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

Class Diagram

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

AGENDA[1]

- **類別圖簡介**
- **類別圖圖示 – Part I**
  - 類別 Class
    - 屬性 Attribute / Member
    - 操作 Operation / Member Function
- **綜合範例**

AGENDA²

- **類別圖圖示 -Part II**
  - 抽象類別
  - 介面
- **類別關係**
  - 關聯
  - 聚合
  - 一般化
  - 相依
  - 具體化

類別圖簡介

類別圖簡介 1

- 目的
  - 塑模系統的靜態模型
  - 塑模問題領域中的找尋物件
  - 表達系統中物件的靜態資料結構
  - 表達系統中物件間的關係

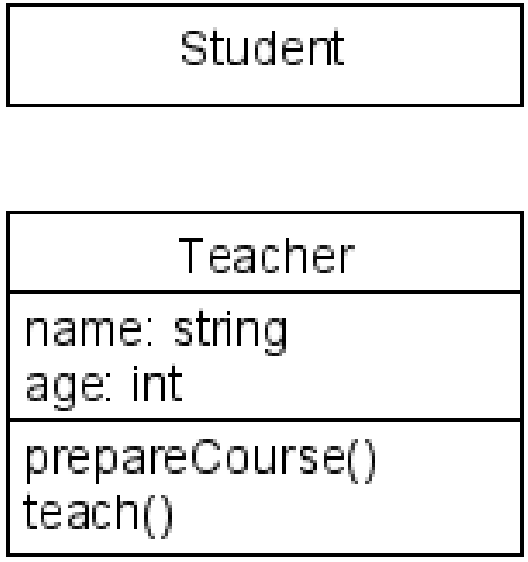
類別圖簡介 2

- **主要做爲物件的資料結構塑模使用**
  - 捕捉問題領域所牽涉的概念
  - 探究各類別之間的關係
- **與結構化分析所使用的實體關係圖 (ERD) 類似**
  - 但是它同時能表達資料與行為
  - 模型與現實物件之間的對應

類別圖圖示
Part I

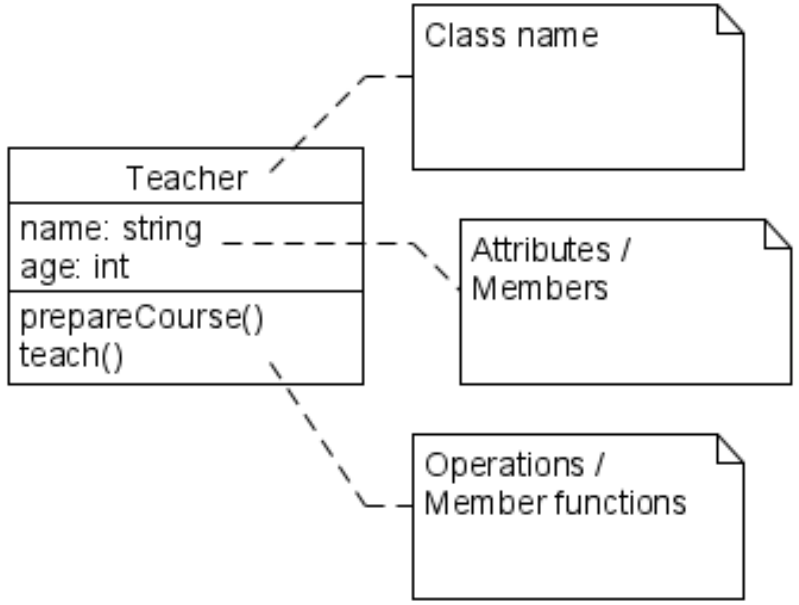
類別圖圖示 [1]

- **類別 (Class)**
  - 最基本的組成元素
  - 用來描述一群具有相同屬性及行為的物件



類別圖圖示 2

- **類別的組成**
  - 類別名稱
  - 類別屬性 (Member)
    - 本質上的特徵或性質
  - 類別行為 (Member Function)
    - 具有的行為或是能力



類別圖圖示 3

- **類別的分類**
  - 依據現實世界中所代表的涵義
    - 跟問題領域相關的類別稱為領域問題類別
    - 用來處理商務邏輯 (business logic) 的類別
    - 跟使用者介面相關的介面類別
    - 用來做資料存取的資料存取類別
    - 抽象化概念

類別圖圖示 4

- **領域問題類別**
  - 在分析階段中，基本上我們只對牽涉到領域問題的類別感興趣
  - 指的是那些在討論的領域中重要的名詞概念
    - 有些需要儲存，有些不需要
  - 必須儲存起來的名詞概念有一個特別的稱呼為永續類別（ Persistent Class ）
    - 資料會被儲存，以做為後續的存取之用

