# 8

# 物件導向軟體工程與開發實務

學習重點

1. 認識物件導向技術
2. 了解物件導向模型的內涵
3. 認識物件導向模型的特性
4. 認識物件導向軟體工程
5. 了解物件導向分析與設計的程序與方法
6. 從 UML 體驗視覺化塑模的過程與方法
7. 學習運用 UML 來輔助物件導向分析與設計工作的進行

學習地圖


8.1 認識物件導向程式設計
8.2 物件導向模型
8.3 物件導向技術的應用
8.4 物件導向軟體工程
8.5 一個比較標準化的物件導向分析程序
8.6 物件導向的設計方法
8.7 塑模工具的魅力：UML

物件導向技術（Object-oriented technologies）是近年來在資料模型、系統分析與設計以及軟體工程等領域應用廣泛的新技術；**對於資料庫應用而言，物件導向技術有兩個主要的影響：應用系統的資料模型與應用系統的開發**。第一個影響來自於新興起的資料庫應用，像電腦輔助設計與製造（CAD／CAM），由於資料的結構複雜，不適合用關聯式的資料模型來表示，物件導向資料模型（Object-oriented data model）可以提供較佳的資料描述的環境與方法。

至於系統開發方面的影響則來自於軟體工程及圖型化使用者介面（GUI, Graphical User Interface）的進展，運用物件導向技術，能提昇軟體元件的再用率（Reuse ratio），設計圖型化介面時，就可以建立在既有的物件基礎之上，所謂的「快速應用系統開發」（RAD, Rapid Application Development）要靠現有軟體元件的堆砌才能達成。

**TIP**

物件導向技術以類別（class）與物件（object）的觀念為基礎，用來描述現實世界的事物，由於類別有強大的抽象化與描述能力，加上物件的互動與動態特性，使得物件導向技術能運用在許多的領域中，而且得到不錯的效果。系統分析與設計是物件導向技術應用的領域之一，也是我們下面要探討的主題。

以物件（Object）為基礎來描述各種事物有很大的彈性，既可以形容實體的物質，又能說明抽象的觀念。目前電腦作業系統中使用的視窗（Windows）環境，就算是物件導向技術的一種應用，視窗裡頭的物件包括按鈕（Button）、功能表單（Menu）等，可以使用滑鼠的指標來觸發其功能，這些視窗物件提供的視覺化使用者介面，可以完全用物件導向的觀念來詮釋。

物件導向技術應用得最廣的領域是在程式設計方面，所謂的「物件導向程式設計」（Object-oriented programming），就是運用物件導向技術來強化程式設計的效率；有組織地把這些技術轉化成軟體工程的工具，能節省系統開發的成本。**在程式設計領域的成功與經驗，使物件導向技術逐漸地被引入各種科技應用上**，例如目前常見的各種視覺化介面開發軟體、分散式系統、資料庫管理系統等。

物件導向技術對於電腦軟體與應用系統的影響最大，傳統的結構化程式設計固然也達到了簡化軟體開發程序的目標，但在效果上並沒有像物件導向技術那樣顯著。

## 8.1 認識物件導向程式設計

所謂的「物件導向程式設計」包括了「物件導向」與「程式設計」兩種技術的整合，在西元 1980 年代，物件導向技術影響了很多資訊科學的領域，產生的結果多半是正面的，其實物件導向的觀念起源得更早，只是到近年來才逐漸地被理論化和系統化，並導入各種領域中加以運用。

程式設計是很多人都曾有過的經驗，目的是要指揮電腦來解決問題，不同的程式語言往往提供了不同的解決問題的方法，而「物件導向程式語言」（Object-Oriented Programming Language）所提供的方法，就是建立在物件導向的基礎上。

