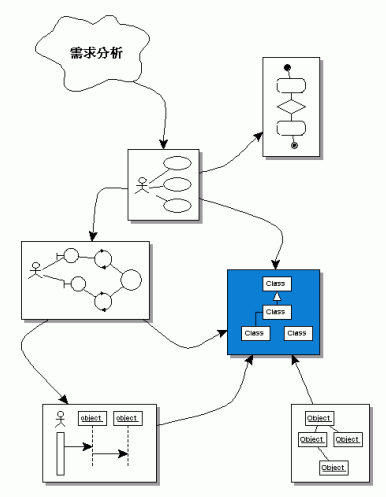
UML簡介_Class/Object Diagram

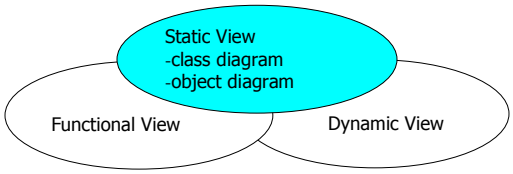
資訊科技系
林偉川

類別圖



類別圖的目的

- 塑模系統的靜態模型。
- 塑模問題領域中所發掘的物件。
- 表示系統中物件靜態的資料結構。



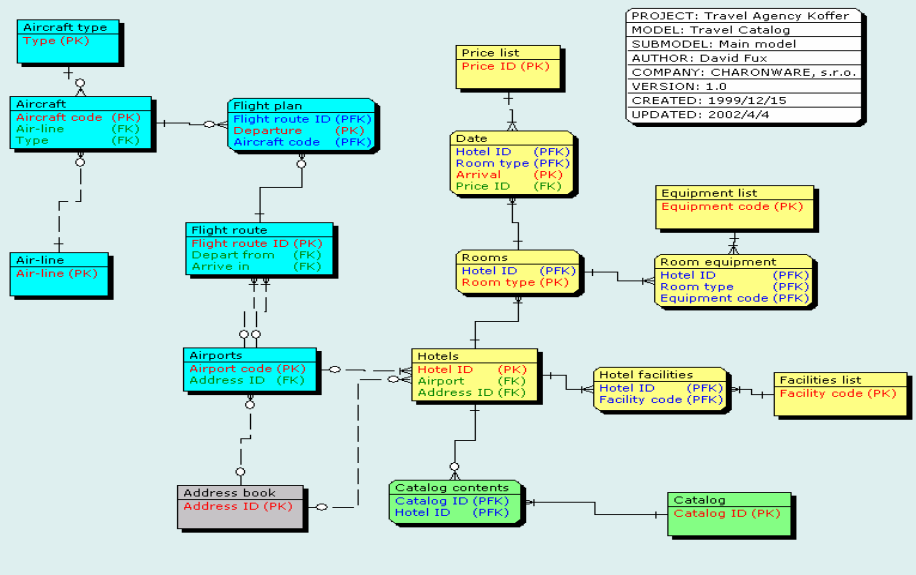
3

類似ERD

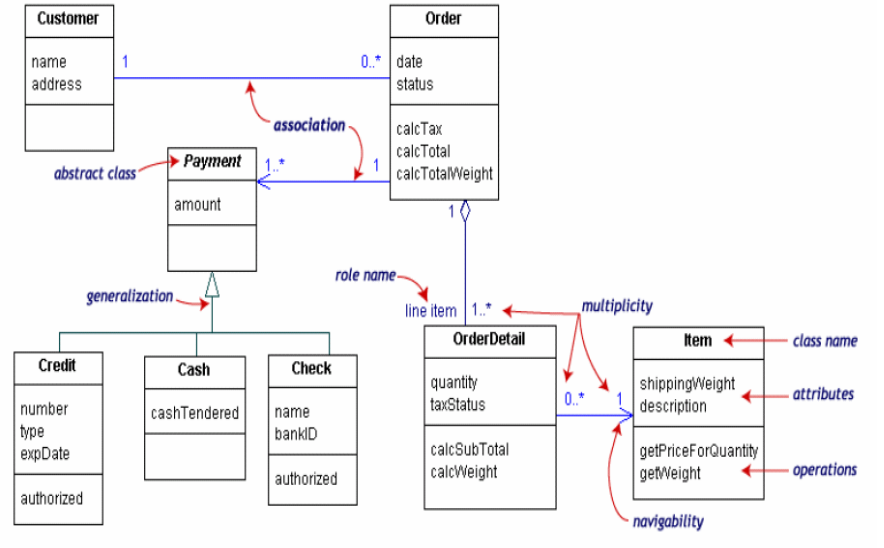
- 類別圖主要是用來做物件的資料結構塑模用的。它有點類似傳統的資料庫的實體關係圖(Entity-Relation Diagram)。 但是， ERD圖並不是以物件導向的觀點來設計。 ERD是針對資料來做設計。
- 傳統的結構化分析與設計過程可以分為處理塑模，以及資料塑模。結構化分析與設計是以處理為中心，利用所謂的"Divide and Conquer"的方式，對於處理做細分的工作。

