物件導向基礎觀念

- 王克明 (Kenming Wang)
- Software Architect @HSDc Inc.
- http://www.kenming.idv.tw
- FB: 軟體設計鮮思維 https://www.facebook.com/groups/softthinking/

Agenda

- 物件 (Object)與類別 (Class)

- 類別的關係
  - 關聯 (association)
  - 整體-局部 (whole-parts)
  - 一般化-特殊化 (generalization-specialization)

Why Object-oriented Thinking？

- 解決軟體複雜度 (complexity)的一種思考模式/方法
- OOP (object-oriented programming)語言的完整支援 (C++, .NET, Java, PHP, Ruby)
- 把軟體作軟 (Keep Software Soft)

解決軟體複雜度的物件導向設計思維

- 封裝 (encapsulation)
  - 黑箱的角度封裝複雜事務。
- 介面設計 (interface design)
  - 定義標準規格 (specification)，以達成可動態抽換易變元件的效果。
- 多型 (polymorphism)
  - 「一視同仁」的角度看待多個特殊化類別的共同行為。

What is Object?

- 物件的實例 - 個體
  - 看得到的：小魯的筆電、妹妹的iPhone、小明養的玄鳳粉圓。
  - 抽象的事物：A飯店E812訂房紀錄、編號10112的訂單交易、單號F1688的保險契約。
- 簡單解釋：到處都是物件 (Everything is Object)
- 對系統而言：只找出有意義/價值的物件

Object Definition

"An Object is anything to which a concept applies. It is an instance of a concept."
(物件是概念可以被應用的任何事物，它是概念所呈現的個體)。

看待物件的角度/面向

以樹木為例，是一個物件還是多個物件？

- 聚焦在整體：一個物件。
  從遊客的角度欣賞樹木的茂盛與宏偉。
- 聚焦在組成樹木的局部組件：多個物件。
  植物學家研究組成樹木各組件 (樹葉、樹枝、樹幹、樹根)的功能與生命週期。

物件的特徵 (Features)

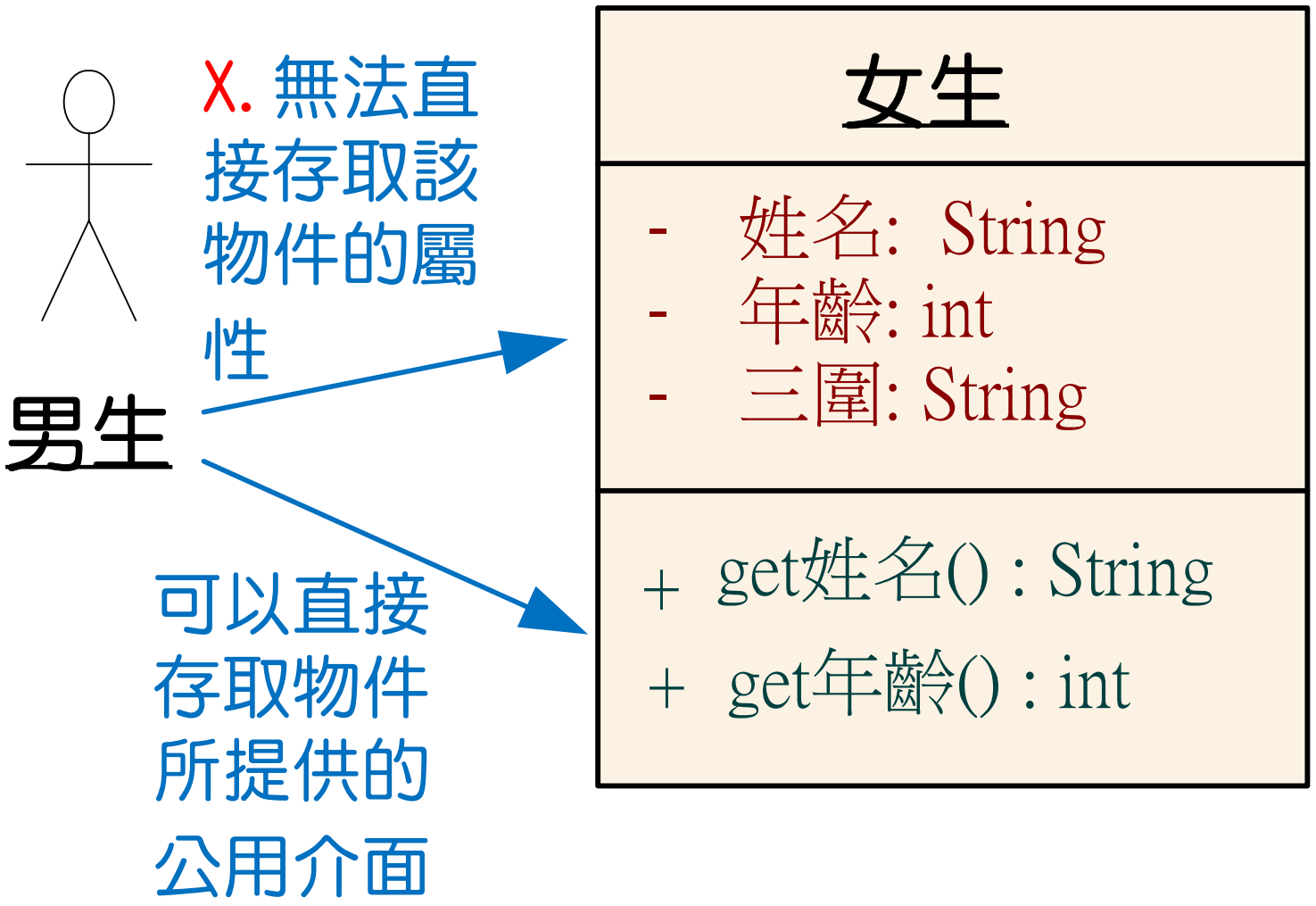
物件必然要有其特徵 (features)，始可以讓外界能辨識甚而操作它。

- 屬性 (attribute) - 物件所知道的 (knowing)
  - 物件所具有的特定型態 (type)
  - 以書本為例，屬性可能就有書名、作者、出版日期、ISBN等。
- 行為 (behavior) - 物件需要做的 (doing)
  - 物件所能執行的動作
  - 以車子為例，行為可能就有發動、轉動方向盤、踩油門、煞車等。
  - 行為 (behavior)
    - 軟體規格模型的表達 (如 UML)稱為「操作 (operation)」
    - 軟體實作的程式語言 (如 C#.NET)成為「方法 (method)」

