Part III 物件導向設計

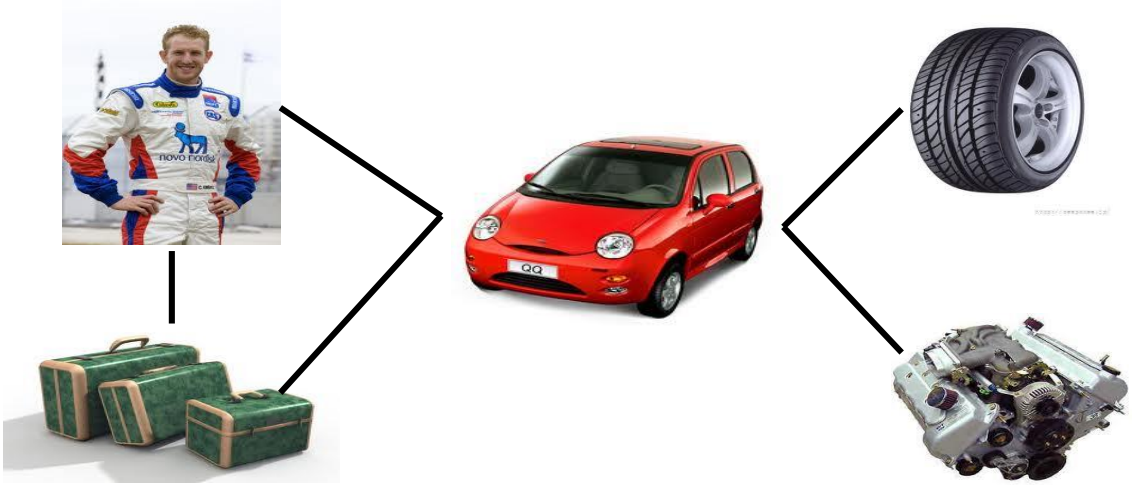
國立成功大學 資訊工程系
李信杰 副教授

*本教材大量引用自國立台北科技大學陳偉凱教授編著之物件導向程式設計軟工輔助教材

基本物件觀念

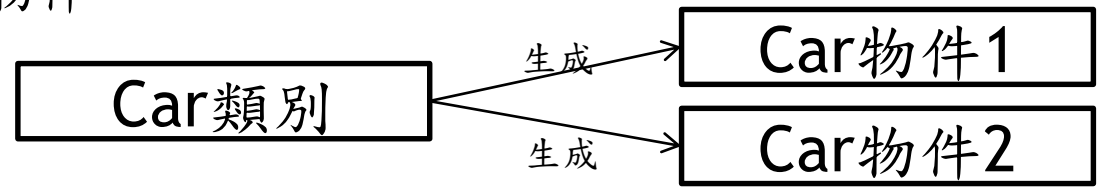
物件(Object)

- 物件導向程式設計的主體是物件，他們合作完成一個系統的運作。
- 物件導向程式(object-oriented program)可視為多個具互動關係之物件的集合



類別(Class)與物件(Object)

- 類別(class)
  - 類別是抽象的模板(template)，不具備實際的值(value)。
  - 一群具相同屬性(attribute)、操作(operation)或方法(method)、關聯(association)與行為(behavior)的物件之模版，描述具相同特徵的一群物件。