4

ERD範例圖



類別圖範例



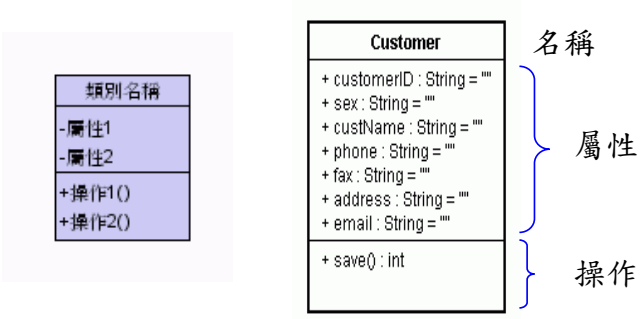
符號-類別(class)

- 類別用來描述一群具有相同屬性(attribute)與操作(operation)的物件。在UML中，類別是以一個長方形來表示，且此長方形被分為三個部份。第一部分表示類別的名稱。第二部份表示類別所具有的屬性。第三部份則是表示類別所具有的操作。
- 類別的名稱要唯一。名稱是用來識別類別用的。類別的屬性用來表達物件本質上所具有的特徵或是性質，類別的操作則表示物件所具有的行為。

7

類別圖結構

- 類別封裝了屬性及操作



8

分類

- 一個系統所牽涉的類別可能有成千上百個。一種將類別分類的方法是依據在現實世界中所代表的涵義。
- 類別的分類:
  - 跟問題領域相關的類別稱為領域問題類別。
  - 用來處理商務邏輯(business logic)的類別。
  - 跟使用者介面相關的介面類別。
  - 用來做資料存取的資料存取類別。
  - 抽象化概念。

9

領域問題類別

- 在分析階段，只對領域問題類別有興趣。
- 領域問題類別指的是那些在討論的領域中重要的名詞概念。這種類別也稱為永續類別(persistent class)。Persistent的意思表示這些物件所代表的資料會被儲存在資料庫裡面以做為後續的存取之用。一個購物系統中，顧客不僅是此領域中重要的概念，並且要將其資料儲存起來以供後續之用。可以利用類別來捕捉這個重要的概念。

10

永續

- 上圖只顯示出類別名稱。在分析階段，可以只寫出名稱。在計畫的進行中，會發覺出類別的相關屬性以及操作。
- 要強調一個類別是persistent class時，可以在類別名稱前面加上<<persistent>>這個stereotype 來標記一個類別是屬於永續類別。

11

用英文的好處

- 在塑模類別時，會儘量使用英文。如果所使用的CASE工具具有產生程式碼的功能時，使用英文會省去後面的翻譯步驟。

12

類別與實例(Class and Instance)

- 所謂的實例(instance)代表著一個類別的實現化。實現化也就是具體化的意思。一間房子的藍圖就比如是類別。而依據藍圖所蓋出來的房子就是實例。因此，類別是抽象的，實例是具體的。由類別所建構出來的實例稱為物件(object)。

13

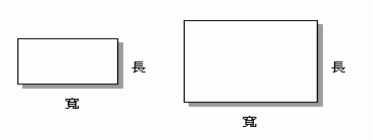
屬性(Attribute)

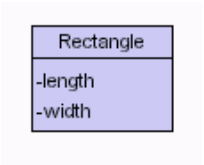
- 問題領域中所探討的類別所具有的性質、特徵、狀態稱為屬性。例如，跟雇員有關的一些屬性包括名字，雇員在哪個部門工作等等。一個雇員的膚色似乎就沒有那麼重要。在分析的階段，對於問題領域是重要的屬性需要被包含在類別中。

14

形狀的類別

- 若在設計一個繪圖軟體，它可以繪製長方形(Rectangle)。會去定義一個類別就叫做長方形。對於任意一個長方形，它所擁有的性質諸如面積，周長皆可由長(length)跟寬(width)來決定。那麼長跟寬均可算是長方形類別的屬性。不同的長或是寬代表的是不同的長方形物件，也就是長方形類別的實例。長方形的類別圖





15

屬性的能見度(visibility)

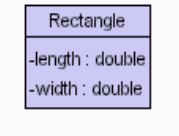
- 屬性可以宣告其能見度。這是物件導向所提供的封裝機制 (encapsulation)。封裝的機制用來保護一個類別的屬性。使得類別的屬性不會被任意修改。
- 能見度的種類有以下三種：
  - 公共的(public)
  - 保護的(protected)
  - 私有的(private)

