第八章
認識類別

1

補充

物件導向觀念說明

2

補充

程式語言的發展分為3個時期

(1) 非程序導向
(2) 程序導向
(3) 物件導向

補充

(1) 非程序導向

- 早期的程式語言，沒有內儲副程式（又稱程式庫）
  - 當開發新的應用程式，如果某一功能與之前寫過的程式相近，則會將此段已完成的程式整段複製，並稍加修改即可重新加以利用
  - 這些程式的分身包括本尊，自從複製出來以後就開始以各自的方式繁殖
  - 結果造成各**版本的差異越來越大**，這些程式**很難弄清楚誰複製誰**
  - **彼此之間也難再共用某些程式碼**
  - 當**遇到錯誤**，或欲新增功能時，更是**很難逐一修改所有的程式**

4

補充

(2) 程序導向

- 為了解決以上**程式共用問題**
  - 各編譯器廠商便開始提供一些大家**常用的函式**
  - 比較有規模的軟體設計公司亦會將一些常用的函式集中在一個**函式庫**
  - 旗下的軟體產品一律呼叫這些標準的函式庫
  - 而不是從函式庫複製出來修改，此即為**程序導向的程式設計**。

- 程式導向與非程序導向相比，的確**解決了程式共用的問題**，但有些問題還是不夠順暢，例如有些函式庫會隨著人們需求的增加而有不同版本，當**某些函式功能增加時，我們只好重新取函式庫的函式加以修改**，並賦與新的版本名稱
- 如此日積月累，我們的函式庫已存有多許函式
  - 這些函式，有的功能相近、有的是前後版本不同
  - 有的函式裏的變數來龍去脈不明，造成使用者的混亂
  - 於是有物件導向的發展，以解決程序導向程式設計的瓶頸。

5

補充

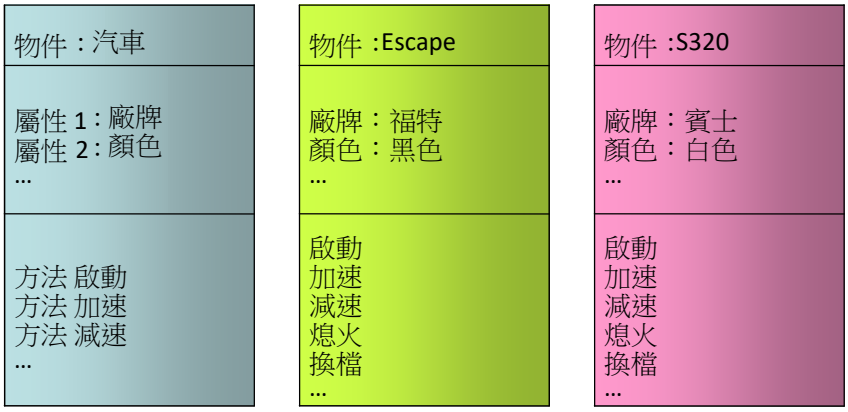
(3) 物件導向

- 程序導向中的函式，存有許多解決問題的函式，**這些函式都是解決問題的方法**，它是偏重在方法的解決，但是真實的世界是屬於較廣義的"**物件**"。
- 訂定一些共通的架構(類別)，可稱為範本。有需要時，再運用這些已訂的類別來建立需要的物件。
  - 例如，人、汽車、貓、房子或成績的計算等，都可視為**一個物件，這些物件都包含屬性及行為**
  - **描述一個人**
    - 他的名字是洪國勝，身高172、體重70。
    - 具有滾進、游泳及跑步的行為
  - **描述一輛車子**
    - 它的名字是SENTRA，排氣量是1600c.c.，耗油量是每公里0.1公升
    - 有每小時120公里的移動行為。
- 以上人與車即稱為"物件"
  - 名字、身高、排氣量則稱為"屬性"(Property)
  - 滾進、游泳、跑步、移動則稱為"行為"(Behavior)，

6

補充

生活上的物件導向概念



4-7

補充

- 既然真實世界是以物件描述物種，程式設計亦不應侷限在狹隘的方法，而是應以物件的宏觀角度撰寫程式，所以基於物件導向的新觀念，程式開發工具即制定一種新的型別，此型別稱為類別，每一類別均有一些屬性與行為。

- 而Java的類別，則採用Class宣告
  - 其內部成員則含資料(Data)與方法(Method)，其中資料代表屬性，方法(Method)代表行為。
  - 其次，類別變數即稱為物件（附註：物件亦稱為類別的實現、類別的樣例(Instance)實體。
  - 這些類別的集合即稱為類別庫，每一類別都有屬於自己的方法，也就是我們已將眾多的函式依照不同的類別分類存放，如此可解決目前日益龐大的函式命名與函式取用的困擾。

8

補充

- 程序導向
  - 關於**開門**的函式即有電梯開門、汽車開門、房子開門等數種開門的方法，如此徒增命名與取用的困擾
- 物件導向
  - 開門這個方法是附在相對應的類別裏，
    電梯類別有電梯的開門
    汽車類別有汽車的開門
    房子類別裏有房子的開門方法，大家的方法名稱都叫開門
  - 撰寫程式時也是電梯.開門，汽車.開門，或是房子.開門（物件與方法、屬性之間以點（.）運算子連結），如此既可簡化程式的撰寫，亦可減少程式出錯的機會
- **物件導向亦提出了三個觀念**
  - **物件的封裝(Encapsulation)**
  - **繼承(Inheritance)**
  - **多型(Polymorphism)**

9

補充

封裝(Encapsulation)

- 類別的封裝
  - Jave提供**private**、**protected**及**public**等存取範圍修飾子
  - **使得這些成員有不同的封裝等級**
  - **以避免程式與類別庫之間的干擾**
  - 就像電視機或汽車的一些開關與旋鈕，有些是開放給一般的使用者調整，有的是讓維修工程師調整，有些則永遠不讓任何人調整。

10

補充

繼承(Inheritance)

- 繼承是為了達成**重覆使用**目的所採取的一種策略
  - 例如：一個滑鼠類別只要加上滾輪裝置，就變成了滾輪滑鼠
  - 但滾輪滑鼠也同樣可以上下左右移動改變指標位置，也可以按兩下執行程式
  - 只不過現在又多了一個滾輪使得瀏覽網頁時更加方便
  - 事實上，這個滾輪滑鼠就是由滑鼠繼承而來的。
  - 有了這項特性，在開發大型程式時，我們就可以延續已經完成的技術，再加以擴充。

11

補充

多型(Polymorphism)

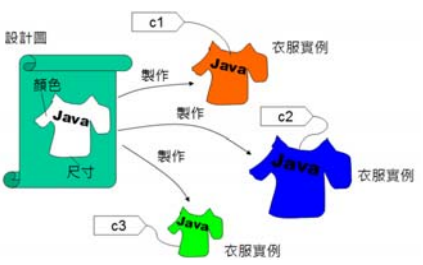
- 多型是一個非常抽象的名詞。
  - 舉例來說，所謂開水人人會煮，各有巧妙不同
  - 基本上，瓦斯爐、電磁爐、熱水瓶都可以煮開水，所以『煮開水』其實是一個相當抽象的名詞，使用瓦斯爐、電磁爐、熱水瓶的煮開水方式都不一樣，但「**煮開水」一詞卻涵蓋了這些差異性的行為**
  - 而此種特質則稱為「多型性」(polymorphism)，多型性使得程式設計具有更大的彈性。
  - 本書中舉出的三種多型的例子：多載(Overloading)、改寫(Overriding) 及介面(Interface)
  - 多載(Overloading): 相同名稱，有不同的意義。