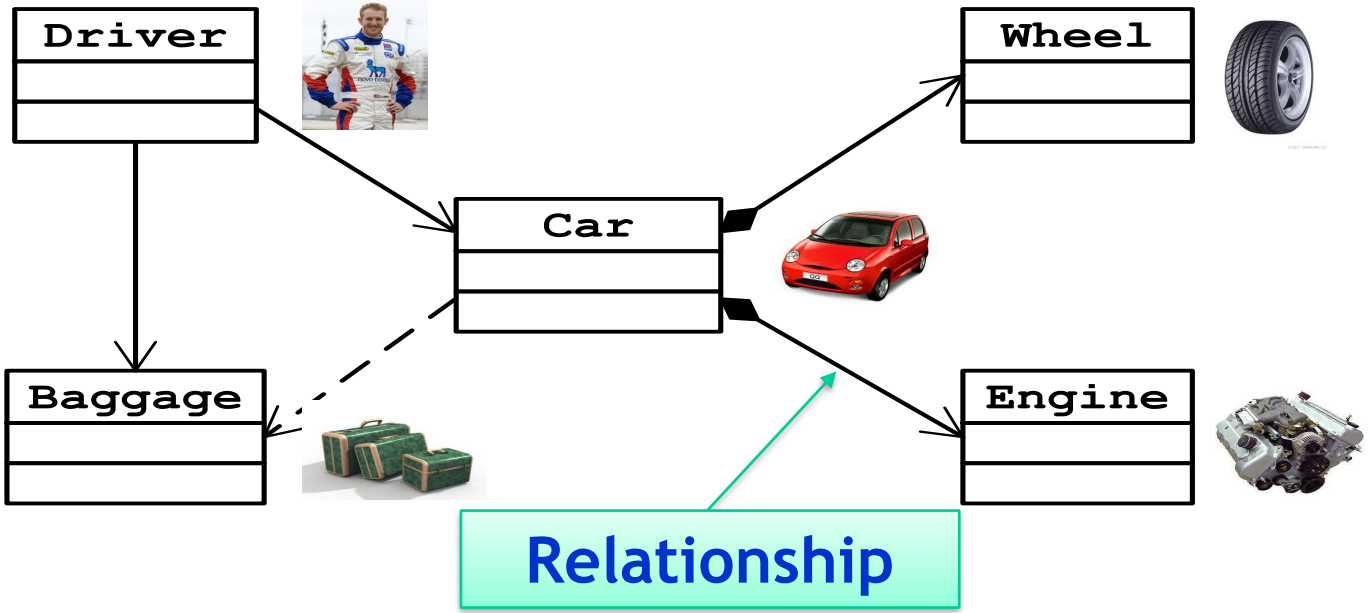
- 物件(object)
  - 又稱實例(instance)，是具象的實體。
  - 由類別產生，具有實際的值。

屬性(Attribute)與方法(Method)

- 一個類別所產生的物件，都具有相同的特性，包含屬性(attribute)與方法(method)。
- 類別定義了屬性，但物件會有不同的屬性內容。
  - 每部車子的車牌號碼不同、油量不同、車速不同。
- 物件雖然有相同的方法（或演算法），但表現出來的行為不一定會一樣。
  - 它在執行的過程中會參考到物件本身的資料（狀態），所以表現出來的行為還是不同的。
  - 每部車子都有啟動、煞車、加速等功能，但每部車會呈現出不同的車速。

UML Class Diagram: 範例

- 統一塑模語言(UML, Unified Modeling Language)可用來表達物件內涵和物件間的互動關係



Class in UML

Class Diagram:

ClassName
-attribute1 #attribute2 +attribute3
-method1() #method2() +method3()

Example:

Car
-color -speed
+start() +brake() +speedUp()

IntArray
-array -size
+sort() +getSize() +printOutput()

```
+    public
#    protected
-    private
```

Lab 3-1

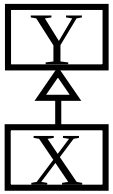
- 請依據底下程式碼繪製出Class Diagram

```
class Dog {
    private int age;
    public int color;

    public void run(){…}
    private void sleep(){…}
}
```

類別關聯性(Relationship)

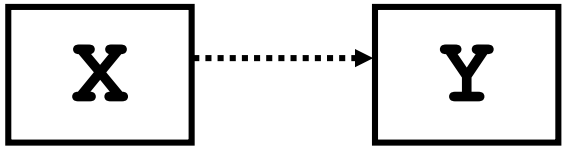
- Dependency  (uses a)
- Association  (knows a)
- Composition  (has a)

- Inheritance  (is a)