物件的資訊隱藏

- 物件可以決定，哪些資訊 (屬性)可以被外界 (client)來取用，所以一般會將其「封裝 (encapsulate)」起來，不直接被外界來取用。此稱為「資訊隱藏 (information hiding)」。
- 物件透過提供「公用的操作方法 (public operation)」，藉以讓外界來呼叫取得需要的資訊。
  所以物件與物件之間 (Client 物件與 提供公用方法的物件)，係透過訊息 (message)的傳遞，來達成溝通的作用。

資訊隱藏的範例


X. 無法直接存取該物件的屬性
男生
可以直接存取物件所提供的公用介面
女生
- 姓名: String
- 年齡: int
- 三圍: String
+ get姓名() : String
+ get年齡() : int

What is Class?

- 將具有共同特性 (包括屬性與操作)的一群物件的集合 (sets)，抽象化後稱之為類別 (class)—**物以類聚**

- 分類是解決軟體複雜度的本能
—但懂得觀察與抽象，確實作好分類，才有機會得以解決軟體的複雜度議題。

What is Class?

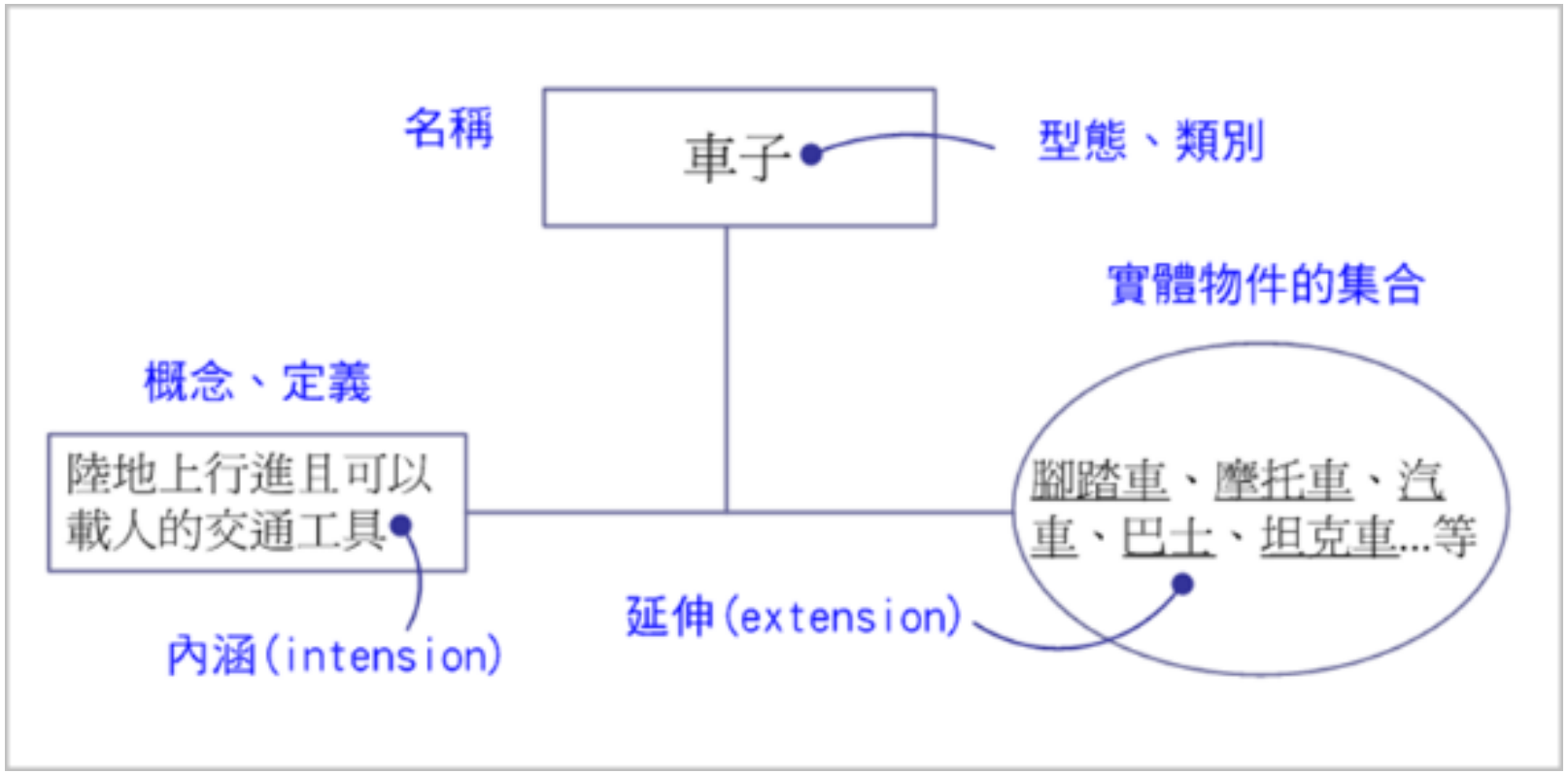
- 將物件“分門別類”，才能層次分明、井然有序的管理好在系統內各司其職、各負其責的眾生物件。

