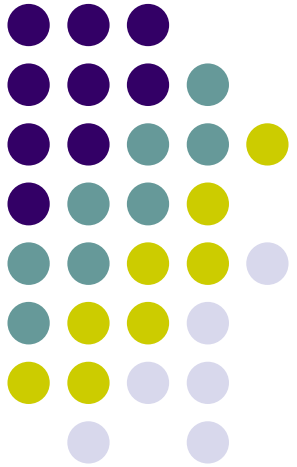
第八章
認識類別



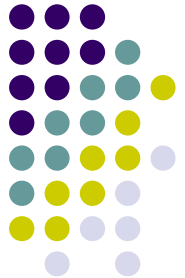
認識類別的基本架構

在類別裡使用資料成員與成員函數

學習this關鍵字的用法

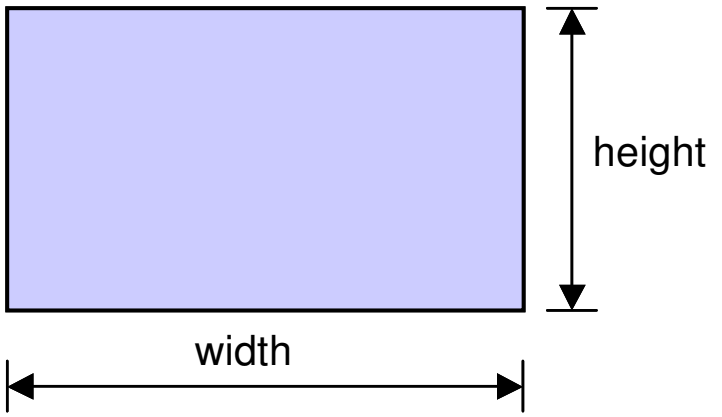
在類別裡設計method的多載

學習如何使用類別裡的公有與私有成員



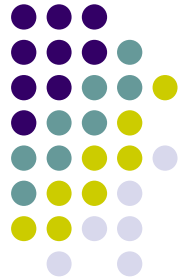
類別的基本概念

- 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)