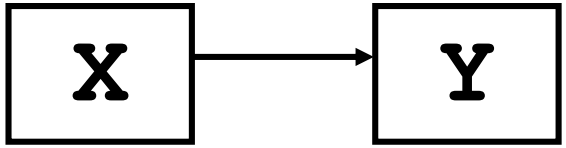
相依與關聯

“Uses a” ⇔ “Knows a” Relationship

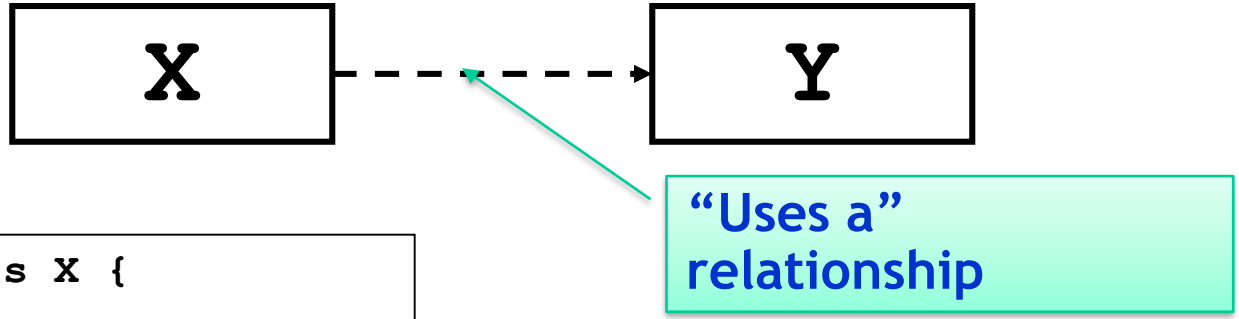
- “Uses a” (使用)
  - Dependency (相依)
  - X使用Y：X呼叫Y的method
  - **Short term relationship (短期關係)**



- “Knows a” (熟識)
  - Association (關聯)
  - X熟識Y：X包含Y的reference (Y是X的一個資料成員)
  - **Long term relationship (長期關係)**



Dependency程式範例1




```
class X {

    void f1(Y y) {
        y.foo();
    }
}
```

Dependency程式範例2


```
class Car {
    private double weight;

    public void addBaggageWeight
        (Baggage baggage){…}
}
```

Car uses Baggage

Lab 3-2

- 請依據底下程式碼繪製出Class Diagram

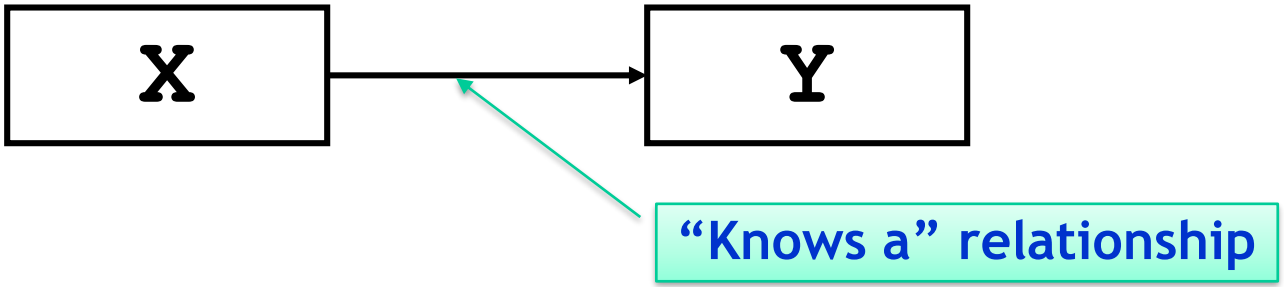
```
class Person {

    void travel(Train t) {
        t.take();
    }
}
```

```
class Train {

    void take() {
        …
    }
}
```

