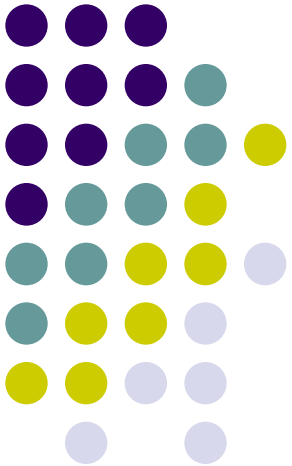
第八章
認識類別



認識類別的基本架構

在類別裡使用資料成員與函數成員

學習this關鍵字的用法

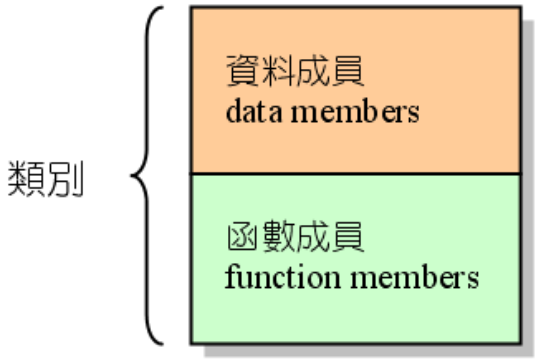
在類別裡設計函數的多載

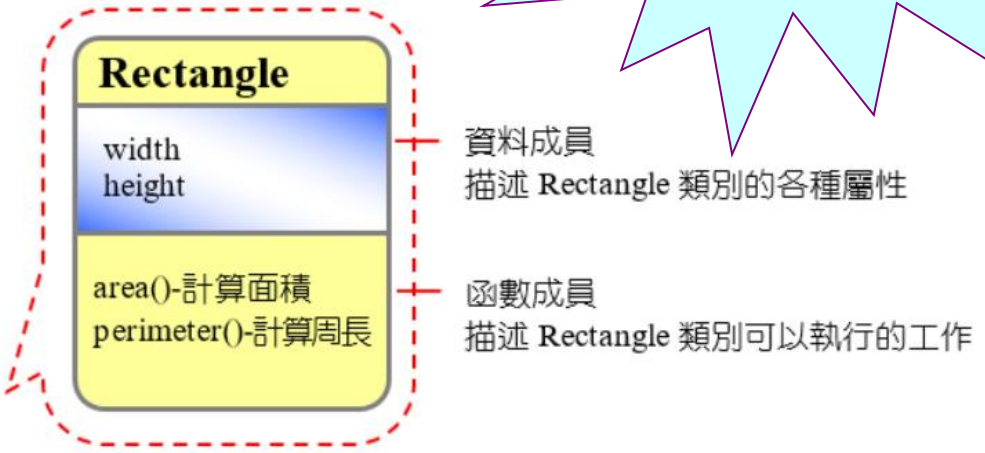
學習如何使用類別裡的公有與私有成員

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函數成員封裝而成
- 類別內的資料成員稱為field（範疇）
- 在oop裡，函數成員是封裝在類別之內

類別是由「資料成員」與「函數成員」封裝而成

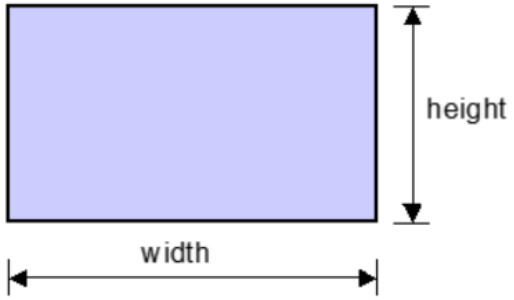




Rectangle 類別的定義

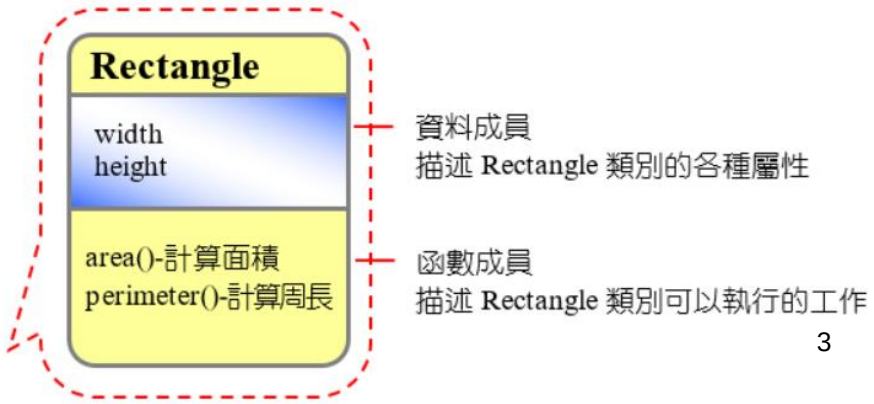
資料成員與函數成員

- 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）
- 計算面積與周長的函數可視為類別的「函數成員」（member function）
  - 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）

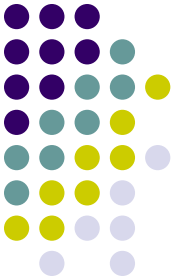


面積(area) = width*height

周長(perimeter) = 2(width+height)



