
分析類別、循序、設計 類別圖介紹



什麼是分析類別圖？

是 UML 的一種，他透過一個系統中的物件、物件的屬性、物件擁有的方法和物件與物件之間的關係來描述其結構。

類別圖符號

類別圖的符號分為兩大類

- 描述物件本身
- 描述物件與物件的關係

類別圖(Class Diagram)：

- 特色：用於描述系統中的類別以及它們之間的關係和結構。
- 類別圖通常顯示類別之間的靜態關係，例如聚合、繼承、關聯等。
- 可以顯示類別的屬性和方法。
- 適用場景：用於分析和設計系統的靜態結構。


設計類別圖

其主要目的是描述系統的設計結構。與分析類別圖類似，設計類別圖也使用類別、屬性、方法等元素。

設計類別圖通常在系統設計階段創建，用於指導開發人員實現系統的組件和模組。

設計類別圖不僅提供了對系統架構的視覺化表示，還可以作為軟體設計文檔的一部分，幫助團隊成員理解系統的結構和設計思路。

物件 Object

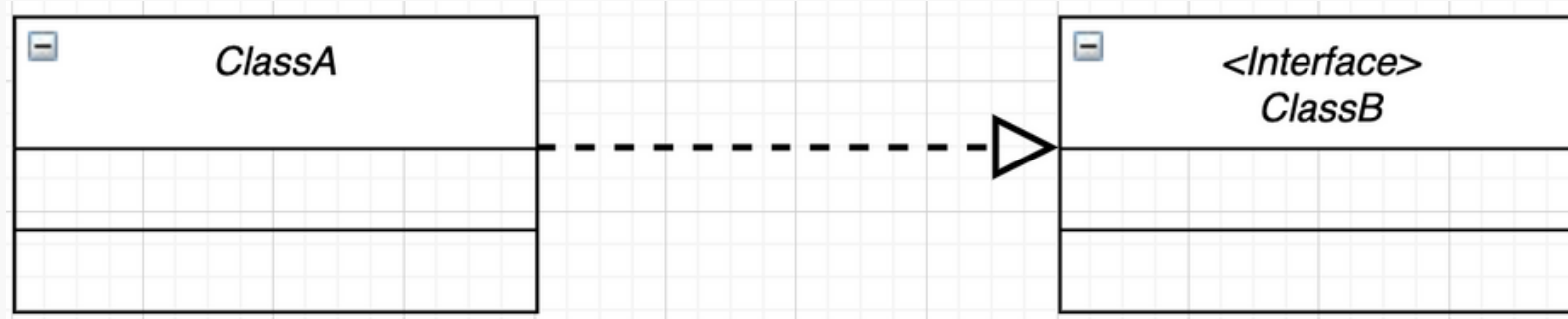
 <i>ClassName</i>
+ attributes1: int
- attributes2: float
attributes3: string
+ method1: string
- method2(float): array<int>
method3(int, string): int

總共分為三大格

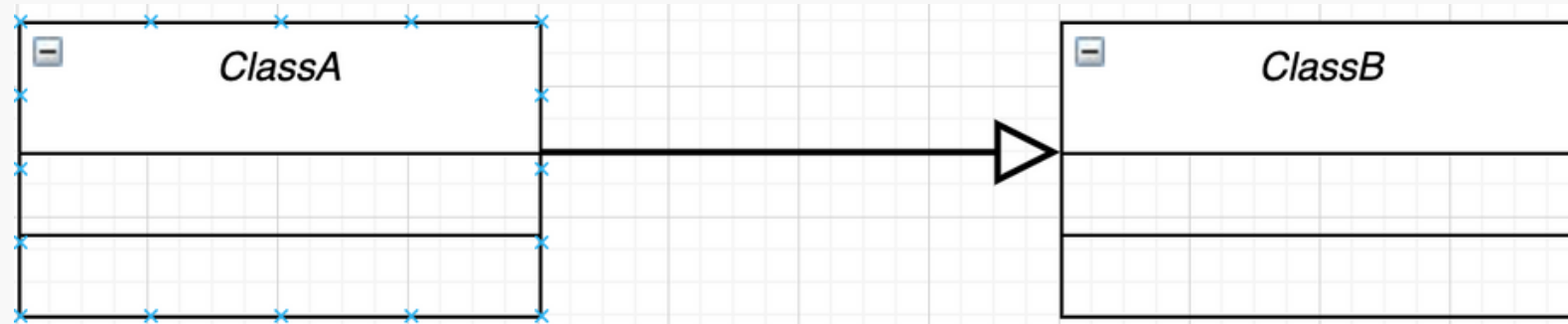
- I. 物件名字
2. 物件屬性
 - 前面為屬性名字
 - 冒號後面為屬性型別
- 3 物件方法
 - 前面為物件名字
 - 括號內為參數和參數型別
 - 冒號後面為回傳值型別