Association程式碼範例1



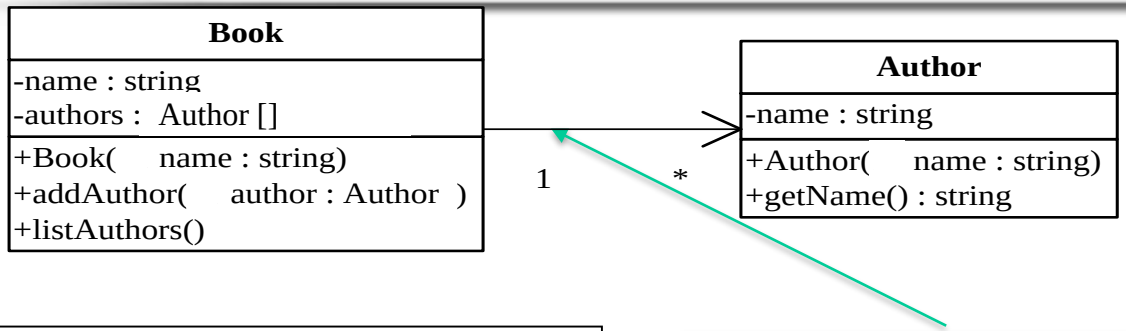

```
class X {
    // X keeps a reference to Y
    private Y y = new Y();
    private int age = 20;


}
```

```
class Y {


}
```

Association程式碼範例2

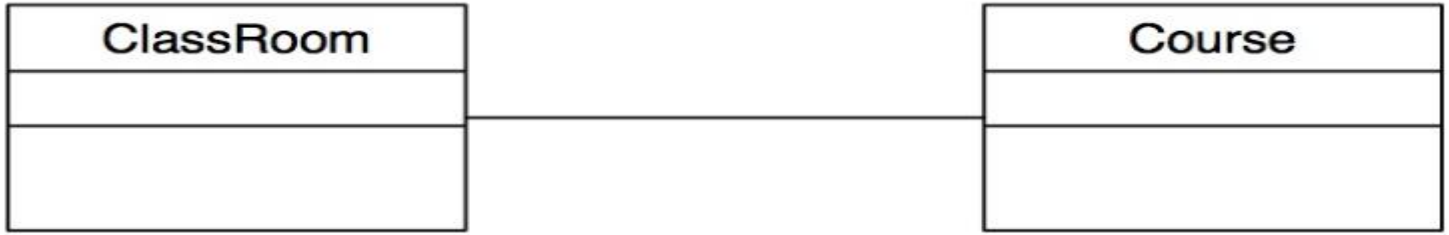



```
class Book {
    private string name;
    private Author[] authors;

    public Book(string name);
    void addAuthor(Author author) {…}
    void listAuthors() {
        …
    }
}
```

Navigation1

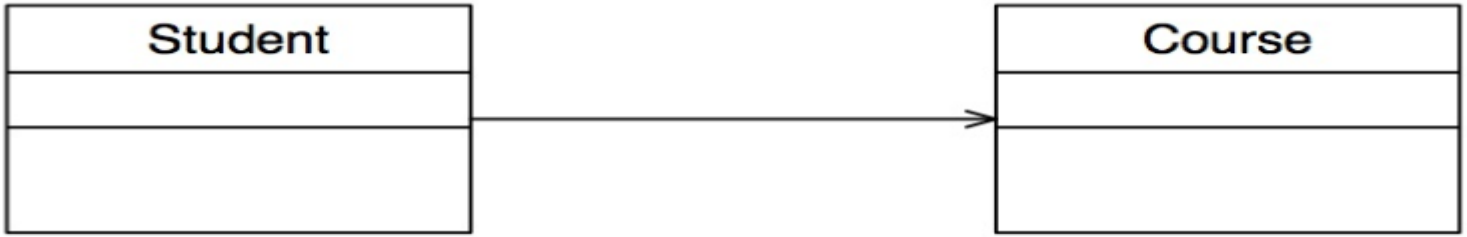
- 當兩邊的類別彼此都可看到對方時，我們可以用沒有箭頭的關聯線將兩端連接起來。




```
class ClassRoom {
    Course course;
}
class Course {
    ClassRoom room;
}
```