16

屬性資料型態

- 每個屬性都有特定的資料型態。屬性的資料型態有基本資料型態或是類別型態。所謂的基本資料型態，以Java語言為例，指的是：boolean、int、 long、double、float、char、String。
- 顯示出一個長方形的長跟寬都是double的型態。UML類別中的屬性宣告格式為

  – 能見度 變數名稱：資料型態



17

操作(Operation)

- 操作是一個類別可以執行的動作或是功能。操作代表著這個類別能接收到的訊息(message)。 一旦接受到訊息，類別就有責任對這個訊息做相對的處理或是運算。操作可以看成是一個類別所應具有的責任。
- 在分析階段，類別可以執行的動作稱為操作。到了後面的設計階段，操作會被轉換成方法(method)。因為方法跟實作比較有關，所以在分析階段，大部分都使用操作來描述類別的行為。

18

例子

- 只關心跟問題領域相關的操作。例如：課程這個類別可以有"計算成績"這個操作。購物車這個類別可以有"加入訂購項目"這個操作。

19

基本型態的操作

- 一個典型的類別會包含有三種基本型態的操作，但這並不是必須的。
- 建構子(constructor operation)
- 詢問的操作(query operation)：詢問的操作有時候也叫做 getter。
- 更新的操作(update operation)：更新的操作有時候也稱為 setter 。

20

操作的注意事項

- 在分析階段只寫出敘述性的操作。
- 敘述性的操作在設計階段會再被分解與檢討。在分析階段儘量避免去描述操作的細部邏輯。在分析階段，不要把焦點放在如何去實現操作。
- 操作也跟屬性一樣可以宣告能見度。這是物件導向所提供的封裝機制 。封裝的機制用來保護一個類別的操作。使得類別的操作不會被其他的物件任意呼叫。

21

操作的注意事項

- 在分析階段，不需要描繪出一個類別的所有操作。在檢視一個使用案例時，可以只描繪出跟該使用案例相關的操作即可。使用案例描述是採用主詞+動詞+受詞的形式，那麼，動詞這一部份就可能是某個物件的操作。某個物件是指受詞。

22

舉例來說

- ”客戶新增訂購項目”使用案例。
- 顧客可以新增訂購項目至購物車中。顧客要把訂購的東西加入到購物車，所以購物車這個類別應該提供一個操作”增加訂購項目” - addLineItem()。因為對於購物車是接受”增加訂購項目”命令的受詞。

23

關係(Relationship)

- 類別圖除了表示領域問題中出現的類別之外，它還可以顯示類別與類別之間的關係。對於領域中的問題，會藉由許多的物件一起合作以提供解答。因此，塑模類別之間的關係是類別圖很重要的工作。

24

物件之間的關係

- 在物件導向的世界中，可以由幾個觀點來看物件之間的關係：
  - 從類別的角度
  - 從物件的角度來看

25

從類別的角度

- 有一些關係是用來表達結構上類別與類別之間的關係。例如:繼承(inheritance)的關係，介面(interface)實現化的關係。這一類的關係可以把它看作是固定的，不會因時間的改變而有所改變的關係。用一個現實生活中的比喻，父子關係不會因為兒子成家立業搬出去住了而有所改變。

26

從物件的角度

- 物件們是利用什麼關係來一起合作的。
- 物件之間的關係並沒有如上所述那種無法改變的觀念。除了上述之家庭的關係之外，可能有在學校修課。學生跟學校之間建立了的關係。一旦學生畢了業，這個關係就消失了；學生可能是某個組織，社團的一員，學生跟此組織或是社團就有會員的關係。這些關係與學生家人的那種固定不變的關係是不一樣。在物件導向的世界中，稱這種為關聯關係(association)。
- 現時世界中物件與物件的關係相當的多而且很複雜，不會只有上述概括的幾種而已。

27

關聯(Association)

- 關聯關係依其結合的程度(degree of coupling)而分為不同形式的關聯關係。所謂的結合度是用來量度兩個物件的依賴程度。
- 當參與關係的兩個物件彼此必須依靠對方，無法單獨存在，是高度結合。如果一個物件可以獨立存在，但是如果沒有另一個物件的幫忙無法完成某些事項，可以說這些物件比較不高度依賴。如果兩個物件互不溝通，也沒有必要分享資料，這些物件沒有結合度。