  也因為系統就是藉由各類型物件的互動、分工合作，才能完成越來越形複雜、所被賦予的一個個工作與任務。

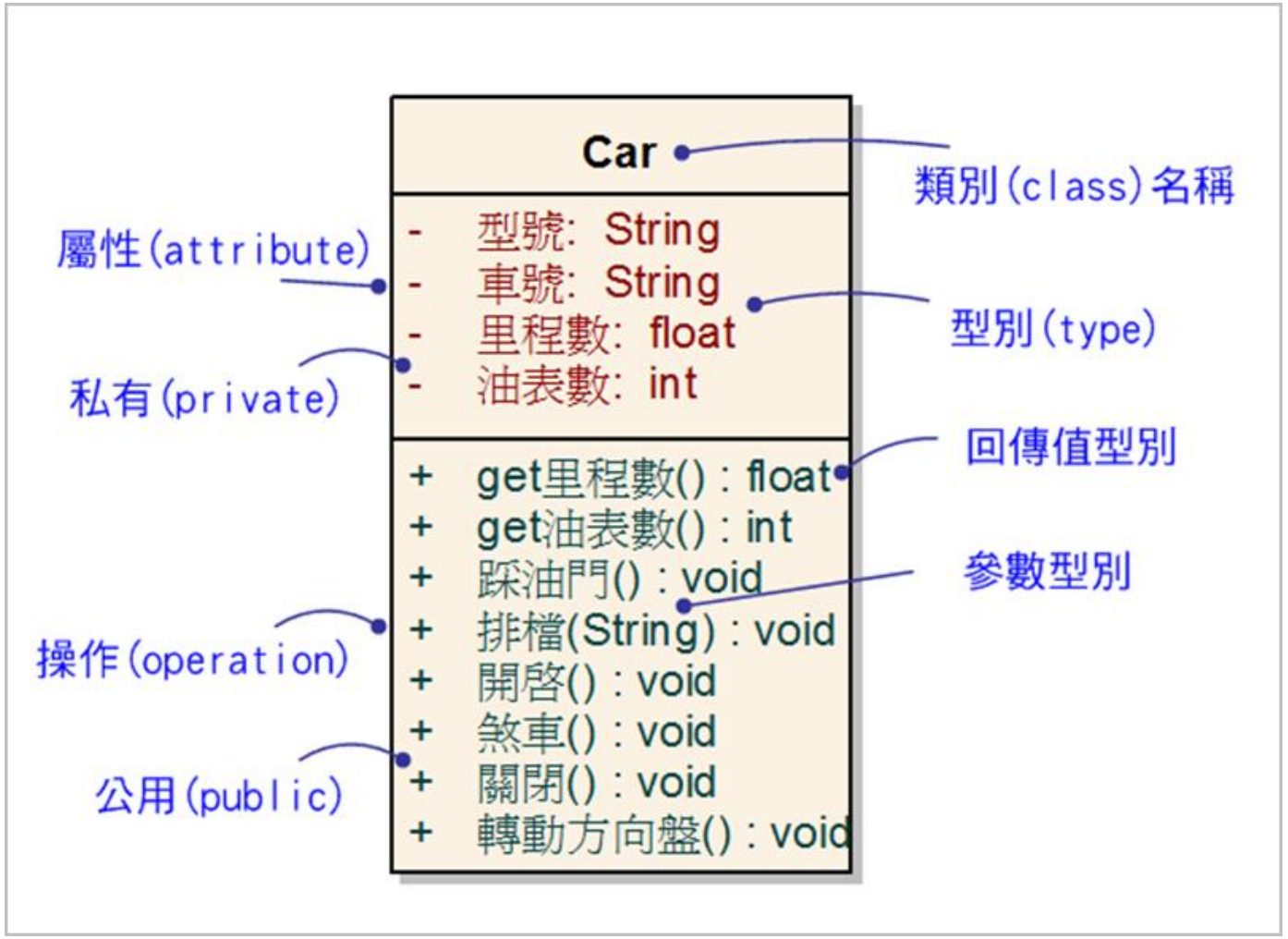
類別的定義 – 概念三部曲

- 依據 J.Martin & J.Odell 的「概念三部曲 (concept triad)」，可以利用三角形的圖形表示法，將類別以內涵 (intension)與延伸 (extension)兩種表達方式，來作更為明確的定義及描述。
  - 內涵 - 以定義的方式來表達類別。
  - 延伸 - 以實際的實體為例，來說明類別。

範例－定義類別的概念三部曲



類別的 UML 表示法


Car
類別(class)名稱
屬性(attribute)
- 型號: String
- 車號: String
- 里程數: float
- 油表數: int
型別(type)
私有(private)
回傳值型別
+ get里程數() : float
+ get油表數() : int
+ 踩油門() : void
參數型別
操作(operation)
+ 排檔(String) : void
+ 開啓() : void
+ 煞車() : void
公用(public)
+ 關閉() : void
+ 轉動方向盤() : void

