第十章
類別的繼承

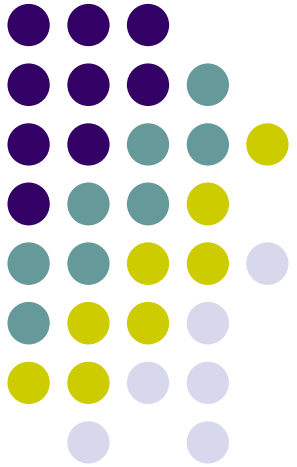
學習繼承的基本概念

瞭解子類別與父類別之間的關係

認識method的改寫

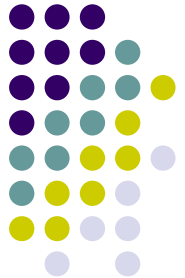
區分super() 與this() 的用法

認識Object類別



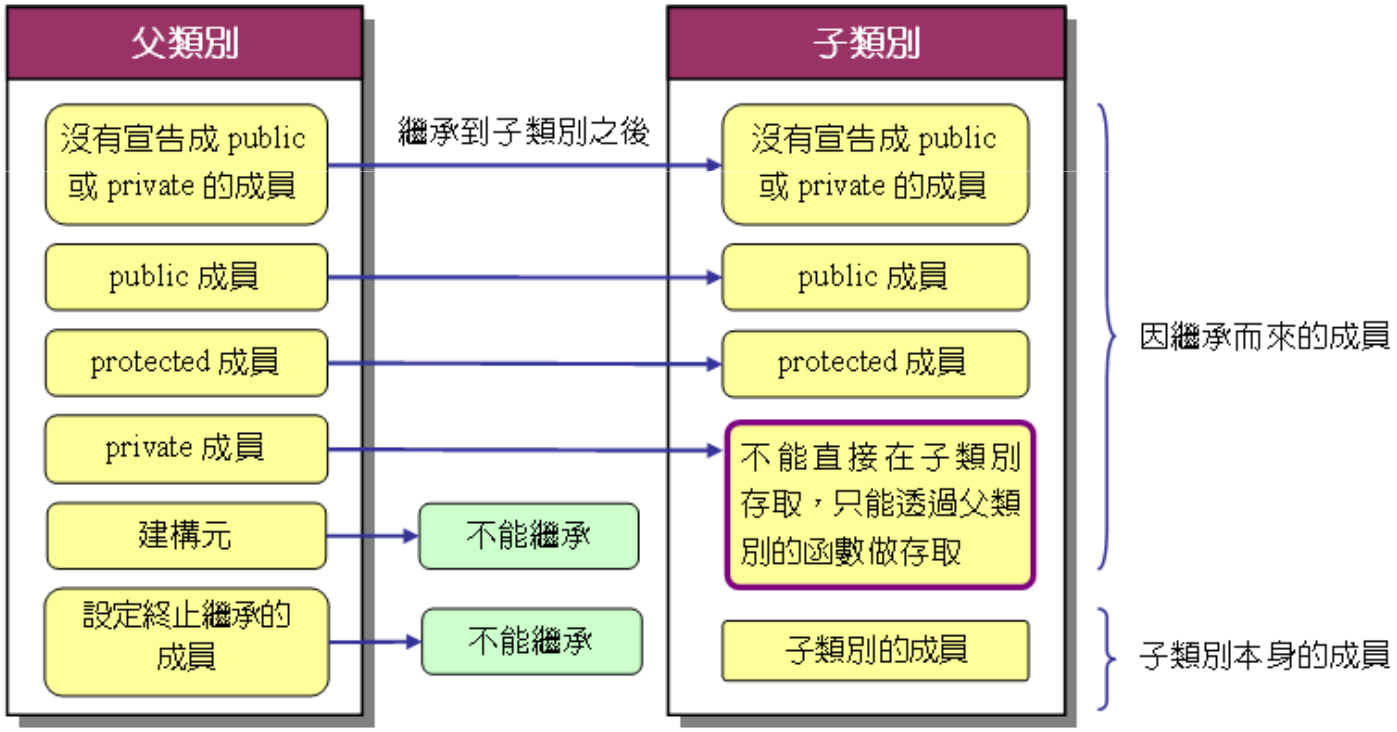
認識繼承 (1/2)

- 根據**既有類別**爲基礎，進而衍生出另一類別，這種概念稱爲類別的繼承
  - **既有類別**稱爲**父類別**（super class）或**基底類別**（basis class）
  - **衍生出的類別**稱爲**子類別**（sub class）或**衍生類別**（derived class）
- 每一個類別只能有一個父類別，這是所謂的**單一繼承**（single inheritance）
- Java的**介面**（interface）可以實現**多重繼承**的概念