Rectangle 類別的定義



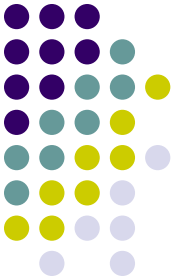
類別的定義

- 類別定義的語法如下：

定義類別的語法

```
class 類別名稱{
    資料型別 field 名稱;          } 宣告 field
    ...

    傳回值的資料型別  函數名稱(引數 1,引數 2,...){
        程式敘述 ;                } 函數的本體
        return 運算式;
    }                                                  } 定義函數
    ...
}
```



矩形類別的範例(1/2)

- 以矩形為例，可定義如下的矩形類別：

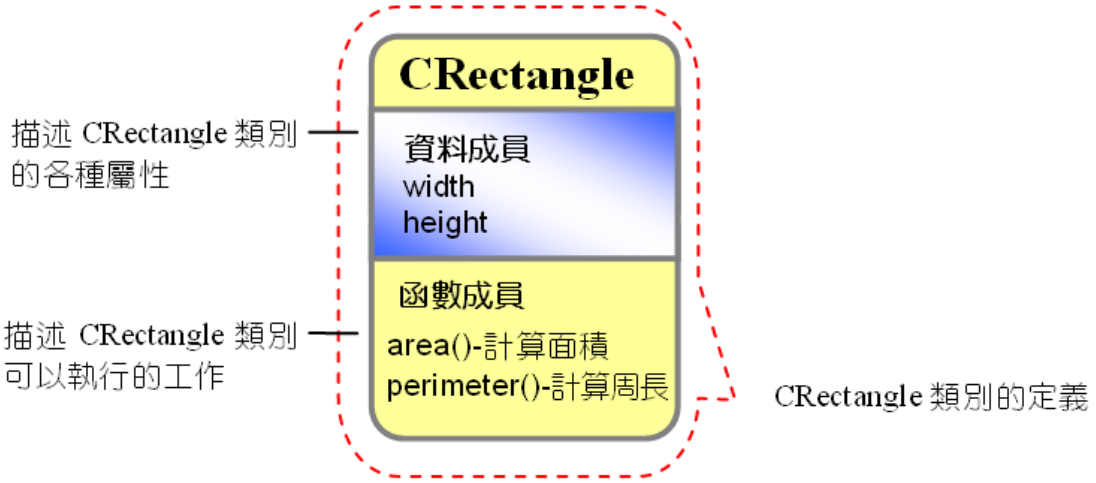
```
01  // 定義矩形類別
02  class Rectangle{              // 定義矩形類別 Rectangle
03     int width;                 // 宣告資料成員 width
04     int height;                // 宣告資料成員 height
05
06     int area(){                   // 定義函數成員 area(), 用來計算面積
07        return width*height;       // 傳回矩形的面積
08     }
09   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

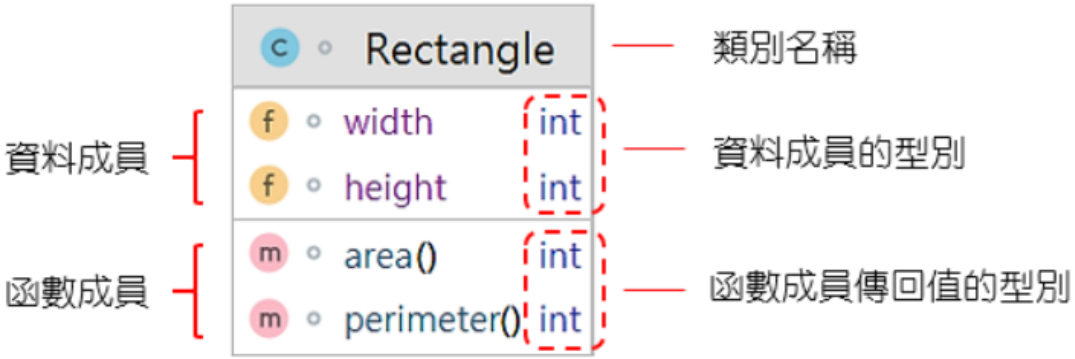