### 8.1.1 從物件與類別談起

物件導向的程式設計是以類別與物件的觀念為核心的，一般來說，要談類別與物件的觀念並不需要使用任何的語言或軟體工具。物件代表我們所要描述的世界中的事物，類別把這些物件分門別類，事物之間有關聯，所以物件之間也會有各種關係與互動，任何的分類都會產生架構，所以眾多的類別也會形成類別的架構。針對一個問題分析以後得到的類別與物件就形成了問題本身的物件導向模型。這就是物件導向的基本觀念！

#### 「物件」是什麼？

比較正統的說法會以三個主要的特徵來描述物件（Object）：物件的身份（Identity）、物件的狀態（State）與物件的行為（Behavior），「身份」用來標示一個物件，可能是一個名稱或是號碼，「物件的狀態」指物件各種特性的現況，「物件的行為」則代表其功能，或是對於外來刺激會產生的回應。圖 8-1 是物件的圖示，通常物件的狀態不直接暴露給外部的世界，必須經由物件的行為來了解其狀態，這是物件導向中有名的「封裝」（Encapsulation）特性。

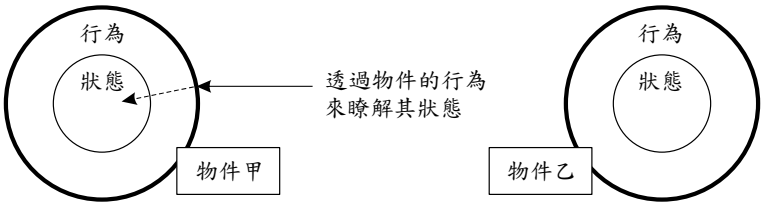


圖 8-1　物件的主要特徵

其實，從程式語言的觀點來看，物件的狀態要用資料或資料結構來表示，這些資料在數值上的變化就代表物件狀態的改變，物件的行為可以用程序（Procedure）來表示，也就是程式的片段，在物件導向的領域中，這些代表行為的程序常被稱為物件的「方法」（Method），而物件的狀態則慣稱為「屬性」（Properties 或 Attributes）。以 Java 程式語言為例，所有的運算都是以物件為核心的。

## ●「類別」是什麼？

分類可以將複雜的事物釐清，「類別」（Class）是用來將物件分類的，屬於同一類別的物件具有很多相同的特徵，從另一個角度來看，我們也可以說類別是鑄造物件的模子，物件則是類別的案例（Instance）。當我們定義新的類別時，可能會發現新類別的很多特性已經存在於現有的類別中，此時可以透過繼承（inheritance）的方式來定義新類別，新類別和舊類別之間就產生了 subclass 與 superclass 的關係，而實質上，新類別承繼了舊類別的定義，包括類別的屬性與方法。繼承的好處是「再用」（Reuse），軟體再用可提升軟體開發的生產力，類別繼承衍生出來的再用模式，使物件導向技術大幅地改善了軟體開發的效率。

在學習程式設計的時候，除了語法語意之外，得花最多功夫去了解的就是現有的類別庫（Class library），這些類別庫裡的類別除了定義一些屬性之外，還有含程式碼的方法（Method），透過繼承的語法，我們可讓新的類別自動具有既存類別的特性，不必再重新定義屬性或撰寫方法的程式碼，而且新類別能進一步地擴增或修改屬性與方法，讓程式設計者透過現有類別的再用來提升軟體開發的效率。當然，程式執行時真正在作用並展現功能的是物件，類別之間的關係形成了類別架構（Class hierarchy），物件之間除了承繼了類別之間的關係之外，應用系統的功能主要是靠物件之間的交互作用來描述的。

### 8.1.2 物件導向程式設計的基本觀念

經由物件導向程式設計寫出來的程式和一般程序導向的程式有很大的差異，程序導向的程式可以看成是一連串依序執行的程序的組合，物件導向的程式則是一群互動物件的組合，圖 8-2 簡略地畫出上述兩種程式設計觀念的特徵。

