

第九章 類別的進階認識

[9.1 認識建構元與建構元的多載](#)

[9.2 認識「類別變數」與「類別函數」](#)

[9.3 認識類別型態的變數](#)

[9.4 學習利用陣列來儲存物件](#)

[9.5 認識內部類別](#)

9.1 認識建構元與建構元的多載



9-2

9.1 建構元

在建立物件的同時，一併設定資料成員，其方法是利用「建構元」(constructor)。

9.1.1 建構元的基本認識

建構元可視為一種特殊的method，其語法如下：

可以是public
或private
 建構元的名稱必須
和類別名稱相同
 格式9.1.1
建構元的定義格式
 修飾子 類別名稱(型態1 引數1, 型態2 引數2,...)
 {
 程式敘述;

 }
 建構元沒有傳回值

3

9-3

建構元的名稱必須與其所屬之類別的類別名稱相同。

以CCircle類別為例，如果想利用建構元來設定資料成員radius的值，可把CCircle類別的建構元撰寫如下：

```

01 public CCircle(double r) // 定義建構元CCircle()
02 {
03     radius=r;           // 設定資料成員radius的值
04 }
  
```

4

9-4

9.1.2 建構元的呼叫時機

在建立物件時，便會自動呼叫建構元，並執行建構元的內容，因此可利用它來做初始化（initialization）的設定。

5

9-5

下面的例子說明了建構元的使用：

```

01 // app9_1, 建構元的使用
02 class CCircle // 定義類別CCircle
03 {
04     private double pi=3.14;
05     private double radius;
06
07     public CCircle(double r) // 定義建構元CCircle()
08     {
09         radius=r;
10     }
11     public void show()
12     {
13         System.out.println("radius="+radius+", area="+pi*radius*radius);
14     }
15 }
16 public class app9_1
17 {
18     public static void main(String args[])
19     {
20         CCircle cir1=new CCircle(4.0); // 建立物件並呼叫CCircle()建構元
21         cir1.show();
22     }
23 }

```

```

/* app9_1 OUTPUT---
radius=4.0, area=50.24
-----*/

```

6

9.1.3 建構元的多載

9-6

```

01 // app9_2,建構元的多載
02 class CCircle // 定義類別CCircle
03 {
04     private String color;
05     private double pi=3.14;
06     private double radius;
07
08     public CCircle() // 沒有引數的建構元
09     {
10         System.out.println("constructor CCircle() called");
11         radius=1.0;
12         color="Green";
13     }
14     public CCircle(String str, double r) // 有兩個引數的建構元
15     {
16         System.out.println("constructor CCircle(String,double) called");
17         color=str;
18         radius=r;
19     }
20     public void show()
21     {
22         System.out.println("color="+color+", Radius="+radius);
23         System.out.println("area="+pi*radius*radius);
24     }
25 }

```

```

26 public class app9_2
27 {
28     public static void main(String args[])
29     {
30         CCircle cir1=new CCircle(); // 呼叫沒有引數的建構元
31         cir1.show();
32
33         CCircle cir2=new CCircle("Blue",4.0); // 呼叫有引數的建構元
34         cir2.show();
35     }
36 }

```

```

/* app9_2 OUTPUT-----
constructor CCircle() called
color=Green, Radius=1.0
area=3.14
constructor CCircle(String,double) called
color=Blue, Radius=4.0
area=50.24
-----*/

```

7

9.1.4 從某一建構元呼叫另一建構元

9-8

下面的例子是在沒有引數的CCircle() 建構元裡，利用this() 來呼叫有引數的建構元：

8

```

01 // app9_3, 從某一建構元呼叫另一建構元
02 class CCircle // 定義類別CCircle
03 {
04     private String color;
05     private double pi=3.14;
06     private double radius;
07     public CCircle() // 沒有引數的建構元
08     {
09         this("Green",1.0); // 此行會呼叫第13行的建構元
10     }
11     System.out.println("constructor CCircle() called");
12 }
13 public CCircle(String str, double r) // 有引數的建構元
14 {
15     System.out.println("constructor CCircle(String,double) called");
16     color=str;
17     radius=r;
18 }
19 public void show()
20 {
21     System.out.println("color="+color+", Radius="+radius);
22     System.out.println("area="+pi*radius*radius);
23 }
24 }
25 public class app9_3
26 {
27     public static void main(String args[])
28     {
29         CCircle cir1=new CCircle();
30         cir1.show();
    }
}

```

/* app9_3 OUTPUT-----
constructor CCircle(String,double) called
constructor CCircle() called
color=Green, Radius=1.0
area=3.14
-----*/

9

9-10

- ✓ 於某一建構元呼叫另一建構元時，必須以this() 來呼叫，而不能以建構元直接呼叫。
- ✓ this() 必須寫在建構元內第一行的位置。
- ✓ 呼叫沒有引數的建構元時，在this() 的括號裡不必填上任何引數：

this(); // 呼叫沒有引數的建構元

10

9-11

9.1.5 建構元的公有與私有

若建構元為public，則可以在程式的任何地方被呼叫。

如果建構元被設成private，則無法在該建構元所在的類別以外的地方被呼叫。

11

```

01 // app9_4, 公有與私有建構元的比較
02 class CCircle // 定義類別CCircle
03 {
04     private String color;
05     private double pi=3.14;
06     private double radius;
07
08     private CCircle() // 私有建構元
09     {
10         System.out.println("private constructor called");
11     }
12     public CCircle(String str, double r) // 公有建構元
13     {
14         this();
15         color=str;
16         radius=r;
17     }
18     public void show()
19     {
20         System.out.println("color="+color+", Radius="+radius);
21         System.out.println("area="+pi*radius*radius);
22     }
23 }
24 public class app9_4
25 {
26     public static void main(String args[])
27     {
28         CCircle cir1=new CCircle("Blue",1.0);
29         cir1.show();
30     } }

```

```

/* app9_4 OUTPUT-----
private constructor called
color=Blue, Radius=1.0
area=3.14
-----*

```

12

9-14

9.1.6 建構元的省略

如果省略建構元，Java會呼叫預設的建構元（default constructor）。

預設建構元的格式如下：

```
public CCircle()    // 預設的建構元
{
}

```

格式9.1.2

預設的建構元。如果沒有事先定義好建構元，則Java會使用此一版本的建構元

如果自行撰寫了建構元，無論是否有引數，則Java會假設已備妥好所有的建構元，不會再提供預設的建構元。

13

問題1：錯在哪裡？

```

01 //
02 class CRectangle
03 {
04     int width;
05     int height;
06
07     public CRectangle(int w,int h)
08     {
09         width=w;
10         height=h;
11         System.out.println("constructor CRectangle(int w,int h) called");
12
13     public CRectangle()
14     {
15         System.out.println("constructor called");
16         CRectangle(10,8);
17     }

```

14

問題2：錯在哪裡？

```

01 //
02 class CRectangle
03 {
04     int width;
05     int height;
06
07     public CRectangle(int w,int h)
08     {
09         width=w;
10         height=h;
11         System.out.println("constructor CRectangle(int w,int h) called");
12     }
13     public CRectangle()
14     {
15         System.out.println("constructor called");
16         this(10,8);
17     }