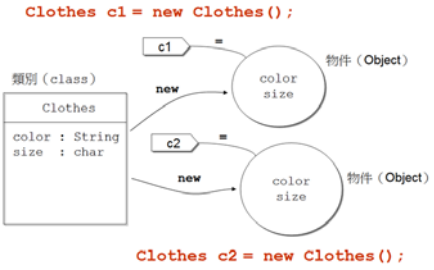
8.1 認識類別

13

定義類別

正式開始說明如何使用Java定義類別之前，先來看看，如果要設計衣服是如何進行的？你會有個衣服的設計圖，上頭定義了衣服的款式與顏色、尺寸，你會根據設計圖製作出實際的衣服，每件衣服都是同一款式，但會擁有自己的顏色與尺寸，你會為每個衣服別上一個名牌，這個名牌只能別在同款式的衣服上。

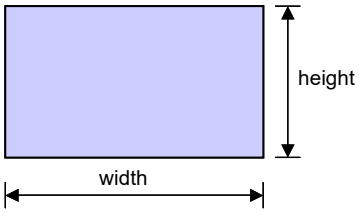




https://openhome.cc/Gossip/Java/Class.html 物件（Object）=實例（Instance）

類別的基本概念

- 每一個Java程式，至少會存在一個或一個以上的類別
- 類別是由資料成員與成員函數封裝而成
- 矩形有寬（width）與高（height）兩個基本屬性
  - 根據這兩個屬性，可求出面積（area）與周長（perimeter）



面積(area) = width*height

周長(perimeter) = 2(width+height)

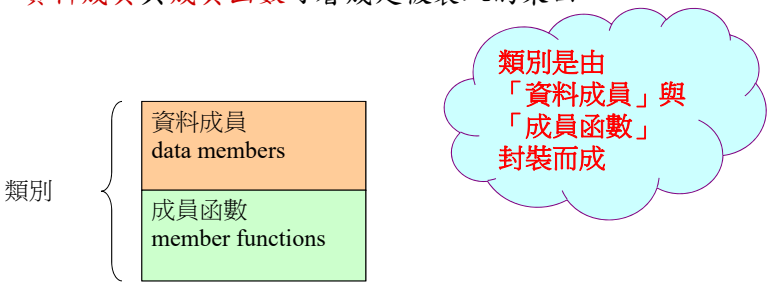
資料成員與成員函數

- 矩形具有「寬」與「高」等屬性，這些屬性也就是矩形類別的「資料成員」（data member）
- 類別內的資料成員稱為field（範疇）
- 計算面積與周長的函數可視為類別的「成員函數」（member function）
- 在OOP(object-oriented programming)裡，成員函數是封裝在類別之內

8.1 認識類別

資料成員、成員函數與封裝

- 「類別」就是把事物的資料與相關功能「封裝」（encapsulate）在一起
  - 「encapsulate」的原意是「將...裝入膠囊內」
- 類別可看成是「膠囊」
  - 資料成員與成員函數可看成是被裝入的東西

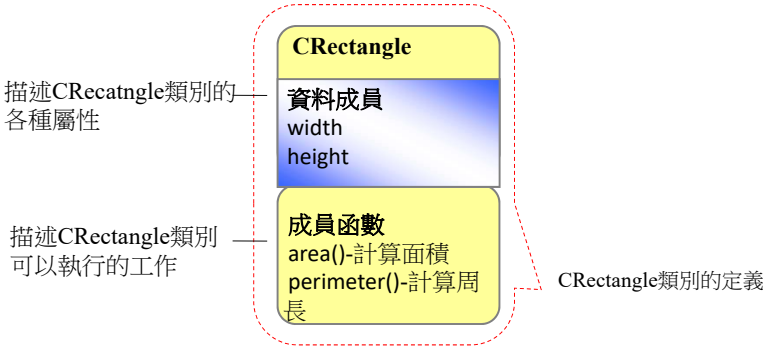


17

8.1 認識類別

矩型類別的認識

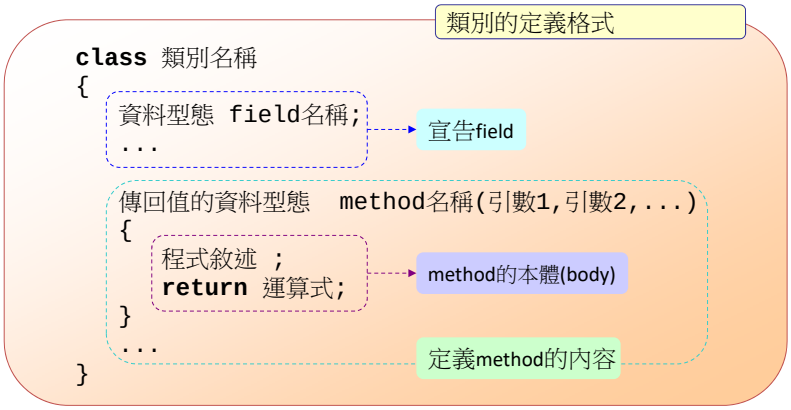
- 矩型類別：
  - 資料成員為 width 與 height
  - 成員函數為 area() 與 perimeter()



18

類別的定義

- 類別定義的語法如下：



矩形類別的範例

- 以矩形為例，可定義如下的矩形類別：

```
01  class CRectangle                  // 定義矩形類別 CRectangle
02  {
03     int width;                     // 宣告資料成員 width
04     int height;                    // 宣告資料成員 height
05
06     int area()                     // 定義成員函數 area(), 用來計算面積
07     {
08        return width*height;        // 傳回矩形的面積
09     }
10     int perimeter()                // 定義成員函數perimeter(),用來計算周長
11     {
12        return 2*(width+height);    // 傳回矩形的周長
13     }
14  }
```

本書大寫C為開頭的識別字做為類別的名稱，方便和其它變數做區隔

圓形類別的範例

- 將圓面積的計算納入圓形類別的成員函數：

```
01  class CCircle                  // 定義 CCircle 類別
02  {
03     double radius;              // 宣告資料成員 radius
04
05     double area()               // 定義成員函數 area(), 用來傳回圓面積
06     {
07        return 3.14*radius*radius;    // 傳回圓面積
08     }
09  }
```