類別 UML 語法說明

- 類別在 UML 圖示中是以長方形來表示，以橫隔線將長方形分為上、中、下三塊，從最上方依序填入「類別名稱」、「屬性」與「操作」。

「屬性」的表示語法為：(其中，型態與初始值可以省略)：

屬性名稱：型態=初始值

「操作」的表示語法為：(參數串列與傳回值型態可以省略)：

操作名稱(參數串列)：傳回值

其中，參數串列中，參數名稱與型態為一組，並以逗號來區隔表示。

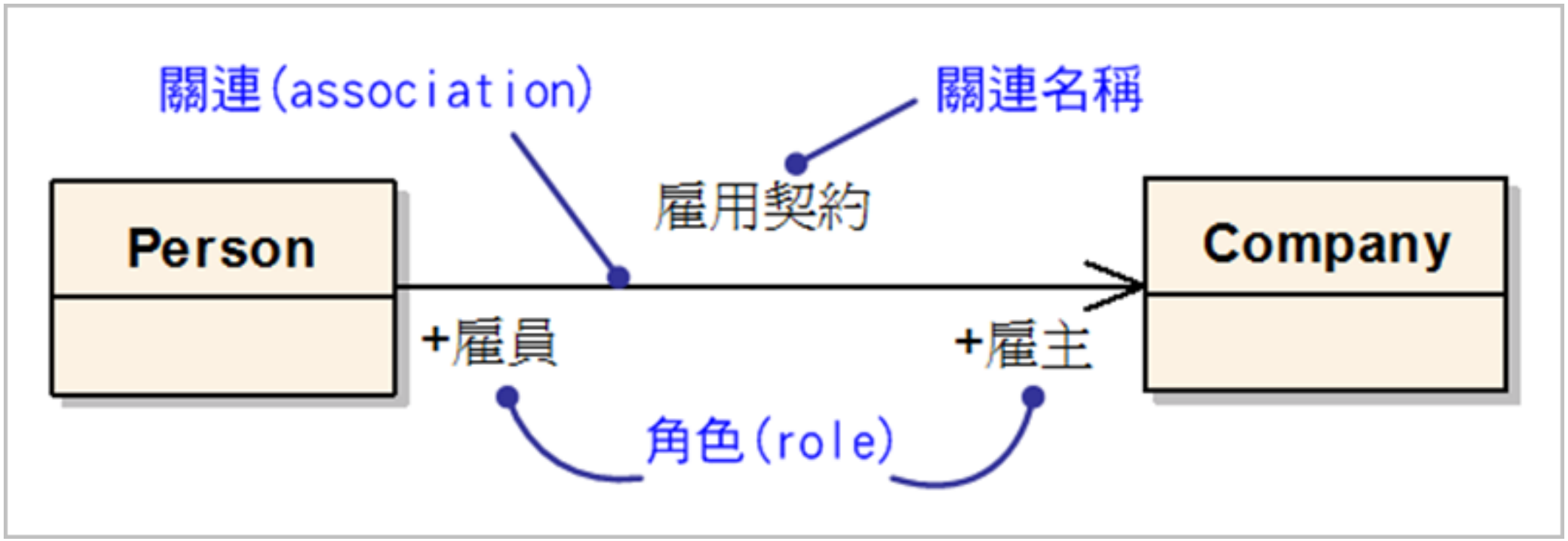
類別之間的關係

- 觀察物件之間的關係，再予以抽象化 (abstraction)，也就是說再進而一步地來觀察這些物件所屬類別(class)之間的關係；依據其關係的性質與其結構性，可以發掘出類別之間的關係有三種：
  - 關聯 (Association)
  - 整體－局部 (Whole-parts)
  - 一般化－特殊化 (Generalization-Specialization)

類別－關聯

- 關聯 (association)代表著需要記憶類別之間的關係。通常是在某個情境當中，我們需要保存其相關資訊一段時間 (視領域與需求而定，一般而言，可能保持幾秒到數十年的情況都有可能)。