類別圖圖示 5

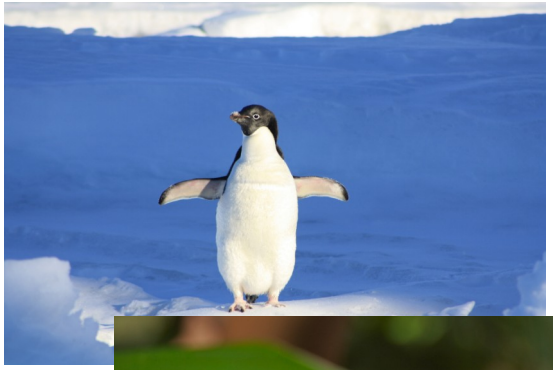
- **範例：購物系統**
  - 顧客是領域中重要的概念
    - 我們希望將其資料儲存起來以供後續之用 (使用 <<presistent>>)
    - 利用類別來捕捉這個重要的名詞概念

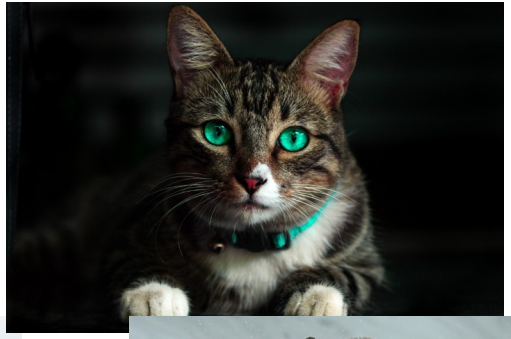
類別圖圖示 6

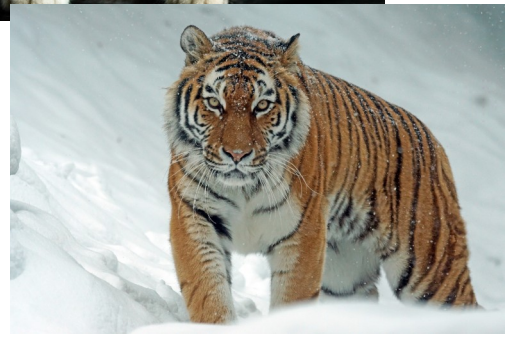
- **類別與實例**
  - 類別（ Class ）所代表的是一個抽象層次的概念
  - 「類別」與「物件」這兩個詞彙常常被互相混用，但它們所描述的概念並不一樣
  - 所謂的實例（ Instance ）代表著一個類別的實現化

類別圖圖示 7

還記得這些圖嗎？類別與實例…







類別圖圖示 8

- **實現化也就是具體化的意思**
  - 所謂的 instance 就是利用類別所建構出來的物件
  - 以蓋房子為例
    - 房子的藍圖就好比如是類別
    - 依據藍圖所蓋出來的房子就是實例
- **類別是抽象的，實例是具體的**
  - 由類別所建構出來的實例稱為物件 (Object)

類別圖圖示 9

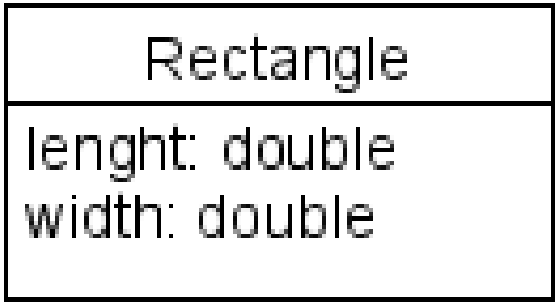
- **屬性**
  - 類別所具有的性質、特徵，或是狀態
  - 又稱為「 Member 」
  - 在分析的階段中，只要是對於問題領域是重要的屬性，就需要被包含在類別中
  - 例如，前述的購物系統中，與顧客有關的屬性...
    - 姓名、聯絡電話、寄件地址...

類別圖圖示 10

- **範例：形狀的類別**
  - 設計一個繪圖軟體， 它可以繪製長方形（ Rectangle ），那麼我們可以定義一個類別就叫做長方形
    - 任意一個長方形，它所擁有的性質諸如面積，周長皆可由長（ Length ）跟寬（ Width ）來決定
  - 長跟寬均可算是長方形類別的屬性，不同的長或是寬代表的是不同的長方形物件，也就是長方形類別的實例