21

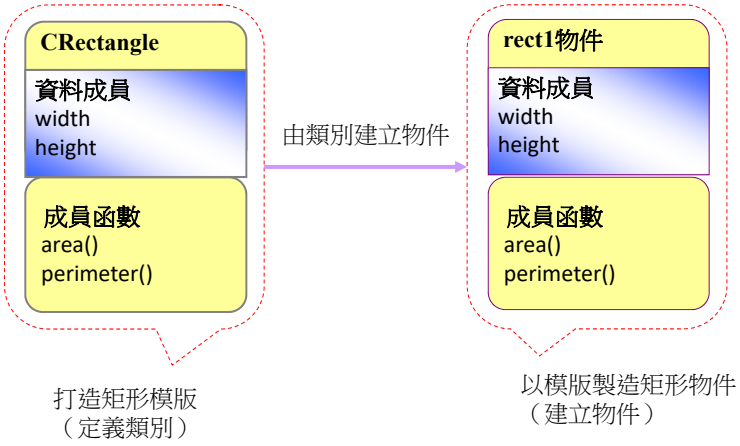
建立新物件

- 類別只是一個模版：
  - 利用它才能建立屬於該類別的物件（object）
- 以矩形類別來說，從定義類別到建立物件，可想像成：
  - 先打造一個矩形模版(定義類別)
  - 再以此模版製造矩形(建立物件)
- 由類別所建立的物件稱為該類別的 instance(例子)

22

矩形類別的物件

- 下圖是由矩形類別所建立的矩形物件rect1：



宣告與建立物件

- 欲建立某類別的物件，可藉由下面兩個步驟來達成：

  (1) 以類別名稱宣告變數

  (2) 利用new建立新的物件，並指派給先前所建立的變數

- 例如：

```
CRectangle rect1;
rect1=new CRectangle();
```

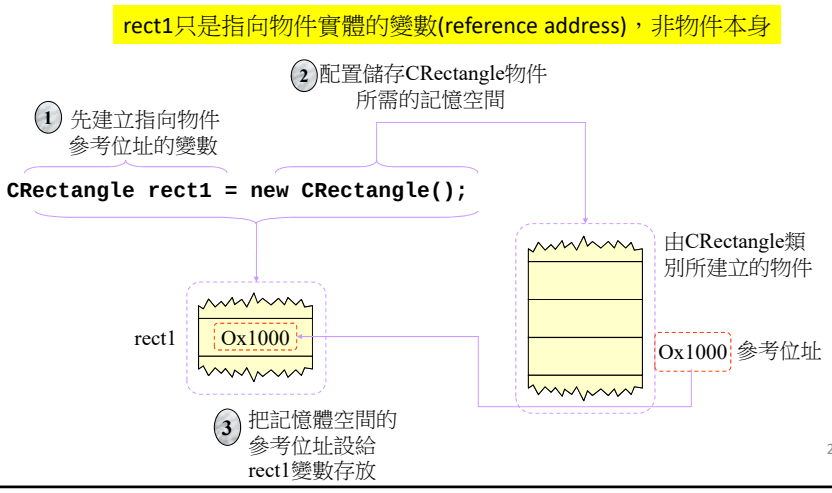
- 或是縮減成一行：

```
CRectangle rect1= new CRectangle(); // 建立新物件，並讓rect1指向它
```

**類似於：int A[]; A=new int[4]; 或者 int A[]=new int[4];

指向物件的變數

- 建立新的物件，並讓變數指向它的過程：



25

存取物件的內容

- 存取物件裡的特定資料成員，可透過下面語法來達成：

存取物件裡特定的資料成員

物件名稱.資料成員名稱

- 舉例來說，存取物件rect1的寬與高，可用下列方式：

```
rect1.width;
rect1.height;
```

26

設定物件的寬與高

- 想要將物件寬與高設值，其程式碼的撰寫如下：

```
01  public static void main(String args[])
02  {
03    CRectangle rect1;                    // 宣告變數rect1
04    rect1=new CRectangle();      // 建立新物件,並將變數rect1指派給它
05
06    rect1.width=10;                 // 設定矩形物件rect1的寬為10
07    rect1.height=5;                 // 設定矩形物件rect1的高為5
08  }
```

27

設計完整的程式

- 下面的程式碼為建立物件與field(變數)的存取之範例

```
01  // app8_1, 建立物件與field的存取
02  class CRectangle      // 定義CRectangle類別
03  {
04    int width;                    // 宣告資料成員width
05    int height;                   // 宣告資料成員height
06  }
07
```

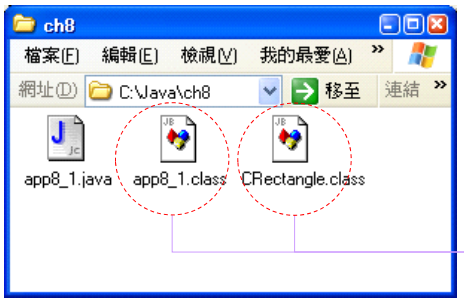
```
08  public class app8_1
09  {
10    public static void main(String args[])
11    {
12      CRectangle rect1;
13      rect1=new CRectangle();      // 建立新的物件
14
15      rect1.width=10;              // 設定矩形rect1的寬
16      rect1.height=5;              // 設定矩形rect1的高
17
18      System.out.println("width="+rect1.width); // 印出rect1.width
19      System.out.println("height="+rect1.height); // 印出rect1.height
20    }
21  }
```

```
/* app8_1 OUTPUT---
width=10
height=5
-------------------*/
```

28

編譯類別後的檔案

- Java會將每個類別編譯成獨立的.class檔案 ：



Java原始檔裡的每一個類別會被編譯成獨立的.class檔案

29

同時建立多個物件的範例

```
01 // app8_2, 同時建立兩個物件
02 class CRectangle
03 {
04   int width;      // 定義資料成員width
05   int height;     // 定義資料成員height
06 }
07
08 public class app8_2
09 {
10   public static void main(String args[])
11   {
12     CRectangle rect1,rect2;          // 宣告指變數rect1,rect2
13     rect1=new CRectangle();          // 建立物件rect1
14     rect2=new CRectangle();          // 建立物件rect2
15
16     rect1.width=10;                  // 設定矩形rect1的寬
17     rect1.height=5;                  // 設定矩形rect1的高
18
19     rect2.width=12;                  // 設定矩形rect2的寬
20     rect2.height=rect1.height+3;     // 設定矩形rect2的高
21
22     System.out.println("rect1.width="+rect1.width);
23     System.out.println("rect1.height="+rect1.height);
24     System.out.println("rect2.width="+rect2.width);
25     System.out.println("rect2.height="+rect2.height);
26   }
27 }
```

```
/* app8_2 OUTPUT--
rect1.width=10
rect1.height=5
rect2.width=12
rect2.height=8
----------*/
```

30

8.1 課堂練習

ex8_1_1.java

設類別Caaa 的定義為：

```
class Caaa
{
   int a;
   int b;
   int c;
}
```

試在程式碼裡完成下列各敘述：

(a) 試在main() method 裡建立一個Caaa 類別型態的物件obj;

(b) 將obj 資料成員a 的值設為5，b 的值設為3。

(c) 計算a*b 之後設給成員c。

(d) 印出a、b 與c 的值。

```
/* output----
obj.a=5
obj.b=3
obj.c=15
-----------*/
```

31

8.1 課堂練習

ex8_1_2.java

設類別Cddd 的定義為：

```
class Cddd
{
  String name;
  double height;
  double weight;
}
```

```
/* output---------------
name=Sandy
height=165.5cm
weight=58.2kg
BMI=21.248436943802993
----------------------*/
```