關係 Relation

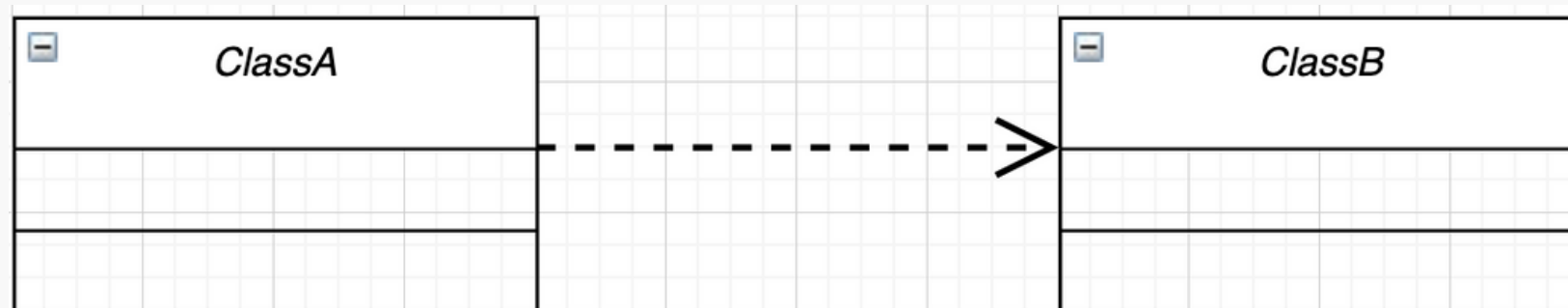
Realization



Generalization



Dependency



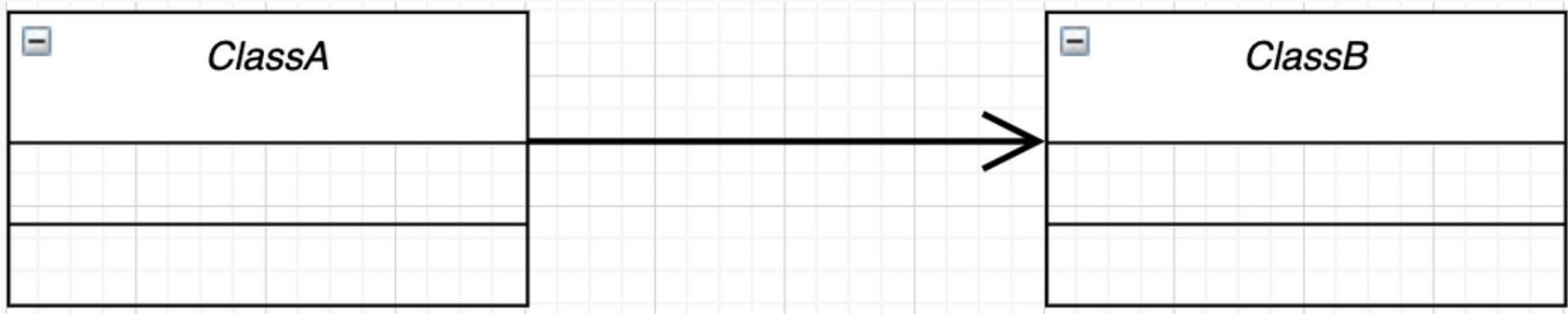
- A implement B , **A 實作 B 為介面**

- A extend B, A **繼承** B
- B 為父類別

- A references B, A **使用** B
- A **在參數或回傳時有用到** B

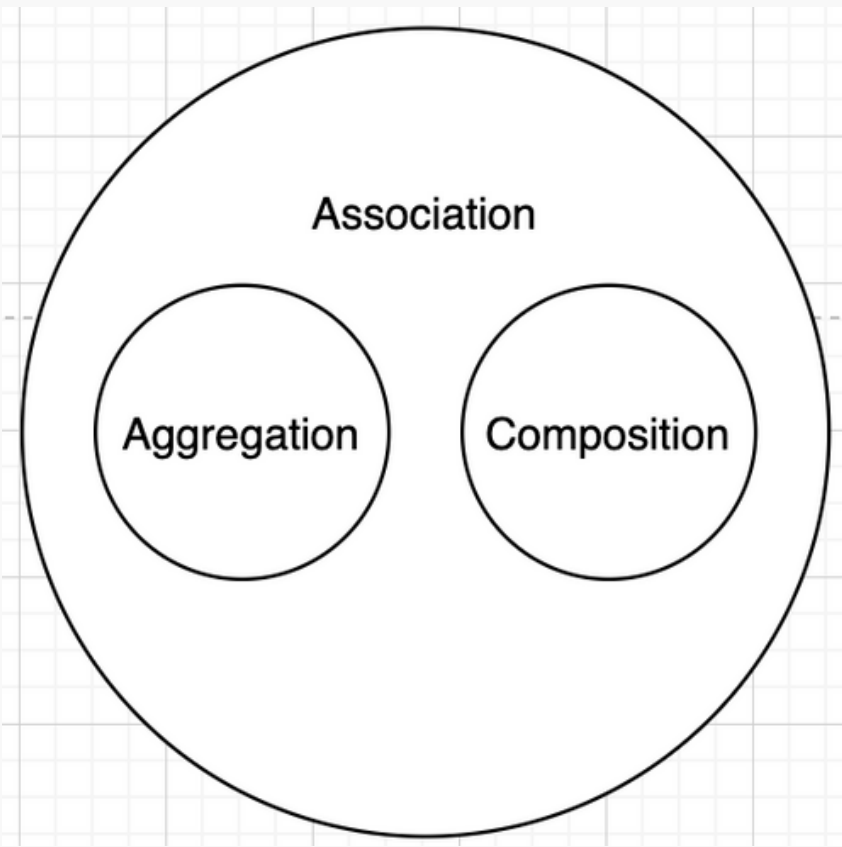