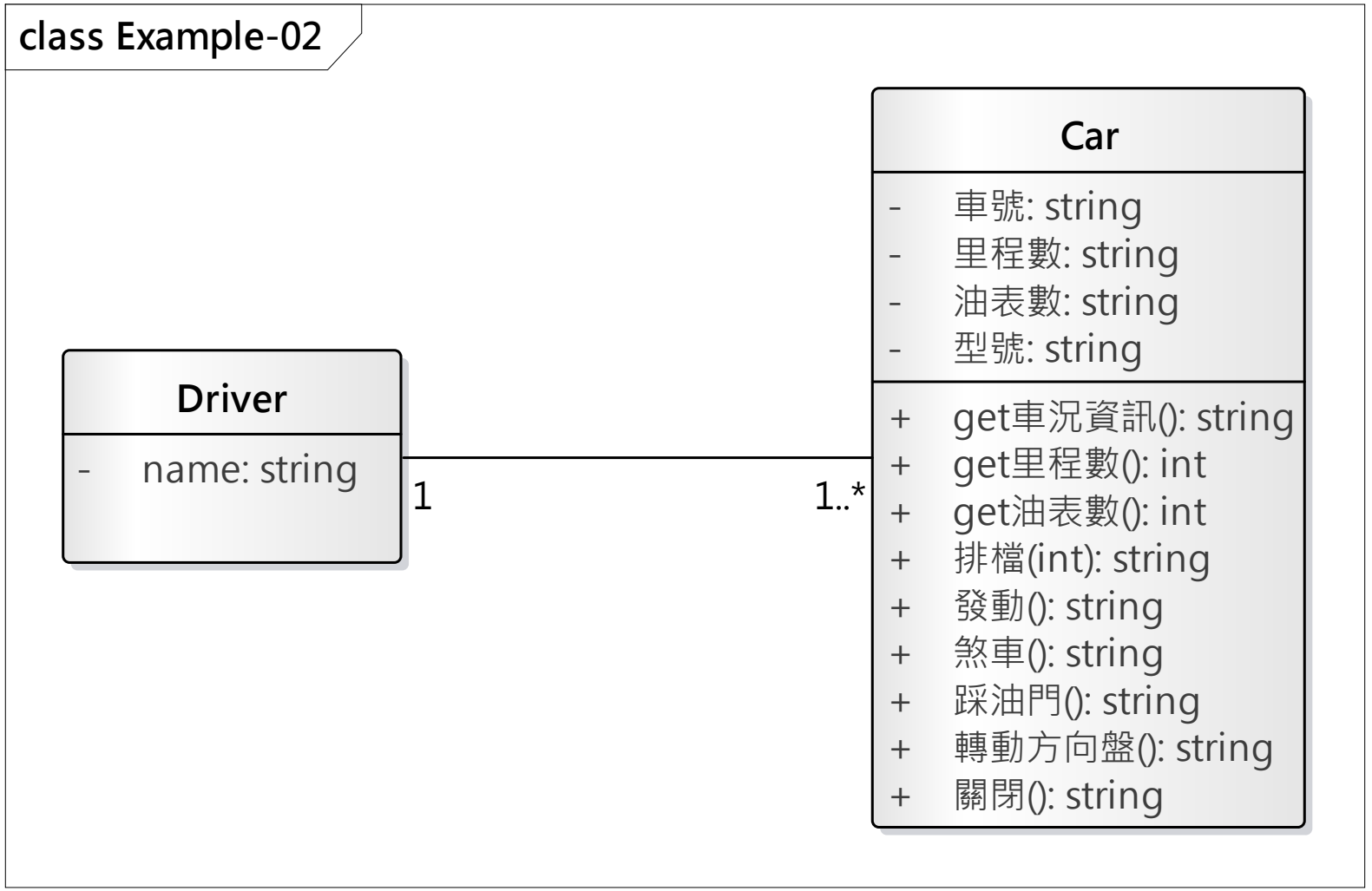
範例－類別之間的關聯



範例說明－類別之間的關聯

- 在結合關係內的每一個類別，都有其各自所擔任的角色(role)。
  - Person 類別在與 Company 類別的結合關係中，是擔任了「雇員」的角色；反之，Company類別則是擔任著在這個結合關係中的「雇主」角色。
  - 類別所擔任的角色，是表示在這個關連靠近類別的端點旁邊。
- 關連也可以為其命名，來凸顯兩個類別之間的一種行為。
  - 上圖關連名稱被命名為「雇用契約」，代表著 Person 類別與 Company 類別是一種「雇用」 的結合關係。

範例－駕駛員與汽車的關聯 (UML)



範例說明－類別的多重性

上圖為「Driver」類別與「Car」類別的 UML 表示法。
其中，1 對 1…* 為多重性 (multiplicity)的表示法，多重性定義了類別 A 會有多少個個體 (instance)與類別 B 多少個個體會關連在一起。上圖的多重性定義即表示為：一個駕駛員可以擁有1到多台車子。

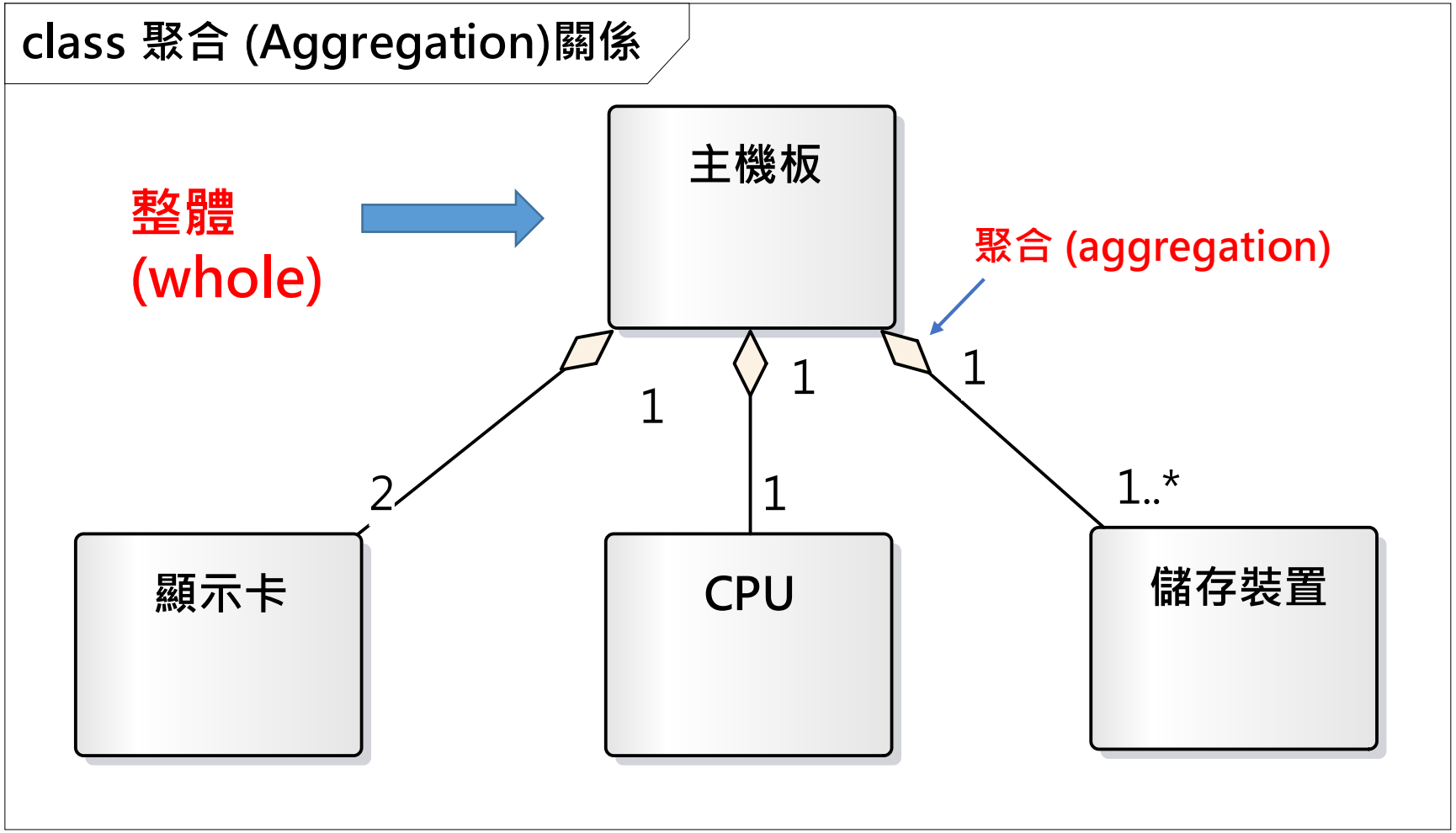
汽車的範例程式碼 for C#.NET (使用 VS.NET Community 2015)

原始碼位置 (可使用 Git Clone 整個專案)：
https://github.com/kenming/object-foundation-csharp

類別－整體 - 局部

- 降低物件複雜度 (complexity)的最有效機制。
  - 將物件視為一個整合體 (whole)，不用探究內部組成的細節。
  - 從外部觀點來看待一個整體物件，是一種「用」的態度。
  - 「用」的角度即為一種「封裝 (encapsulation)」的效果。
  - 「用」= 需求分析，從外部觀點看待整體物件所提供的服務。
- 當需自行製造 (實現)某一整體物件，需探究其內部的組成元素，使其更具彈性度與延展性。
  - 探究 (設計)整體物件內部的組成元素，此為「結構設計 (structure design)」的範疇。
  - 一個基本的設計原則－內部組成元素的變動，不要影響到整體物件對外所提供的功能/服務。