試在程式碼裡完成下列各敘述：

(a) 試在main() method 裡建立Cddd 類別型態的物件student。

(b) 將student 資料成員name 設值為"Sandy"，height 的值設為165.5（單位為公分），weight 的值設為58.2（單位為公斤）。

(c) 利用BMI=weight(Kg)/(height*height) (M) 計算此學生的身體質量指數BMI 值。

(d) 印出student 的資料及BMI 值。

**注意：公式撰寫不同，取出來的小數點位數也不同。

32

8.1 回家作業 認識類別

hw8_1_1.java

設類別Cbbb 的定義為：

```
class Cbbb
{
double x;
double y;
}
```

```
/* output--------------------
obj1.x=5.2, obj1.y=3.9
obj2.x=6.5, obj2.y=4.6
avg.x=5.85, avg.y=4.25
-------------------------------*/
```

試在程式碼裡完成下列各敘述：

(a) 試在main() method 裡建立Cbbb 類別型態的物件obj1、obj2 與avg。
(b) 將obj1 資料成員x 的值設為5.2，y 的值設為3.9。
(c) 將obj2 資料成員x 的值設為6.5，y 的值設為4.6。
(d) 將obj1 與obj2 的x 值平均後，指定給avg 的x 存放。
(e) 將obj1 與obj2 的y 值平均後，指定給avg 的y 存放。
(f) 印出obj1、obj2 與avg 的值。

8.1 回家作業 認識類別

hw8_1_2.java

請在下面的程式中填上適當的程式碼，使得物件box 的length 成員可被設為15，width成員可被設為10，height=25。

```
/* output----------
length= 15
width=10
height=25
----------------*/
```

```
class CBox
{
 int length;
 int width;
 int height;
}
public class hw8_1_2
{ public static void main(String arge[])
 {
      CBox box;
      box=new CBox();
      // 請於此處填上程式碼
      System.out.println("length=
     "+box.length);
      System.out.println("width="+box.width);
      System.out.println("height="+box.height);
 } }
```

8.2 成員函數的使用

35

定義與使用method

- 類別裡的method可用下面的語法來定義：

成員函數定義的語法

```
傳回值型態 method名稱(型態 引數1, 型態 引數2,...)
{
    程式敘述 ;
    return 運算式;
}
```

method的本體(body)

- 呼叫封裝在類別裡的method的語法：

呼叫封裝在類別內的method

```
物件名稱.method名稱(引數1, 引數2,...)
```

36

method的建立 (1/2)

```
01  // app8_3, method的建立
02  class CRectangle
03  {
04    int width;
05    int height;
06    int area()      // 定義成員函數area(), 用來計算面積
07    {
08      return width*height;            // 傳回矩形的面積
09    }
10
11    int perimeter()   // 定義成員函數perimeter(), 用來計算周長
12    {
13      return 2*(width+height);        // 傳回矩形的周長
14    }
15  }
17  public class app8_3
18  {
19    public static void main(String args[])
20    {
21      CRectangle rect1;
22      rect1=new CRectangle();   // 建立新的物件
23
24      rect1.width=10;   // 設定矩形rect1的寬
25      rect1.height=5;   // 設定矩形rect1的高
26
27      System.out.println("area="+rect1.area());
28      System.out.println("perimeter="+rect1.perimeter());
29    }
30  }
```

```
/* app8_3 OUTPUT--
area=50
perimeter=30
----------*/
```


再一個簡單的範例

- 下列的範例建立了一個圓形類別CCircle：

```
// app8_4, 圓形類別CCircle
02  class CCircle  // 定義類別CCircle
03  {
04    double pi=3.14;              // 將資料成員設定初值
05    double radius;
06
07    void show_area()             // show_area() method, 顯示出圓面積
08    {
09      System.out.println("area="+pi*radius*radius);
10    }
11  }
12  public class app8_4
13  {
14    public static void main(String args[])
15    {
16      CCircle cir1=new CCircle();          // 建立cirl物件
17      cir1.radius=2.0;                     // 設定radius的值
18      cir1.show_area();                    // 呼叫show_area()函數
19    }
20  }
```

```
/* app8_4 OUTPUT--
area=12.56
----------*/
```

field於記憶體內的配置關係 (1/2)

```
// app8_5, 圓形類別CCircle之field於記憶體內的配置關係
class CCircle        // 定義類別CCircle
{
  double pi=3.14;    // 將資料成員設定初值
  double radius;
  void show_area()   // show_area() method, 顯示出圓面積
  {
    System.out.print("pi="+pi);
    System.out.println(", area="+pi*radius*radius);
  }
}
public class app8_5
{
  public static void main(String args[])
  {
    CCircle cir1=new CCircle();    // 建立cir1物件
    CCircle cir2=new CCircle();    // 建立cir2物件

    cir1.radius=cir2.radius=2.0;   // 設定資料成員的值
    cir2.pi=3.0;                   // 更改cir2的pi值
    cir1.show_area();
    cir2.show_area();
  }
}
```

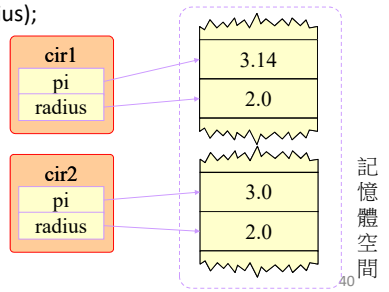
```
/* app8_5 OUTPUT---
pi=3.14, area=12.56
pi=3.0, area=12.0
----------*/
```


field於記憶體內的配置關係 (2/2)

```
01  // app8_5, 圓形類別CCircle之field於記憶體內的配置關係
02  class CCircle   // 定義類別CCircle
03  {
04    double pi=3.14;                  // 將資料成員設定初值
05    double radius;
06
07    void show_area()                 // show_area() method, 顯示出圓面積
08    {
09      System.out.print("pi="+pi);
10      System.out.println(", area="+pi*radius*radius);
11    }
12  }
```

```
/* app8_5 OUTPUT---
pi=3.14, area=12.56
pi=3.0, area=12.0
----------*/
```

不同物件的資料成員在記憶體中的配置情形



資料成員的存取方式

- 在**main()** 內存取**field**時，可透過 物件名稱.資料成員名稱

```
01 class app
02 {
03   public static void main(String args[])
04   {
05     ....
06     cir1.radius=2.0;
07     cir1.pi=3.0;
08   }
09 }
```

radius與pi均為cir1的field

- 在類別的內部使用資料成員，可直接取用它的名稱：

```
01 class CCircle
02 {
03   double pi=3.14;
04   double radius;
05
06   void show_area()
07   {
08     System.out.println("area="+pi*radius*radius);
09   }
10 }
```

可直接取用field的名稱

this的使用

- 要強調「物件本身的field」時，可在field前面加上this：

  this.資料成員名稱

  – 下面的程式碼片段是冠上this的寫法：

```
01 class CCircle
02 {
03   double pi=3.14;
04   double radius;
05
06   void show_area()
07   {
08     System.out.println("area="+this.pi*this.radius*this.radius);
09   }
10 }
```

在資料成員前面加上this，此時的this即代表取用此一資料成員的物件

但不論我們在程式中是否出現this關鍵字，Java編譯器在編譯類別時，都會自動多出隱含的參考變數this.