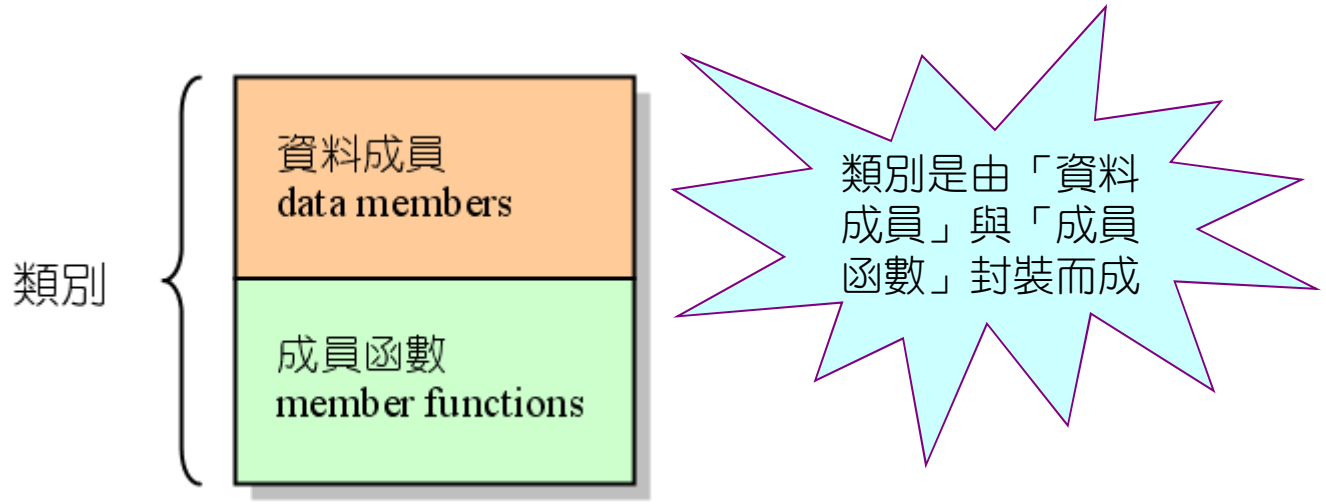


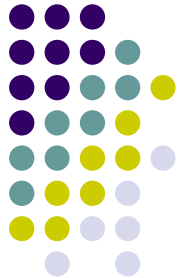
資料成員與成員函數

- 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裡，成員函數是封裝在類別之內

資料成員、成員函數與封裝

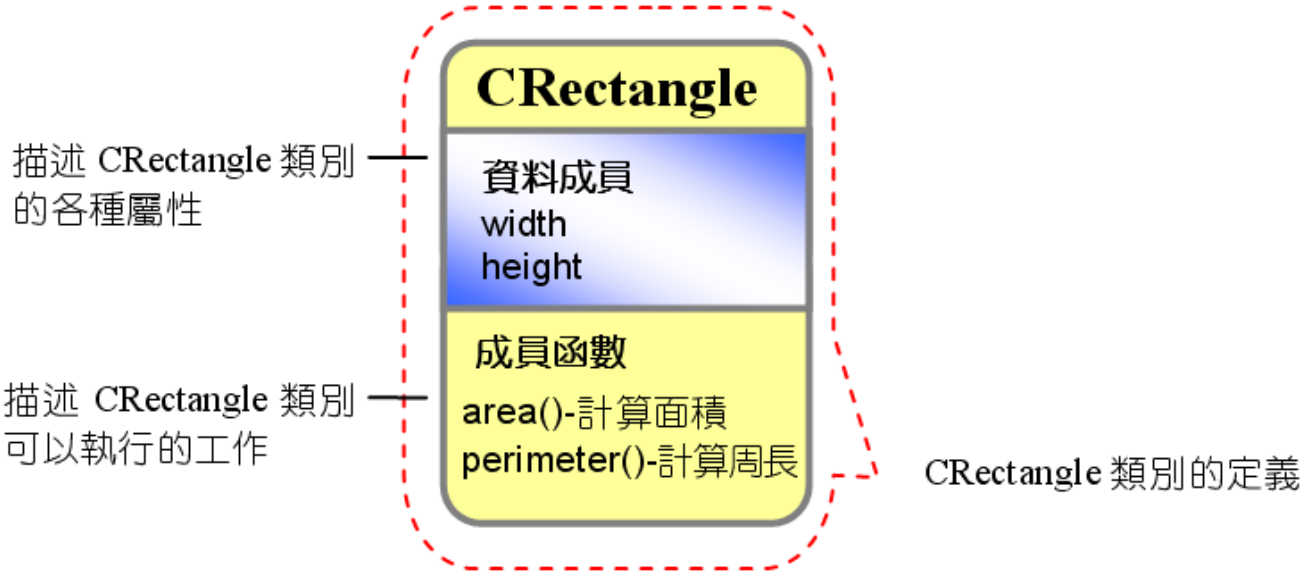
- 「類別」就是把事物的資料與相關功能「封裝」（encapsulate）在一起
  - 「encapsulate」的原意是「將...裝入膠囊內」
- 類別可看成是「膠囊」
  - 資料成員與成員函數可看成是被裝入的東西