28

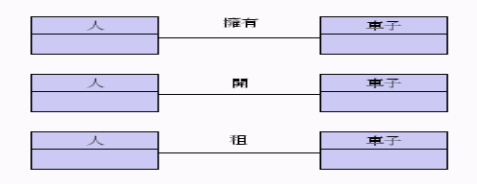
設計的原則

- 在所建立的物件中，讓它們之間的關係僅可能地鬆散，但是又能夠有效率地完成它們所該執行的工作。這是在物件導向分析與設計中必須謹記的一個原則。

29

關聯關係

- 關聯關係代表物件與物件在結構上的連結。關聯關係是用一條直線來連結相關的物件。然後在關聯的線上寫上兩者之間的關係名稱。
- 人跟車子可以有：人擁有車、人開車、人租車等關係。參與者相同但是之間的關係不同。要表達的關係語意很清楚時，可以省略掉關係名稱。



學生修課

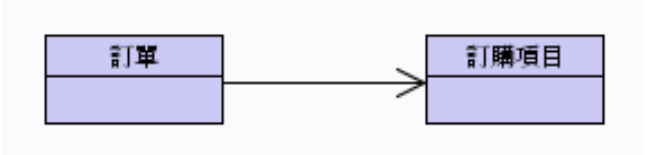
- 學生修課表示了"學生"與"課程"之間的關係。對於課程，還有"老師教導課程"的關係。可以將此三個類別及其關係繪製如下：

- 給定一個學生，可以找出所修的課程。給定一個課程。可以找出修此課程的學生。同樣的解釋情況也適用於老師與課程關係。

31

航行方向(navigation)

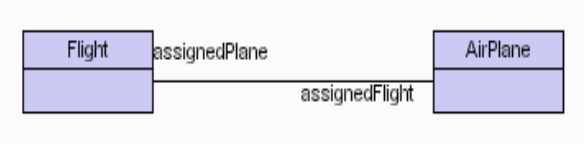
- 關聯關係中，沒有箭頭的直線代表著彼此雙方都知道對方的存在。有時候，希望表達單一的航行的方向，那麼可以用只帶有單方向箭頭的直線。例如，一個訂單可以有許多訂購項目。給定一個訂單，可以找出訂購項目，但訂購項目不需要知道它是屬於哪個訂單。此種情況可以塑模如下：



32

角色名稱

- 在關聯關係中，可以給定參與物件在關係中所扮演的角色。
- 對於一個航空班機資訊系統，可能有的兩個類別為航班和飛機。飛機被指定某個航班，而一個航班也有指定的飛機。此種情況可以塑模如下：



- 值得一提的是，有些CASE工具會將角色名稱用來作為產生程式碼時類別變數的名稱。

33

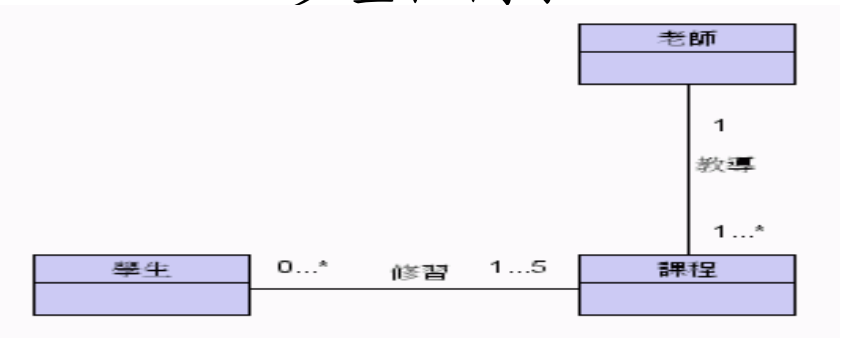
多重性

- 在關係連線的兩端，可以給出多重性。多重性用來表示參與此關係之物件的數量。其表法有下列幾種:

名稱	表法	例子
恰好一個	1	一個系有一個系主任
零個或是更多	0…*	教師有零個或是多個行政工作
一個或是更多	1…*	學生主修一個或是多個學位
零或是一個	0…1	教師有一個或是零個計畫補助
指定範圍	2…4	職員一年可以享有兩個到四個假期

34

多重性例子

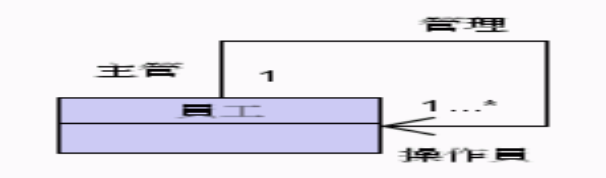


- 此類別圖：一個學生可以修1到5門課程。一門課程可能沒有人休(0)或是有很多人修(*)。老師可以教導一門以上的課程。一門課程只有1位老師。

35

反身關聯(reflexive association)