成員函數的相互呼叫 (1/2)


```
01  // app8_6, 在類別內部呼叫method
02  class CCircle                    // 定義類別CCircle
03  {
04    double pi=3.14;  // 將資料成員設定初值
05    double radius;
06
07    void show_area()               // show_area() method, 顯示出圓面積
08    {
09      System.out.println("area="+pi*radius*radius);
10    }
11    void show_all()   // show_all() method, 同時顯示出半徑與圓面積
12    {
13      System.out.println("radius="+radius);
14      show_area();    // 於類別內呼叫show_area() method
15    }
16  }
17  public class app8_6
18  {
19    public static void main(String args[])
20    {
21      CCircle cir1=new CCircle();  // 宣告並建立新的物件
22      cir1.radius=2.0;
23      cir1.show_all();  // 用cir1物件呼叫show_all()
24    }
25  }
```

app8_6示範如何呼叫在類別內部的成員函數

```
/* app8_6 OUTPUT--
radius=2.0
area=12.56
----------*/
```


成員函數的相互呼叫 (2/2)

- app8_6的show_all() 改成下面的敘述，可得相同的結果：

```
void show all()
{
   System.out.println("radius="+radius);
   this.show area();     // 於類別內呼叫 show area() method
}
```

在類別的定義內呼叫其它的 method，可在該 method 之前加上 this，此時的 this 即代表取用此一 method 的物件

- 假設在main() method裡有這一行敘述：

```
cir1.show_all();              // 用 cir1 物件呼叫 show_all()
```

this關鍵字即代表cir1

8.2 課堂練習

ex8_2_1.java

假設我們要設計一個CBox 類別，用來表示立體的箱子（box）。此類別內含長（length）、寬（width）與高（height）三個資料成員，其類別程式碼的撰寫如下：

```
class CBox {
        int length;
        int width;
        int height; }
```

```
/* output----------
length=3
width=6
height=9
surface area=198
volume=162
-----------------*/
```

(a) 試在main() method 裡，以CBox 類別建立一個box 物件，並將其length、width、height 三個資料成員的值設為3,6,9。

(b) 試在CBox 類別裡，定義volume() method，用來傳回box 物件的體積。

(c) 試在CBox 類別裡，定義surfaceArea () method，用來傳回box 物件的表面積。

(d) 試在CBox 類別裡，加入showData() method，用來顯示box 物件length、width、height 三個資料成員的值。

(e) 試在CBox 類別裡，加入showAll() method，用來顯示box 物件length、width、height 三個資料成員的值，以及其表面積與體積。

45

8.2 回家作業 method的使用

hw8_2_1.java

參考程式app8_3，除了保有原來的成員之外，於CRectangle 類別內再加入一個double型態的資料成員cost，並設值為3.17，代表矩形物件（例如，布匹或LCD 面板）每平方單位的價錢，然後再定義一price() method，用來計算購買此矩形物件所需的金錢。

```
/* output--------------
area=50
perimeter=30
price=158.5
--------------------*/
```

```
// app8_3, method的建立
class CRectangle
{
  int width;
  int height;
  int area()        // 定義成員函數area(), 用來計算面積
  {
    return width*height;     // 傳回矩形的面積
  }

  int perimeter()  // 定義成員函數perimeter(), 用來計算周長
  {
    return 2*(width+height);    // 傳回矩形的周長
  }
}

public class app8_3
{
  public static void main(String args[])
  {
    CRectangle rect1;
    rect1=new CRectangle();     // 建立新的物件

    rect1.width=10;          // 設定矩形rect1的寬
    rect1.height=5;         // 設定矩形rect1的高

    System.out.println("area="+rect1.area());
    System.out.println("perimeter="+rect1.perimeter());
  }
}

/* app8_3 OUTPUT---
area=300
perimeter=70
----------------*/
```

8.2 回家作業 method的使用

hw8_2_2.java

請撰寫一個程式, 由鍵盤輸入一個數值 n, 傳入一個方法, 並由該方法顯示出 1~n 中可以被 13 除的數值。

*請採用建立物件的方法

```
請輸入1-n的n值：80
13可被 13 整除
26可被 13 整除
39可被 13 整除
52可被 13 整除
65可被 13 整除
78可被 13 整除
```

8.3 引數的傳遞與傳回值

48

method的引數

- method不傳遞引數時，method的括號內什麼也不填：
  show_area( )　沒有傳遞任何引數，因此不需填上任何文字

- 傳遞引數時，引數是置於method的括號內，如：
  show_area(10);

呼叫method並傳遞引數

```
01  // app8_7, 呼叫method並傳遞引數
02  class CCircle   // 定義類別CCircle
03  {
04    double pi=3.14;   // 將資料成員設定初值
05    double radius;
06
07    void show_area()              // show_area() method, 顯示出半徑及圓面積
08    {
09      System.out.println("radius="+radius);
10      System.out.println("area="+pi*radius*radius);
11    }
12    void setRadius(double r)     // setRadius() method, 可用來設定半徑
13    {
14      radius=r;                   // 設定radius成員的值為r
15    }
16  }
17  public class app8_7
18  {
19    public static void main(String args[])
20    {
21      CCircle cir1=new CCircle();  // 宣告並建立新的物件
22      cir1.setRadius(2.0);  // 設定cir1的半徑為2.0
23      cir1.show_area();
24    }
25  }
```

```
/* app8_7 OUTPUT--
radius=2.0
area=12.56
-------------------*/
```

method裡的區域變數

- 區域變數若離開該method，變數即會失去效用：

```
01  // app8_7,呼叫method並傳遞引數
02  class CCircle   // 定義類別CCircle
03  {
      .....
12    void setRadius(double r)
13    {
14       radius=r;
15    }
16  }
```

r是區域變數，一離開此範圍，變數r即屬無效

傳遞多個引數

- 下面的程式是傳遞多個引數的範例：

```
01  // app8_8, 圓形類別CCircle
02  class CCircle         // 定義類別CCircle
03  {
04    double pi;          // 將資料成員設定初值
05    double radius;
06
07    void show_area()    // show_area() method, 顯示出圓面積
08    {
09      System.out.println("area="+pi*radius*radius);
10    }
11    void setCircle(double p,double r)  // 擁有兩個引數的method
12    {
13      pi=p;
14      radius=r;
15    }
16  }
17  public class app8_8
18  {
19    public static void main(String args[])
20    {
21      CCircle cir1=new CCircle();   // 宣告並建立新的物件
22      cir1.setCircle(3.1416,2.0);   // 呼叫並傳遞引數到setCircle()
23      cir1.show_area();
24    }
25  }
```

```
/* app8_8 OUTPUT--
area=12.5664
----------*/
```

沒有傳回值的method

- 若method沒有傳回值，則在定義的前面加上關鍵字void：

若method本身沒有傳回值，則必須在前面加上void

```
void show_area(){    // show_area() method, 顯示出圓面積
{
        System.out.println("area="+pi*radius*radius);
}
```

- method沒有傳回值，return敘述可以省略

```
void show_area(){    // show_area() method, 顯示出圓面積
  {
     System.out.println("area="+pi*radius*radius);
     return;
  }
```