```

15

ex9_1_1.java; ex9_1_2.java

假設CRectangle類別的定義如下：

```

class CRectangle
{
    int width;
    int height;
}

```

- (a) 試設計一個建構元CRectangle(int w, int h)，當此建構元呼叫時，便會自動設定width=w，height=h。 **ex9_1_1.java**
- (b) 請接續 (a) 的部份，請再設計一個沒有引數的建構元CRectangle()，使得當此建構元呼叫時，便會自動設定width=10，height=8（請不要使用this()來設定）。
ex9_1_1.java
- (c) 同 (b) 小題，但width與height的值請用this()來設定。 **ex9_1_2.java**

```

(a) (b)
/* output-----
constructor CRectangle(int w,int h) called
width=5
height=2
constructor CRectangle() called
width=10
height=8
-----*/

```

```

(c)
/* output-----
constructor CRectangle(int w,int h) called
width=5
height=2
constructor CRectangle(int w,int h) called
constructor CRectangle() called
width=10
height=8
-----*/

```

16

9.1 回家作業 建構元

hw9_1_1.java

於下列的程式碼中，當您編譯它時會有錯誤訊息。試嘗試找出錯誤之所在，並修正之。

class CCircle // 定義類別CCircle

```
{
    private double pi=3.14;
    private double radius;
    public CCircle(double r) // 定義建構元CCircle()
    {
        radius=r;
    }
    public void show()
    {
        System.out.println("radius=" + radius);
    } }
```

```
/* output-----
   radius=1.0
   -----*/
```

```
public class hw9_1_1
{
    public static void main(String args[])
    {
        CCircle cir1=new CCircle();
        cir1.show();
    } }
```

17

9.1 回家作業 建構元

hw9_1_2.java

於下列的程式碼中，當您編譯它時會有錯誤訊息。試嘗試找出錯誤之所在，並修正之。

class CSquare // 定義類別CSquare

```
{
    private int length;
    public void CSquare()
    {
        length=10;
    }
    public void CSquare(int len)
    {
        length=len;
    } }
```

```
/* output-----
   length=5
   area=25
   -----*/
```

```
public void show()
{
    System.out.println("length="+length);
    System.out.println("area="+length*length);
} }
public class hw9_1_2
{
    public static void main(String args[])
    {
        CSquare squ=new CSquare(5);
        squ.show();
    } }
```

18

9.2

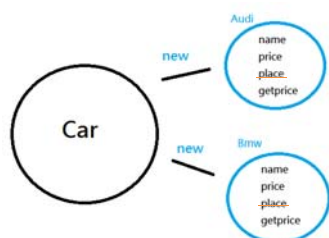
認識「類別變數」與「類別函數」



重點摘要

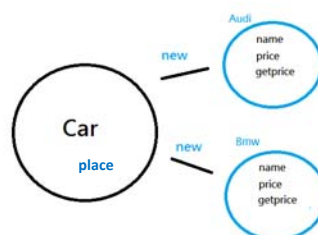
- 新生成的物件稱為instance(實例)
 - 實例變數Instance variable:
 - 物件建立時，變數各自獨立，擁有不同的記憶體空間稱之。
 - 實例方法Instance method:
 - 必須建立物件，再利用物件呼叫他，具有這些特性的method，稱為instance method

類別變數(Static)的概念-再整理



但是如果某個變數不需要每個物件都各自擁有，而是固定的，或者需要共享，就可以宣告static，例如發售地點(place)，每台車都在台灣，怎麼做？

被宣告static的變數(place)或方法，不會讓物件(Bmw、Audi)各自擁有，而是類別(Car)所擁有，簡單來說"只要這個類別被載入，不用物件化便可以使用"。



21

9.2類別變數(class variable)與類別函數(class method)

9-15

9.2.1實例變數與實例函數(instance variable and instance method)

app9_5是一個很簡單的程式，可以認識實例變數與實例函數。

```
01 // app9_5, 簡單的範例,實例變數與實例函數
02 class CCircle // 定義類別CCircle
03 {
04     private double pi=3.14;
05     private double radius;
06
07     public CCircle(double r) // CCircle()建構元
08     {
09         radius=r;
10     }
11     public void show()
12     {
13         System.out.println("area="+pi*radius*radius);
14     }
15 }
16 public class app9_5
17 {
18     public static void main(String args[])
19     {
20         CCircle cir1=new CCircle(1.0);
21         cir1.show(); // show()必須透過物件來呼叫
22         CCircle cir2=new CCircle(2.0);
23         cir2.show(); // show()必須透過物件來呼叫
24     }
25 }
```

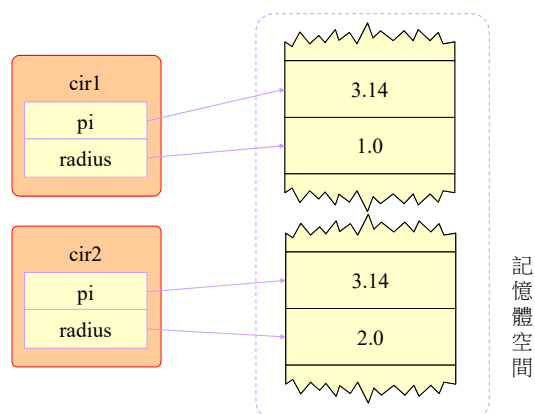
```
/* app9_5 OUTPUT--
area=3.14
area=12.56
-----*
```

22

9-17

實例變數(instance variable)

物件各自擁有儲存資料成員的記憶體空間，不與其它物件共用：



23

9-18

實例函數(instance method)

必須透過物件來呼叫的函數稱為實例函數：

```
CCircle cir1=new CCircle(1.0);    // 建立物件cir1
cir1.show();                      // 由物件cir1呼叫show() method
CCircle cir2=new CCircle(2.0);    // 建立物件cir2
cir2.show();                      // 由物件cir2呼叫show() method
```

24

9.2.2 類別變數(class variable)

