物件導向有哪些元素

什麼是物件(object)？

你身邊看到、聽到、摸到的—有形、無形的事物—通通都是物件！
一個物件可以代表人、地方、事件，或交易
物件也可以是無生命的個體，比如一輛汽車或一台電腦
物件也可以是一件抽象的觀念，如天氣的變化，或滑鼠所產生的事件

類別（class）

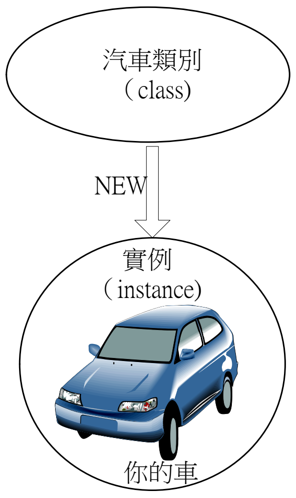
在真實世界裡，有許多同“種類”的的物件。而這些同“種類”的物件可被歸類為一個“類別”
例如我們可將世界上所有的汽車歸類為汽車類別。所有的動物歸為動物類別。

實例（instance）

汽車類別有些共通的狀態（汽缸排氣量，排檔數，顏色，輪胎數…）和行為（換檔，開燈，開冷氣…)。

但每一台汽車個別的狀態及方法可不同於於其它汽車。

你的汽車只是這世界中許多汽車中的一個。我們就稱你的汽車物件是汽車類別中的一個實例（instance）



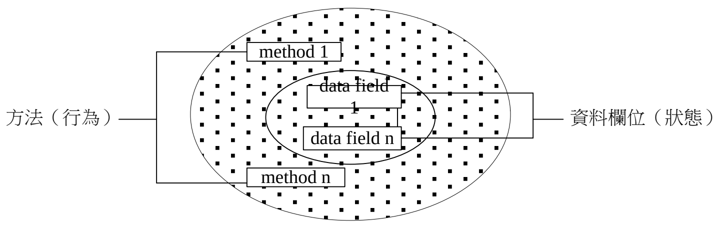
Source: Java 2 王建捷著 碁峰出版

物件有何特徵？

物件有兩個特徵：狀態（state）和行為（behavior)。

一個人有他的身高或體重等狀態，他的行為－如唱歌、打球、騎摩托車、開汽車。

一隻狗有牠的顏色作狀態，也有牠的行為，如吠叫，或跳躍



Source: Java 2 王建捷著 碁峰出版

如何將物件表達在程式中呢？

軟體物件的定義－**物件是由資料欄位（變數）及相關方法所組成**

在程式設計中，軟體物件的觀念由真實世界物件而來。

- 狀態—以變數(variables)、或屬性(attributes)、或特性(properties)、或欄位(fields)表達。資料成員(Data member)
- 行為則以方法(methods)來達成。成員函數(Member function)

屬性 (Attributes)

- 屬性是描述物件的各種特質

- 要有那些屬性？取決於使用者的業務需求
- 物件的屬性在系統設計時定義

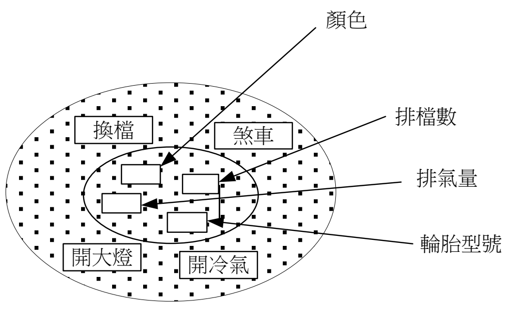
方法 (Methods)

描述一個物件做些什麼及如何作(*what* and *how)*

- **建構方法**(constructor method)：產生一個物件的新實例
- **更新方法**(update method)改變現有資料
- **查詢方法**(query method) 查詢某個物件屬性相關資訊的方法
- …還有很多…

如何描述一個物件？

- 如何描述一台車子？
  - 編號：
  - 廠牌是：
  - 顏色是：
  - 目前速度：
- 車子有什麼行為動作？
  - 發動
  - 加速
  - 左右轉向
  - 超車
  - 停車



Source: Java 2 王建捷著 碁峰出版

類別圖 Class diagram

每個人都有不同的車子…

王小明有一台車是三菱 LV1234
李大同有一台車是豐田 MM1828
需要針對每個人的車子各寫一個操作車子的程式嗎？

車子有共同之處—共通的類別

Car
Attributes number speed color

Methods
accelerate()
turnRight()
stop()

物件導向特色

Java is an object oriented language because it provides the features to implement an object oriented model. These features includes encapsulation 封裝, inheritance 繼承 and polymorphism 多型.

OOPS is about developing an application around its data, i.e. objects which provides the access to their properties and the possible operations in their own way.

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.

Principles of OOPs

1) Encapsulation 封裝

Below is a real-life example of encapsulation. For the example program and more details on this concept refer encapsulation in java with example.

Encapsulation is:

- Binding the data with the code that manipulates it.
- It keeps the data and the code safe from external interference

Looking at the example of a power steering mechanism of a car. Power steering of a car is a complex system, which internally have lots of components tightly coupled together, they work synchronously to turn the car in the desired direction. It even controls the power delivered by the engine to the steering wheel. But to the external world there is only one interface is available and rest of the complexity is hidden. Moreover, the steering unit in itself is complete and independent. It does not affect the functioning of any other mechanism.

Similarly, same concept of encapsulation can be applied to code. Encapsulated code should have following characteristics:

- Everyone knows how to access it.
- Can be easily used regardless of implementation details.
- There shouldn't any side effects of the code, to the rest of the application.

The idea of encapsulation is to keep classes separated and prevent them from having tightly coupled with each other.

2) Inheritance 繼承

Below is a theoretical explanation of inheritance with real-life examples. For detailed explanation on this topic refer inheritance with examples and types of inheritance in java.

- Inheritance is the mechanism by which an object acquires the some/all properties of another object.
- It supports the concept of hierarchical classification.

For example: Car is a classification of Four Wheeler. Here Car acquires the properties of a four-wheeler. Other classifications could be a jeep, tempo, van etc. Four Wheeler defines a class of vehicles that have four wheels, and specific range of engine power, load carrying capacity etc. Car (termed as a sub-class) acquires these properties from Four Wheeler (termed as a super-class), and has some specific properties, which are different from other classifications of Four Wheeler, such as luxury, comfort, shape, size, usage etc.

A car can have further classification such as an open car, small car, big car etc, which will acquire the properties from both Four Wheeler and Car, but will still have some specific properties. This way the level of hierarchy can be extended to any level. Java Swing and Awt classes represent best examples for inheritance.

3) Polymorphism 多型

Below is a real-life example of polymorphism. For the example program and more details on this OOP concept refer polymorphism in java and runtime & compile time polymorphism.

- Polymorphism means to process objects differently based on their data type.

- In other words it means, one method with multiple implementation, for a certain class of action. And which implementation to be used is decided at runtime depending upon the situation (i.e., data type of the object)
- This can be implemented by designing a generic interface, which provides generic methods for a certain class of action and there can be multiple classes, which provides the implementation of these generic methods.

Let us look at same example of a car. A car have a gear transmission system. It has four front gears and one backward gear. When the engine is accelerated then depending upon which gear is engaged different amount power and movement is delivered to the car.

Polymorphism could be static and dynamic both. Overloading is static polymorphism while, overriding is dynamic polymorphism.

- Overloading in simple words means two methods having same method name but takes different input parameters. This called static because, which method to be invoked will be decided at the time of compilation
- Overriding means a derived class is implementing a method of its super class.