Navigation2

- 當Association關係中只有一方看得到另外一方，我們則用有箭頭的關聯線表達這樣的可視性(navigation)。




```
class Student {
    Course course;
}
```

```
class Course {
}
```

Role Name (角色名稱)

- 在 Association 的端點上的 role name，可轉換為類別中的屬性，此屬性記錄參與此關係的另一個物件。


```
class Professor {
}
```

```
class Course {
    Professor instructor;
}
```

Multiplicity (數量)

- 關聯端點上可寫上數量，代表此參與此關聯關係之物件個數。

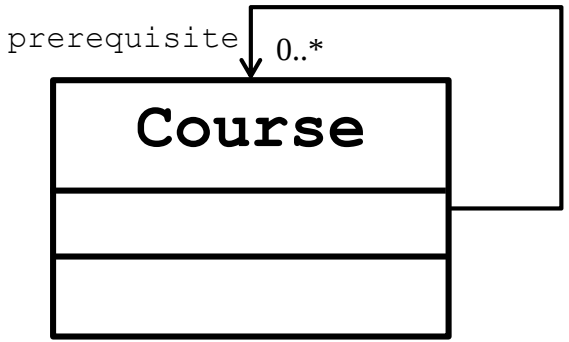

```
class Student {
    Course[] courses = new Course[10];
}
```

```
class Course {
}
```

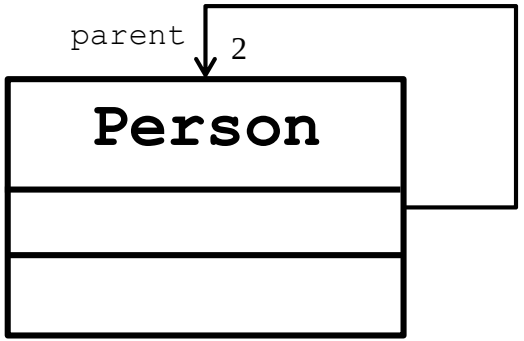
(一個學生可以上0到10門課)

Self-Association（自我關聯）

- 同一個類別的物件彼此有關係。
  - 例如：課與課之間有『先修』的關係、Person的parent仍是Person




```
class Course {
    ArrayList<Course> prerequisites
= new ArrayList<Course>();
}
```




```
class Person {
    Person parents []
= new Person [2];
}
```

Lab 3-3

- 請依據底下程式碼繪製出Class Diagram

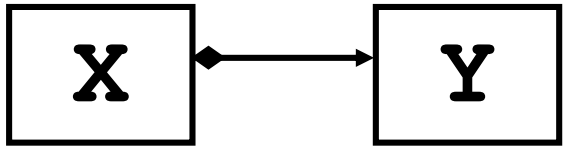
```
class Person {
    CellPhone phone =
        new CellPhone();
}
```

```
class CellPhone {
    int status;
}
```