- 反身關聯指的是相同類別中的物件間彼此關係。



- 上圖在表達一位主管管理多位操作員。主管以及操作員都是由員工類別來表示。

36

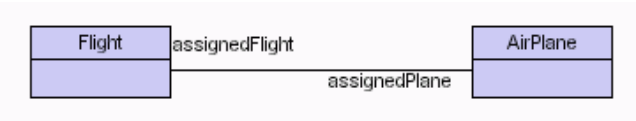
程式碼

- 先來看看關聯關係在程式碼中是如何體現的。相信對於有物件導向程式設計基礎的讀者會有一個比較具體的了解。關聯關係代表物件與物件在結構上的連結。那麼，在程式碼中，這種關係看起到底是像什麼？

37

程式碼看關聯關係

- 利用Java語言來解說關聯的概念。上面所提到的航班跟飛機之間的關聯關係。一架飛機會有它所屬的指定航班，而一個航班也會指派某一架飛機來飛航。



- 它的對應程式碼則為:

```
public class Flight{
private AirPlane assignedPlane;
}
```

38

程式碼看關聯關係

- 關聯關係表示一個類別被當作是另一個類別的類別變數。可以說一個類別被當作是另一個類別的屬性或是資料成員。因此當類別A使用類別B來當作其類別變數時，它所隱含的意思就是類別A與類別B是相關聯的。這是一種類別之間結構上的關係。

39

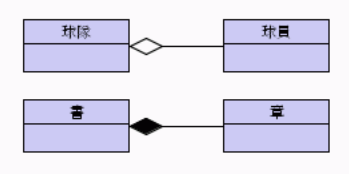
聚合(Aggregation)

- 聚合是一種特殊的關聯關係。它是用以表達“整體和部份”的關係。 聚合關係可以看作是包含(include)的關係。因此，聚合的關係可以用英文中的”is-part-of”、“has-a” 或是”has-parts” 語意上的關係來表示。聚合的關係不只是指出物件相互了解的關係， 它們被組合起來以行成一個新的更複雜的物件。
- 在UML中，聚合關係以一個空的菱形來表示代表整體的一方。

40

組合(Composition)

- 組成關係是一種比聚合關係更強的包含關係。在一個聚合的關係中，如果整體的消失會造成部分(parts)的消失，那麼這個聚合是一種組合關係。一本書包含有很多章節。如果書沒了，章節也就沒了(組合)。一個球隊有球員，球隊如果沒了，球員還可以到別的球隊打球(聚合)



41

程式碼

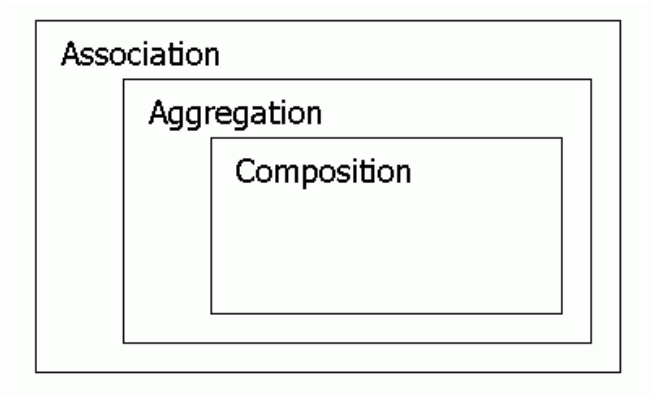
- 利用Java語言來看一下在程式碼中這種關係是如何體現的。

```
public class Team{
   private Player[] players;
}

public class Player{
// 屬性
}
```

42

關係的強弱



43

一般化的關係(generalization)

- 一般化的關係是一種分類的關係(taxonomic relationship)。此關係將比較一般的元素(稱為父類別)跟比較特別的元素(稱為子類別)分類。子類別除了跟父類別完全一致之外，它還另外具有其他的父元素所沒有的成分。除了類別圖可以有一般化的關係之外，一般化的關係也可以應用在使用案例等相關的UML圖中。

44

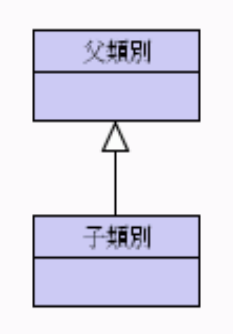
繼承的關念

- 一般化講的其實就是物件導向程式語言中的繼承觀念。一般化可以用英文中的"is-a"或是"is a-kind-of" 關係來表示。一個判斷一般化的簡單原則就是：在中文裏如果可以用"物件A是物件B的一種"這一句話來表達時，A跟B之間就可能存在著一般化的關係。例如：秘書跟工程師都是雇員的一種。雇員跟客戶都是人的一種。