「物件」(Object)的概念是物件導向技術的核心，物件導向的程式可以看成是物件的組合，程式描述應用領域的世界，程式中的物件則代表實際的事物或抽象的觀念，圖 8-2 中物件之間的連結代表訊息的傳送，訊息會觸發物件內含的功能，而程序導向的程式設計中，程序間的連結表示程序執行的先後順序，除此之外，程序和物件在功能與結構上有很大的差異。

圖 8-2 所描述的比較接近程式在執行時期(runtime)的動態行為。光從靜態的程式內容可能還不太容易看出一個物件導向程式的特徵。以 Java 來說，Java 程式是以類別的定義為主，含有 main( )方法類別定義了主程式，程式中會利用類別的定義來產生物件，執行時期的活動是以物件為主軸的。物件的屬性代表處理的資料或是記錄的狀態，物件的方法是運算與系統邏輯的化身。

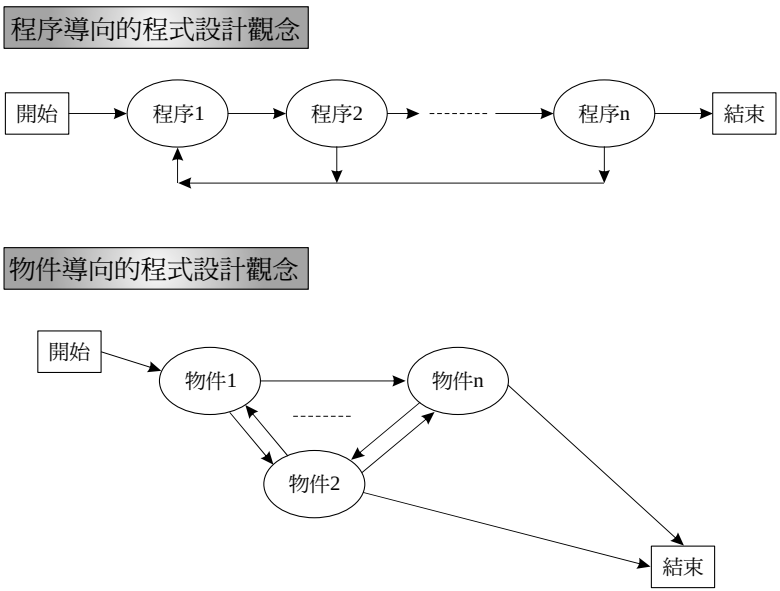


圖 8-2　程序導向與物件導向的程式設計觀念

## 程式語言中類別的觀念

物件導向程式設計的精髓在於類別（class）與物件（object）的運用，通常程式設計所產生的是對於現實事物的一種描述。一般人都能以口語的方式表達得很好，但是程式語言的描述往往更為嚴謹而精準，或許有人說這種描述太抽象了，很難意會。不過，電腦系統在目前的科技下就是得接受這樣的描述，我們姑且稱之為「抽象描述」（abstraction）。幸好抽象化能以簡御繁，程式語言自然具有支援抽象描述的能力，所用的就是類別與物件。

基本上，抽象描述針對的是所描述的物件，類別則實際地將抽象描述給定義出來，包括物件的屬性（property）與行為（behavior）的定義，類別所下的定義可衍生出很多物件。這些物件屬於同一種類別，而且表現出類別定義中的屬性與行為。屬性像是內涵，行為則像外在的表現，在 Java 裡頭，我們用程式變數（program variable）來定義屬性，行為用 Java 的方法（method）來定義。有時候屬性也稱做特性（attributes），行為也稱做操作（operations）。

類別定義出物件可提供的對外介面，表明物件可提供的服務，同時也定義了內部的實作（implementation），記載了詳細的操作步驟。從程式語言的觀點來看，外界可透過介面來呼叫物件內部的實作程序。

## 程式中物件的觀念

物件是類別的案例（instance），也就是說，類別像是建立物件的架構或模子，實際在 Java 程式中擔當大任的主要還是物件，不過在使用前，物件必須先被生產出來，物件的建立包括兩個主要的步驟：