因沒有傳回值，所以可在method結束前加上return敘述，但不接任何的運算式，其執行結果與前例相同

53

有傳回值的method

- 下面的範例裡增加一個傳回物件半徑的method：

```
01  // app8_9, 圓形類別CCircle
02  class CCircle   // 定義類別CCircle
03  {
04    double pi;   // 將資料成員設定初值
05    double radius;
06
07    double getRadius()              // getRadius(), 用來傳回物件的半徑
08    {
09      return radius;
10    }
11    void setCircle( double p, double r)
12    {
13      pi=p;
14      radius=r;
15    }
16  }
17  public class app8_9
18  {
19    public static void main(String args[])
20    {
21      CCircle cir1=new CCircle();  // 宣告並建立新的物件
22      cir1.setCircle(3.1416,2.0);
23      System.out.println("radius="+cir1.getRadius());
24    }
25  }
```

```
/* app8_9 OUTPUT--
radius=2.0
----------*/
```

method的本體，傳回物件的半徑radius

傳回值radius的型態為double，因此getRadius() 之前要冠上double

54

8.3 課堂練習

ex8_3_1.java

試設計一類別**CTest**，內含一**test() method**，可以用來判別傳入的值為奇數還是偶數，如果為奇數則印出"此數為奇數"，反之若為偶數則印出"此數為偶數"；若輸入的數為**0**，則印出"此數為 **0**"。

```
/* output----------
n= 3, 此數為奇數
n= 8, 此數為偶數
n= 0, 此數為 0
-----------------*/
```

55

8.3 回家作業
引數的傳遞與回傳值

hw8_3_1.java

請參考ex8_2_1，試於CBox 類別中加入下列的method，用來設定與取得物件的資料成員：

(a) get_length() method，可傳回CBox 物件的length 成員。

(b) get_width() method，可傳回CBox 物件的width 成員。

(c) get_height() method，可傳回CBox 物件的height 成員。

(d) set_length(int len) method，可設定CBox 物件的length 成員為len。

(e) set_width(int wid) method，可設定CBox 物件的width 成員為wid。

(f) set_height(int hei) method，可設定CBox 物件的height 成員為hei。

```
/* output----------
length=5
width=2
height=4
surface area=76
volume=40
For box object:
length=5
width =2
height=4
----------*/
```

8.3 回家作業
引數的傳遞與回傳值

hw8_3_2.java

試設計一個CCalculator 類別，並完成下列的各method 的程式設計：
(a) 定義add(int a, int b) method，可用來傳回二數之和。
(b) 定義sub(int a, int b) method，可用來傳回二數之差。
(c) 定義mul(int a, int b) method，可用來傳回二數之乘積。
(d) 定義div(int a, int b) method，可用來傳回a/b，傳回值型態請設為double。

```
/* output----------
8+3=11
8-3=5
8*3=24
8/3=2.6666666666666665
----------------*/
```

8.4 成員函數的多載(overloading)

- 同名的method，以多種型態出現，利用參數個數與型態區別呼叫哪一個方法
- 好處在於省去命名的麻煩，可使用相同的函數來針對不同的參數做不同的動作。

58

多載的認識


```
01  // app8_10, 函數的多載(一)
02  class CCircle   // 定義類別CCircle
03  {
04    String color;
05    double pi=3.14;
06    double radius;
07
08    void setColor(String str)  // 設定color的method
09    {
10      color=str;
11    }
12    void setRadius(double r)  // 設定radius method
13    {
14      radius=r;
15    }
16    void setAll(String str, double r) // 設color/radius
17    {
18      color=str;
19      radius=r;
20    }
21    void show()  // 列印半徑、顏色與圓面積
22    {
23      System.out.println("color="+color+",     Radius="+radius);
24      System.out.println("area="+pi*radius*radius);
25    }
26  }
```

```
27  public class app8_10
28  {
29    public static void main(String args[])
30    {
31      CCircle cir1=new CCircle();
32
33      cir1.setColor("Red");    // 設定cir1的color
34      cir1.setRadius(2.0);     // 設定cir1的radius
35      cir1.show();
36
37      cir1.setAll("Blue",4.0);  // 設定cir1的color和radius
38      cir1.show();
39    }
40  }
```

```
/* app8_10 OUTPUT---
color=Red, Radius=2.0
area=12.56
color=Blue, Radius=4.0
area=50.24
----------*/
```

59

函數的多載 (1/2)


```
01  // app8_11, 函數的多載(二)
02  class CCircle   // 定義類別CCircle
03  {
04    String color;
05    double pi=3.14;
06    double radius;
07
08    void setCircle(String str)  // 設定color成員
09    {
10      color=str;
11    }
12    void setCircle(double r)    // 設定radius成員
13    {
14      radius=r;
15    }
16    void setCircle(String str, double r)// 設定color與radi
17    {
18      color=str;
19      radius=r;
20    }
21    void show()
22    {
23      System.out.println("color="+color+", Radius="+radius);
24      System.out.println("area="+pi*radius*radius);
25    }
26  }
27
```

```
28  public class app8_11
29  {
30    public static void main(String args[])
31    {
32      CCircle cir1=new CCircle();
33
34      cir1.setCircle("Red");// 呼叫第8行的setCircle()
35      cir1.setCircle(2.0); // 呼叫第12行的setCircle()
36      cir1.show();
37
38      cir1.setCircle("Blue",4.0);// 呼叫第16行的setCircle()
39      cir1.show();
40    }
41  }
```

```
/* app8_11 OUTPUT---
color=Red, Radius=2.0
area=12.56
color=Blue, Radius=4.0
area=50.24
----------*/
```

60

函數的多載 (2/2)

- 使用多載時，編譯器會根據引數的個數與型態，來呼叫相對應的method

```
cir1.setCircle("Red");
```
→
```
void setCircle(String str){
   color=str;
}
```

```
cir1.setCircle(2.0);
```
→
```
void setCircle(double r){
   radius=r;
}
```

```
cir1.setCircle("Blue",4.0);
```
→
```
void setCircle(String str, double r){
   color=str;
   radius=r;
}
```


使用多載常犯的錯誤

- 多載不能是引數個數或引數型態完全相同，而只有傳回型態不同。下面的程式碼是錯誤的：

```
void setCircle(double radius){ ... };
int setCircle(double radius){ ... };
```
} 這兩個method的引數個數和型態完全相同，但傳回型態不同

→ 呼叫setCircle()時，程式無法判斷是哪一個method被呼叫

- 下列多載的程式碼在Java裡是合法的：

```
void setCircle(String color,double radius);
int setCircle(double radius);
```
} method的引數個數和型態不同，且傳回型態也不相同

8.4 課堂練習

ex8_4_1.java

試問在下列哪一個選項可以呼叫void set(int r) 這個method？

(a) set("hello");

(b) set(50);

(c) set(10,25);

(d) set(3.14);

63

8.4 課堂練習

ex8_4_2.java

設有一CTriangle 類別，可用來表示一個三角形。此類別內含base、height 二個資料成員，代表三角形的底與高，此外還有一個color 成員，用來表示三角形的顏色。試在程式碼裡完成下列各敘述：