本書以大寫為開頭的識別字做為類別的名稱，方便和其它變數做區隔

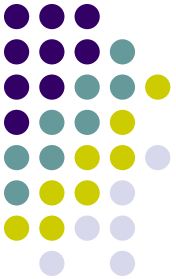
- 資料成員為 width 與 height
- 函數成員為 area() 與 perimeter()

矩形類別的範例(2/2)

- 矩形類別說明：



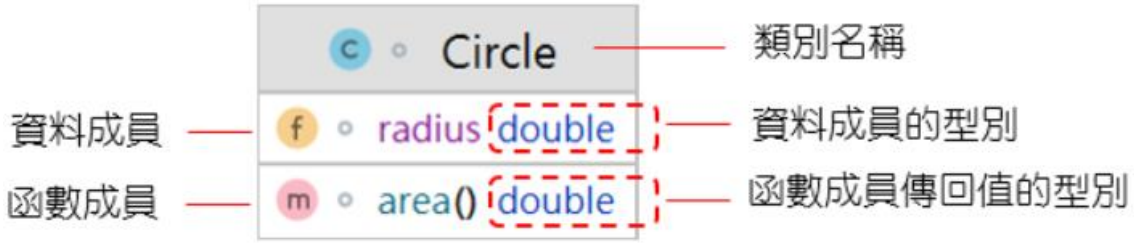


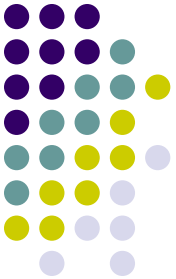


圓形類別的範例

- 以圓形為例，可定義如下的圓形類別：

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



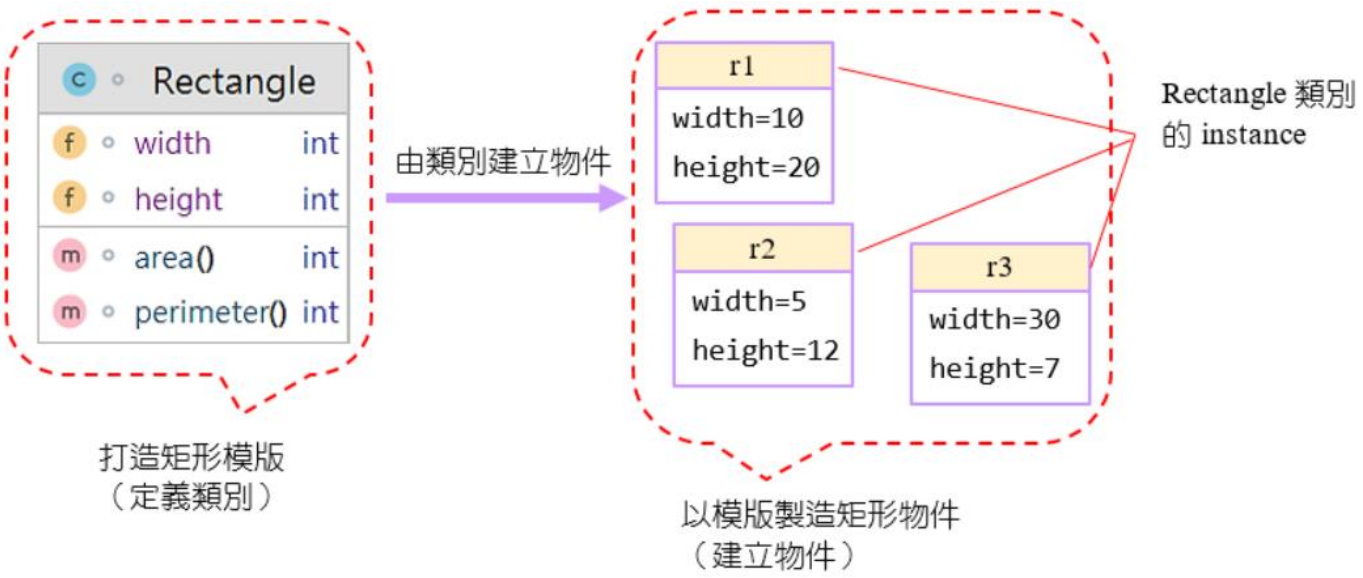


建立新物件

- 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：



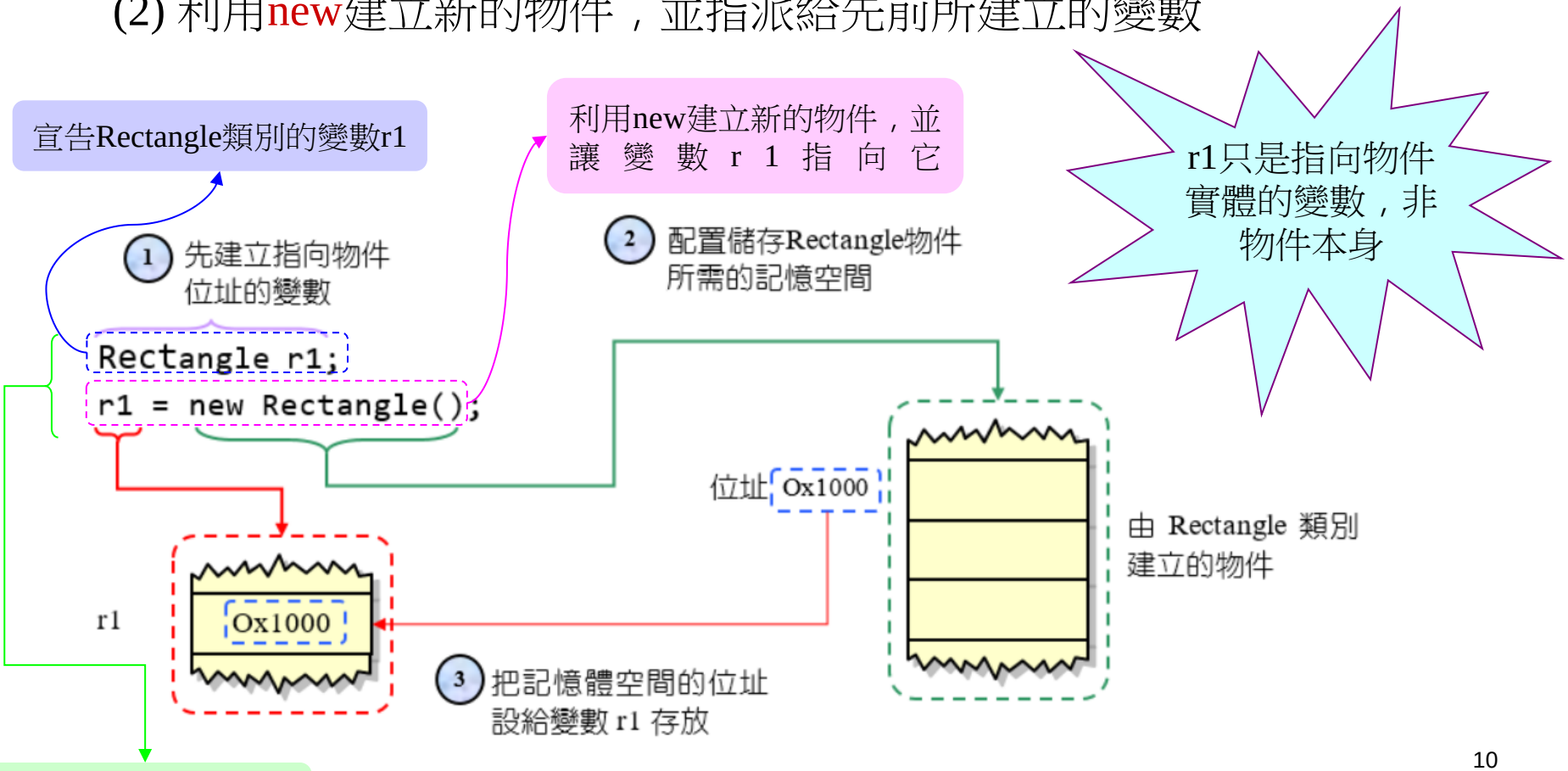
- 由類別所建立的物件稱為該類別的 instance

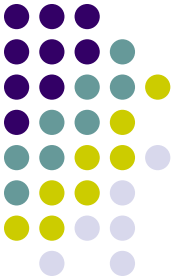
宣告與建立物件

- 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





存取物件的內容

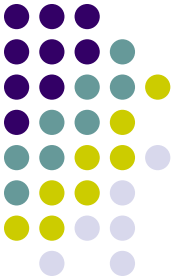
- 存取物件裡的特定資料成員，可透過下面語法來達成：

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，設定物件r1的寬與高之值，可用下列方式：