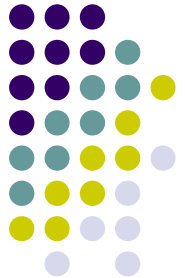




矩型類別的認識

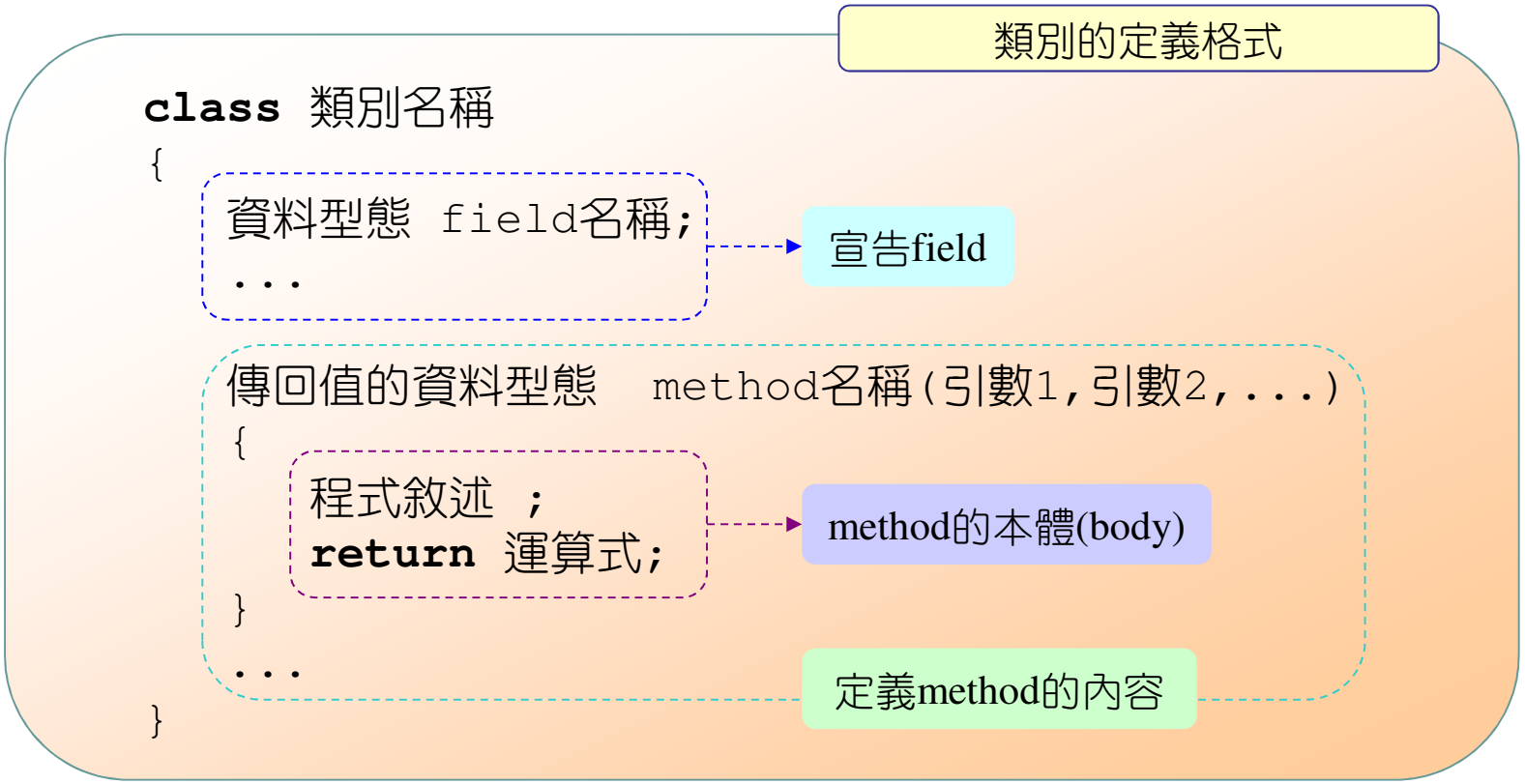
- 矩型類別：
  - 資料成員為 width 與 height
  - 成員函數為 area() 與 perimeter()