(a) 試定義setB()、setH()與setColor()，使得它們可以分別用來設定CTriangle 物件的base、height 與color 成員的值。

(b) 試加入setTriangle(int b, int h) method，使得它可以同時設定CTriangle物件的base 與height。

(c) 接續上題，請多載setTriangle(int b,int h, String c) method，使得它可以同時設定CTriangle 物件的base、height 與color 三個資料成員。

```
/* output-------------
Color=Red
base=5, height=8

Color=Blue
base=3, height=9

Color=Green
base=4, height=6
----------------------*/
```

8.4 回家作業
函數成員的多載

hw8_4_1.java

假設我們要設計一個CWin 類別，用來表示一個視窗（window）的基本外觀。此類別內含寬（width）、高（height）與名稱（name）三個資料成員，部份程式碼撰寫如下：

```
class CWin
{
 int width;
 int height;
 String name;

 void setW(int w) // 設定寬度的method
 { // 請在此處填上程式碼 }
 void setH(int h) // 設定高度的method
 { // 請在此處填上程式碼 }
 void setName(String s) // 設定視窗名稱的method
 { name=s;}

 public void show()
 {
  System.out.println("Name="+name);
 System.out.println("W="+width+", H="+height);
 }
 }
```

```
public class hw8_4_1
 {
 public static void main(String args[])
 {
   CWin cw=new CWin();
   cw.setName("My Windows");
   cw.setW(5);
   cw.setH(3);
   cw.show();
 }
 }
```

(a) 於上面的程式碼中，setW() 與setH() 兩個method 並沒有填上程式碼。試將它們完成，使得它們可以分別用來設定CWin 物件的width 與height 成員的值。

(b) 試加入setWindows(int w, int h) method，使得它可以同時設定CWin 物件的width 與height。

(c) 接續上題，請多載setWindows() method，使得它可以同時設定CWin 物件的
width、height 與name 三個資料成員。

```
/* output----------
Name=1st Windows
W=5, H=3
Name=2nd Windows
W=6, H=8
Name=3rd Windows
W=4, H=2
----------*/
```

8.5 公有成員與私有成員

- 好處在於重要變數不會任意被變更
- 例如：app8_12.java中，pi和radius可任意在ccircle類別外部更改，雖然方便，但是不安全。
- 例如：app8_12.jajva中，雖邏輯對，但是觀念錯誤，需要一個機制來設定類別中資料的存取，以免出錯。

67

資料成員的潛在危險

- app8_12的18行將cir1物件的radius成員設成-2.0：

```
01  // app8_12, 圓形類別CCircle
02  class CCircle   // 定義類別CCircle
03  {
04    double pi=3.14;   // 將資料成員設定初值
05    double radius;
06
07    void show_area()
08    {
09      System.out.println("area="+pi*radius*radius);
10    }
11  }
12
13  public class app8_12
14   {
15    public static void main(String args[])
16    {
17      CCircle cir1=new CCircle();
18      cir1.radius=-2.0;
19      cir1.show_area();
20    }
21  }
```

CCircle類別內部

CCircle類別外部

從類別外部存取資料成員時，如果沒有一個機制來限定存取的方式，很可能導致安全上的漏洞，而讓臭蟲（bug）進駐程式碼

```
/* app8_12 OUTPUT---
area=12.56
----------------------*/
```

68

建立私有成員

- 透過私有成員（private member）的設定，可限定類別中資料成員的存取。設定的方式如下：

```
01  class CCircle
02  {
03    private double pi=3.14;
04    private double radius;
05     ....
06  }
```

設定field為私有成員

- 在field宣告的前面加上private，則無法從類別以外的地方設定或讀取到它，可達到資料保護的目的

69

私有成員的範例

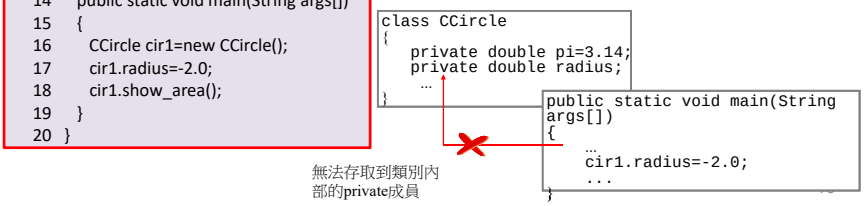
- app8_13在field之前加上private：

```
01  // app8_13, 私有成員無法從類別外部來存取的範例
02  class CCircle   // 設定field為私有成員
03  {
04    private double pi=3.14;   // 將資料成員設定初值
05    private double radius;
06
07    void show_area()
08    {
09      System.out.println("area="+pi*radius*radius);
10    }
11  }
```

```
12  public class app8_13
13  {
14    public static void main(String args[])
15    {
16      CCircle cir1=new CCircle();
17      cir1.radius=-2.0;
18      cir1.show_area();
19    }
20  }
```

編譯時將會得到下列的錯誤訊息，這個訊息說明私有成員無法從類別外的地方存取：

```
radius has private access in CCircle
    cir1.radius=-2.0;
        ^
1 error
Process completed.
```



建立公有成員 (1/2)

- 在類別內加上公有成員setRadius()與私有成員函數area()：

```
01  // app8_14, 公有成員(method)的建立
02  class CCircle   // 定義類別CCircle
03  {
04    private double pi=3.14;   // 將資料成員設定為private
05    private double radius;
06
07    private double area()  // 定義私有的成員函數area()
08    {
09      return pi*radius*radius;
10    }
11    public void show_area()   // 定義公有的成員函數show_area()
12    {
13      System.out.println("area="+ area());  // 呼叫私有成員area()
14    }
15    public void setRadius(double r) // 定義公有的成員函數setRadius()
16    {
17      if(r>0)
18      {
19        radius=r;     // 將私有成員radius設為r
20        System.out.println("radius="+radius);
21      }
22      else
23        System.out.println("input error");
24    }
25  }
```

```
/* app8_14 OUTPUT---
input error
area=0.0
----------*/
```

```
27  public class app8_14
28  {
29    public static void main(String args[])
30    {
31      CCircle cir1=new CCircle();
32      cir1.setRadius(-2.0);// 呼叫公有的setRadius() method
33      cir1.show_area();    // 呼叫公有的show_area() method
34    }
35  }
```

建立公有成員 (2/2)


```
27  public class app8_14
28  {
29    public static void main(String args[])
30    {
31       CCircle cir1=new CCircle();
32       cir1.setRadius(-2.0);// 呼叫公有的setRadius() method
33       cir1.show_area();     // 呼叫公有的show_area() method
34    }
35  }
```

```
/* app8_14 OUTPUT---
input error
area=0.0
----------*/
```

透過公有成員setRadius()，私有成員radius的值才得以修改

```
class CCircle
{
   private double pi=3.14;
   private double radius;
     ...
}
```