```
r1.width=12;          // 矩形物件 r1 的寬
r1.height=30;         // 矩形物件 r1 的高
```



設計完整的程式

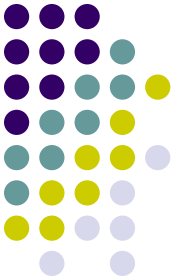
- 下面的程式碼為建立物件與field的存取之範例

```
01  // Ch8_1, 建立物件與資料成員的設定
02  class Rectangle{              // 定義 Rectangle 類別
03     int width;                 // 宣告資料成員 width
04     int height;                // 宣告資料成員 height
05  }
06
07  public class Ch8_1{
08     public static void main(String[] args){
09         Rectangle r1;
10        r1=new Rectangle();     // 建立新的物件
11
12        r1.width=20;       // 設定矩形 r1 的寬
13        r1.height=15;      // 設定矩形 r1 的高
14        System.out.println("width="+r1.width);     // 印出 r1.width
15        System.out.println("height="+r1.height);   // 印出 r1.height
16     }
17  }
```

執行結果：

```
width=20
height=15
```

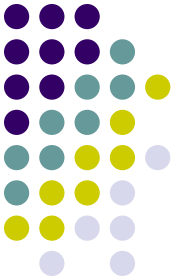
同時建立多個物件的範例



```
01  // Ch8_2, 同時建立兩個物件
02  class Rectangle{
03     int width;          // 定義資料成員 width
04     int height;         // 定義資料成員 height
05  }
06
07  public class Ch8_2{
08     public static void main(String[] args){
09        Rectangle r1,r2;            // 宣告指變數 r1,r2
10        r1=new Rectangle();         // 建立物件 r1
11        r2=new Rectangle();         // 建立物件 r2
12
13        r1.width=20;                // 設定矩形 r1 的寬
14        r1.height=15;               // 設定矩形 r1 的高
15        r2.width=25;                // 設定矩形 r2 的寬
16        r2.height=r1.height+3;      // 設定矩形 r2 的高
17
18        System.out.println("r1.width="+r1.width);
19        System.out.println("r1.height="+r1.height);
20        System.out.println("r2.width="+r2.width);
21        System.out.println("r2.height="+r2.height);
22     }
23  }
```

執行結果：

```
r1.width=20
r1.height=15
r2.width=25
r2.height=18
```



定義與使用函數

- 類別裡的函數可用下面的語法來定義：

定義函數成員的語法

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

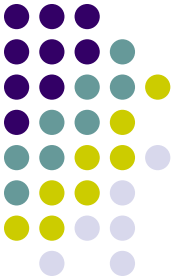
函數的主體

- 呼叫封裝在類別裡的函數的語法：

呼叫函數成員