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

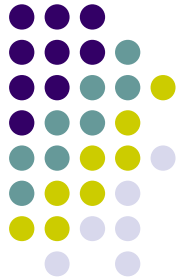
圓形類別的範例

- 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  }
```

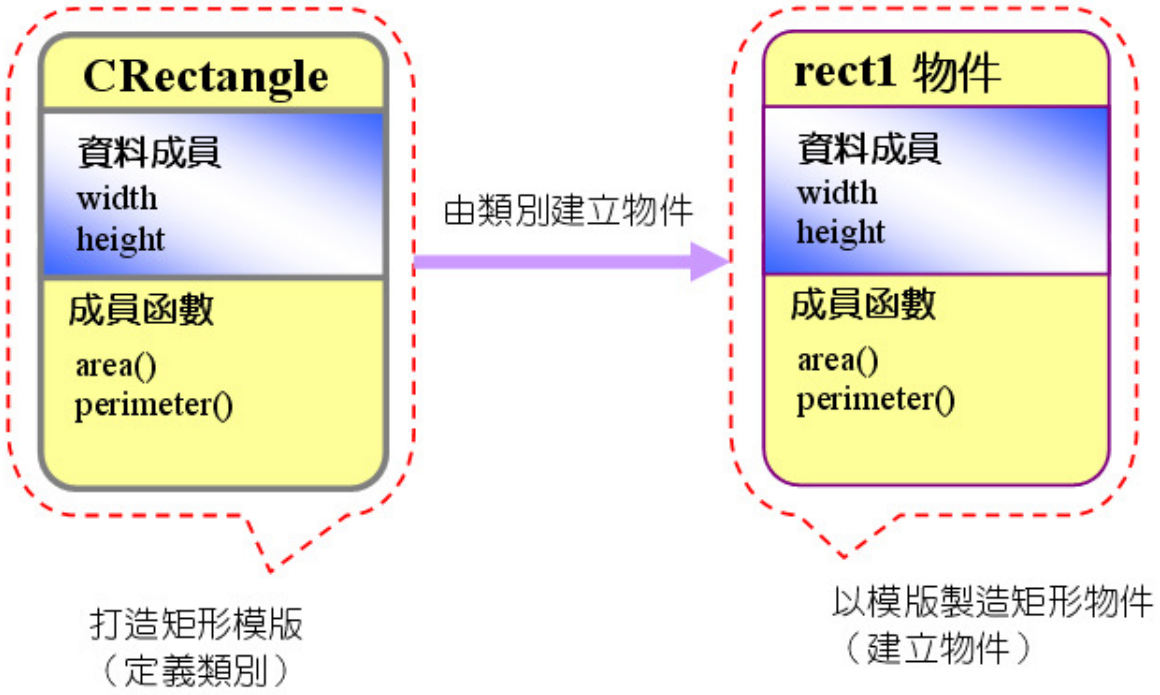
建立新物件

- 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



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：

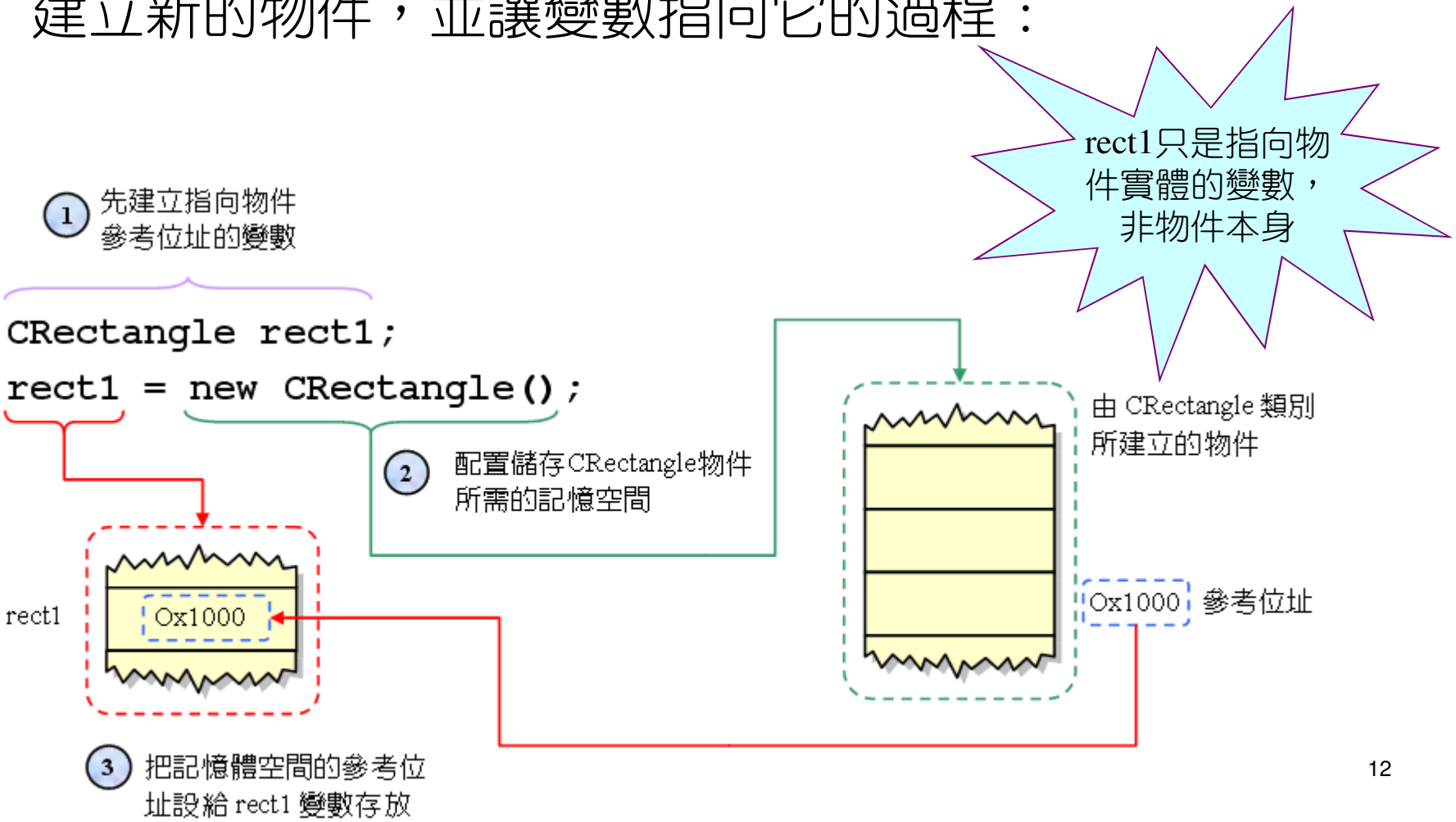
```
CRectangle rect1;           // 以類別名稱 CRectangle 宣告變數 rect1
rect1=new CRectangle();     // 利用 new 建立新的物件，並讓變數 rect1 指向它
```

  或是縮減成一行：

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

指向物件的變數

- 建立新的物件，並讓變數指向它的過程：



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;                    // 矩形物件 rect1 的寬
rect1.height;                   // 矩形物件 rect1 的高
```