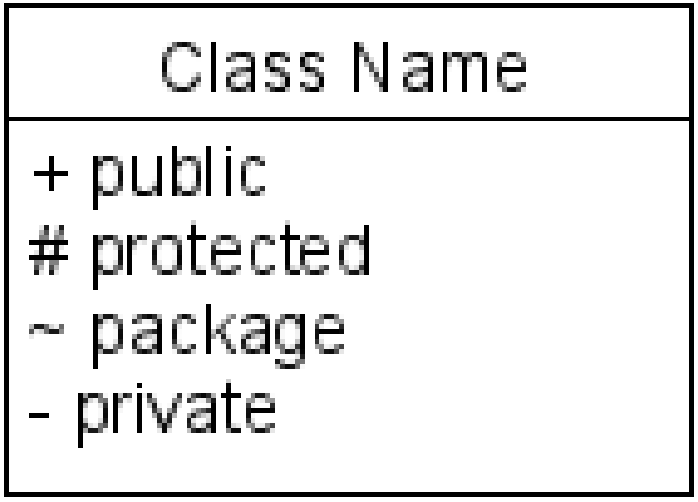
類別圖圖示 11

- 範例：形狀的類別



類別圖圖示 12

- **屬性的能見度**
  - 物件導向所提供的封裝機制
  - 主要是用來保護一個類別的屬性，使得類別的屬性不會被任意修改
  - 會與程式語言的支援有關

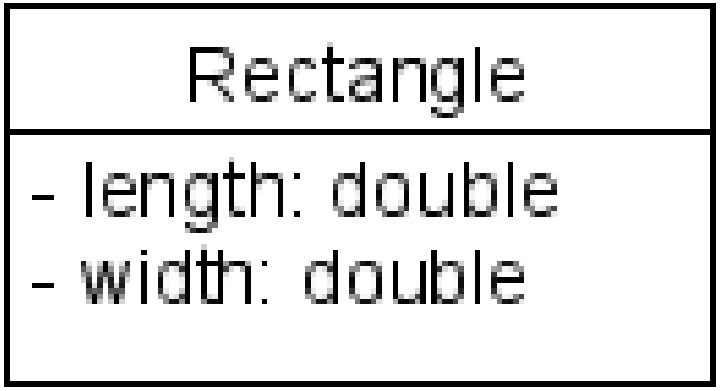


類別圖圖示 13

- **屬性的能見度**
  - Public
    - 所有的物件都可以存取與設定它的值
    - 在屬性的前面用一個加號 (+) 號表示
  - Protected
    - 只有子類別可以存取與設定
    - 們在屬性的前面用一個井號 (#) 號表示
  - Package
    - 只有相同類別庫之類別可以存取與設定它的值
    - 在屬性的前面用一個取代符號 (~) 號表示
  - Private
    - 該屬性只屬於它自己，任何其他的類別都不可以存取它
    - 在屬性的前面用一個減號 (-) 表示

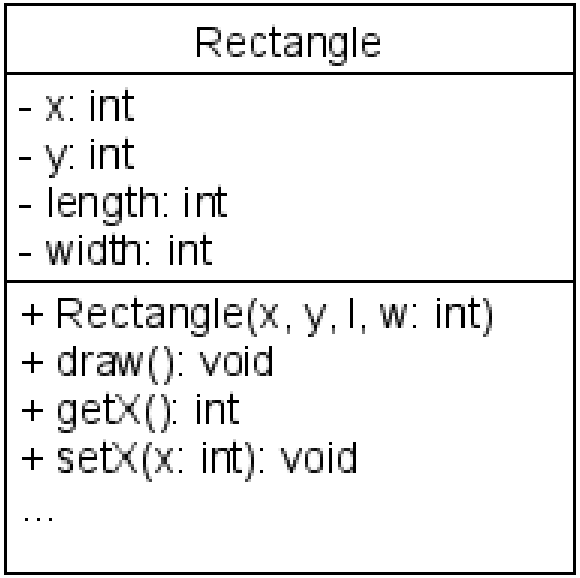
類別圖圖示 14

- **屬性的資料型態**
  - 屬性都有特定的資料型態
    - 基本資料型態
    - 類別型態
  - 與使用的程式語言相關



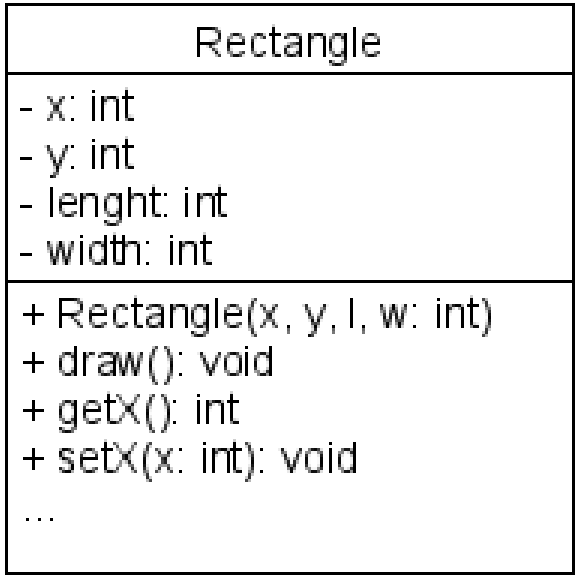
類別圖圖示 15

- **操作 (Operation)**
  - 類別所具有的行為或是能力
  - 又稱為「 Member Function 」
  - 是類別對外界刺激的回應



類別圖圖示 16

- **操作的基本型態**
  - Constructor
    - 建構 / 初始化一個實例 / 物件
  - Getter
    - 取得一個類別的屬性值
  - Setter
    - 更新一個類別的屬性值
- **與領域相關的行為**