```
物件名稱.函數名稱(型別 引數 1, 型別 引數 2,...)
```

函數成員的建立



```
01  // Ch8_3, 函數的建立
02  class Rectangle{
03     int width;
04     int height;
05
06     int area(){                    // 定義函數成員 area(), 用來計算面積
07        return width*height;        // 傳回矩形的面積
08     }
09     int perimeter(){               // 定義函數成員 perimeter(), 用來計算周長
10        return 2*(width+height);    // 傳回矩形的周長
11     }
12  }
13
14  public class Ch8_3{
15     public static void main(String[] args){
16        Rectangle r1;
17        r1=new Rectangle();         // 建立新的物件
18
19        r1.width=20;                // 設定矩形 r1 的寬
20        r1.height=15;               // 設定矩形 r1 的高
21        System.out.println("area="+r1.area());
22        System.out.println("perimeter="+r1.perimeter());
23     }
24  }
```

建立area()

建立perimeter()

執行結果：

```
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 Circle：

```
01  // Ch8_4, 圓形類別 Circle
02  class Circle{              // 定義類別 Circle
03    double pi=3.14;          // 將資料成員 pi 設定初值
04    double radius;
05
06    void show_area(){        // show_area() 函數, 顯示出圓面積
07       System.out.printf("area=%6.2f",pi*radius*radius);
08    }
09  }
10  public class Ch8_4{
11     public static void main(String[] args){
12        Circle c1=new Circle();     // 建立 cirl 物件
13        c1.radius=2.0;              // 設定 radius 的值
14        c1.show_area();             // 呼叫 show_area() 函數
15     }
16  }
```

資料成員

函數成員

執行結果：

```
area= 12.56
```

資料成員於記憶體內的配置

```
01  // Ch8_5, 資料成員於記憶體內的配置關係
02  class Circle{           // 定義類別 Circle
03     double pi=3.14;      // 設定資料成員的初值
04     double radius;
05
06     void show_area(){         // show_area() 函數, 顯示出圓面積
07        System.out.printf("pi=%5.2f, area=%6.2f\n",pi,pi*radius*radius);
08     }
09  }
10  public class Ch8_5{
11     public static void main(String[] args){
12        Circle c1=new Circle();          // 建立 c1 物件
13        Circle c2=new Circle();          // 建立 c2 物件
14
15        c1.radius=c2.radius=2.0;
16        c2.pi=3.0;
17        c1.show_area();
18        c2.show_area();
19     }
20  }
```

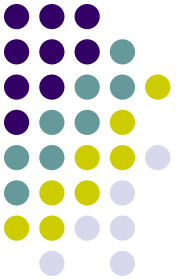
執行結果：

```
pi= 3.14, area= 12.56
pi= 3.00, area= 12.00
```

不同物件的資料成員在記憶體中的配置情形

物件的資料成員在記憶體中有各自的存放空間

資料成員的存取方式



- 在main() 內存取資料成員時，可透過 物件名稱.資料成員名稱

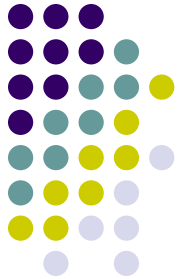
```
01  class Test{
02     public static void main(String[] args){
03        ....
04        c1.radius=2.0;
05        c1.pi=3.0;
06     }
07  }
```

radius 與 pi 均為 c1 的資料成員

- 在類別的內部使用資料成員，可直接取用它的名稱：

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

可直接取用資料成員的名稱



this的使用

- 要強調「物件本身的成員」時，可在成員前面加上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 Circle{
02     double pi=3.14;
03     double radius;
04
05     void show_area(){
06        System.out.println("area="+this.pi*this.radius*this.radius);
07     }
08  }
```

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

函數成員的相互呼叫

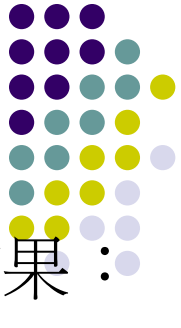
Ch8_6示範如何呼叫在類別內部的函數成員

```
01  // Ch8_6, 在類別內部呼叫函數
02  class Circle{
03     double pi=3.14;
04     double radius;
05
06     void show_area(){          // show_area() 函數, 顯示出圓面積
07        System.out.printf("area=%6.2f\n",pi*radius*radius);
08     }
09     void show_all(){           // show_all() 函數, 同時顯示出半徑與圓面積
10        System.out.printf("radius=%5.2f\n",radius);
11        show_area();            // 於類別內呼叫 show_area() 函數
12     }
13  }
14  public class Ch8_6{
15     public static void main(String[] args){
16        Circle c1=new Circle();
17        c1.radius=2.0;
18        c1.show_all();          // 用 c1 物件呼叫 show_all()
19     }
20  }
```

執行結果：

```
radius= 2.00
area= 12.56
```

以this呼叫函數成員



- Ch8_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=%5.2f\n",radius);
    this.show_area();      // 於類別內呼叫 show_area() 函數
}
```