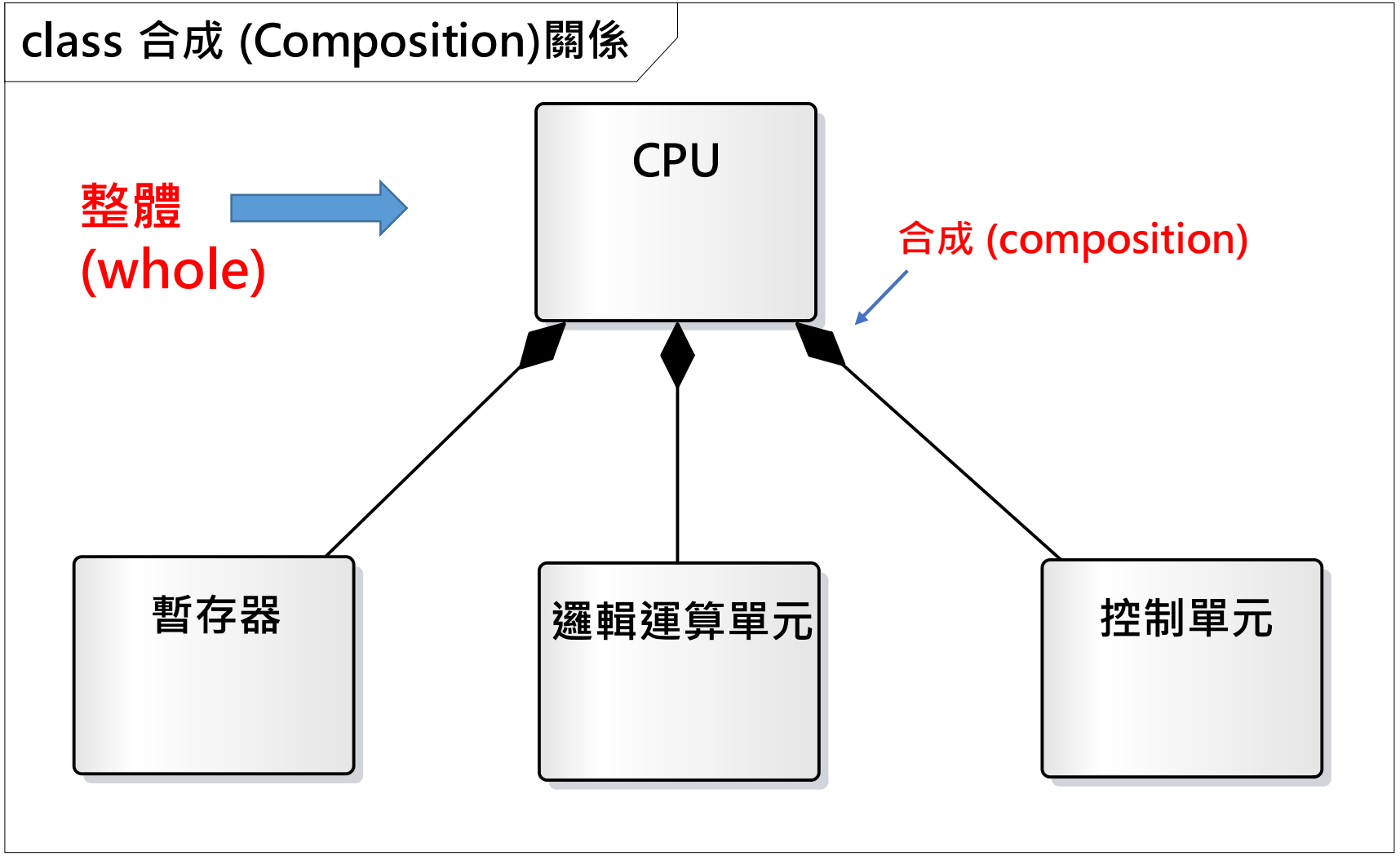
範例－整體 - 局部的聚合關係



整體 - 局部聚合關係說明

- 聚合 (aggregation)是一種較鬆散的整體－局部關係。
- 整體物件 (主機板)與局部 (顯示卡/CPU/儲存裝置)組件可以是不同的生命週期。
- 局部組件可以被抽換，以延續整體物件的「可重用度 (reuse)」性。

