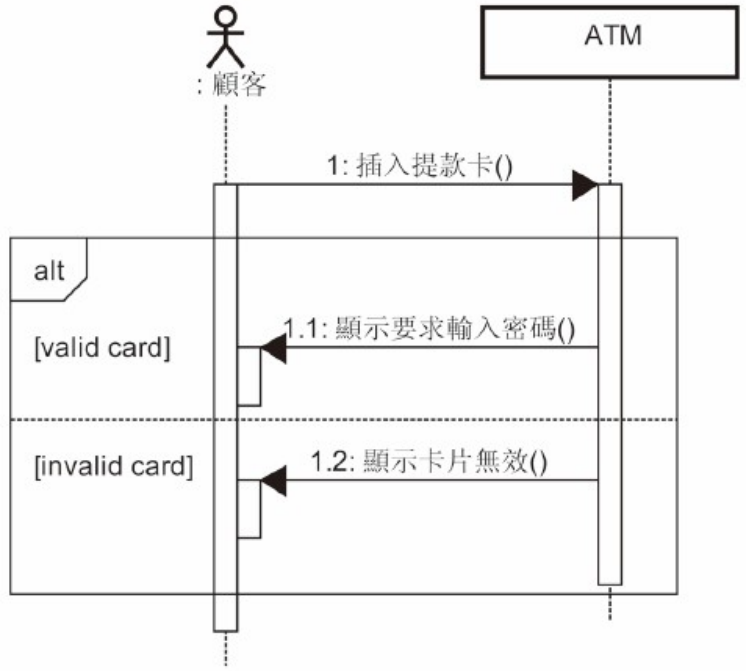
循序圖圖示

- **組合片段： opt**
  - 當條件成立時，才會執行單一運算元
  - E.g. ATM 操作
    - 卡片有效時，才會要求輸入密碼



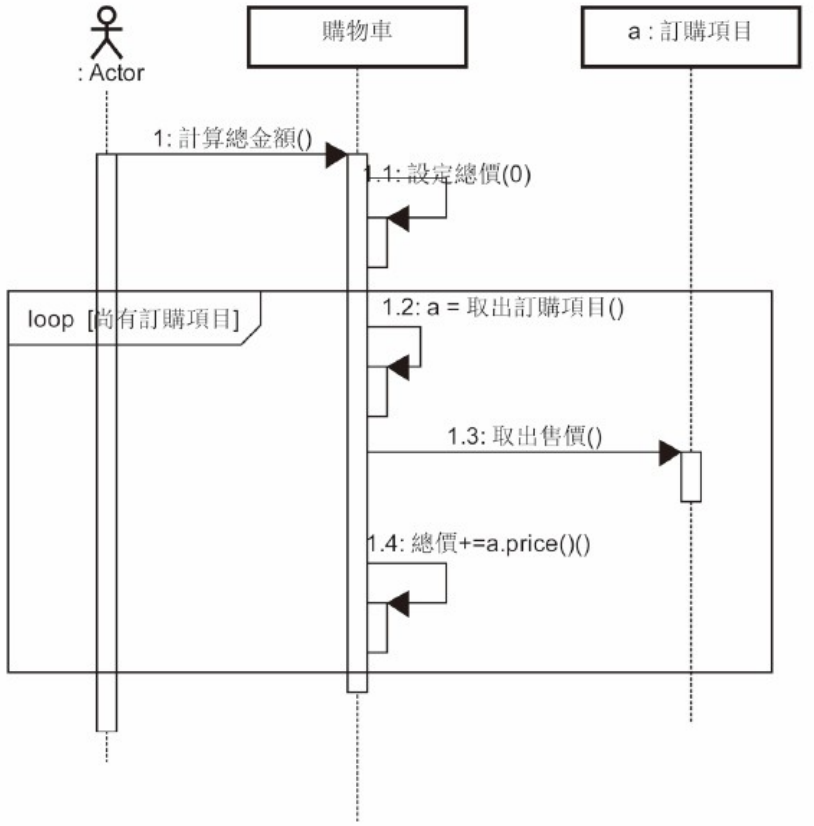
循序圖圖示

- **組合片段： alt**
  - 用於表達多種選擇
  - 條件成立時，才會執行相關的運算元
  - 類似 if...then...else if... 的結構