類別圖圖示 17

- **操作可以宣告能見度**
  - 與使用的語言相關
- **分析階段我們只寫出敘述性的操作**
  - 要儘量避免去描述操作的細部邏輯
  - 設計階段時，敘述性的操作還會再次地被分解與細化
  - 不需要描繪所有的操作
    - 只描繪出與目前檢視的使用案例相關的操作
    - 逐步加入操作

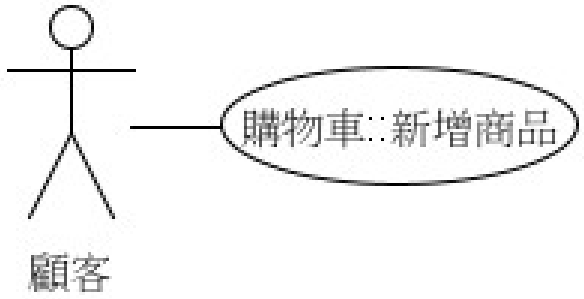
綜合範例

綜合範例 1

- **購物網站 - 新增商品至購物車**
  - REQ-A001 ：作為顧客，我想把商品加至購物車，以便我可以完成選購商品

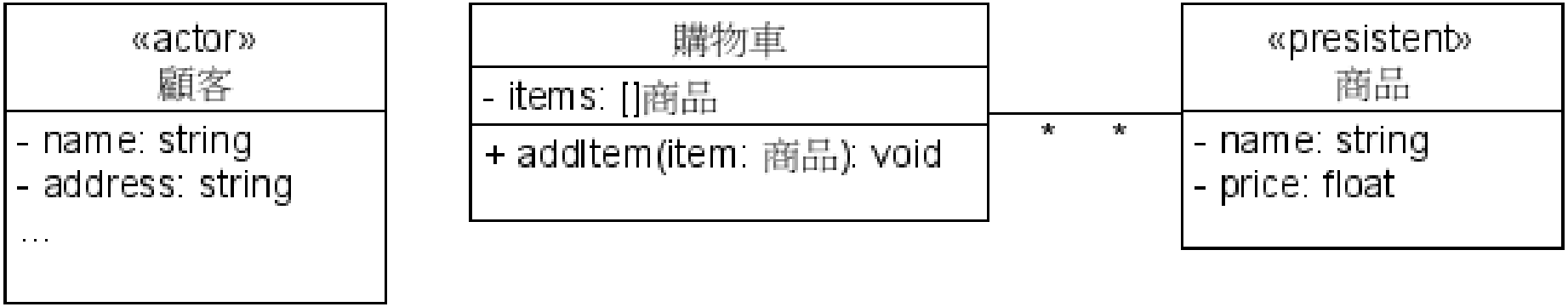
綜合範例 2

- **購物網站 - 新增商品至購物車**
  - Step I ：繪製 Use Case Diagram



綜合範例 3

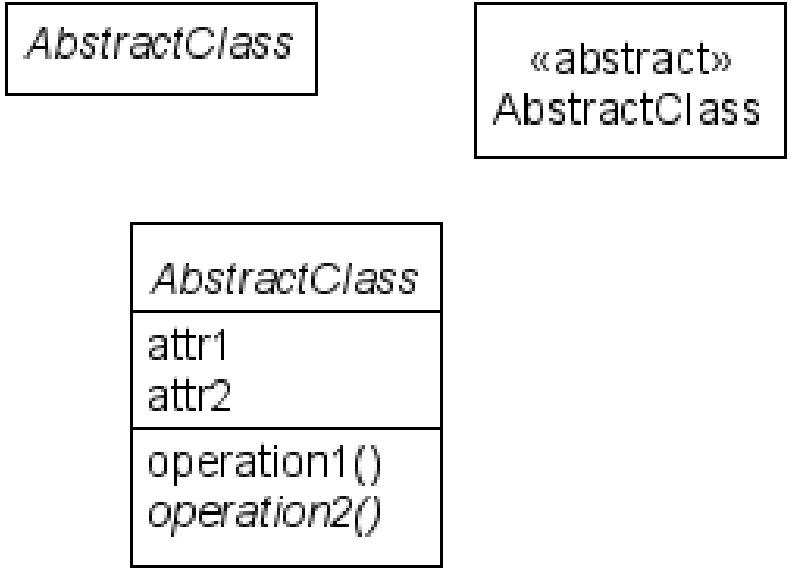
- **購物網站 - 新增商品至購物車**
  - Step II ：找出其中的類別



類別圖圖示
Part II

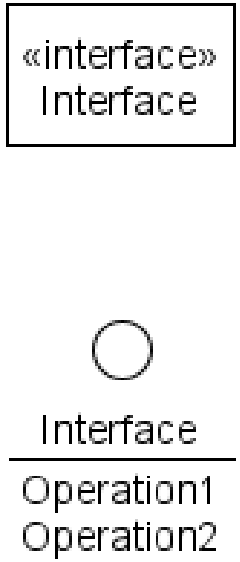
類別圖圖示 [1]