9-19

「實例變數(instance variable)」是各別物件所有，彼此之間不能共享。

「類別變數(class variable)」是由所有的物件共享。

如要把變數宣告為「類別變數」，必須在變數之前加上 **static** 修飾子。

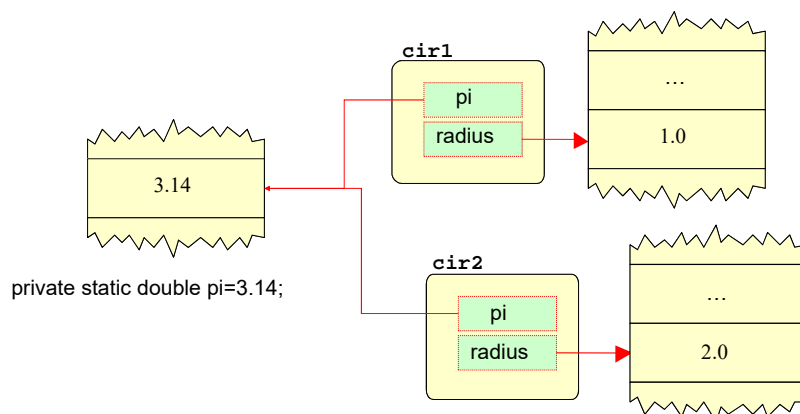
25

假設CCircle類別裡的變數pi，想要改為「類別變數」，可將它宣告成：

9-20

```
private static double pi=3.14; // 將pi宣告為「類別變數」
```

下圖是把pi宣告成static之後，變數與記憶體之間的配置關係：



26

```
// app9_6, 「類別變數」的使用
01 class CCircle // 定義類別CCircle
02 {
03     private static int count=0; // 宣告count為「類別變數」
04     private static double pi=3.14; // 宣告pi為「類別變數」
05     private double radius;
06
07     public CCircle() // 沒有引數的CCircle()建構元
08     {
09         this(1.0); // 呼叫第12行的建構元，並傳入1.0
10     }
11     public CCircle(double r) // 有一個引數的CCircle()建構元
12     {
13         radius=r;
14         count++; // 當此建構元被呼叫時，count便加1
15     }
16     public void show()
17     {
18         System.out.println("area="+pi*radius*radius);
19     }
20     public void show_count()// show_count(),顯示目前物件建立的個數
21     {
22         System.out.println(count+" object(s) created");
23     }
24 }
```

```
1 object(s) created
3 object(s) created
3 object(s) created
3 object(s) created
```

```
public class app9_6
{
    public static void main(String args[])
    {
        CCircle cir1=new CCircle(); // 呼叫第7行的建構元
        cir1.show_count(); // 用cir1物件呼叫show_count() method
        CCircle cir2=new CCircle(2.0); // 呼叫第12行的建構元
        CCircle cir3=new CCircle(4.3); // 呼叫第12行的建構元
        cir1.show_count(); // 用cir1物件呼叫show_count() method
        cir2.show_count(); // 改用cir2物件呼叫show_count() method
        cir3.show_count(); // 改用cir3物件呼叫show_count() method }
}
```

27

9.2.3 類別函數(class method)

9-23

若將method定義成類別函數，則可以直接由類別來呼叫：

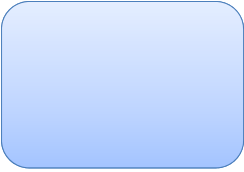
- static跟non-static的差異
 - 就是載入記憶體的時間。
 - 程式剛載入還沒執行的時候，static的方法已經載入記憶體且隨時可以被呼叫，但這個時候一般變數的成員(non-static)還沒被初始化，還沒在記憶體中佔有空間，就是根本還不存在。
 - 所以static的方法在程式裡面限定只能存取static的欄位，因為要確保存取的到東西。

28

```

01 // app9_7, 「類別變數」的使用
02 class CCircle // 定義類別CCircle
03 {
04     private static int num=0;    // 宣告num為「類別變數」
05     private static double pi=3.14; // 宣告pi為「類別變數」
06     private double radius;
07
08     public CCircle()            // 沒有引數的CCircle()建構元
09     {
10         this(1.0);              // 呼叫第12行的建構元，並傳入1.0
11     }
12     public CCircle(double r)    // 有一個引數的CCircle()建構元
13     {
14         radius=r;
15         num++;                // 當此建構元被呼叫時，num便加1
16     }
17     public void show()
18     {
19         System.out.println("area="+pi*radius*radius);
20     }
21     public static void count() // count(), 用來顯示目前物件建立的個數
22     {
23         System.out.println(num+" object(s) created");
24     }
25 }

```



```

26 public class app9_7
27 {
28     public static void main(String args[])
29     {
30         CCircle.count(); // 用CCircle類別呼叫count() method
31         CCircle cir1=new CCircle(); // 呼叫第8行的建構元
32         CCircle.count(); // 用CCircle類別呼叫count() method
33         CCircle cir2=new CCircle(2.0); // 呼叫第12行的建構元
34         CCircle cir3=new CCircle(4.3); // 呼叫第12行的建構元
35         cir3.count(); // 用cir3物件呼叫count() method
36     } }

```

9.2.4 「類別函數」使用的限制

9-25

「類別函數」無法存取「實例變數」與「實例函數」

如果在app9_7中撰寫如下的程式碼：

```

public static void count()
{
    System.out.println(num+" object(s) created");
    System.out.println("radius="+radius); // 錯誤
    show();    // 錯誤
}

```

編譯時將產生錯誤。

30

「類別函數」內部不能使用this關鍵字

9-26

下面的程式碼是錯誤的：

```
public static void count()
{
    // 錯誤，不可使用this
    System.out.println(this.num+" object(s) created");
    System.out.println("radius="+radius);
}
```

31

```
01 // app9_7,「類別變數」的使用
02 class CCircle // 定義類別 CCircle
03 {
04     private static int num=0; // 宣告 num 為「類別變數」
05     private static double pi=3.14; // 宣告 pi 為「類別變數」
06     private double radius;
07
08     public CCircle() // 沒有引數的 CCircle() 建構元
09     {
10         this(1.0); // 呼叫第 12 行的建構元，並傳入 1.0
11     }
12     public CCircle(double r) // 有一個引數的 CCircle() 建構元
13     {
14         radius=r;
15         num++; // 當此建構元被呼叫時，num 便加 1
16     }
17     public void show()
18     {
19         System.out.println("area="+pi*radius*radius);
20     }
21     public static void count() // count(), 用來顯示目前物件建立的個數
22     {
23         System.out.println(num+" object(s) created");
24     }
25     System.out.println("radius="+radius);
26     show();
27
28 public class app9_7
29 {
30     public static void main(String args[])
31     {
32         CCircle.count(); // 用 CCircle 類別呼叫 count() method
33         CCircle cir1=new CCircle(); // 呼叫第 8 行的建構元
34         CCircle.count(); // 用 CCircle 類別呼叫 count() method
35         CCircle cir2=new CCircle(2.0); // 呼叫第 12 行的建構元
36         CCircle cir3=new CCircle(4.3); // 呼叫第 12 行的建構元
37         cir3.count(); // 用 cir3 物件呼叫 count() method
38     }
39 }
```

```
/* app9_7 OUTPUT----
0 object(s) created
1 object(s) created
3 object(s) created
*****
```

32

ex9_2_1.java

試撰寫一個可以計算 $1+2+\dots+n$ 的類別函數
add2n(int n) method。

```
/* output-----
1+2+...+5=15
1+2+...+10=55
-----*/
```

33

ex9_2_2.java

試依題意回答下列各題：

- 試設計類別CCount，內含cnt變數（初值設為0）與count() method，只要每建立一個物件，cnt的值便加1。也就是說，cnt可用來追蹤CCount物件建立的個數。
- 試設計setZero() method，當此method呼叫時，cnt的值會被歸零。
- 試設計setValue(int n) method，當此method呼叫時，cnt的值會被設為n。
- 於本例中，cnt變數應該利用「實例變數」還是「類別變數」？為什麼？
- 於本例中，count() method應該宣告成「實例函數」還是「類別函數」？或者是兩者都可以？

```
/* output-----
cnt=2
using setZero()...
cnt=0
using setValue(10)...
cnt=10
-----*/
```

34

9.2 回家作業

類別變數與類別函數

hw9_2_1.java

試撰寫一個類別函數 `power(int x, int n)` method，用來計算 `x` 的 `n` 次方。

```
/* output-----
2 的5 次方=32
3 的2 次方=9
-----*/
```

35

9.2 回家作業

類別變數與類別函數

hw9_2_2.java

試依題意回答下列各題:

- 試設計類別 `CWin`，內含 `cnt` 變數(初值設為 0)與 `count()` method，只要每建立一個物件，`cnt` 的值便加 1。
也就是說，`cnt` 可用來追蹤 `CWin` 物件建立的個數。
- 試設計一個建構元 `Cwin(String str, int w, int h)`，當此建構元呼叫時，便會自動設定 `color=str`, `width=w`, `height=h`。
- 請接續 (a) 的部份，請再設計一個沒有引數的建構元 `CWin()`，使得當此建構元呼叫時，便會自動設定 `color="Red"`, `width=2`, `height=2`。
- 試設計 `setZero()` method，當此 method 呼叫時，`cnt` 的值會被歸零。
- 試設計 `setValue(int n)` method，當此 method 呼叫時，`cnt` 的值會被設為 `n`。
- 於本例中，`cnt` 變數應該利用「實例變數」還是「類別變數」？為什麼？
- 於本例中，`count()` method 應該定義成「實例函數」還是「類別函數」？或者是兩者都可以？

```
/* output-----
cnt=2
using setZero()...
cnt=0
using setValue(10)...
cnt=10
cnt=11
-----*/
```

36

9.3 認識類別型態的變數



9.3 類別型態的變數

9-27

由類別宣告而得的變數，稱之為「類別型態的變數」，它是屬於「非基本型態的變數」的一種。

下列的程式碼則是宣告了cir1為CCircle類別型態的變數：

```
CCircle cir1;  
cir1=new CCircle();
```

38

9.3.1 設值給類別型態的變數

即使沒有用new產生新的物件，依然可對類別型態的變數設值：

```

01 // app9_8, 設值給類別型態的變數
02 class CCircle // 定義類別CCircle
03 {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle(double r)
08     {
09         radius=r;
10     }
11     public void show()
12     {
13         System.out.println("area="+pi*radius*radius);
14     }
15 }
16 public class app9_8
17 {
18     public static void main(String args[])
19     {
20         CCircle cir1,cir2; // 宣告cir1,cir2為類別型態的變數
21         cir1=new CCircle(1.0); // 建立新的物件，並將cir1指向它
22         cir1.show();
23
24         cir2=cir1; // 將cir1設給cir2，此時這兩個變數所指向的內容均相等
25         cir2.show();
26
27         CCircle cir3=new CCircle(2.0); // 建立新的物件，並將cir3指向它
28         cir3.show(); }

```

```

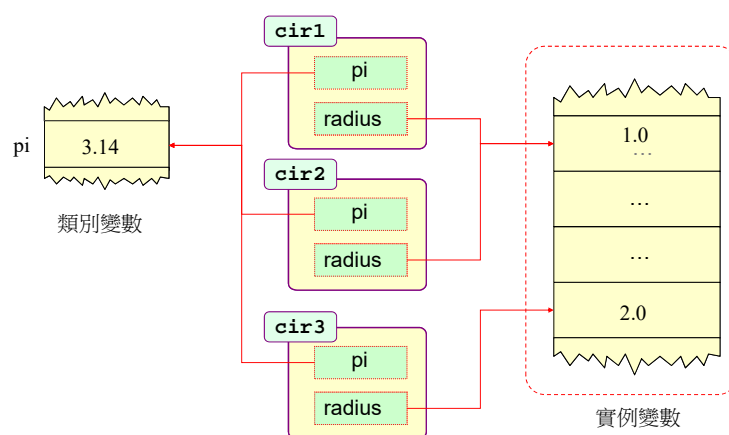
/* app9_8 OUTPUT--
area=3.14
area=3.14
area=12.56
-----*/

```

39

9-30

透過24行的設定，即可將二個不同名稱的變數指向同一個物件：



40

透過其中一個變數對物件做更動，另一變數所指向之物件內容也會隨著更改，下面的例子說明了其中的道理：

```

01 // app9_9, 類別型態之變數的應用
02 class CCircle // 定義類別CCircle
03 {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle(double r) // CCircle建構元
08     {
09         radius=r;
10     }
11     public void setRadius(double r)
12     {
13         radius=r;    // 設定radius成員的值
14     }
15     public void show()
16     {
17         System.out.println("area="+pi*radius*radius);
18     } }
20 public class app9_9
21 {
22     public static void main(String args[])
23     {
24         CCircle cir1,cir2;
25         cir1=new CCircle(1.0);
26         cir1.show();
27
28         cir2=cir1; // 將cir1設給cir2，此時這兩個變數所指向的內容均相等
29         cir2.setRadius(2.0); // 將cir2物件的半徑設為2.0
30         cir1.show();
31     } }

```

```

/* app9_9 OUTPUT--
area=3.14
area=12.56
-----*/

```

41

```

20 public class app9_9
21 {
22     public static void main(String args[])
23     {
24         CCircle cir1,cir2;
25         cir1=new CCircle(1.0);
26         cir1.show();
27
28         cir2=cir1; // 將cir1設給cir2，此時這兩個變數所指向的內容均相等
29         cir2.setRadius(2.0); // 將cir2物件的半徑設為2.0
30         cir1.show();
31     }
32 }

```

```

/* app9_9 OUTPUT--
area=3.14
area=12.56
-----*/

```

42

9-33

9.3.2 以類別型態的變數傳遞引數

想撰寫compare() method，用來比較呼叫物件cir1與compare()裡的引數cir2的資料成員是否完全相同，可用下面的敘述來完成：

```
cir1.compare(cir2);
```

compare() method的定義必須以下面的格式來撰寫：

傳回值型態 compare(**CCircle**obj)

```
{
    ....
}
```

引數型態為CCircle

格式9.3.1

傳遞類別型態的變數

43

下面的範例設計了compare() method，用來比較二個物件是否相等：

```
01 // app9_10, 傳遞類別型態的變數
02 class CCircle
03 {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle(double r) // CCircle()建構元
08     {
09         radius=r;
10     }
11     public void compare(CCircle cir) // compare() method
12     {
13         if(this.radius==cir.radius) // 判別物件的radius成員是否相等
14             System.out.println("radius are equal");
15         else
16             System.out.println("radius are not equal");
17     }
18 }
19 public class app9_10
20 {
21     public static void main(String args[])
22     {
23         CCircle cir1=new CCircle(1.0);
24         CCircle cir2=new CCircle(2.0);
25         cir1.compare(cir2); // 比較cir1與cir2的radius是否相等
26     }
27 }
```

```
/* app9_10 OUTPUT---
radius are not equal
-----*/
```

44

下面的範例設計了compare() method，用來比較二個物件是否相等：

```

01 // app9_10_1, 傳遞類別型態的變數
02 class CCircle
03 {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle(double r) // CCircle()建構元
08     {
09         radius=r;
10     }
11     public void compare(CCircle cir) // compare() method
12     {
13         if(this==cir) // 判別this和cir是否指向同一個物件
14             System.out.println("two objects have the same contents");
15         else
16             System.out.println(" two objects have different hash values");
17     }
18 }
19 public class app9_10
20 {
21     public static void main(String args[])
22     {
23         CCircle cir1=new CCircle(1.0);
24         CCircle cir2=new CCircle(2.0);
25         cir1.compare(cir2); // 比較cir1與cir2的radius是否相等
26     }
27 }