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：

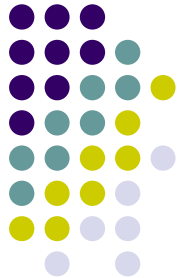
```
01  public static void main(String args[])
02  {
03     CRectangle rect1;              // 宣告變數 rect1
04     rect1=new CRectangle();        // 建立新的物件，並將變數 rect1 指派給它
05
06     rect1.width=20;                // 設定矩形物件 rect1 的寬為 20
07     rect1.height=15;               // 設定矩形物件 rect1 的高為 15
08  }
```

設計完整的程式

- 下面的程式碼為建立物件與field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=20;          // 設定矩形 rect1 的寬
16        rect1.height=15;         // 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=20
height=15
------------------*/
```



編譯類別後的檔案

- Java會將每個類別編譯成獨立的.class檔案 ：

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=20;                    // 設定矩形 rect1 的寬
17        rect1.height=15;                   // 設定矩形 rect1 的高
18
19        rect2.width=25;                    // 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=20
rect1.height=15
rect2.width=25
rect2.height=18
-------------------*/
```

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;
}
```

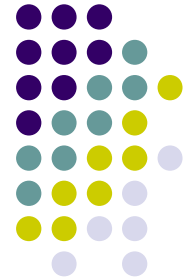
method的本體(body)

- 呼叫封裝在類別裡的method的語法：

呼叫封裝在類別內的method

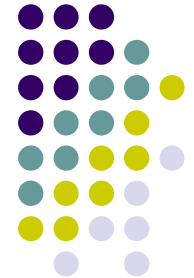
```
物件名稱.method名稱(引數1, 引數2,...)
```

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=20;               // 設定矩形 rect1 的寬
25        rect1.height=15;              // 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=300
perimeter=70
--------------------*/
```



再一個簡單的範例

- 下列的範例建立了一個圓形類別CCircle：

```
01  // 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 cir1 物件
17        cir1.radius=2.0;              // 設定 radius 的值
18        cir1.show_area();             // 呼叫 show_area() method
19     }
20  }
```

```
/* app8_4 OUTPUT---

area=12.56
------------------*/
```

field於記憶體內的配置關係 (1/2)