關係 Relation

Association

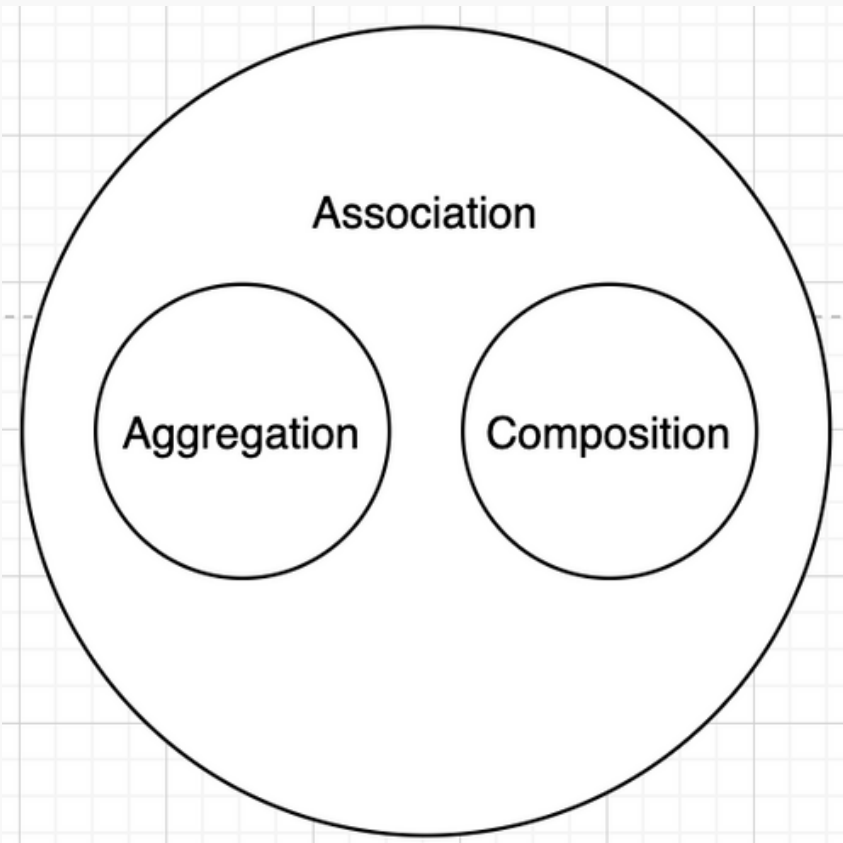


- A has-a B object, A 擁有 B
- B 為 A 擁有的變數
-

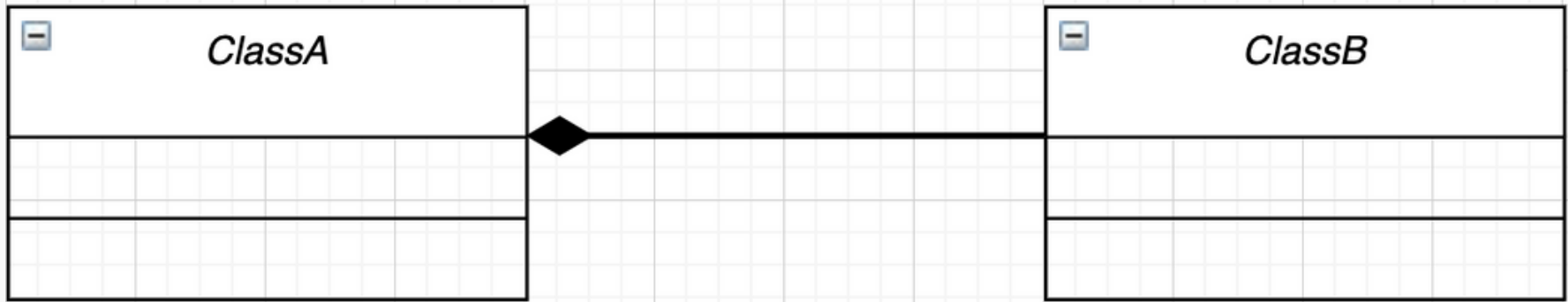
Aggregation 和 Composition 為 Association 的一種特例



關係 Relation



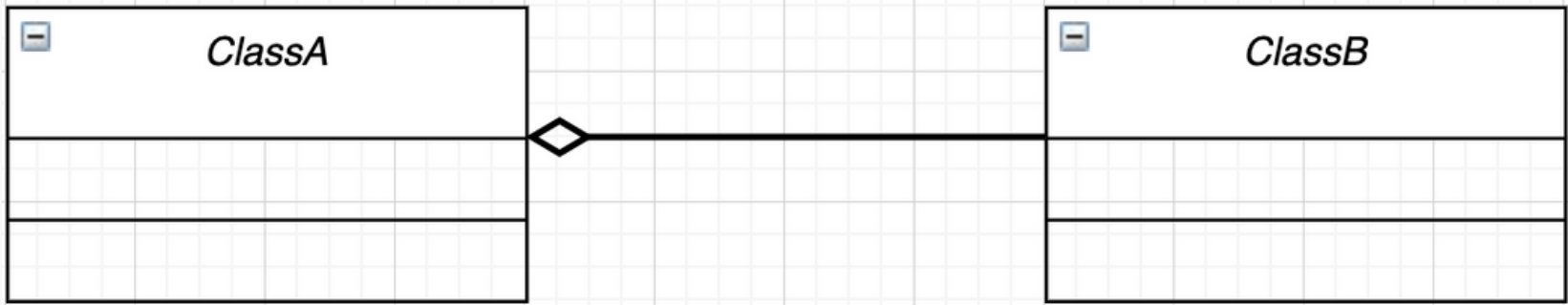
Composition



Aggregation

A 是由 B 組合而成，且為弱關係
A 和 B 擁有自己的獨立的生命週期，B 可單獨存在

舉例
訂單 擁有 商品
商品可以獨立存在



- A 是由 B 組合而成，且為強關係
- B 只要離開 A 便不具意義，無法單獨存在，且生命週期與 A 一樣

-
-

舉例
人 擁有 手、腳、頭
手、腳、頭獨立存在時無意義

名稱	相依關係 (Dependency)	結合關係 (Associations)	聚合關係 (Aggregation)	組合關係 (Composition)	一般關係 (Generalization)
符號	◄.....	————	◊——	◼——	◄——