```

若this and cir是指不同的物件，即使所有成員的內容都相同，比較結果仍然是不同。

/* app9_10 OUTPUT----
radius are not equal
-----*/

45

9.3.3 由method傳回類別型態的變數

9-36

以compare() method為例，如要傳回CCircle類別型態的變數，可利用下面的語法來撰寫：

傳回型態為CCircle類別的變數

```

CCircle compare( CCircle obj)
{
    ....
}

```

格式9.3.2
由method傳回類別
型態的變數

46

下面的範例以compare()比較物件半徑的大小，並傳回半徑較大的物件：

```

01 // app9_11, 由method傳回類別型態的變數
02 class CCircle // 定義類別CCircle
03 {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle(double r)    // CCircle建構元
08     {
09         radius=r;
10     }
11     public CCircle compare(CCircle cir) // Compare() method
12     {
13         if(this.radius>cir.radius)
14             return this;        // 傳回呼叫compare() method的物件
15         else
16             return cir;         // 傳回傳入compare() method的物件
17     }
18 }
19 public class app9_11
20 {
21     public static void main(String args[])
22     {
23         CCircle cir1=new CCircle(1.0);
24         CCircle cir2=new CCircle(2.0);
25         CCircle obj;
26
27         obj=cir1.compare(cir2);    // 呼叫compare() method
28         if(cir1==obj)
29             System.out.println("radius of cir1 is larger");
30         else
31             System.out.println("radius of cir2 is larger");
32     }

```

```

/* app9_11 OUTPUT-----
radius of cir2 is larger
-----*/

```

47

簡化問題

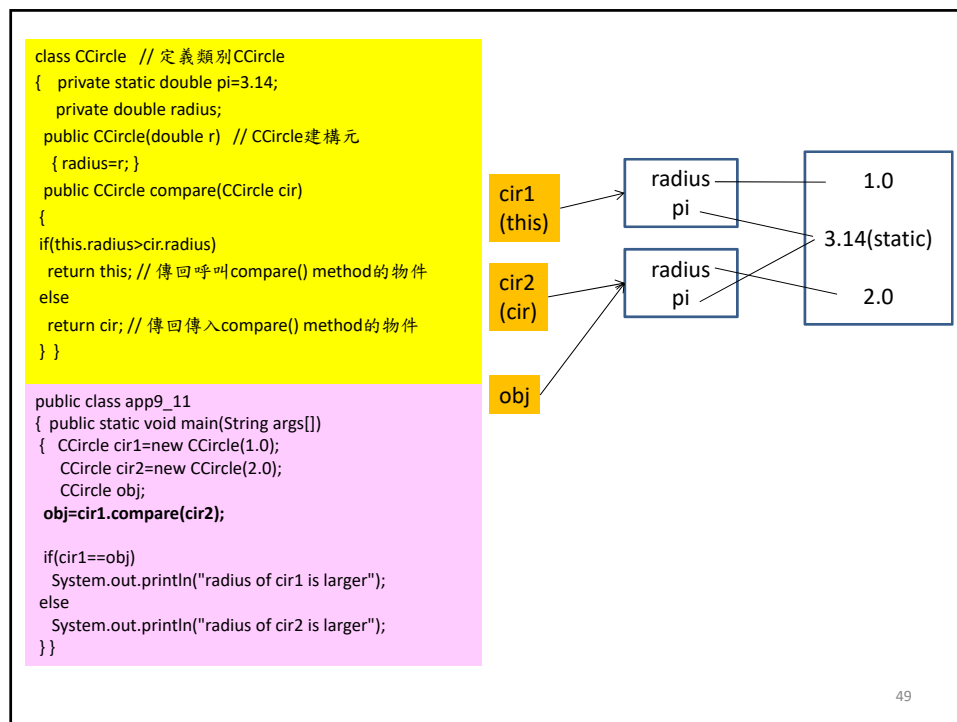
```

class CSphere
{
    int show(int c)
    {
        return c;
    }
}

public class ex
{
    public static void main(String[] argv)
    {
        int a;
        CSphere s=new CSphere();
        a=s.show(3);
        System.out.println(a);
    }
}

```

48



9.3.4 回收記憶體

9-39

如果此物件不再使用了，可收回被它所佔用的記憶體空間，
如下面的程式碼片段：

```

01 class app
02 {
03     public static void main(String args[])
04     {
05         CCircle cir1=new CCircle(1.0); // 建立物件，並配置記憶體給它
06         ....
07         cir1=null; // 將cir1指向null,代表cir1已不再指向任何物件
08         ....
09     }
10 }

```

一經設定為null，該變數便不指向任何物件。

50

9-40

於下面的程式碼中，Java的蒐集殘餘記憶體機制不會回收。

```

01 class app
02 {
03     public static void main(String args[])
04     {
05         CCircle cir1=new CCircle();
06         CCircle cir2;
07         cir2=cir1; // 設定cir2與cir1均指向同一個物件
08         ....
09         cir1=null; // 將cir1指向null,但cir2仍指向該物件，因此不會被回收
10     }

```

51

ex9_3_1.java

app9_11的compare() method 是撰寫在CCircle 類別內。
試修改compare() method，使得它是類別 app9_11 裡的
函數成員，而不是CCircle 類別的函數成員。

// ex9_3_1, 由method傳回類別型態的變數

class CCircle // 定義類別CCircle

```

{
    private static double pi=3.14;
    private double radius;

    public CCircle(double r) // CCircle建構元
    {
        radius=r;
    }

    public CCircle compare(CCircle cir) // Compare() method
    {
        if(this.radius>cir.radius)
            return this; // 傳回呼叫compare() method的物件
        else
            return cir; // 傳回傳入compare() method的物件
    }
}

```

```

public class app9_11
{
    public static void main(String args[])
    {
        CCircle cir1=new CCircle(1.0);
        CCircle cir2=new CCircle(2.0);
        CCircle obj;

        obj=cir1.compare(cir2); // 呼叫compare()
        method
        if(cir1==obj)
            System.out.println("radius of cir1 is larger");
        else
            System.out.println("radius of cir2 is larger");
    }
}

```

提示：obj=compare(cir1,cir2);

```

/* output-----
radius of cir2 is larger
-----*/

```

52

ex9_3_2.java

假設我們想設計一類別CRational，
可用來處理分數的一些相關運算。
CRational 類別初步的撰寫如下：

```
class CRational // 分數類別
{
    public int n;
    public int d;
    public void setN(int num) // 設定分子
    { n=num; }
    public void setD(int num) // 設定分母
    { d=num; }
    public void show()
    {}
}
```

```
(a)
/* output---
2/5
3/7
-----*/
```

```
public class ex9_3_2
{
    public static void main(String args[])
    {
        CRational aaa=new CRational();
        aaa.setN(2);
        aaa.setD(5);
        aaa.show();
    }
}
```

上面的程式碼初步定義了CRational 類別，它具有兩個資料成員，分別為分子（numerator）n 與分母（denominator）d，以及三個method，分別為用來設定分子與分母的setN()與setD()，和用來顯示分數的show()。在main() 裡，我們設定了分子為2，分母為5，最後並利用show() 顯示分數。如果執行此一程式，可得到如下的執行結果：2/5

(a) 在CRational 類別裡，setN() 與setD() method 分別是用來設定分子與分母。試撰寫另一method，**public void setND(int num, int den)**，可用來同時設定分數的分子與分母。