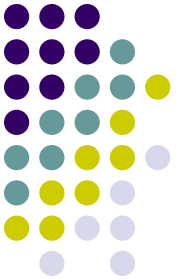
在類別內呼叫其它函數時，可在該函數之前加上 this，此時的 this 即代表呼叫此一函數的物件

- 假設在main() 函數裡有這一行敘述：

```
c1.show_all();              // 用 c1 物件呼叫 show_all()
```

this關鍵字即代表c1

呼叫函數並傳遞引數



```
01  // Ch8_7, 呼叫函數並傳遞引數
02  class Circle{              //   類別 Circle
03     double pi=3.14;         // 將資料成員設定初值
04     double radius;
05
06     void show_area(){    // show_area() 函數, 顯示出半徑及圓面積
07        System.out.printf("radius=%5.2f, ",radius);
08        System.out.printf("area=%6.2f\n",pi*radius*radius);
09     }
10     void setRadius(double r){     // setRadius() 函數, 可用來設定半徑
11        radius=r;                  // 設定 radius 成員的值為 r
12     }
13  }
14  public class Ch8_7{
15     public static void main(String[] args){
16        Circle c1=new Circle();     // 宣告並建立新的物件
17        c1.setRadius(4.0);          // 設定 c1 的半徑為 4.0
18        c1.show_area();
19     }
20  }
```

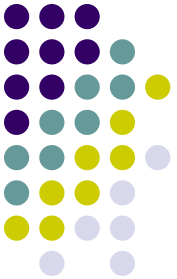
傳遞引數 r

r 是區域變數，一旦離開此範圍，變數 r 即屬無效

執行結果：

```
radius= 4.00, area= 50.24
```

傳遞多個引數



- 下面的程式是傳遞多個引數的範例：

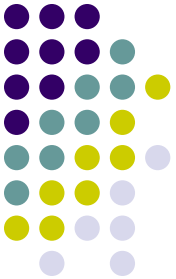
```
01  // Ch8_8, 圓形類別 Circle
02  class Circle{
03     double pi;
04     double radius;
05
06     void show_area(){          // show_area() 函數, 顯示出圓面積
07        System.out.printf("area=%6.2f",pi*radius*radius);
08     }
09     void setCircle(double p,double r){     // 擁有兩個引數的函數
10        pi=p;
11        radius=r;
12     }
13  }
14  public class Ch8_8{
15     public static void main(String[] args){
16        Circle c1=new Circle();     // 宣告並建立新的物件
17        c1.setCircle(3.1416,2.0);  // 呼叫並傳遞引數到 setCircle()
18        c1.show_area();
19     }
20  }
```

若函數本身沒有傳回值，必須在前面加上void

函數沒有傳回值，return敘述可以省略

執行結果：

```
area= 12.57
```



沒有傳回值的函數

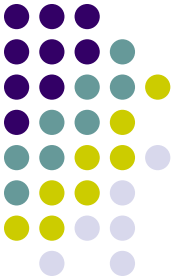
若函數本身沒有傳回值，
則必須在前面加上 void

```
void show_area(){          // show_area() 函數, 顯示出圓面積
   System.out.printf("area=%6.2f",pi*radius*radius);
}
```

```
void show_area(){          // show_area() 函數, 顯示出圓面積
   System.out.printf("area=%6.2f",pi*radius*radius);
   return;
}
```

因為沒有傳回值，所以可在函數結束
前加上 return 敘述，但不連接任何的
運算式，其執行結果與前例相同

有傳回值的函數



- 下面的範例裡增加一個傳回物件半徑的函數：

```
01  // Ch8_9, 圓形類別 Circle
02  class Circle{                  // 定義類別 Circle
03     double pi;                  // 將資料成員設定初值
04     double radius;
05
06     double getRadius(){         // getRadius(), 用來傳回物件的半徑
07        return radius;
08     }
09     void setCircle( double p, double r){
10        pi=p;
11        radius=r;
12     }
13  }
14  public class Ch8_9{
15     public static void main(String[] args){
16        Circle c1=new Circle();     // 宣告並建立新的物件
17        c1.setCircle(3.1416,2.0);
18        System.out.printf("radius=%5.2f",c1.getRadius());
19     }
20  }
```

執行結果：

```
radius= 2.00
```

函數的本體，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

多載的認識 (1/2)

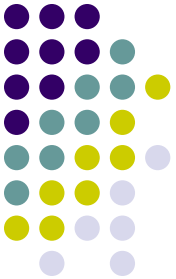
- 本節將以Circle類別做延伸：

```
01  // Ch8_10, 函數的多載(一)
02  class Circle{
03     String color;                    // 資料成員 color
04     double pi=3.14;
05     double radius;
06
07     void setColor(String str){       // 定義設定 color 的函數
08        color=str;
09     }
10     void setRadius(double r){        // 定義設定 radius 的函數
11        radius=r;
12     }
13     void setAll(String str, double r){  // 同時設定 color 與 radius
14        color=str;
15        radius=r;
16     }
17     void show(){                     // 列印半徑、顏色與圓面積
18        System.out.printf("color=%s, Radius=%5.2f\n",color,radius);
19        System.out.printf("area=%6.2f\n",pi*radius*radius);
20     }
21  }
```