- **抽象類別 (Abstract Class)**
  - 只存在特定的程式語言中 (Java, C++)
  - 類別中宣告了抽象方法，但不實作它
  - 使用 «abstract»
  - 無法建立 instance



類別圖圖示 2

- **介面 (Interface)**
  - 類似抽象類別，但只宣告抽象方法
  - 規範了方法能被外部使用的規則
  - 利用介面將實作分離，解偶類別間的相依性
  - 使用 «interface»



類別圖示 3

- **介面 (Interface)**
  - 以 USB 介面為例
    - 現今大部份的電腦設備都提供的介面規格
    - 只要外接的設備符合 USB 介面的規格，大都能插上就使用
    - 不需要管這些設備的內容設計為何

類別關係

類別關係 1

- **類別圖顯示…**
  - 領域問題中出現的類別
  - 類別與類別之間的關係
- **對於領域中的問題，會藉由許多的物件一起合作以提供解答**
- **塑模類別之間的關係，是類別圖很重要的工作**

類別關係 2

- **物件導向世界中的關係**
  - 在物件導向的世界中，我們可以由幾個觀點來看物件之間的關係：
    - 從類別的角度
    - 從物件的角度來看

物件導向世界中的關係 1

- **從類別角度**
  - 在類別中，有一些關係是用來表達結構上類別與類別之間的關係
    - 類別繼承 (inheritance) 的關係
    - 介面 (interface) 實現化的關係
  - 這一類的關係你可以把它看作是固定的，不會因時間的改變而有所改變的關係
    - 例如，父子關係不會因為成家立業搬出去住了而有所改變

物件導向世界中的關係 2

- **從物件的角度**
  - 也就是說物件們是利用什麼關係來一起合作的
  - 在本質上，物件之間的關係並沒有如上所述那種無法改變的概念
    - 與學校之間建立了學生的關係，畢了業就消失了
    - 可能是組織社團的一員，跟組織社團就有會員的關係
  - 在物件導向的世界中，我們稱這種關係為關聯關係 (association)

物件導向世界中的關係 3

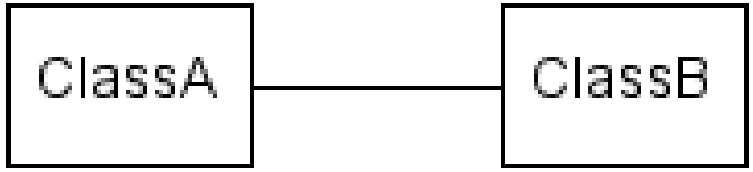
- **現實世界中物件與物件的關係相當的多而且很複雜**
  - 不會只有上述概括的幾種而已
  - 如果實在很難將關係歸納於哪一種類的話，不要覺得奇怪
- **關係的表示法是在類別與類別之間用實線或是虛線連結起來，再配以不同形式的箭頭**
- **對於實線的表示法，在線的上面可以寫下此關係的名字，或是在連結線的兩端寫上類別在關係中所代表的角色**

關聯 Association[1]

- 物件導向的世界中，物件與物件彼此互相傳遞訊息來完成工作，猶如現實生活中的情境
- 關聯關係所代表建立類別之間，彼此互通訊息的管道
- 這個管道可以讓物件之間互相傳遞訊息，完成所需完成的工作

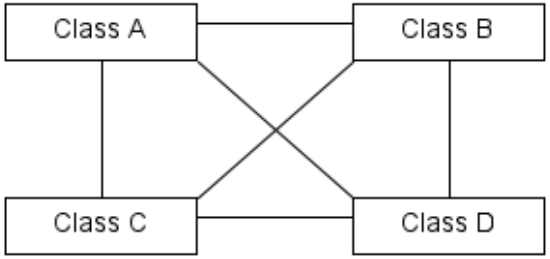
關聯 Association[2]

- 關聯關係代表著類別之間結構上的連結
- 在參與該關係的兩方之間，使用一條實線來連結



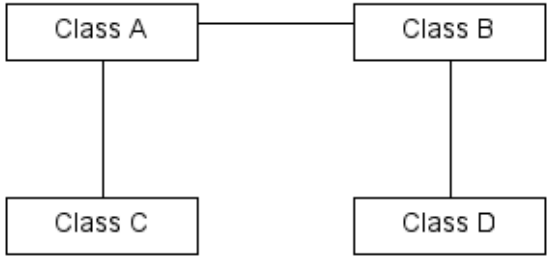
關聯 Association[3]

關聯緊密



最好避免

關聯鬆散



關聯 Association[4]

- **ClassC 被刪除時會造成什麼樣的影響呢？**
  - 右圖中只有類別 B 與類別 C 之間的關聯消失了
  - 左圖顯示類別 A 與類別 C ，類別 B 與類別 C ，類別 D 與類別 C 之間的關聯都消失了
    - 右圖所帶來的破壞力比左圖來的大
    - 類別間相互關聯，具有高耦合度 ← **儘量避免**