### 一、參考變數（reference variable）的宣告（declaration）

物件的內容可多可少，當我們指名物件時，無法以其內容來分辨，因此，Java 採用物件參考值（object reference value）來指名某一個物件，有點像 C/C++ 裡頭指標和位址的觀念。當然，參考值還是得存在變數裡頭，在建立物件以前，我們得用物件參考變數來宣告物件在程式裡頭的名稱。

## 二、產生物件

這個步驟也叫做類別案例化（class instantiation），利用建構子（constructor）來產生類別案例，也就是物件。關鍵字 new 會指示系統傳回一個指定類別所屬物件的參考值，當做參考變數的值。由於建構子是一個程序（即一小段可呼叫執行的程式），物件產生過程中物件本身的一些成員是利用建構子的執行而建立的。宣告和案例化兩個步驟在語法上也可以合併在一起。由於物件移到記憶體之後占用了空間，最好能回收，在 Java 的環境裡頭，不再被引用的 Java 物件會被系統移除，這種程序就是有名的「垃圾回收」（garbage collection），這裡的垃圾其實是指物件所占有的記憶體空間。

### 物件的成員與類別的成員

物件是從類別的模子裡塑造出來的，既然物件是類別的案例，自然就擁有了類別中定義的屬性與方法，這些都是物件的成員（可稱之為 instance member）。不過，物件的屬性成員具有指定的值，方法成員則是同一類別的所有物件都一樣的，所以物件的屬性成員決定了物件的狀態（state），每個物件的狀態決定於其屬性成員所具有的值。類別可以具有不屬於任何物件的屬性成員，我們把它叫做靜態成員（static member）。靜態成員的處理可以透過類別名稱或是該類別的物件的名稱來進行。

程式語言中有一些術語是一般人常感到混淆的。圖 8-3 列出幾個很重要的名詞，案例成員和靜態成員是純粹在物件和類別的觀念中討論的說法，在 Java 語法裡頭，則是以案例變數（instance variable）、案例方法（instance method）、靜態變數（static variable）與靜態方法（static method ）來表示。

若是從程式執行時所發生的情況來觀察，任何物件建立之前，其所屬的類別會先被系統載入（load），之後才能開始建立物件。利用物件參考變數來存取物件的屬性，或是呼叫物件的方法成員，至於**靜態成員的部分，可經由類別名稱來引用。若是透過物件來引用，基本上可以看成是所有物件共同的變數，假如某個物件修改了某個靜態成員的值，則其他屬於相同類別的物件都會看到這個改變**。

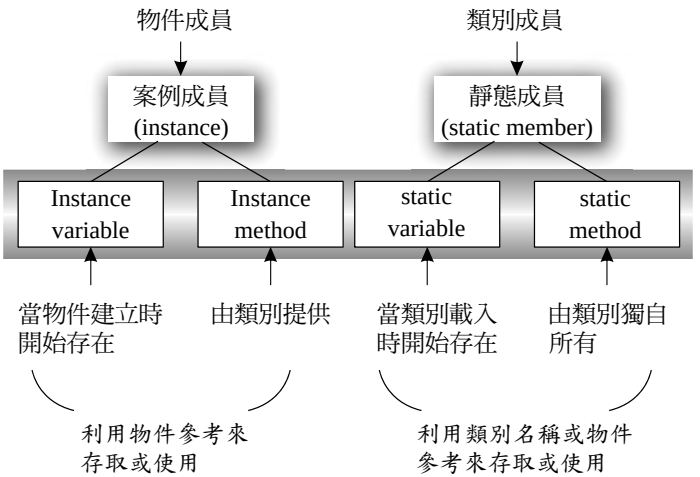


圖 8-3　與物件成員和靜態成員相關的名詞

程式中對於各種成員的利用，必須遵循語法與規則，通常我們可以透過方法的呼叫（method invocation）來觀察物件所表現的行為，所謂的「句點表示法」（dot notation）就是常用的呼叫方法的語法。