45

一般化關係表法

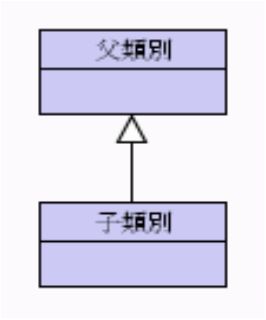
- 在UML中，一般化關係的表法是一條有著空心三角型的直線。畫法則是從子類別連接到父類別，如下所示：



46

程式語言中

- 在Java語言中，一般化關係是利用 extends 來表示。

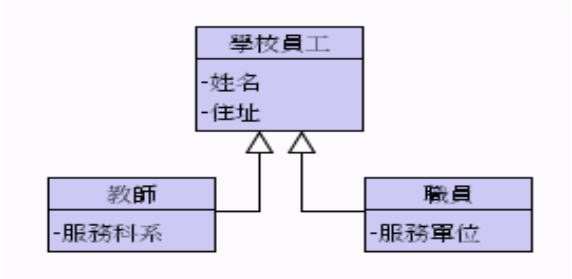



```
public class Parent{
//
}
public class Child extends Parent{
//
}
```

47

例子

- 對於一個校務資訊系統，它的使用者可能為學校授課的教授或者是各不同層級的職員等等。不管是哪種使用者，相同的是他們都會有姓名、住址等屬性。不同的地方，則是不同的角色所服務的場所。



48

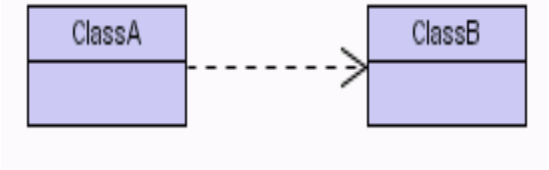
相依 (Dependency)

- 從UML的定義，相依關係是指兩個類別之間語意上的相依關係。就是當一個類別"使用"到其他類別所提供的服務時，我們稱這兩個類別有相依關係。當一個類別A會傳送訊息(passing message)給另一個類別B時，即為類別A使用到類別B。因此，類別A相依於類別B。

49

傳遞訊息

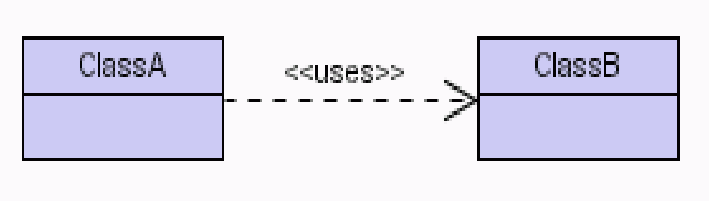
- 類別A傳送訊息給類別B，也就是類別A呼叫了類別B的操作。
- 相依關係的符號表法是用一條有箭頭的虛線來表示。相依關係的箭頭是由使用類別指向被使用類別。下圖顯示出類別A(ClassA)相依於類別B(ClassB)。



50

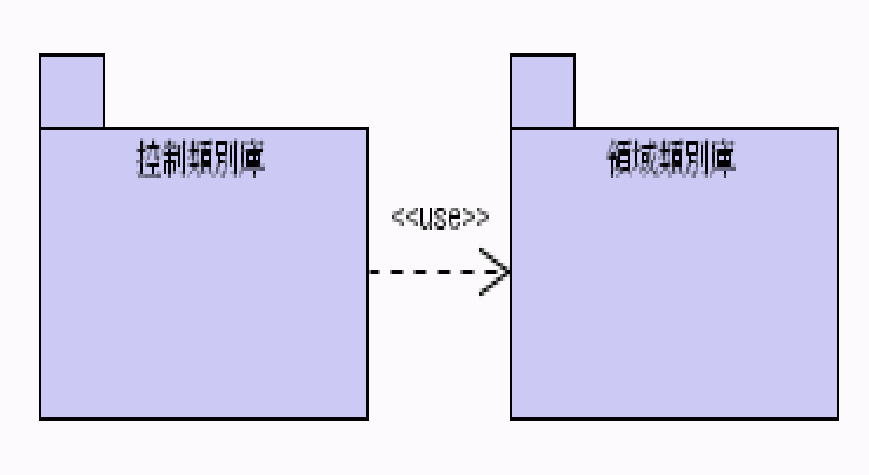
<<uses>>

- 相依關係的上面可以加上stereotype的字來說明此相依關係。UML本身定義了以下幾個關鍵字(keyword)：access, bind, derive, import, refine, trace, use。所以，上圖可以繪製如下