關聯 Association[5]

- **依其結合的程度（Degree of Coupling）而分爲不同形式的關聯關係。所謂的結合度是用來量度兩個物件（類別）之間的依賴程度，又稱爲耦合度**
  - 當參與關係的兩個物件彼此必須依靠對方，無法單獨存在，稱為高耦合度
  - 一個物件可以獨立存在，但是如果沒有另一個物件的幫忙會無法完成「某些」事項，這些物件具有低耦合度
  - 兩個物件互不溝通，也沒有必要分享資料，這些物件是零耦合度

關聯 Association[6]

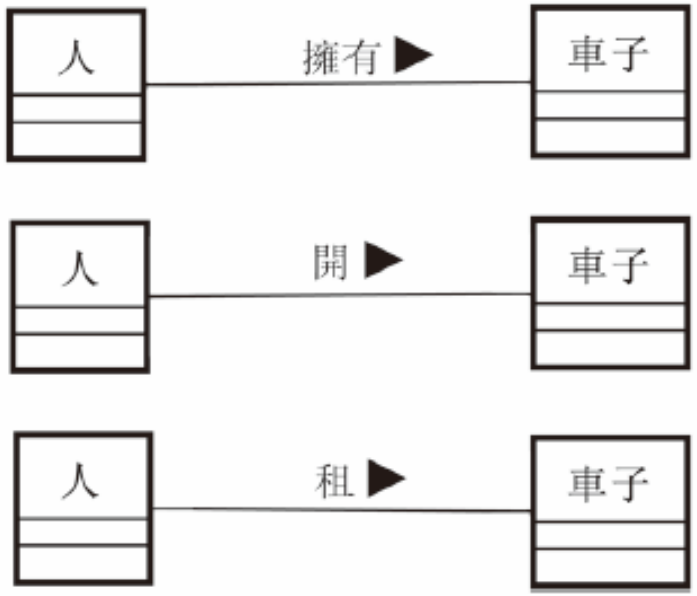
- **與耦合度相關的觀念稱爲凝聚力**
  - 或稱為**內聚力**
  - 由機能相關的程式組合成一個模組
  - 高凝聚力的模組，代碼的可讀性會提高，可再利用的可能性也會增加
  - 會影響到系統的設計，以及將來的維護和功能擴充

關聯 Association[7]

- **關係的名稱**
  - 可以在關係上加上名稱
  - 語意上可以恰當地表達該關係的解釋，才不至於造成閱讀者的混淆以及疑惑
  - 關係很明確的時候可以被省略

關聯 Association[8]

- 關係的名稱



關聯 Association[9]

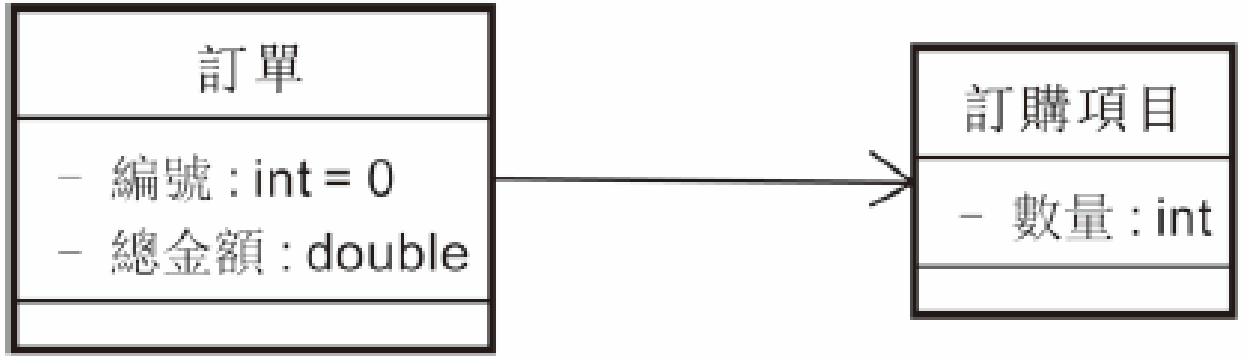
- **範例：課程開設**
  - 學生「修課」表示了「學生」與「課程」之間的關係
  - 對於課程，我們還有「老師教授課程」的關係

關聯 Association[10]

- **關係的互通性**
  - 沒有箭頭的直線代表著彼此雙方都知道對方的存在，彼此雙方都可以互通訊息
  - 假如希望表達訊息的傳遞是單向的，可以用只帶有單方向箭頭的直線來繪製此概念

關聯 Association[11]

- **關係的互通性**
  - 例如，一個訂單可以有許多訂購項目
  - 但訂購項目不需要知道它是屬於哪個訂單



關聯 Association[12]