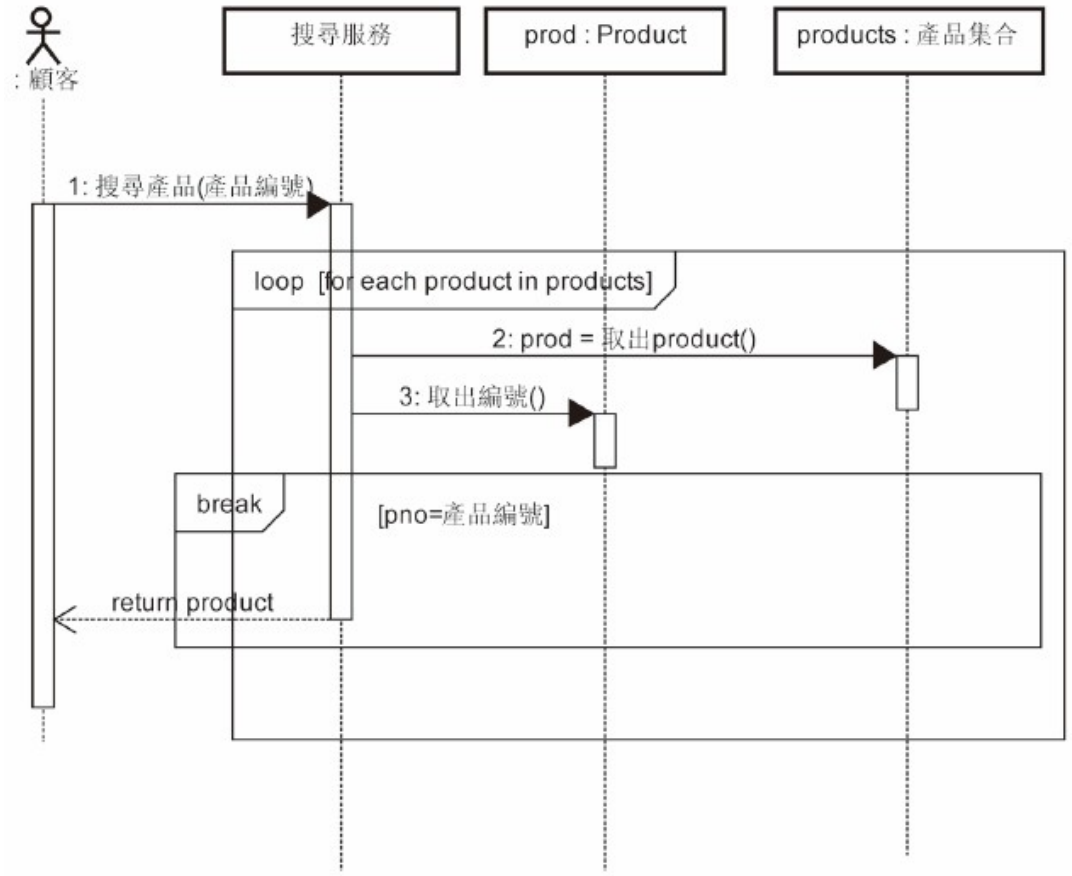


循序圖圖示

- **組合片段： loop**
  - 用於表達迴圈
  - 被包含之互動運算元會被執行許多次
- **組合片段： break**
  - 條件成立後，就跳出迴圈