愈弱的類別關係

○ ○ ○

愈強的類別關係

一般關係
繼承關係
箭頭指向 父類別

相依關係

- 通常用在方法參數、回傳值
- 箭頭指向 被使用者

結合關係







- 通常用在屬性、全域變數
- 箭頭指向 被擁有者

聚合關係

- 聚合，弱 整體與部分的關係
- 整體可以脫離部分而單獨存在
- 整體與部分具有各自的生命周期
- 菱形指向 整體

組合關係

組合，強 整體與部分的關係
整體不可脫離部分而存在
整體的生命周期結束也就意味著部分的生命周期結束
菱形指向 整體

	概念			符號
Dependency	A uses a B	使用	參數、回傳	A  B
Association	A has a B	有	全域變數	A  B
Aggregation	A owns a B	擁有	整體、部分	A  B
Composition	B is a part of A	一部分	整體、部分	A  B
Realization	B implements A	實作	介面	B  A
Generalization	B extends A	繼承	父子類別	B  A

I. 分析類別圖 (Analysis Class Diagram) :

- 在需求階段，用於建立領域模型 (Domain Model) 。
- 重點在於識別類別的職責 (Responsibilities) 。
- 屬性和操作的具體實現通常不在此階段決定。
- 用於理解系統需求和領域知識。

2. 設計類別圖 (Design Class Diagram) :

- 在設計階段，用於建立設計模式 (Design Model) 。
- 關注具體的實現細節，包括屬性、操作和能見度 (Visibility) 。
- 指定類別之間的關係，例如繼承、關聯、聚合等。
- 用於實現系統的具體設計。

I. 分析類別圖：

- 主要關注系統需求和用例。
- 描述系統的問題領域，包括所涉及的實體、其屬性和行為。
- 通常在系統分析階段創建，用於了解和溝通系統需求，並與利害關係人討論。
- 通常不包含具體的實現細節，著重於系統的概念模型。

2. 設計類別圖：

- 著重於系統的實現細節和設計決策。
- 包含更多與軟體實現相關的細節，例如方法、關聯、介面等。
- 基於分析類別圖，並將其轉化為可實現的程式碼或軟體組件。
- 通常在系統設計階段創建，以支援開發人員進程式碼的實現。

什麼是循序圖？

循序圖是普遍被開發者使用的工具，在單例使用者和物件的互動中做建模。它們說明系統不同的部分如何進行互動以及如何執行功能，以及執行特定案例中的發生的前後順序。

循序圖符號

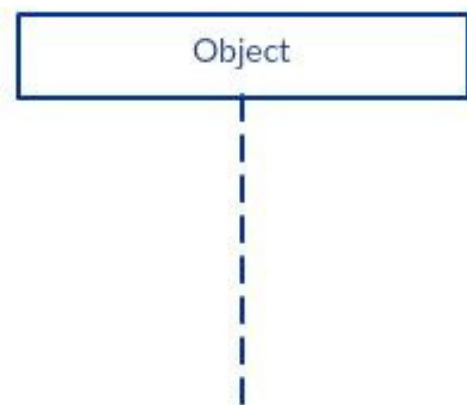
時間軸可以代表循序圖的結構方式，該時間軸從頂部開始並逐漸下降以標記互動的順序，每一個物件擁有一個列，它們之間交互的訊息用箭頭表示。

循序圖(Sequence Diagram)：

- 特色：用於描述系統中的物件之間的動態交互。
- 循序圖顯示物件之間的消息傳遞順序。
- 可以顯示不同物件之間的互動模式和訊息流。
- 適用場景：用於描述系統的動態行為，尤其是在物件之間有訊息交換的情況下。
- 通常用於詳細設計階段，幫助開發人員理解和實現系統的動態行為。
- 可以用於測試和驗證系統的設計，確保系統的交互遵循所需的邏輯。

循序圖各個部分的快速概述:

生命線符號(Lifeline Notation) :