- **角色名稱**
  - 在關聯關係中，我們可以給定參與物件在關係中所扮演的角色（ Role ）
  - 一個航空班機資訊系統，我們可能有的兩個類別 為航班和飛機，飛機被指定某個航班，而一個航班也有指定的飛機

關聯 Association[13]

- **多重性**
  - 在關係連線的兩端，用來表示參與此關係之物件的數量

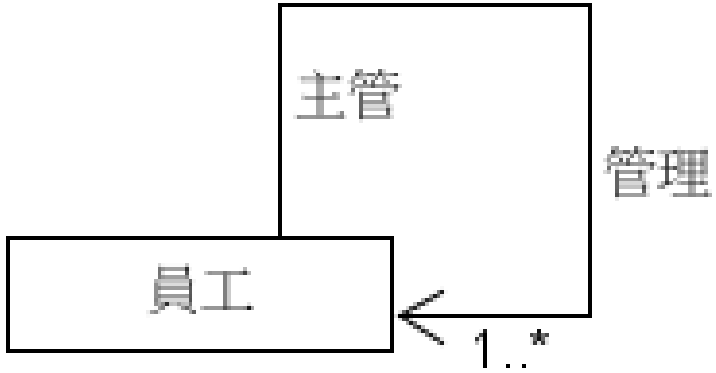
名稱	表示法	範例
恰好一個	1	一個課程有一位老師
零或一個	0..1	
零或多個	0..*	課程有零或多個學生選修
一或多個	1..*	學生主修一或多個學位
指定範圍	m..n	

關聯 Association[14]

- **範例：課程開設**
  - 一位老師可以開設一到多門課程，一門課程需要一位老師
  - 學生可以選修一到五門課程，一間課程可以有零到多位學生選修

關聯 Association[15]

- **反身關聯 (reflexive association)**
  - 反身關聯指的是相同類別中的物件之間彼此關係
  - 例如，一位主管管理多位操作員。主管以及操作員都是由員工類別來表示



關聯 Association[16]

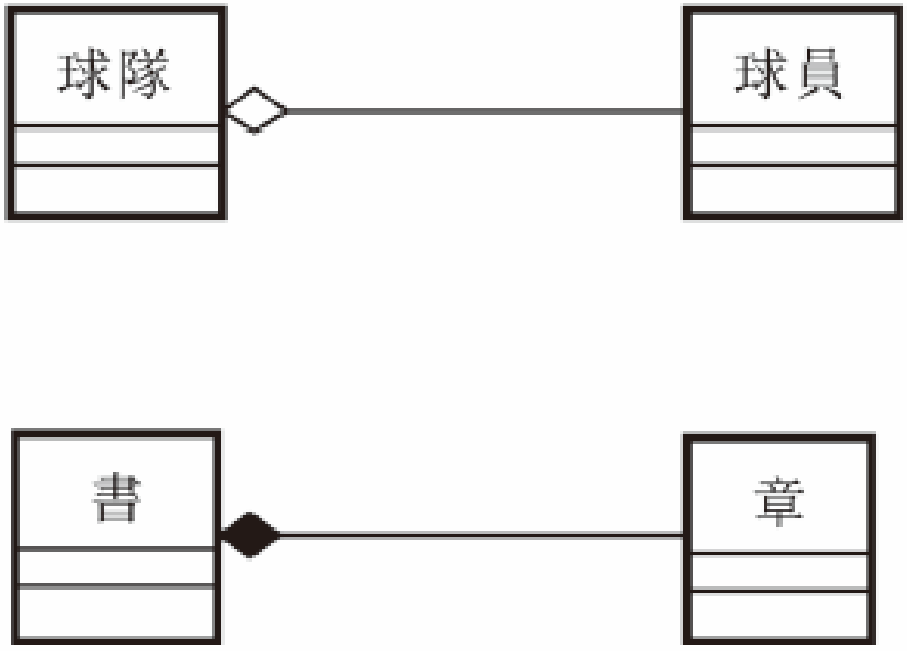
- **聚合 Aggregation**
  - 一種特殊的關聯關係，用以表達「整體和部份」的概念
  - 物件被組合起來以形成一個新的更複雜的物件
  - 整體消失時，和它有關的物件還是可能會繼續存在
  - 聚合關係以一個空的菱形來表示代表整體的一方

關聯 Association[17]

- **組合 Composition**
  - 是一種比聚合關係更強的包含關係
  - 如果整體不存在，那與這個整體相關的其他部份也會跟著消失
  - 在整理端以實心菱形代表