- app8_5說明類別的field於記憶體內之配置關係：

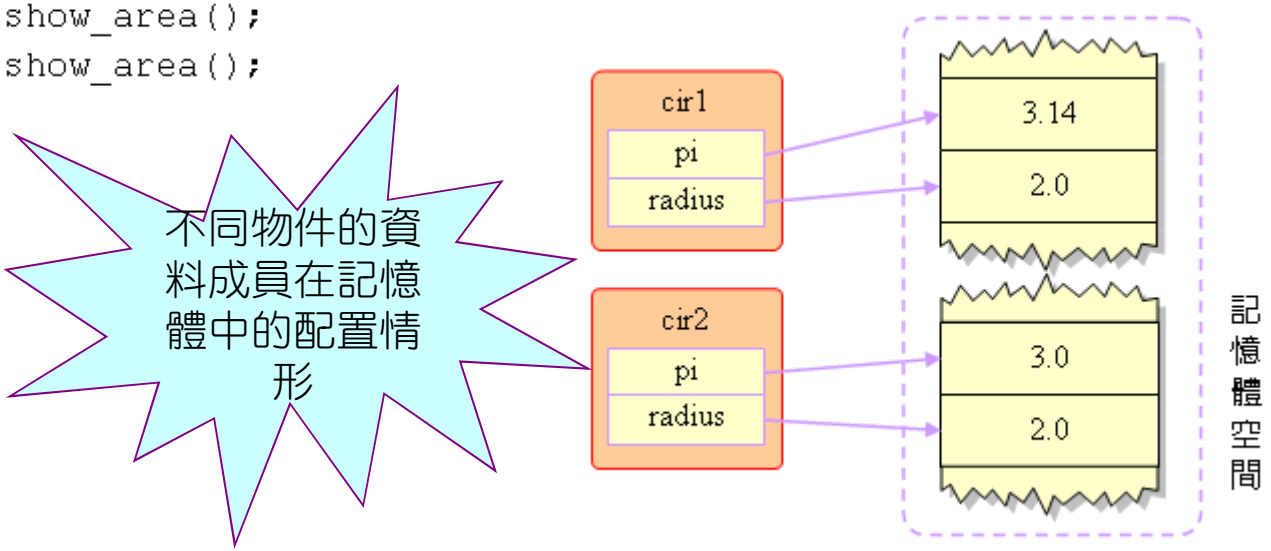
```
01  // app8_5, 圓形類別 CCircle 之 field 於記憶體內的配置關係
02  class CCircle                  // 定義類別 CCircle
03  {
04     double pi=3.14;             // 將資料成員設定初值
05     double radius;
06
07     void show_area()            // show_area() method, 顯示出圓面積
08     {
09        System.out.print("pi="+pi);
10     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

```
/* app8_5 OUTPUT----

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

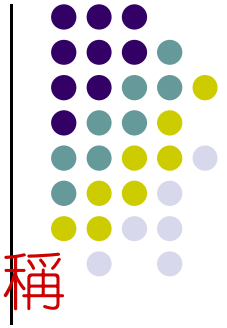
field於記憶體內的配置關係 (2/2)

```
13  public class app8_5
14  {
15     public static void main(String args[])
16     {
17        CCircle cir1=new CCircle();        // 建立 cir1 物件
18        CCircle cir2=new CCircle();        // 建立 cir2 物件
19
20        cir1.radius=cir2.radius=2.0;       // 設定資料成員的值
21        cir2.pi=3.0;                       // 更改 cir2 的 pi 值
22
23        cir1.show_area();
24        cir2.show_area();
25     }
26  }
```

```
/* app8_5 OUTPUT----

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



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

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 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

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          //   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(4.0);               // 設定 cir1 的半徑為 4.0
23        cir1.show_area();
24     }
25  }
```

```
/* app8_7 OUTPUT---

radius=4.0
area=50.24
------------------*/
```

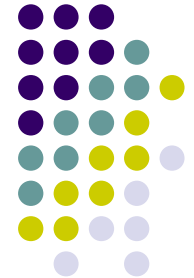
method裡的區域變數

- 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);

}
```

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

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
------------------*/
```

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

method 的 本 體 ， 傳回物件的半徑radius

多載的認識 (1/2)

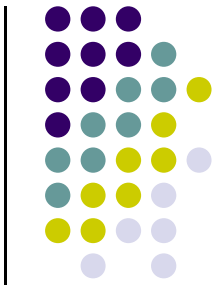
- 本節將以CCircle類別做延伸：

```
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 的 method
13     {
14        radius=r;
15     }
16     void setAll(String str, double r)      // 同時設定 color 與 radius
17     {
18        color=str;
19        radius=r;
20     }
```