這些函數功能相近，卻有不同的函數名稱，使用起來很麻煩

接續下一頁

多載的認識(2/2)



```
22  public class Ch8_10{
23     public static void main(String[] args){
24        Circle c1=new Circle();
25        c1.setColor("Red");          // 設定 c1 的 color
26        c1.setRadius(2.0);           // 設定 c1 的 radius
27        c1.show();
28
29        c1.setAll("Blue",4.0);       // 同時設定 c1 的 color 和 radius
30        c1.show();
31     }
32  }
```

執行結果：

```
color=Red, Radius= 2.00
area= 12.56
color=Blue, Radius= 4.00
area= 50.24
```

這些函數功能相近，名稱不同，使用起來較麻煩

函數的多載 (1/2)

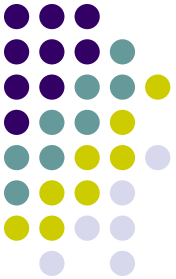
- 下面的例子是函數多載的範例：

```
01  // Ch8_11, 函數的多載(二)
02  class Circle{
03     String color;
04     double pi=3.14;
05     double radius;
06
07     void setCircle(String str){          // 設定 color 成員
08        color=str;
09     }
10     void setCircle(double r){            // 設定 radius 成員
11        radius=r;
12     }
13     void setCircle(String str, double r){  // 同時設定 color 與 radius
14        color=str;
15        radius=r;
16     }
17     void show(){                         // 列印半徑、顏色與圓面積
18        System.out.printf("color=%s, Radius=%5.2f\n",color,radius);
19        System.out.printf("area=%6.2f\n",pi*radius*radius);
20     }
21  }
```

```
25        c1.setCircle("Red");        // 呼叫第 7 行的 setCircle()
26        c1.setCircle(2.0);          // 呼叫第 10 行的 setCircle()
27        c1.show();
28
29        c1.setCircle("Blue",4.0);   // 呼叫第 13 行的 setCircle()
```


接續下一頁

函數的多載 (2/2)



```
22  public class Ch8_11{
23     public static void main(String[] args){
24        Circle c1=new Circle();
25        c1.setCircle("Red");        // 呼叫第 7 行的 setCircle()
26        c1.setCircle(2.0);          // 呼叫第 10 行的 setCircle()
27        c1.show();
28
29        c1.setCircle("Blue",4.0);   // 呼叫第 13 行的 setCircle()
30        c1.show();
31     }
32  }
```

執行結果：

```
color=Red, Radius= 2.00
area= 12.56
color=Blue, Radius= 4.00
area= 50.24
```

```
c1.setCircle("Red");
```

→

```
void setCircle(String str){
   color=str;
}
```

```
c1.setCircle(2.0);
```

→

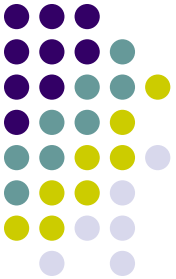
```
void setCircle(double r){
   radius=r;
}
```

```
c1.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，來呼叫相對應的函數



使用多載常犯的錯誤

- 多載不能是引數個數與型態完全相同，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){ ... };
```

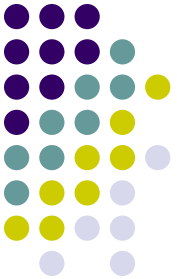
這兩個函數的引數個數和型態完全相同，但傳回型態不同

呼叫setCircle()時，程式無法判斷是哪一個函數被呼叫

- 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);
```

函數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

- Ch8_12的18行將c1物件的radius成員設成-2.0：

```
01  // Ch8_12, 圓形類別 Circle
02  class Circle{                    // 定義類別 Circle
03     double pi=3.14;          // 將資料成員設定初值
04     double radius;
05
06     void show_area(){
07        System.out.printf("area=%6.2f\n",pi*radius*radius);
08     }
09  }
10  public class Ch8_12{
11     public static void main(String[] args){
12        Circle c1=new Circle();
13        c1.radius=-2.0;
14        c1.show_area();
15     }
16  }
```

Circle 類別內部

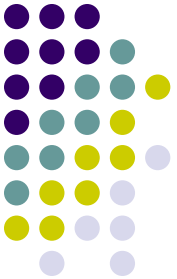
在 Circle 類別外部可以直接更改成員資料

Circle 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

執行結果：

```
area= 12.56
```



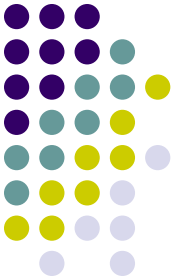
建立私有資料成員

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

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

設定 pi 和 radius 為私有成員

- 把資料成員宣告成private，就無法從類別外部存取到它，可達到資料保護的目的



私有資料成員的範例(1/2)

- Ch8_13在資料成員之前加上private：