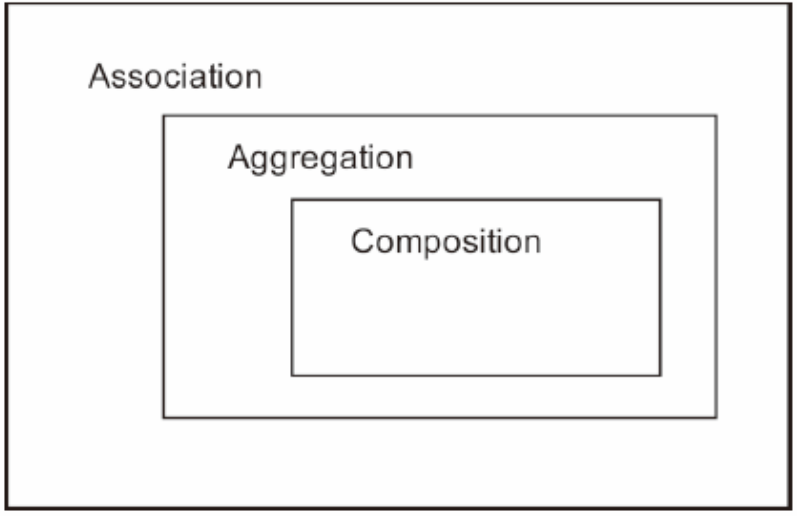
關聯 Association[18]

- **聚合與組合**
  - 球隊沒了，球員還會在嗎？可否去他隊打球？
  - 書本沒了，章節還能獨立存在嗎？



關聯 Association[19]

- **關係的强弱程度**
  - 越是核心，强度越大

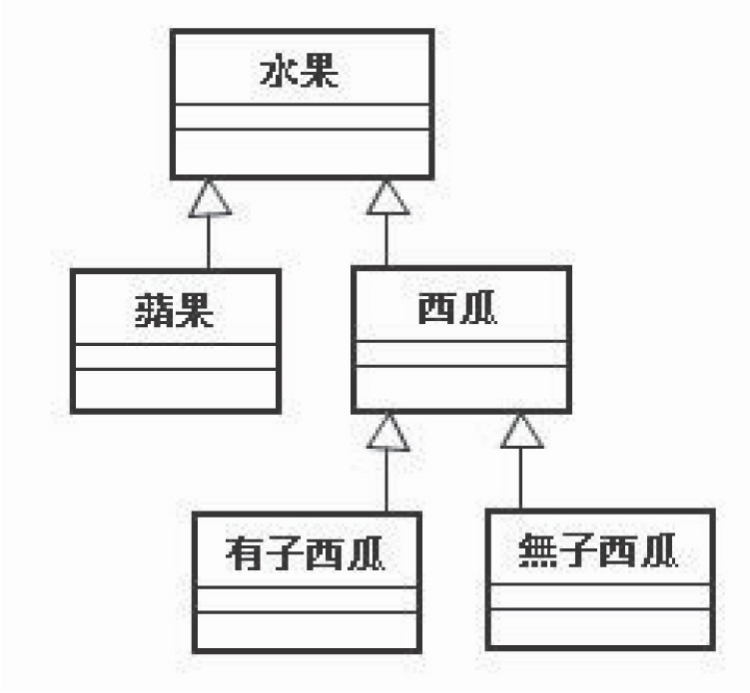


關聯 Association[20]

- 一般化 Generalization
  - 是一種分類的關係
  - 子類除了有父類的特色外（繼承），可能還具備了一些父類沒有的元素
  - 簡單的說「物件 A 是物件 B 的一種」
  - 可以降低模型的複雜度

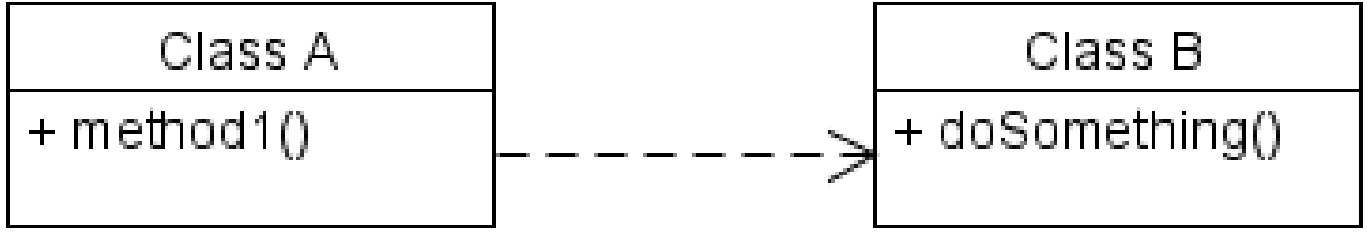
關聯 Association[21]

- **範例：水果的一般化**
  - 蘋果、西瓜怎麼分？



關聯 Association[22]

- **相依 Dependency**
  - 是一種依賴關係
  - 一個類別使用到其他類別提供的服務時
    - ClassA.method1 calls ClassB.doSomething()



關聯 Association[23]

- **具體化 Realization**
  - 用來表達一個類別之行為是由另一類別來描述定義
  - 被具體化的類別型態一定是介面（ Interface ）型態，它是一種特殊的類別型態
  - 表示法為一條虛線，以空心三角形當箭頭，指向被具體化的類別