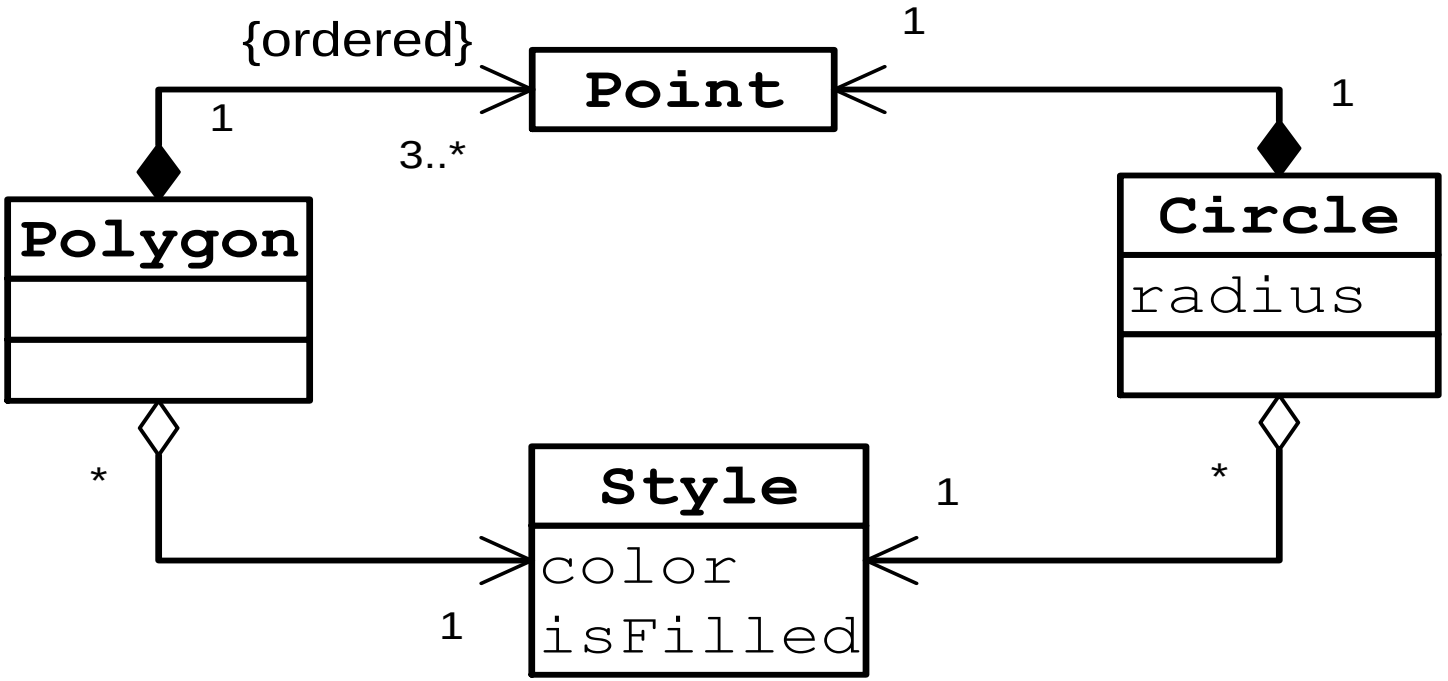
組合

Composition

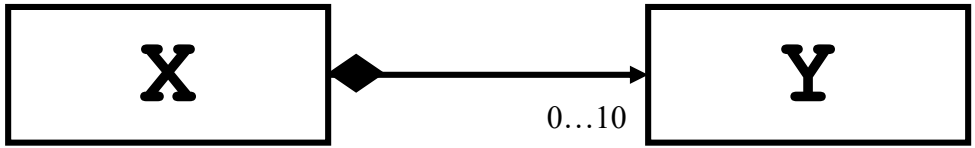
- 組合(composition)
  - “Has a” 或“part-of”之關係
  - 在組合關係中，若Y包含於X，則Y不可被其他物件包含
  - X與Y之生命週期一致：若刪除X，則Y隨之被刪除。



範例: "has a" 關係


{ordered}
1
1
3..*
1
Point
Polygon
Circle
radius
*
1
*
1
Style
color
isFilled

Composition程式範例




```
class X {
    Y[] y;    // 0..10; Composition
    public X() { y = new Y[10]; }
    protected void finalize(){ y = null; }
    …
}
```

Lab 3-4

- 請依據底下程式碼繪製出Class Diagram

```
class Directory {
    File[] files;

    delete(){
        for each f in files
                f.delete();
    }

}
```

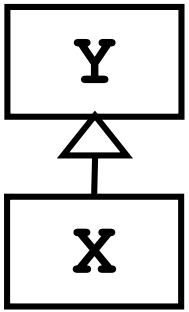
```
class File {

    delete(){
        …
    }
}
```