## 8.2　物件導向模型

物件導向模型以物件與類別的觀念為基礎，運用在資料塑模上的稱為物件導向資料模型（object-oriented data model），一般系統分析與設計裡頭用的物件導向模型比較強調類別的架構與物件的描述能力。

### 8.2.1　物件導向資料模型

資料模型可以用來描述應用系統，物件導向資料模型（Object-Oriented Data Model）以物件為基礎，具有繼承（Inheritance）、包裝（Encapsulation）與同形異功（Polymorphism）等特性。我們可以用類別（Class）與型式（Type）來分類物件，而物件本身的特性則是由屬性（Attributes）與方法（Methods）所構成的。圖 8-4 把物件與類別之間的關係描繪出來。這裡有幾個要釐清的觀念：

1. **根型式（root type）有什麼作用**：每個物件導向系統都會有一個所謂的 root class 或 root object，其實都可以看成是一種 root type，對於系統

來說，最簡單的看法是把所有的處理的東西 看成是物件，所以都是從 root type 衍生出來的，這樣可以簡化系統的設計。

2. **型式（type）與類別（class）有什麼差別**：一般把 type 看成比較抽象的定義，而類別則含有實作（implementation）的部分，當然不見得大家都有這樣的看法。

3. **類別與物件的關係**：類別是物件的定義，從另外一個觀點來看，類別是製造物件的模子。這個觀念要在物件導向的程式語言中比較容易體會，因為我們需要真正地去定義類別然後由類別的定義來產生物件，這是和其他的程式語言比較不一樣的地方。

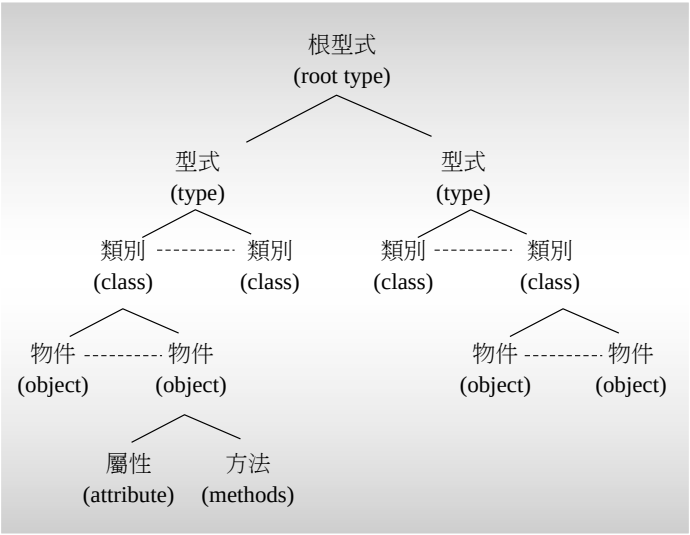


圖 8-4　物件和類別的關係

型式與類別的差異在於型式比較抽象，定義某一類物件的一般特性與行為，這些特性或是行為可能會因為詮釋的方法不同而產生各種類別。**所謂的根型式（Root Type）則是集合了各種型式的特徵，讓一般的型式可以透過繼承來取得共通的特徵，不必再重新定義**。某些型式之間，也可能有繼承的關係，形成所謂的型式架構（Type hierarchy）。物件可以看成是類別的具體化型態，直接對應到現實世界裡頭的實體或是概念。

物件的屬性及方法定義在所屬的型式與類別中，物件的內涵包括這些屬性或是方法的具體涵義或是數值。和關聯式資料模型比較起來，物件導向資料模型的特點在於描述複雜結構的資料，雖然關聯式資料模型的觀念簡單，對於複雜資料的描述不像物件導向資料模型那麼自然，而且常需要做昂貴的 Join。