51

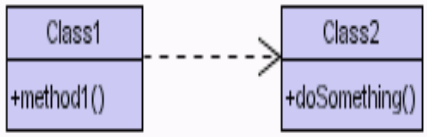
類別庫相依



52

程式碼

- 假設有一個類別Class1與Class2有相依的關係。並且給出這兩個類別的一些操作。

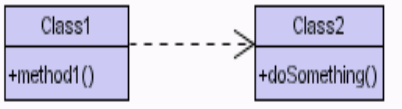


- 由定義上，可以得出相依的關係可能存在於以下幾種情況：

53

一個其他類別的區域變數

- 這種情形可用下列程式碼來表達以說明Class1相依於Class2。




```
public class  Class1{
    public void method1() {
        Class2 p = new Class2();
        // 其他敘述
        p.doSomthing();
    }
}
```

54

操作的參數中使用到其他類別

```
public class  Class1{
     public void method1(Class2 p) {
        //
        p.doSomthing();
     }
}
```

55

使用到一個類別的static method

```
public class  Class1 {
     public void method1() {
        //
        Class2.staticMethod();
     }
}
```

56

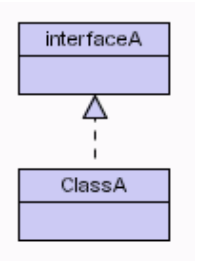
相依關係與關聯關係

- 相依關係是一種較弱的關係。它並沒有關聯關係中那種"擁有"的涵義。
- 舉例說，打電話要外帶Pizza。"要求者"跟"Pizza店"有相依的關係。因為要求者使用了Pizza店所提供的服務。可是，要求者不需要去"擁有"Pizza店。Pizza店不會是要求者的一部份！如果這家Pizza店的東西不好吃，下次要求者可以換另一家店，要求者可以從此不再向這家Pizza店買Pizza。所以，"要求者"跟"Pizza店"不會是關聯關係。"要求者"跟"Pizza店"沒有結構上的關連。

57

實現化(Realization)

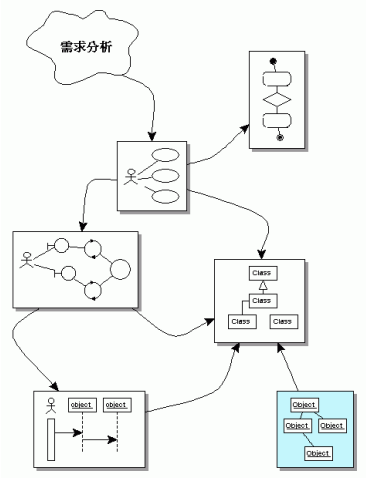
- 實現化關係是用來表達一類別之行為是由另一類別來描述定義。在Java中，它是用implements來表示。被實現的類別型態一定是介面(interface) 型態，而不是類別(class)型態。




```
public class  ClassA implements interfaceA {
//
}
```

58

物件圖



59

物件圖目的

- 塑模出問題領域中所參與的實際物件，藉以幫助發覺新的類別。
- 檢驗類別圖的準確度。

60

實例(instances)

- 物件圖是由實例(instances)所組成的圖。所以，物件圖有時也稱為實例圖(instance diagram)。實例就是由類別所建構出來的實體物件。
- 物件圖與類別圖很類似，圖中表示的是物件而不是類別。因此，物件圖包含參與的物件、相關屬性的屬性值、還有物件與物件之間的關係，關係表法中不包含多重性。

61

物件圖中並不表達操作

- 物件圖中並不表達操作。這是因為對於任一個由同一類別所建構的物件，其操作都是相同的。操作的表達在物件圖中變得多餘，因此可以省略。

62

物件圖表達系統的靜態結構

與類別圖的功能相同，物件圖也是用來表達系統的靜態結構。物件圖的建立並不是一定需要的。但是，利用物件圖可以幫助：

- 用來塑模關於特定實體的事實。因此，可以用來捕捉需求描述中的候選物件以及描述物件之間的關係。
- 提供問題領域內的範例來了解問題以及發掘新類別。
- 檢驗類別圖的準確度。

63

符號

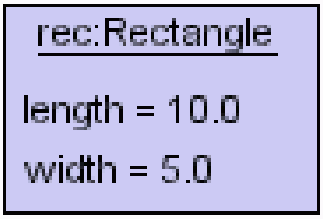
- 一個物件是以長方形來表示，像類別圖一樣。這個長方形分為兩部分：第一部分表示物件的名稱以及它所屬的類別型態。物件名稱與類別名稱之間用依個冒號：分隔開來，並且要將它們加上底線。所以，物件圖中的物件名稱寫法如下所示：

物件名稱：類別名稱

64

範例

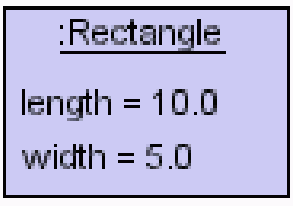
- 譬如要表達一個長10.0，寬5.0的長方形物件，這個物件的名稱是rect，它的型態是長方形(Rectangle)。那麼表達此物件的物件圖則可劃成:



65

範例

- 在物件圖中，物件的名稱可以省略而只使用分號和類別名稱。這種表法代表著一個沒有名字的(anonymous)物件。所以，上面的長方形物件圖例子可以繪製如下：



66

表達屬性

- 物件圖中屬性的表法並不需要顯示出一個物件擁有的所有屬性。在物件圖上，可以只寫出對於所討論的案例中感興趣的屬性即可。

67

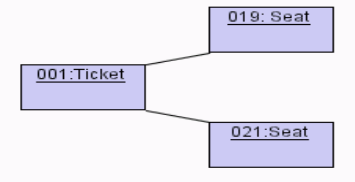
測試類別圖

- 利用一個簡單的範例－電影院訂票系統。假設當購票時，可以同時告知系統這一張票是由多少人所購買。系統會依照人數來指定同等數目的座位於票上。因此，當塑模類別圖時，根據上面的描述，可能會捕捉到如下的關係：

68

測試類別圖

- 用物件圖來表達上述的概念。

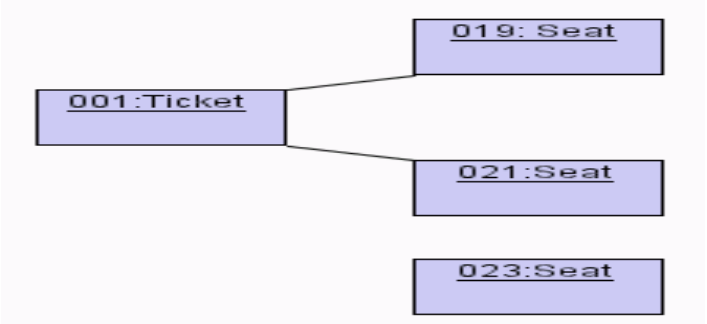


- 上面物件圖中所顯示的號碼只是用來表示不同的物件編號，做為區分不同的實例。所以，這個物件圖表示的是編號001的票，上面指定了兩個座位– 座位號碼為 019以及021。

69

測試類別圖

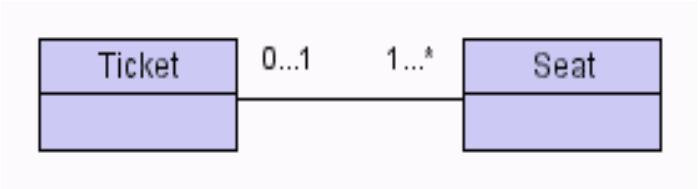
- 真實生活中的情形，是否每個座位都會被指定到某張票呢？對於不賣座的電影，通常都會有一大堆的空位。也就是說，從物件圖的觀點，下面的情形是可能的：



70

測試類別圖

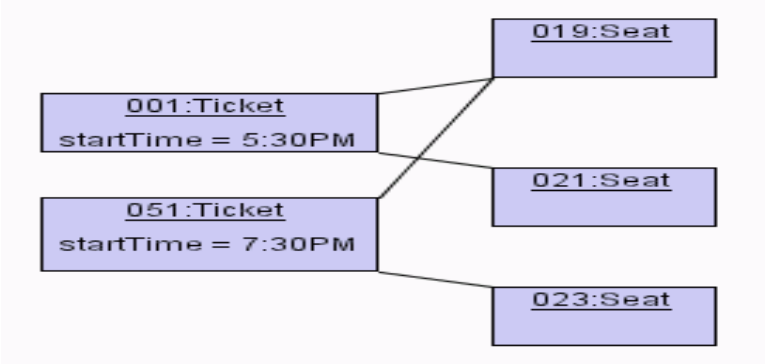
- 上圖表示座位號碼023的座位物件沒有與任何票有關聯，因為它是空位。因此，票與座位的關係可以改進變成：



71

測試類別圖

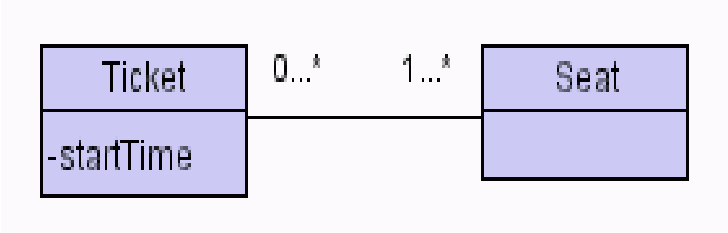
- 一個座位是否只屬於一張票呢？答案是不會的。因為對於下一場戲(播映時段不同了)，同一個座位有可能被指定到不同的票上。



72

測試類別圖

- 類別圖可以再修改為



73