繼承

“Is a” ⇔ “Can do” relationship

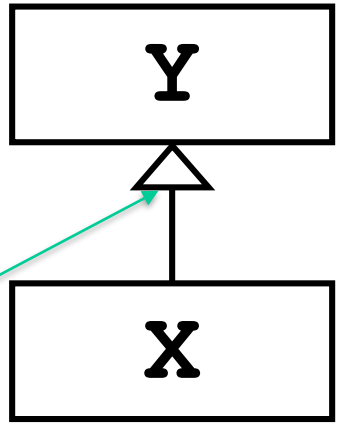
- “Is a” 或”Is a kind of”關係
  - 繼承(inheritance)
  - X is derived from Y



Inheritance程式範例

```
class Y {
    ...
}

class X extends Y {
    ...
}
```



"is a" relationship

Lab 3-5

- 請依據底下程式碼繪製出Class Diagram

```
class Fruit {

}
```

```
class Apple extends Fruit{

}
```

```
class Banana extends Fruit {

}
```

類別圖
(Class Diagram)

使用UML工具發展Class Diagram

- StarUML
  - https://staruml.io

需求敘述

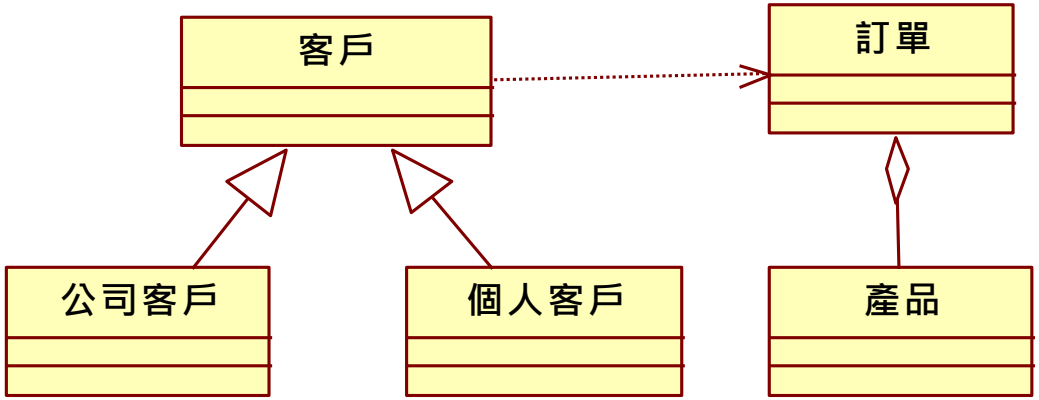
- 客戶有兩種，一類是公司客戶，另外一類為個人客戶。
- 客戶有名稱、住址等屬性資料。
- 個人客戶則額外有信用卡帳號資料。
- 公司客戶額外有信用狀態資料。
- 客戶會下訂單買產品。
- 產品需記錄編號、名稱、價錢等資料。

識別出所有的類別(Class)