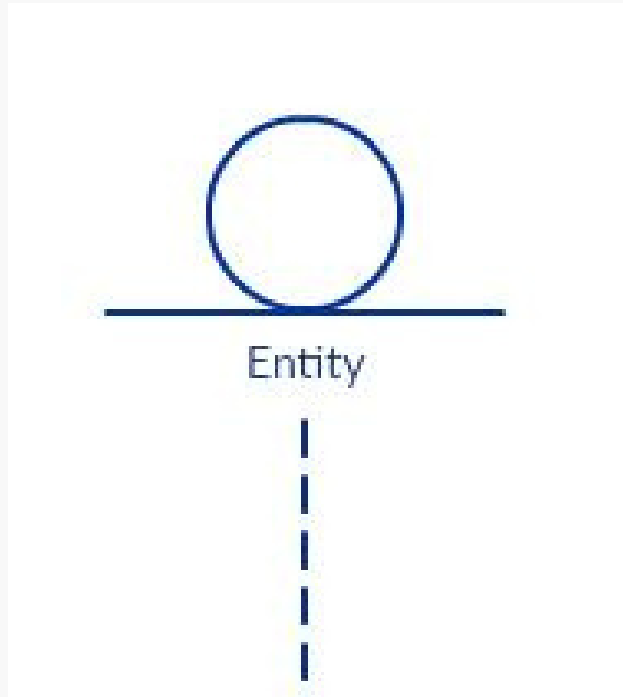
一個循序圖是由數個生命線符號所組成且應該在頂部做水平排列。不可以互相重疊。

Actor(角色)元素:



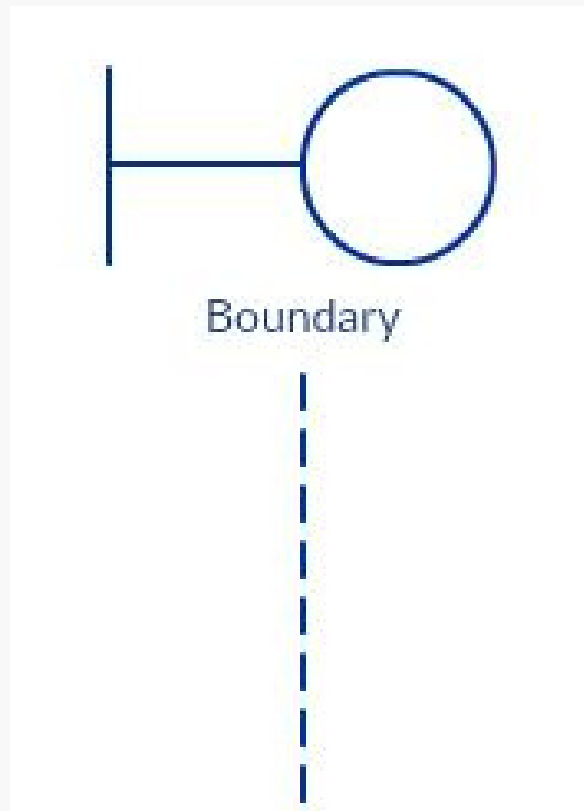
角色元素是對象的一種特殊形式，用於表示參與系統互動的角色或者外部實體

Entity(實體)元素:



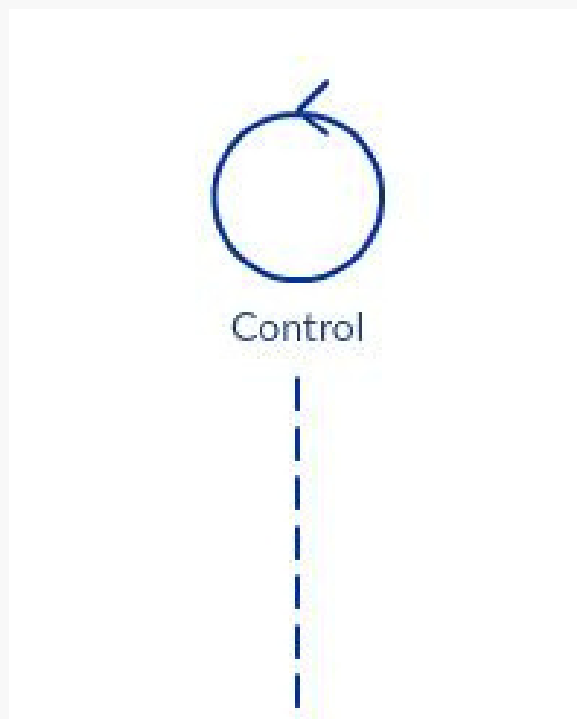
帶有entity(實體) 元素的生命線代表系統資料，例如:在一個顧客服務Application，客戶實體將管理和客戶相關的全部資料。