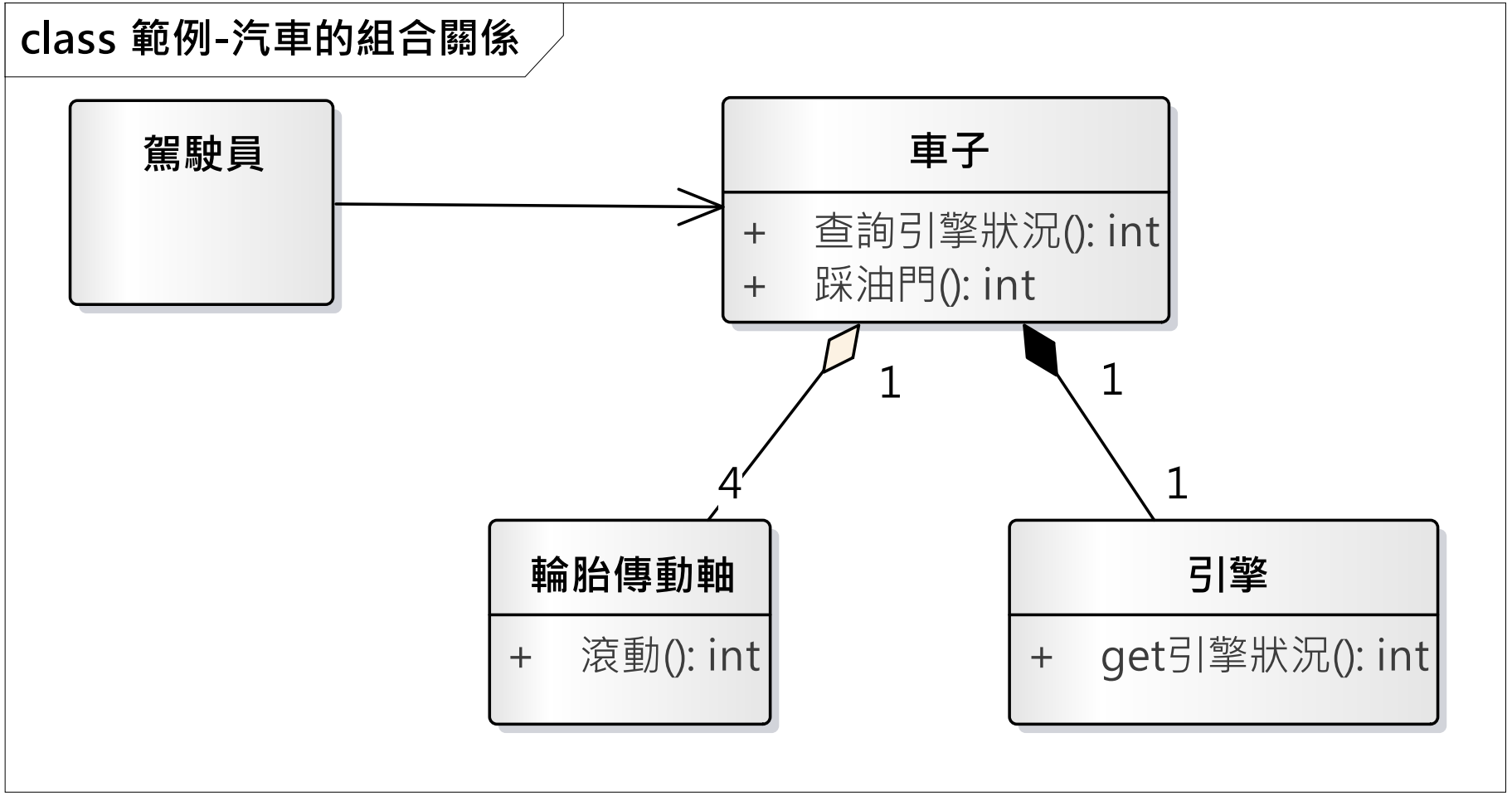
範例－整體 - 局部的聚合關係



整體 - 局部合成關係說明

- 合成 (composition)是一種較嚴謹的整體－局部關係。
- 整體物件 (CPU)與局部 (控制/邏輯運算/暫存器)組件的生命週期是一致的。
- 更強調整體性的黑箱 (black-box)「用」的服務，無法讓外界碰及內部的組成。

範例－汽車的組合關係


class 範例-汽車的組合關係
駕駛員
車子
+ 查詢引擎狀況(): int
+ 踩油門(): int
1
1
4
1
輪胎傳動軸
+ 滾動(): int
引擎
+ get引擎狀況(): int