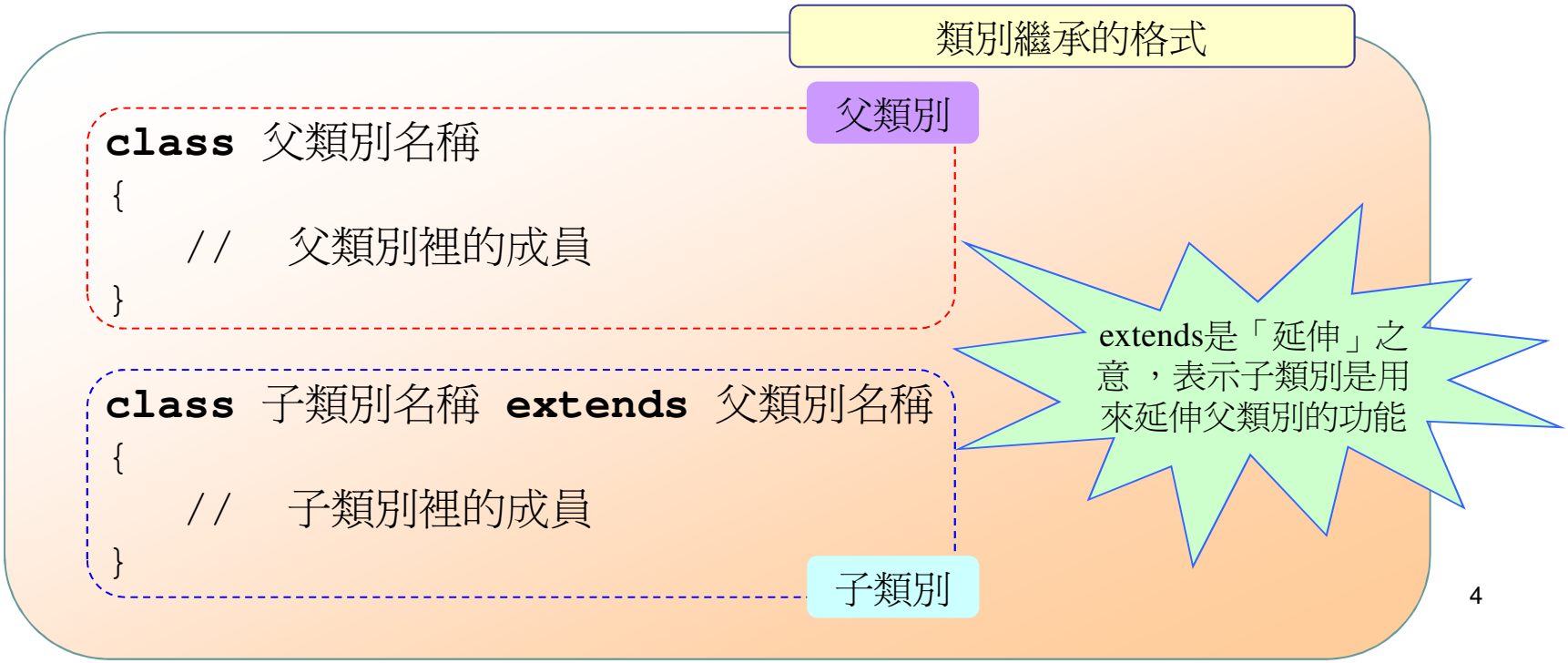
認識繼承 (2/2)