這些函數功能相近，卻有不同的函數名稱，使用起來很麻煩

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

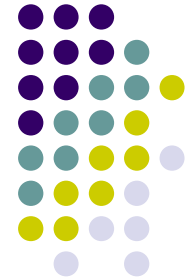
多載的認識(2/2)



```
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);       // 同時設定 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
------------------------*/
```

函數的多載 (1/2)



- 下面的例子是函數多載的範例：

```
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)    // 同時設定 color 與 radius
17     {
18        color=str;
19        radius=r;
20     }
```

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

```
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()
```

函數的多載 (2/2)

```
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");
35         cir1.setCircle(2.0);
36         cir1.show();
37
38         cir1.setCircle("Blue",4.0);
39         cir1.show();
40      }
41   }
```

```
/* app8_11 OUTPUT-------

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

```
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;
}
```

使用多載時，編譯器會根據引數的個數與型態，來呼叫相對應的method

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){ ... };
```

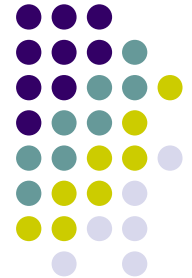
這兩個 method 的引數個數和型態完全相同，但傳回型態不同

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



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

在 CCircle 類別外部可以直接更改成員資料

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

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

建立私有成員

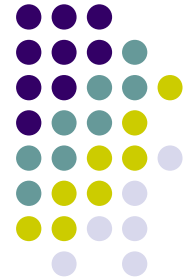
- 透過私有成員（private member）的設定，可限定類別中資料成員的存取。設定的方式如下：

```
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

私有成員的範例



- app8_13在field之前加上private：

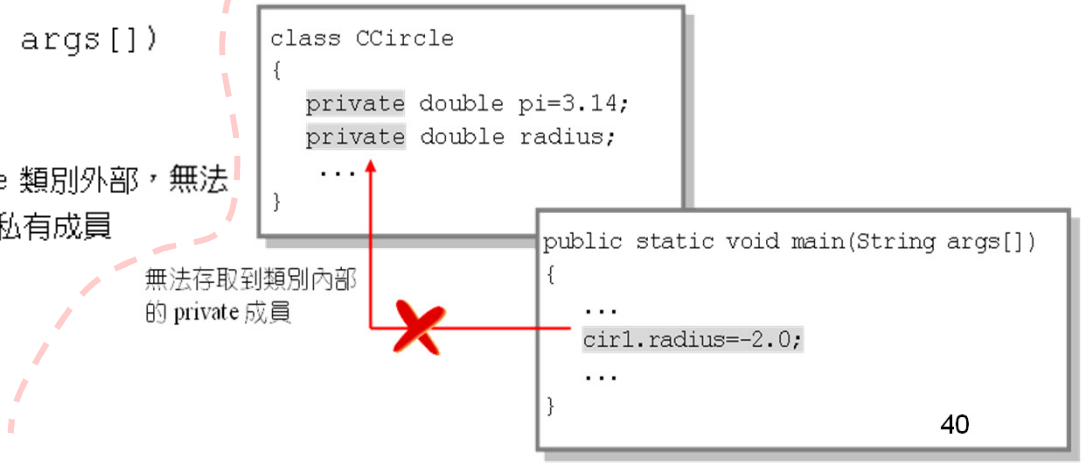
```
01  // app8_13, 私有成員無法從類別外部來存取的範例
02  class CCircle                    // 設定 field 為私有成員
03  {
04     private double pi=3.14;       // 將資料成員設定初值
05     private double radius;
06
07     void show_area()
08     {
09        System.out.println("area="+pi*radius*radius);
10     }
11  }
12
13  public class app8_13
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  }
```

在 CCircle 類別內部，所以可以存取私有成員

在 CCircle 類別外部，無法直接更改私有成員

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     }
```

```
/* app8_14 OUTPUT---

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

```
31   CCircle cir1=new CCircle();
32   cir1.setRadius(-2.0); // 呼叫公有的 setRadius()
33   cir1.show_area();     // 呼叫公有的 show_area()
```

建立公有成員 (2/2)

```
25  }
26
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  // 定義類別 CCircle
{
    ....
    public void setRadius(double r)
    {
        ....
    }
}
```

```
class app8_14
{
    public static void main(String args[])
    {
        CCircle cir1=new CCircle();
        cir1.setRadius(-2.0);
        cir1.show_area();
    }
}
```

可以存取到類別內部的 public 成員

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：