### 學習活動

為什麼要學物件導向的觀念呢？跟系統分析與設計好像沾不上邊ㄟ？現在可能還看不出來，等到了解物件導向分析與設計的方法以後，就會知道物件導向觀念的重要性。

我們下面以一個簡單的例子來比較關聯式資料模型與物件導向資料模型之間的差異。圖 8-5 列出關聯式資料模型描述員工與部門關係的方式，而圖 8-6 則是以物件導向資料模型來描述相同的資料；在圖 8-5 中，主鍵值編號及主鍵值部門編號是用來建立表格之間關係的，員工編號的值可以代入工作關係表格，找到該員工所屬部門的編號，然後依此編號，在部門資料表格中找到部門的名稱及其經理。圖 8-6 則以部門物件和員工物件來表示相同的資訊，員工和部門的關係以各種指標（Pointer）來聯繫，通常指標所占的儲存空間和資料比起來很小，處理的時候效率較高。

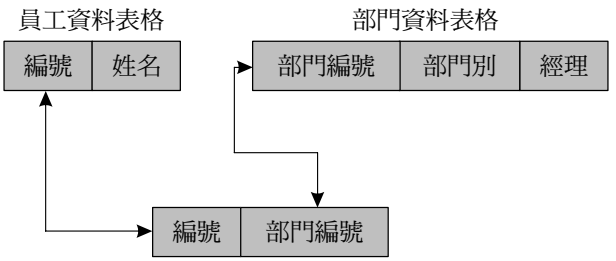


圖 8-5　員工部門關係的關聯式資料模型

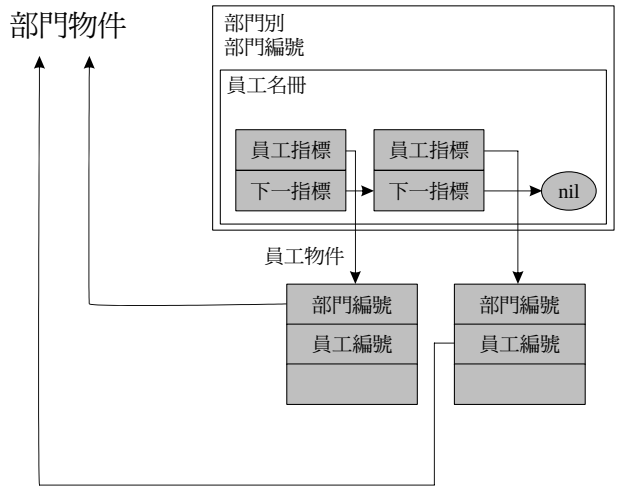


圖 8-6　員工部門關係的物件導向資料模型

程式語言的資料模型也可以描述結構複雜的資料；因此，**物件導向資料模型比較容易和程式語言結合，而關聯式資料模型則需要適當的轉換才能讓程式語言接受以表格為主的資料，而程式語言所描述的複雜資料也無法很自然地透過關聯式資料庫管理系統來處理，這種現象稱為所謂的 Impedance mismatch，代表關聯式資料庫系統與程式語言在資料模型上無法充分結合**。由於物件導向資料庫系統沒有 Impedance mismatch 的問題，所以對於複雜結構資料的描述與處理，在先天上優於關聯式資料庫系統。

以圖 8-5 的資料定義來說，只是很單純的資料表格，在程式語言中可用「集合（Set）」來表示，對於資料庫來說，這些集合可能代表了成千上萬個資料記錄的組合，以程式語言的觀點來看，任何的程式變數的值，都對應到主記憶體中某些位置的儲存值，但顯然資料表格中所有資料記錄全部載入主記憶體中，是不實際的做法，因為主記憶體的空間有限。物件導向資料模型和關聯式資料模型的另一個主要差異在於對資料的處理。

物件導向資料模型可以讓使用者對資料做巡弋式的（Navigational）的查詢，主要是透過指標（Pointer）來建立資料之間的鏈結；因此，不管資料的結構多複雜，經由指標所建立的查詢路徑，可以直接指向使用者所查詢的資料。至於關聯式資料庫系統所用的則是一種平行式的（Associative）的查詢，必須假設資