```
01  // Ch8_13, 私有成員無法從類別外部來存取的範例
02  class Circle{                          // 設定 field 為私有成員
03     private double pi=3.14;             // 將資料成員設定初值
04     private double radius;
05
06     void show_area(){
07        System.out.printf("area=%6.2f\n",pi*radius*radius);
08     }
09  }
10  public class Ch8_13{
11     public static void main(String args[]){
12        Circle c1=new Circle();
13        c1.radius=-2.0;
14        c1.show_area();
15     }
16  }
```

在 Circle 類別內部，所以可以存取私有成員

在 Circle 類別外部，無法直接更改私有成員

私有資料成員的範例(2/2)

- 編譯、執行時的錯誤訊息，說明私有成員無法從類別外的地方存取：

```
10  public cl
        Run | De
11      public
12          Cir
13          c1.radius=-2.0;
14          c1.show_area();
15      }
16  }
```

```
double radius
The field Circle.radius is not visible Java(33554503)
檢視問題  快速修復... (Ctrl+.)
```

類別外部無法看見 private 成員

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The field Circle.radius is not visible.
        at Ch8_13.main(Ch8_13.java:13)
```

執行Ch8_13將會得到錯誤訊息

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

```
public static void main(String[] args){
    ...
    c1.radius=-2.0;
    ...
}
```

類別外部無法存取到類別內部的 private 成員

建立公有函數成員

- 在類別內加上公有函數setRadius()與私有函數area()：

```
01  // Ch8_14, 公有成員(函數)的建立
02  class Circle{                    // 定義類別 Circle
03     private double pi=3.14;       // 將資料成員設定為 private
04     private double radius;
05
06     private double area(){        // 私有的函數成員 area()
07        return pi*radius*radius;
08     }
09     public void show_area(){      // 公有的函數成員 show_area()
10        System.out.printf("area=%6.2f", area());  // 呼叫私有成員 area()
11     }
12     public void setRadius(double r) {  // 定義公有的函數成員 setRadius()
13        if(r>0) {
14           radius=r;               // 將私有成員 radius 設為 r
15           System.out.printf("radius=%5.2f",radius);
16        }
17        else
18           System.out.println("input error");
19     }
20  }
21  public class Ch8_14{
22     public static void main(String args[]){
23        Circle c1=new Circle();
24        c1.setRadius(-2.0);     // 呼叫公有的 setRadius() 函數
25        c1.show_area();         // 呼叫公有的 show_area() 函數
26     }
27  }
```

執行結果：

```
input error
area=  0.00
```

透過公有成員setRadius()，私有成員radius的值才得以修改

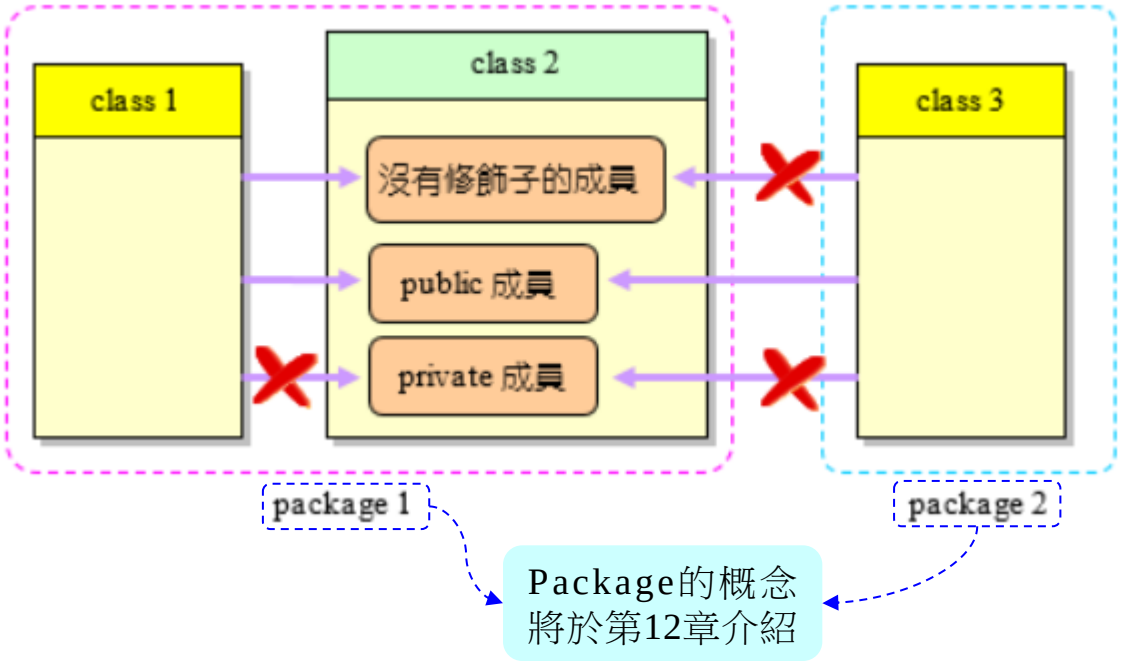
```
class Circle   // 定義類別 Circle{
    ....
    public void setRadius(double r)
    {
        ....
    }
}
```

```
class Ch8_14{
    public static void main(String[] args){
        Circle c1=new Circle();
        c1.setRadius(-2.0);
        c1.show_area();
    }
}
```

c1 物件可以存取到類別內部的 public 成員

public與private

- 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




-The End-