53

ex9_3_3.java

(b) 試將CRational 類別裡的show() method 改寫成類別ex9_3_2 裡的函數成員，而非CRational 類別裡的函數成員。

```
(b)
/* output---
2/5
3/7
-----*/
```

54

9.3 回家作業

類別型態的變數

hw9_3_1.java

下列各題接續習題 ex9_3_2.java，試在 CRational 類別裡加入分數運算的相關 method，其中所有 method 的傳回型態皆為 CRational，且分數運算的結果不需化成最簡分數：

- (a) 加入 add(CRational r) method，它可用來將呼叫它的 CRational 物件與傳入的引數進行分數的相加，並傳回相加後的結果。
- (b) 加入 sub(CRational r) method，它可用來將呼叫它的 CRational 物件與傳入的引數進行分數的相減，並傳回相減後的結果。
- (c) 加入 mul(CRational r) method，它可用來將呼叫它的 CRational 物件與傳入的引數進行分數的相乘，並傳回相乘後的結果。
- (d) 加入 div(CRational r) method，它可用來將呼叫它的 CRational 物件與傳入的引數進行分數的相除，並傳回相除後的結果。

```
/* output-----
(2/5)+(3/7)=29/35
(2/5)-(3/7)=-1/35
(2/5)*(3/7)=6/35
(2/5)/(3/7)=14/15
-----*/
```

55

9.3 回家作業

類別型態的變數

hw9_3_2.java

接續習題 ex9_3_2.java，試於 CRational 類別中撰寫 compare(CRational r1, CRational r2) method，可用來比較分數 r1 與 r2 的大小，並傳回較大者。compare() 請用類別函數來撰寫。

```
/* output-----
2/5
3/7
3/7 is larger
-----*/
```

57

9.4 學習利用陣列來儲存物件



9.4 利用陣列來儲存物件

9-41

用陣列來存放物件，必須經過下面兩個步驟：

1. 宣告類別型態的陣列變數，並用new配置記憶空間給陣列。
2. 用new產生新的物件，並配置記憶空間給它。

如下列的語法：

```
01 CCircle cir[];
02 cir=new Circle[3];
```

} 宣告CCircle類別型態的陣列變數，並用new配置記憶空間

建立好陣列之後，便可把陣列元素指向由CCircle類別所建立的物件：

```
03 cir[0]=new CCircle();
04 cir[1]=new CCircle();
05 cir[2]=new CCircle();
```

} 用new建立新的物件，並配置記憶空間給它

60

也可以把第1行與第2行合併成一行：

9-42

```
01 CCircle cir[]=new Circle[3]; // 建立物件陣列元素，並配置記憶空間
```

或者利用for迴圈來完成指向新建立物件之動作：

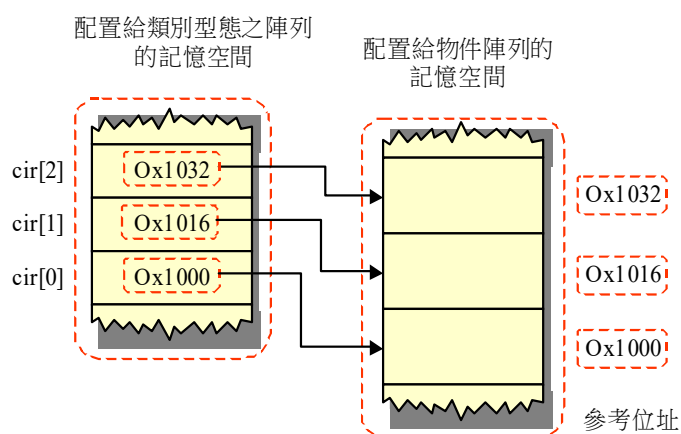
```
03 for(int i=0; i<cir.length; i++)
04 {
05     cir[i]=new CCircle();
06 }
```

```
03 cir[0]=new CCircle();
04 cir[1]=new CCircle();
05 cir[2]=new CCircle();
```

61

下圖為類別型態的陣列與物件陣列的記憶空間配置情形：

9-43



62

9.4.1 建立物件陣列的範例

下面的程式碼是物件陣列的使用範例：

```

01 // app9_12, 建立物件陣列
02 class CCircle // 定義類別CCircle
03 {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle(double r) // CCircle建構元
08     {
09         radius=r;
10     }
11     public void show()
12     {
13         System.out.println("area="+pi*radius*radius);
14     }
15 }

16 public class app9_12
17 {
18     public static void main(String args[])
19     {
20         CCircle cir[]; // 宣告類別型態的陣列。
21         cir=new CCircle[3]; // 並用new配置記憶體空間
22         cir[0]=new CCircle(1.0); // 用new產生新的物件，
23         cir[1]=new CCircle(4.0); // 並配置給陣列元素
24         cir[2]=new CCircle(2.0);
25
26         cir[1].show(); // 利用物件cir[1]呼叫show() method
27         cir[2].show(); // 利用物件cir[2]呼叫show() method
28     }
}

```

/* app9_12 OUTPUT--
area=50.24
area=12.56
-----*/

9.4.2 傳遞物件陣列到method裡

app9_13是傳遞物件陣列的練習：

```

01 // app9_13, 傳遞物件陣列到method
02 class CCircle // 定義類別CCircle
03 {
04     private static double pi=3.14;
05     private double radius;
06     public CCircle(double r)
07     { radius=r; }
08
11     public static double compare(CCircle c[]) // compare() method
12     {
13         double max=0.0;
14         for(int i=0;i<c.length;i++)
15             if(c[i].radius>max)
16                 max=c[i].radius;
17         return max; }
18 }
21 public class app9_13
22 {
23     public static void main(String args[])
24     {
25         CCircle cir[];
26         cir=new CCircle[3];
27         cir[0]=new CCircle(1.0);
28         cir[1]=new CCircle(4.0);
29         cir[2]=new CCircle(2.0);
30         System.out.println("Largest radius = "+CCircle.compare(cir)); // cir是一個類別型態的陣列變數
31     }
32 }

```

引數資料型態為CCircle
傳遞陣列

傳遞陣列時，括號內填上陣列名稱即可

/* app9_13 OUTPUT--
Largest radius = 4.0
-----*/

64

9-48

第11行的語法解說如下：

```
public static double compare(CCircle c[])
```

引數型態為CCircle
傳回型態為double
傳遞陣列

傳遞陣列時，compare()的括號內所填上的是陣列的名稱：

```
CCircle.compare(cir)
```

傳遞陣列時，括號內填上陣列名稱即可

65

ex9_4_1.java

試修改app9_13，加入average(CCircle c[]) method，
用來傳回CCircle 物件陣列裡所有radius 成員的平均值。

```
/* output-----
average radius=2.333333333333335
-----*/
```

66

ex9_4_2.java

利用 CScore 來記錄學生的資料，其類別的定義如下：

```
class CScore
{
    private String name; // 姓名
    private int math;    // 數學成績
    private int eng;     // 英文成績
}
```

試依題意回答下列各題：

(a) 設計一個建構元 CScore(String str, int m, int e)，當此建構元呼叫時，便會自動設定 name=str, math=m, eng=e。