Boundary(邊界)元素:



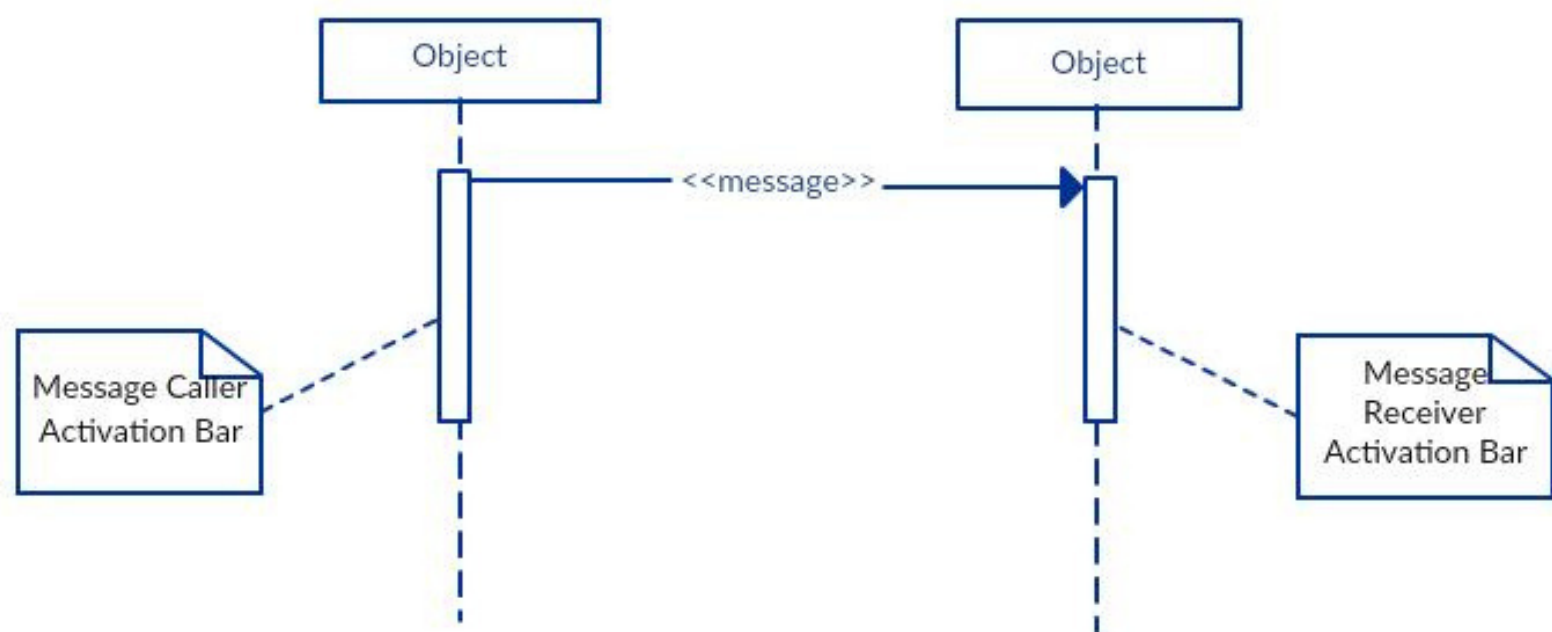
Boundary元素可以用於定義系統的邊界，有助於區分內部系統元素和外部實體之間的界面和互動方式。

Control(控制)元素:



帶有控制(Control)元素的生命線表示控制實體或管理者。它組織和安排邊界和實體之間的互動，並充當它們之間的中介。

活動條(Activation Bars)