- DEMO – C#.NET 程式碼範例

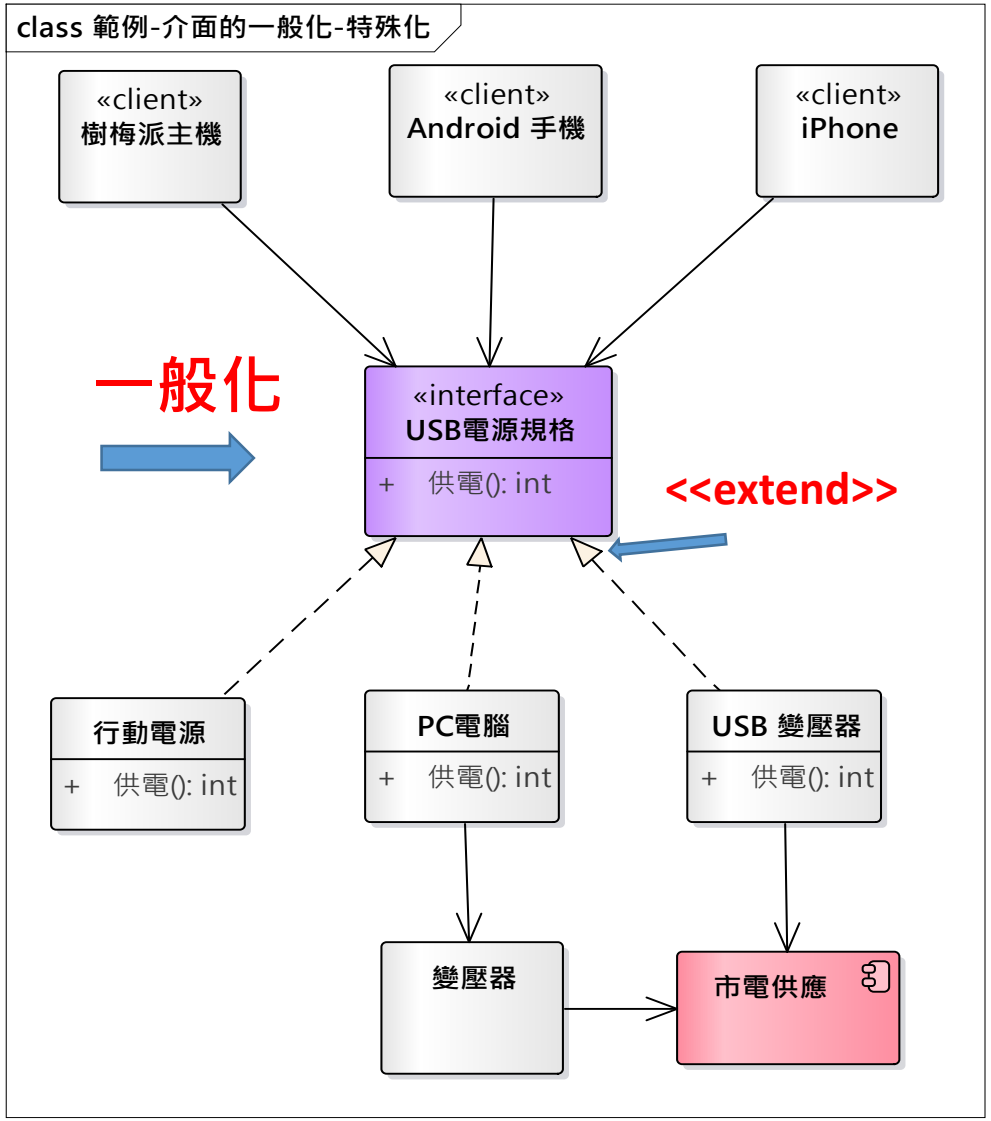
類別－一般化－特殊化

- 觀察多個類別所具有共同的特徵 (尤指行為)，抽離成一般化的類別 (generalization class)，讓外界 (client)只與一般化類別互動，避免直接與特定的類別 (specialization class)耦合 (coupling)。
- 外界 (client)可以「一視同仁」的角度來看待一般化類別，同時可以有效擴展 (extend)與調整特殊化類別的實作內容，且不會影響到外界的操作存取。
- 一般化-特殊化關係強調的是「可被替代性 (substitutability)」，而非繼承 (inheritance)，以為是程式碼的可重用性。
- 特殊化類別的「可被替代性」，有兩種意涵：
  - 覆蓋 (override)－覆蓋一般化類別實作的行為。
  - 擴展 (extend)－擴展一般化類別沒有的行為。

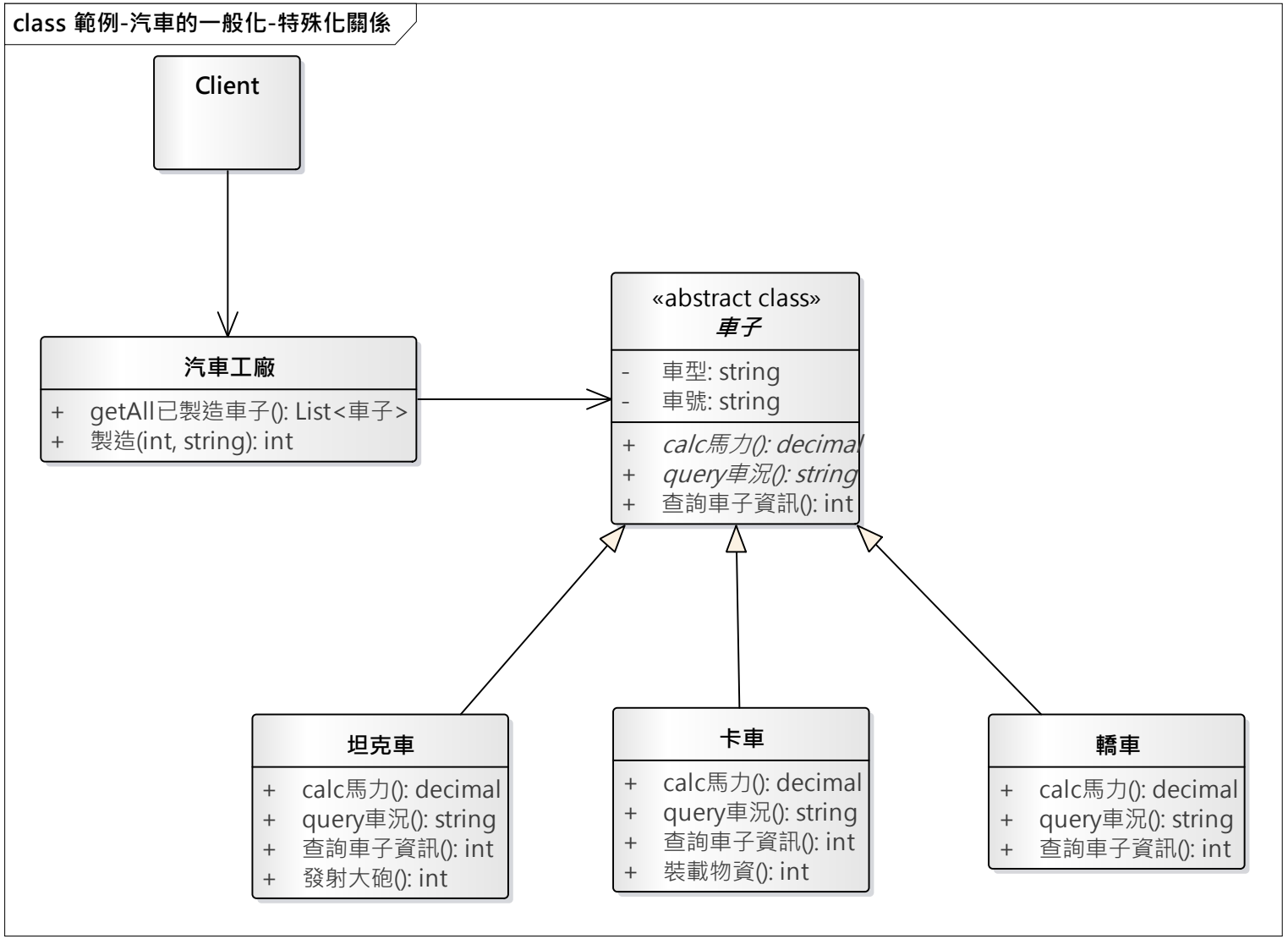
抽象的一般化類型

- 具體類別 (concrete class)
  - 完全實作所有方法的內容。
- 抽象類別 (abstract class)
  - 實作部分方法的內容。
- 介面 (interface)
  - 完全沒有實作，只定義規格 (方法名稱/參數/回傳值)。

範例－介面的一般化設計


class 範例-介面的一般化-特殊化
«client»
樹梅派主機
«client»
Android 手機
«client»
iPhone
一般化
«interface»
USB電源規格
+ 供電(): int
<<extend>>
特殊化
行動電源
+ 供電(): int
PC電腦
+ 供電(): int
USB 變壓器
+ 供電(): int
變壓器
市電供應

範例－汽車的一般化-特殊化



- DEMO － C#.NET 程式碼範例