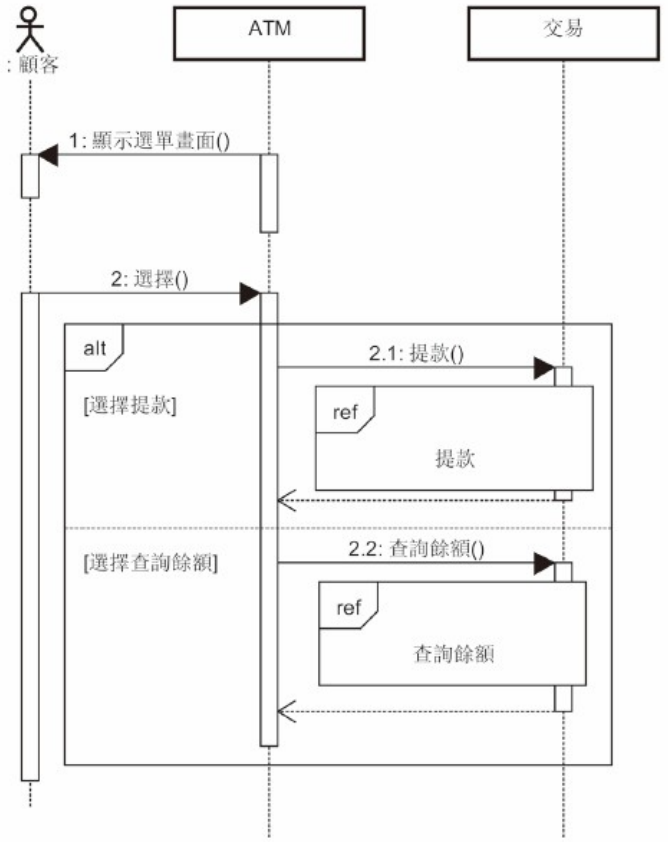


循序圖圖示


: 顧客
搜尋服務
prod : Product
products : 產品集合
1: 搜尋產品(產品編號)
loop [for each product in products]
2: prod = 取出product()
3: 取出編號()
break
[pno=產品編號]
return product