無法存取到類別內部的private成員

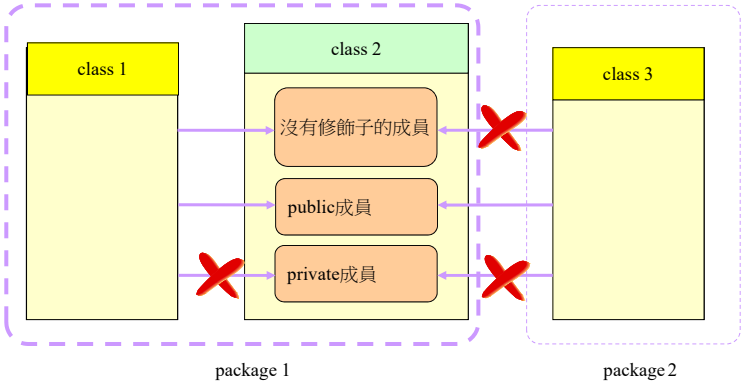
```
public static void main(String args[])
{
    ...
    cir1.radius=-2.0;
    ...
}
```


public與private (1/2)

- 「封裝」(encapsulation)
  - 把field和method依功能劃分為「私有成員」與「公有成員」，並包裝在一個類別內來保護私有成員，使得它不會直接受到外界的存取
- 設定公有與私有成員的「修飾子」(modifier)：
  - public -- 公有
  - private – 私有

public與private (2/2)

- **若省略public與private，則成員只能在同一個package裡被存取**
- **如果冠上public的話，則成員可以被任何一個package所存取**
  - 類別內成員的修飾子與存取等級的關係圖：



74

8.5 課堂練習

ex8_5_1.java

設有一**CSphere** 類別，可用來表示一個圓球。此類別內含**x, y, z** 三個資料成員，用來代表圓心的位置，另外還有一個**radius** 資料成員，代表圓球的半徑。其部份程式碼的撰寫如下：

```
class CSphere
    { private int x; // 圓心的x 座標
      private int y; // 圓心的y 座標
      private int z; // 圓心的z 座標
      private int radius; // 圓球的半徑 }
```

(a) 試在**CSphere** 類別裡加入**setLocation() method**，用來設定圓球之圓心的位置。

(b) 試在**CSphere** 類別裡加入**setRadius() method**，用來設定圓球之半徑。

(c) 試在**CSphere** 類別裡加入**surfaceArea() method**，用來傳回**CSphere** 物件的表面積。($4*PI*R^3$)

(d) 試在**CSphere** 類別裡加入**volume() method**，用來傳回**CSphere** 物件的體積。($(3/4)*PI*R^3$)

(e) 試在**CSphere** 類別裡加入**showCenter() method**，用來顯示**CSphere** 物件之圓心座標。

***radius=1

```
/* output----------
 x=3, y=4, z=5
 surface area=12.56
 volume=2.355
----------*/
```

75

8.5 回家作業
公有成員與私有成員

hw8_5_1.java

設有一CData 類別，用來記錄好友的姓名、電子郵件信箱及生日。其部份程式碼的撰寫如下：

```
class CData
{
 private String name; // 姓名
 private String email; // 電子郵件信箱
 private int mm; // 生日的月
 private int dd; // 生日的日
 private int yy; // 生日的年
}
```

```
/* output----------
姓名:Kevin
電子郵件信箱:Kevin@hot.com
生日:12/8/1998
姓名:Amy
電子郵件信箱:Amy@hot.com
生日:4/19/2000
----------*/
```

(a) 試在CData 類別裡加入setName() method，用來設定好友的姓名。

(b) 試在CData 類別裡加入setEmail() method，用來設定電子郵件信箱。

(c) 試在CData 類別裡加入setBirthday(int m,int d,int y) method，用來設定生日的年、月、日。

(d) 試在CData 類別裡加入private boolean checkDate (int m,int d,int y) method，用來判定生日的年、月、日是否在合法的範圍，其中y 的值要在西元1900~2005 之間。為簡化程式，在此僅考慮2 月有28 天，不考慮閏年的問題。

(e) 試在CData 類別裡加入setAll() method，使得在main() method 裡可以設定所有資料成員的內容。

(f) 試在CData 類別裡加入showData() method，用來顯示好友的所有資料。生日的輸出格式請依照mm/dd/yy 的方式處理。

(g) 請建立CData 類別的物件friend1，利用(a)~(c)的method 為friend1 設值，最後再印出friend1 的所有內容。

(h) 請建立CData 類別的物件friend2，利用(e)的method 為friend2 設值，最後再印出friend2 的所有內容。

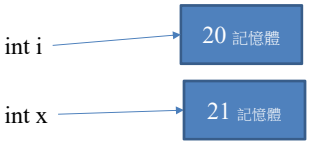
參數的傳遞方式

補充

- 參數是傳值方式(Call By Value)傳遞
  - 參數傳遞時，是把值**複製**給參數。

程式 Argument.java 在方法中更改參數值

```
01 public class Argument {
02   public static void main(String[] argv){
03     Argument a = new Argument(); // 建立測試物件
04     int i = 20;
05
06     System.out.println("呼叫方法前  i = " + i);
07     a.changePara(i);    // 傳入 i
08     System.out.println("呼叫方法後  i = " + i);
09   }
10
11   void changePara(int x) {        // 會修改參數值的方法
12     System.out.println("...方法參數 x = " + x);
13     System.out.println("...修改中");
14     x ++;    // 更改接收到的參數值
15     System.out.println("...現在參數 x = " + x);
16   }
17 }
```



執行結果

```
呼叫方法前  i = 20
...方法參數 x = 20
...修改中
...現在參數 x = 21
呼叫方法後  i = 20
```

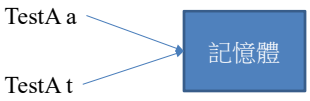
傳遞參照型別的資料 (Call By Reference)

補充

程式 PassReference.java 傳遞參照

```
01 class TestA { // 測試類別
02   int x = 3;  // 設定初始值
03
04   void show() {
05     System.out.println("x = " + x);
06   }
07 }
08
09 class TestB {
10   void changeTestA(TestA t,int newX) {
11     t.x = newX; // 透過參照修改物件內容
12   }
13 }
14
15 public class PassReference {
16   public static void main(String[] argv){
17     TestA a = new TestA();
18     TestB b = new TestB();
19
20     a.show();
21     b.changeTestA(a,20); // 傳入物件參照
22     a.show();
23   }
24 }
```

物件變數是指向物件的**參照**所以變數a和變數t都是指向同一個物件，因此透過t更改物件的內容之後，再透過a存取則是已經讀到修改後的內容了。



執行結果

```
x = 3
x = 20
```

中英文術語對照

成員	member
類別	class
物件	object
實例方法	instance method
實例變數	instance variable
類別變數	class variable
類別方法	class method

類別總整理

- **public** **static** **void** **main (String argv[])**
  變數存取範圍 方法特性 傳回值型態 方法名稱 傳入變數型態

- **public的意義**
  - public：package 內外，所有人都可以取用
  - protected：同一個 package 內所有人，或子孫 (subclass) (不論是否同一個 package) 可以取用。
  - (空白)：預設為同一個 package 所有人都可取用
  - private：只有 class 自己可以取用
- **static的意義**
  - 宣告為static(靜態)的變數，該成員屬於類別 (class) 而非物件 (object) ，因此使用 static 就不需要先建立物件
  - 可以直接以類別的名稱呼叫
- **void的意義**
  - 傳回值的型態
  - 寫void則為無傳回值

81