(b) 請利用 CScore 類別型態的陣列儲存以下的資料：

Name	Math	English
Fiona	86	92
Ryan	95	79
Ariel	81	83

```
/* output-----
Name Math English
Fiona 86 92
Ryan 95 79
Ariel 81 83
-----*/
```

(c) 試撰寫 show(CScore c[]) method，用來顯示類別物件陣列內所有的資料。傳入 show() method 的是一個整個類別物件陣列。輸出的格式請參考(b)題中的表格。

67

9.4 回家作業

利用陣列來儲存物件

hw9_4_1.java

試將 app9_13 改為利用 for 迴圈來輸入資料，cir[0] 傳入 0、cir[1] 傳入 1、cir[2] 傳入 2，依此類推，一直到 cir[5]，最後並列印出輸入的結果與每一個物件的面積。

$\pi * radius * radius$ area=28.259999999999998
 若 $radius * radius * \pi$ area=28.26
 兩者答案均可

```
/* output-----
radius=0.0, area=0.0
radius=1.0, area=3.14
radius=2.0, area=12.56
radius=3.0,
area=28.259999999999998
radius=4.0, area=50.24
radius=5.0, area=78.5
-----*/
```

68

9.4 回家作業 利用陣列來儲存物件

hw9_4_2.java

請撰寫一個程式, 其中包含一個類別 Dates, 並在建構方法中初始化一個含有 7 個元素的字串陣列, 各個元素對應到星期一到星期天的英文縮寫, 並提供一個方法 askDate(), 傳入 1~7 的數字, 傳回對應的英文縮寫。

```
/* output-----  
星期1 Mon  
星期2 Thu  
星期3 Wed  
星期4 Tus  
星期5 Fri  
星期6 Sat  
星期7 Sun  
-----*/
```

69

9.5 認識內部類別



9-49

9.5 內部類別與巢狀類別

類別A的內部再定義一個類別B，此時類別B稱為內部類別（inner class），而類別A則稱為外部類別（outer class）。

下面列出了內部類別的定義格式：

```

修飾子 class 外部類別的名稱
{
    // 外部類別的成員
    修飾子 class 內部類別的名稱
    {
        // 內部類別的成員
    }
}

```

格式9.5.1
定義內部類別

外部類別

內部類別

71

9-50

9.5.1 內部類別的撰寫

app9_14是一個簡單的範例，稍後將以此範例做延伸來介紹內部類別的撰寫方式：

```

01 // app9_14, 類別的複習
02 class Caaa
03 {
04     int num;
05     void set_num(int n)
06     {
07         num=n;
08         System.out.println("num= "+ num);
09     }
10 }
11 public class app9_14
12 {
13     public static void main(String args[])
14     {
15         Caaa aa=new Caaa();
16         aa.set_num(5);
17     }
18 }

```

```

/* app9_14 OUTPUT---
num= 5
-----*/

```

72

9-51

將類別Caaa改寫為內部類別

下面的程式碼是內部類別的撰寫範例：

01 // app9_15, 內部類別的撰寫

02 public class app9_15

03 {

04 public static void main(String args[])

05 {

06 Caaa aa= new Caaa();

07 aa.set_num(5);

08 }

09 }

10 static class Caaa

11 {

12 int num;

13 void set_num(int n)

14 {

15 num=n;

16 System.out.println("num= "+ num);

17 }

18 }

19 }

外部類別

```
/* app9_15 OUTPUT---
num= 5
-----*/
```

內部類別

```
app9_15$Caaa.class
app9_15.class
```

73

9-52

在外部類別的建構元裡建立內部類別的物件

如果不想把內部類別Caaa宣告成static，但卻又要在main()裡存取到它，利用下列的步驟便可以做到：

- 1) 在外部類別的建構元裡建立內部類別的物件
- 2) 在main()裡建立一個外部類別的物件

如下面的範例：

74

9-53

```

01 // app9_16, 在建構元裡建立內部類別的物件
02 public class app9_16
03 {
04     public app9_16()
05     {
06         Caaa aa= new Caaa();
07         aa.set_num(5);
08     }
09
10     public static void main(String args[])
11     {
12         app9_16 obj=new app9_16();
13     }
14
15     class Caaa          內部類別
16     {
17         int num;
18         void set_num(int n)
19         {
20             num=n;
21             System.out.println("num= "+ num);
22         }
23     }
24 }

```

外部類別的建構元

在外部類別的建構元裡建立內部類別的物件

// 呼叫建構元app9_16()建立外部類別的物件

- 1) 在外部類別的建構元裡建立內部類別的物件
- 2) 在main() 裡建立一個外部類別的物件

```

/* app9_16 OUTPUT---
num= 5
-----*/

```

75

ex9_5_1.java

試將app9_8 的CCircle類別改以內部類別的方式來撰寫(超簡單喔～～)

```

/* output---
area=3.14
area=3.14
area=12.56
-----*/

```

76

ex9_5_2.java

試將app9_8 的CCircle類別改以內部類別的方式來撰寫，但CCircle請務必不能宣告成static.

```
/* output--
area=3.14
area=3.14
area=12.56
-----*/
```

77

9.5.2 匿名內部類別

9-54

用途：適合用在只用到一次就不需再使用此物件的情況

建立匿名內部類別並存取成員的語法如下：

<div style="border: 1px solid red; padding: 5px; display: inline-block;">{</div> new 類別名稱(引數) 注 { 傳回值型態 method名稱(引數1, 引數2,...,引數n) 意 { 是 method 敘述; 小 } 括 } 號	格式9.5.2 建立匿名內部類別，並執行所定義的method
<div style="border: 1px solid red; padding: 5px; display: inline-block;">).method名稱(引數1, 引數2, ..., 引數n);</div>	

78

補充

匿名內部類別-說明

- 物件導向程式設計的特性之一, 就是類別的重複使用性。
 - 在一般情況下, 我們先定義好一個類別, 即可在程式中用它建立多個物件。
 - 但有時候我們所設計的類別, 在程式中只會用到一次, 根本不會重複使用。費時去定義一個類別, 再來建立物件, 就顯得沒有必要, 此時就能利用匿名類別來建立此類別物件, 同時也有使程式碼簡化的功效。
- 建立匿名類別的方式
 - 在需要用到匿名類別物件的敘述上, 直接將類別定義放在原本在物件名稱的地方 (同時也建立了類別物件), 就可以建立一個匿名類別物件了。請注意, 匿名類別必須衍生自任一既有類別或介面。

<https://sites.google.com/site/dychen1127/java-gui-index/anonymous-class>

79

匿名內部類別可用來補足內部類別裡沒有定義到的method :

9-55

```

01 // app9_17, 匿名內部類別
02 public class app9_17
03 {
04     public static void main(String args[])
05     {
06         (new Caaa() .....).set_num(5);
07         new Caaa() // 建立匿名內部類別Caaa的物件
08         {
09             void set_num(int n)
10             {
11                 num=n;
12                 System.out.println("num= "+ num);
13             }
14         }.set_num(5); // 執行匿名內部類別裡所定義的method
15     }
16 }
17
18 static class Caaa // 內部類別Caaa
19 {
20     int num;
21 }
22 }
```