料是由相同結構的集合（Set）所組成的，這些結構不能是複雜的，要能由表格（Table）來表示。

以圖 8-5 及圖 8-6 中的資料為例，假設我們想知道在某部門內工作的所有員工的姓名，關聯式資料庫的方法是進行圖 8-5 中三個表格的連結（Join），再選取所指定部門內的所有員工姓名，由於資料表格可能相當龐大，鏈結的成本很高。物件導向資料庫的方式則是先找到所指定的部門物件，然後從員工名冊中的員工指標，逐一地找到對應的員工物件，從中取得員工的姓名，這就是所謂的巡弋式的查詢方式，在上面所舉的例子中，比較有效率。

## 8.2.2 物件導向的特性

所謂的「物件」（Objects），可以用來描述事物或觀念，圖 8-7 畫出一個物件的各主要成份，物件的狀態（State）內容物件的特性，環繞在物件狀態之外的灰色圓形代表物件的行為（Behavior），與這些圓形相連的介面（Interface）則是外界與物件溝通的唯一途徑；以上是物件的抽象觀念。

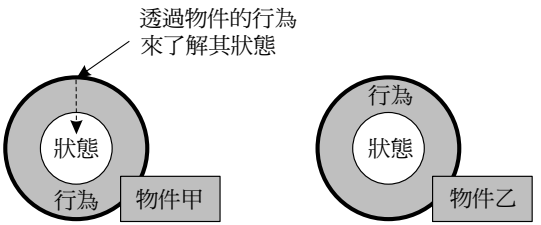


圖 8-7　物件的成份與結構

具體來說，物件可以代表各種事物，但是由於其基本特性的結束，使物件具有所謂「物件導向」的性質：

1. **包裝（Encapsulation）**：狀態必須經由介面與行為來了解，狀態的改變決定於物件的行為。包裝可以提供安全性，因為物件的狀態無法被外界直接控制；另一方面，介面可以簡化對於物件狀態的了解，因為外界不必了解物件狀態的細節。

2. **繼承（Inheritance）**：由於物件常有共通的狀態或是行為，把同類的物件歸納在同一個類別（Class）之下，可以簡化物件的定義。換句話說，類別是物件的定對，而物件則是類別的具體化，以程式語言的觀點來看，類別像是變數的型式宣告（Type Definition），物件則含有變數的實際數值。所謂的繼承是指相似的類別可以把共通點析出，變成一種新類別，原先的類別則可看成是繼承了這個新類別的特徵，再加上個別的特性。圖 8-8 列出類別與類別以及類別與物件之間的關係。繼承的好處是藉再使用（Reuse）來節省重新定義類別的成本。

3. **同形異功（Polymorphism）**：也可稱為多元性，指名稱相同的物件行為，具有不同的功能；這是因為在某些情況下，相同的行為會因物件其他屬性的不同而表現出不同的結果。通常同形異功的特性可以從物件與類別的定義看出來。

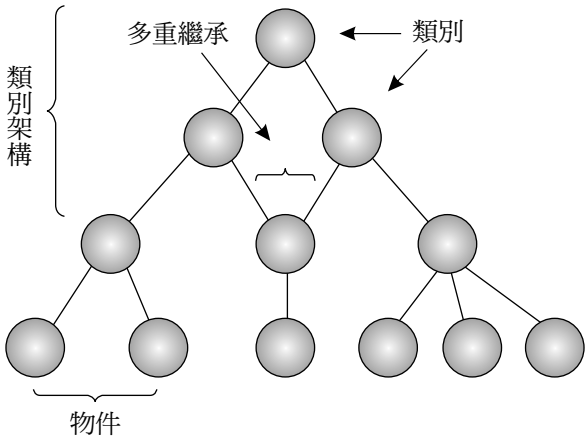


圖 8-8　類別與物件

## 加深印象

物件還有一個很重要的特性，就是識別（identity），object 的 identity 是我們或者是系統辨認 object 的基礎，通常我們會說每個物件都有一個唯一的識別（unique identity），就像身份證一樣。