- 找出可作為類別的名詞：
  - 客戶、公司客戶、個人客戶、訂單、產品

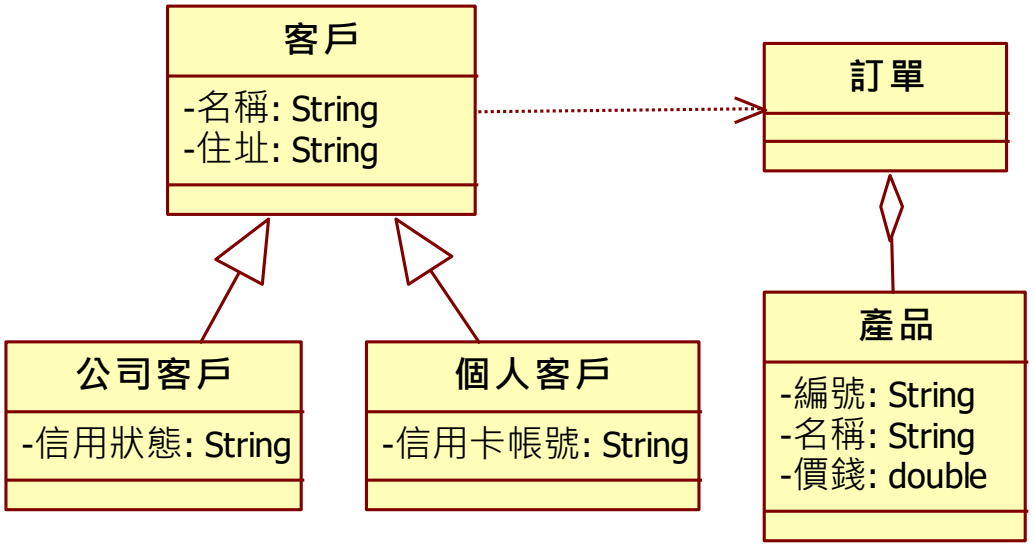
建立類別間的關係

- 客戶有兩種，一類是公司客戶，另外一類為個人客戶。
- 客戶會下訂單買產品。



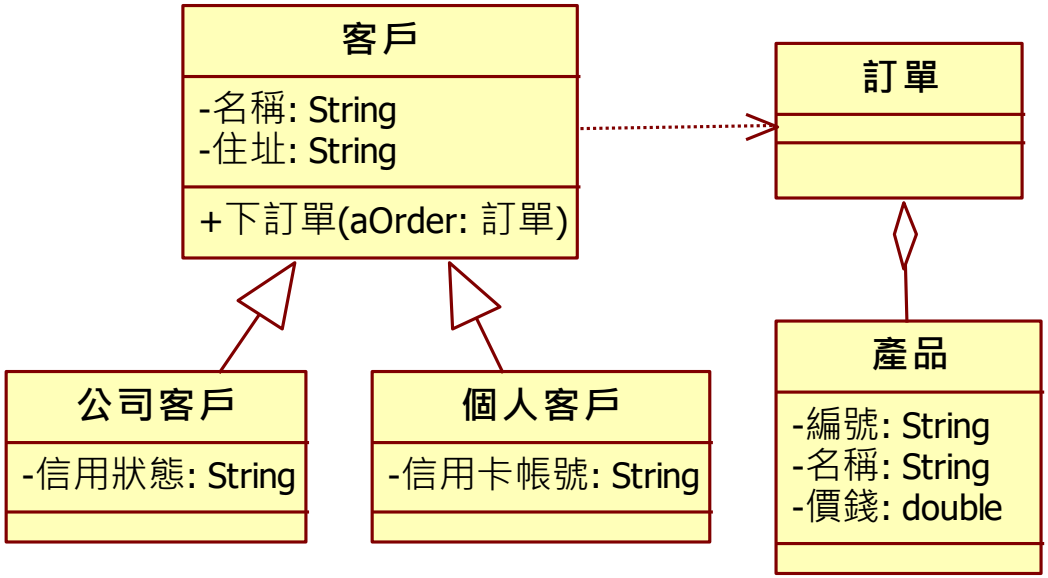
設定屬性(Attribute)

- 客戶有名稱、住址等屬性資料。
- 個人客戶則額外有信用卡帳號資料。
- 公司客戶額外有信用狀態資料。
- 產品需記錄編號、名稱、價錢等資料。



設定方法(Method)

- 客戶會下訂單買產品。



Lab 3-6

請繪製出以下的Class Diagram

- Car ⇔ Engine
- Car ⇔ Baggage
- Driver ⇔ Car
- Driver ⇔ Baggage
- Person ⇔ Cell Phone
- Human ⇔ Brain
- Fighter ⇔ Bomb
- Fighter ⇔ F16
- Bomb ⇔ Nuclear Bomb

Lab 3-7

- 依據以下需求，繪製出Class Diagram
  - 我們希望設計一個電子書店應用軟體，使用者可以從書店入口中選擇一個書店，然後依書名或作者查出書店所庫存的書，放到購物車。

Lab 3-8

- 延續Day1的假想系統，繪製出Class Diagram