建立匿名內部類別Caaa的物件

補足內部類別Caaa裡沒有定義到的method

app9_17.class
app9_17\$Caaa.class
app9_17\$1.class

/* app9_17 OUTPUT---
num= 5
-----*/

80

「匿名內部類別」也是一種類別, 所以它在編譯的過程中一樣會產生「.class」, 命名規則則是根據數量以「\$」字號加上流水編號。

9-56

習慣上會把匿名內部類別的程式碼“擠”在短短的幾行，
如下面的程式碼所示：

```

01 // app9_18, 匿名內部類別
02 public class app9_18
03 {
04     public static void main(String args[])
05     {
06         (new Caaa(){void set_num(int n){num=n;
07             System.out.println("num= "+ num);}}).set_num(5);
08     }
09     static class Caaa
10     {
11         int num;
12     }
13 }

```

建立匿名內部類別的物件，並呼叫set_num(5)

```

/* app9_18 OUTPUT---
num= 5
-----*/

```

81

ex9_5_3.java

於下面的程式碼中，請建立物件，並以show() method，用以顯示出圓面積。

```

02 public class ex9_5_3
03 {
04     public static void main(String args[])
05     {
06         /* 請在此建立物件，並呼叫
07         show() method 來顯示出圓面積 */
08     }
09     static class CCircle
10     {
11         public double pi=3.14;
12         public double radius;
13     }
14     public CCircle(double r)
15     {
16         radius=r;
17     }
18 }
19 }

```

```

/* output---
area=3.14
-----*/

```

82

ex9_5_4.java

於下面的程式碼中，請撰寫CCircle 的匿名內部類別，並用它呼叫 **show() method**，用以顯示出圓面積。

```
02 public class ex9_5_4
03 {
04     public static void main(String args[])
05     {
06         /* 請在此撰寫CCircle 的匿名內部類別，並用它呼叫
07         show() method 來顯示出圓面積 */
08     }
09     static class CCircle
10     {
11         public double pi=3.14;
12         public double radius;
13
14         public CCircle(double r)
15         {
16             radius=r;
17         }
18     }
19 }
```

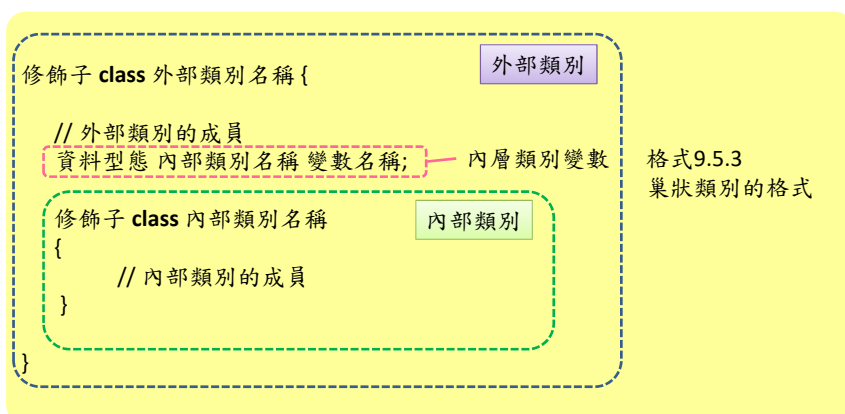
```
/* output---
area=3.14
-----*/
```

83

9.5.3 巢狀類別

9-57

在類別裡含有其它的類別，稱為巢狀類別（Nested Classes）：



84

下面是定義巢狀類別的範例：

9-58

```
class Cbox    // 外部類別
{
    private int length;
    private int width;
    private int height;
    private CColor cr;

    class Ccolor // 內部類別
    {
        String color;
    }
}
```

85

下面示範了如何利用外部類別的成員，存取、呼叫內部類別的成員：

9-59

```
01 // app9_19. 巢狀類別
02 class CBox          // 外部類別
03 {
04     private int length; // CBox類別物件的長
05     private int width;  // CBox類別物件的寬
06     private int height; // CBox類別物件的高
07     private CColor cr;  // CColor類別的物件變數cr，用來表示顏色
08
09     public CBox(int l,int w,int h,String col) // CBox建構元
10     {
11         length=l;
12         width=w;
13         height=h;
14         cr=new CColor(col); // 用new建立CColor物件
15     }
16     class CColor      // 內部類別
17     {
18         private String color;
19
20         public CColor(String clr) // CColor建構元
21         {
22             color=clr;
23         }
24         public void show_color() // 顯示顏色
25         {
26             System.out.println("color="+color);
27         }
28     }
29
30     public void show() // 外部類別CBox的成員函數
31     {
32         System.out.println("length="+length);
33         System.out.println("width="+width);
34         System.out.println("height="+height);
35         cr.show_color();
36         // System.out.println("color="+cr.color);
37     }
38
39 public class app9_19
40 {
41     public static void main(String args[])
42     {
43         CBox box=new CBox(2,3,4,"Blue");
44         box.show();
45     }
46 }
```

```
/* app9_19 OUTPUT---
length=2
width=3
height=4
color=Blue
-----*/
```

86

9-61

巢狀類別在使用上有如下的特點：

1. 當巢狀類別宣告成public時，其內部類別也擁有public的權限。
2. 外部類別的成員可以存取、呼叫內部類別裡的成員，反之亦同，不受private的限制。

87

ex9_5_5.java

假設Data 類別的部份定義如下：

```
class Data
{
    private String name;
    private Test score;
}
```

(a) 其中Test 為類別型態，有english、math 兩個資料成員，皆為int 型態。試將Test類別加入Data 類別，成為巢狀類別。

(b) 請設計一個Test 類別的建構元Test(int eng,int m)，用來設定english 為eng，math為m。

(c) 試在Test 類別裡撰寫一個double avg() method，用來計算並傳回english 與math的平均成績。

(d) 試撰寫一個show() method，用來列印Data 類別裡所有成員的資料，以及平均成績。

(e) 請在main() method 裡加入下面的敘述：

```
Data stu=new Data("Annie",85,92);
```

```
stu.show();
```

使得程式執行的結果如下：

```
學生姓名:Annie
英文成績:85
數學成績:92
平均成績:88.5
```

```
/* output-----
學生姓名:Annie
英文成績:85
數學成績:92
平均成績:88.5
-----*/
```



9.5 回家作業

內部類別與巢狀類別

hw9_5_1.java

試將習題 ex9_5_5.java 的 avg() method 從內部類別 Test 中移出，變成外部類別 Data 裡的成員函數。

```
/* output-----
學生姓名:Annie
英文成績:85
數學成績:92
平均成績:88.5
-----*/
```

89

9.5 回家作業

內部類別與巢狀類別

hw9_5_2.java

假設 Namecard 類別的部份定義如下：

```
class Namecard
{
    private String name;
    private String address;
    private Phone data;
}
```

```
/* output-----
好友姓名:Andy
聯絡地址:123City
公司電話:2345-6789
手機號碼:0911-336600
-----*/
```

- 其中 Phone 為類別型態，有 company、cell 兩個資料成員，皆為 String 型態。試將 Phone 類別加入 Namecard 類別，成為巢狀類別。
- 請設計一個 Phone 類別的建構元 Phone(String s1, String s2)，用來設定 company 為 s1、cell 為 s2。
- 試撰寫一個 show() method，用來列印 Namecard 類別裡所有成員的資料。
- 請在 main() method 裡加入下面的敘述：
 Namecard first=new Namecard("Andy","123City","2345-6789","0911-336600");
 first.show();

90