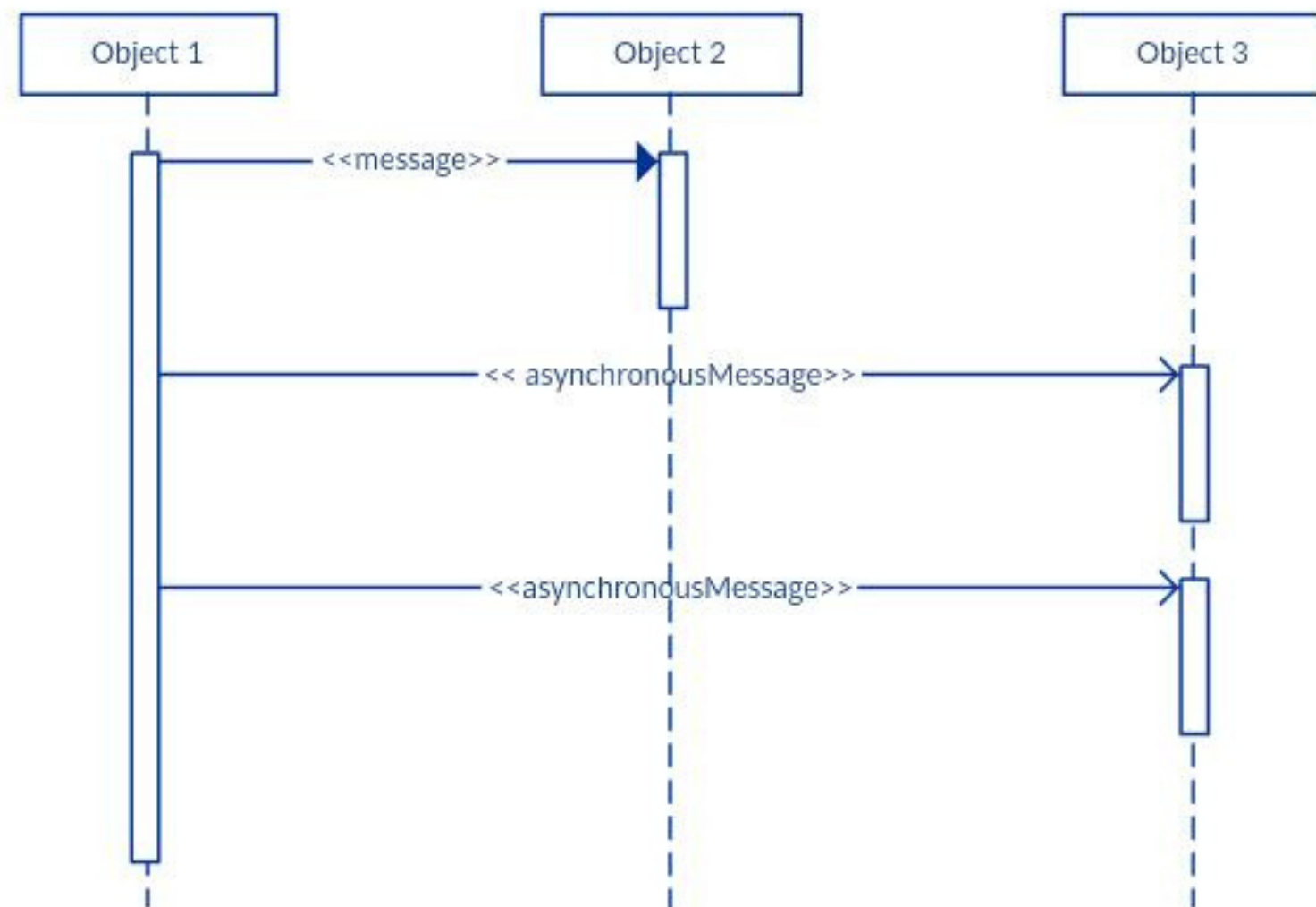
在循序圖中，當一個物件向另一個物件發送消息時，就會發生兩個對象之間的交互互動。

同步訊息(Synchronous Message)



如活動條示例所示，當發送方等待接收方處理該訊息並在繼續處理另一條訊息之前返回時，將使用同步訊息。用來指示這種訊息類型的箭頭是可靠

非同步訊息(Asynchronous Message)



當消息調用方在將其他消息發送到系統內的其他對象之前不等待接收方處理該訊息並返回時



Thank you