循序圖圖示

- **組合片段： ref**
  - 參照其他的互動
  - 對於複雜的流程步驟，可以將它們分別繪製後，再進行組合



循序圖繪製要領

- 先找出參與者 (Participants)
- 再整理主要流程
  - 事件與訊息
- 進一步細化流程與步驟
  - 拆分與組合

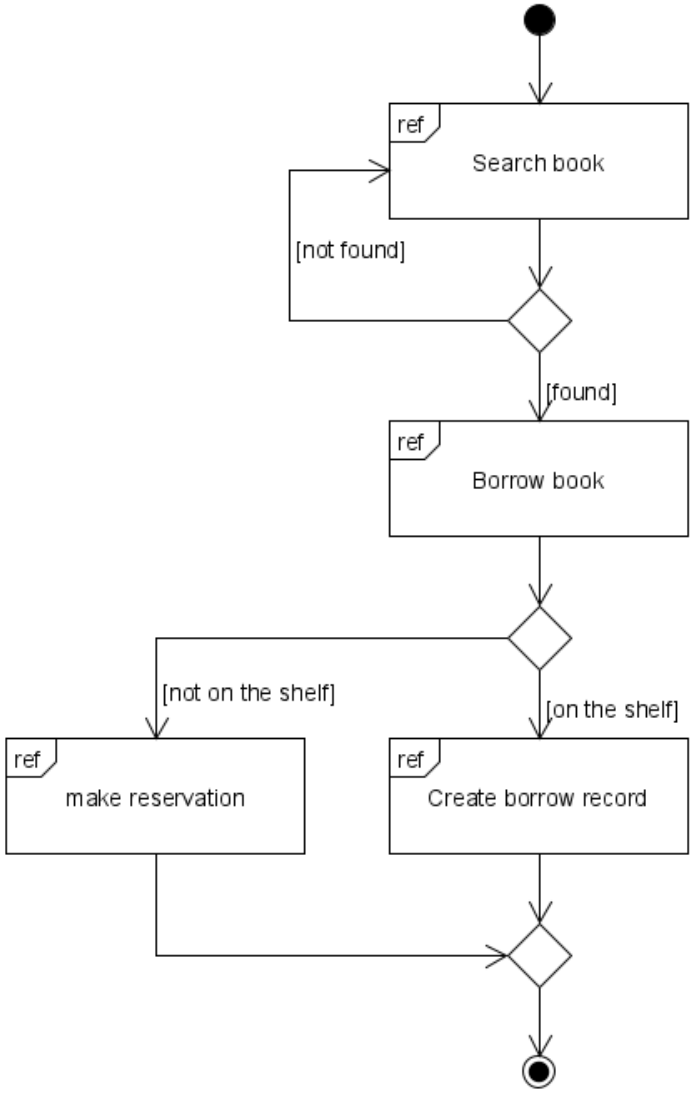
互動概觀圖

互動概觀圖

- 活動圖的一種變形
- 描述高層次的控制流程以及它們之間的互動
  - 除了動作（ Action ）圖形之外，其餘圖示與活動圖相同

互動概觀圖

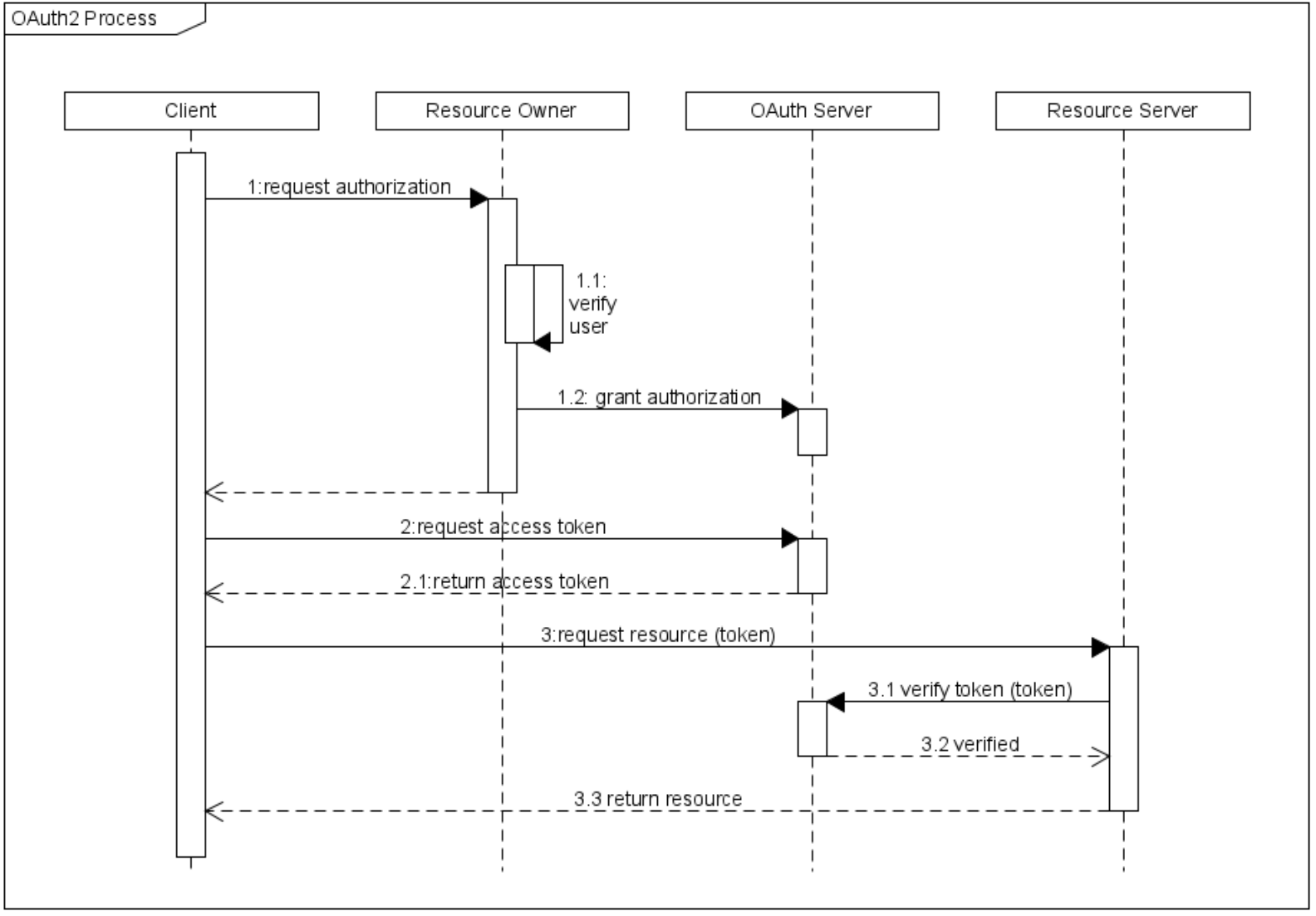
- **範例：圖書館借書**
  - 使用互動概觀圖來描述高層次的作業執行流程



綜合應用

綜合應用

- **範例： OAuth 驗證**
  - 角色
    - Client
    - User
    - Auth Server
    - Resource Server
- **Steps**
  - request authorization
    - verify user (opt.)
    - grant authorization
  - request token
  - request resource
    - verify token


OAuth2 Process
Client
Resource Owner
OAuth Server
Resource Server
1:request authorization
1.1:
verify
user
1.2: grant authorization
2:request access token
2.1:return access token
3:request resource (token)
3.1 verify token (token)
3.2 verified
3.3 return resource