- 類別成員的繼承關係可用下圖來表示：



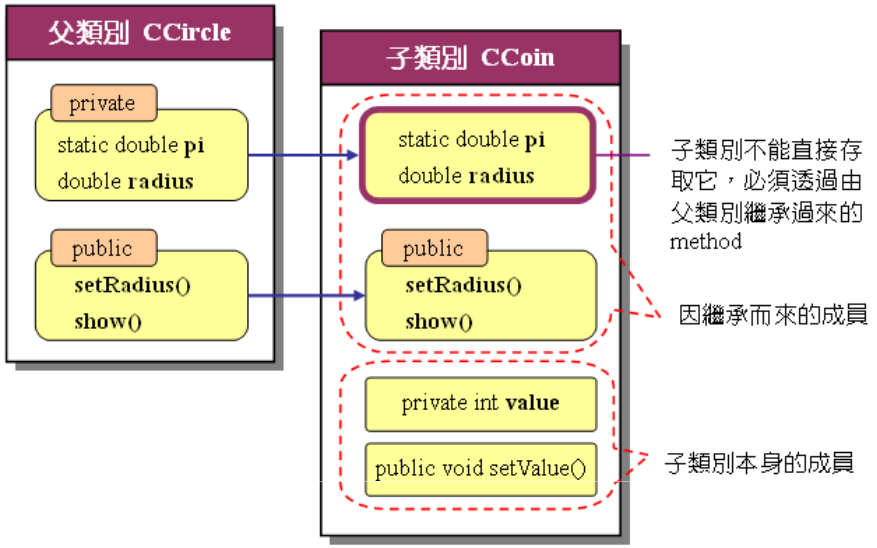
類別的繼承格式

- 類別的繼承
  - 以**extends**關鍵字，將父類別繼承給子類別
  - 類別繼承的格式：



簡單的繼承範例 (1/2)

- 繼承的使用範例：




```
01  // app10_1, 簡單的繼承範例
02  class CCircle                  // 父類別 CCircle
03  {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle()            // CCircle()建構元
08     {
09        System.out.println("CCircle() constructor called ");
10     }
11     public void setRadius(double r)
12     {
13        radius=r;
14        System.out.println("radius="+radius);
15     }
16     public void show()
17     {
18        System.out.println("area="+pi*radius*radius);
19     }
20  }
```

先呼叫父類別的建構元CCircle()，再呼叫子類別的建構元CCoin()後所得的結果

呼叫由父類別繼承而來的method所得的結果

```
/* app10_1 OUTPUT----------
CCircle() constructor called
CCoin() constructor called
radius=2.0
area=12.56
value=5
----------------------*/
```

呼叫子類別的method所得的結果

簡單的繼承範例 (2/2)

```
21  class CCoin extends CCircle          // 子類別 CCoin,繼承自 CCircle 類別
22  {
23     private int value;                // 子類別的資料成員
24
25     public CCoin()                    // 子類別的建構元
26     {
27        System.out.println("CCoin() constructor called ");
28     }
29     public void setValue(int t)       // 子類別的 setValue() method
30     {
31        value=t;
32        System.out.println("value="+value);
33     }
34  }
35  public class app10_1
36  {
37     public static void main(String args[])
38     {
39        CCoin coin=new CCoin();  // 建立 coin 物件
40        coin.setRadius(2.0);     // 呼叫由父類別繼承而來的 setRadius()
41        coin.show();             // 呼叫由父類別繼承而來的 show()
42        coin.setValue(5);        // 呼叫子類別的 setValue()
43     }
44  }
```

本例中學到的觀念：

- 透過extends關鍵字，可將父類別的成員繼承給子類別
- 執行子類別的建構元前，會先呼叫父類別的建構元，目的是要幫助繼承自父類別的成員初始化

先呼叫父類別的建構元CCircle()，再呼叫子類別的建構元CCoin()後所得的結果

```
/* app10_1 OUTPUT----------
CCircle() constructor called
CCoin() constructor called
radius=2.0
area=12.56
value=5
-------------------------*/
```

呼叫由父類別繼承而來的method所得的結果

呼叫子類別的method所得的結果

建構元的呼叫 (1/2)

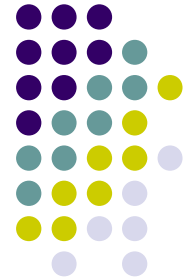
- 下面的範例是透過super() 呼叫父類別中特定的建構元：

```
01  // app10_2, 呼叫父類別中特定的建構元
02  class CCircle    // 定義父類別 CCircle
03  {
04     private static double pi=3.14;
05     private double radius;
06
07     public CCircle()  // 父類別裡沒有引數的建構元
08     {
09        System.out.println("CCircle() constructor called");
10     }
11     public CCircle(double r)    // 父類別裡有一個引數的建構元
12     {
13        System.out.println("CCircle(double r) constructor called");
14        radius=r;
15     }
16     public void show()
17     {
18        System.out.println("area="+pi*radius*radius);
19     }
20  }
```

```
/* app10_2 OUTPUT----------
CCircle() constructor called
CCoin() constructor called
CCircle(double r) constructor called
CCoin(double r, int v) constructor called
area=0.0
area=19.625
-----------------------------------------*/
```

執行第39行所得的結果

執行第40行所得的結果



建構元的呼叫 (2/2)

```
21  class CCoin extends CCircle  // 定義子類別 CCoin，繼承自 CCircle 類別
22  {
23     private int value;
24     public CCoin()                        // 子類別裡沒有引數的建構元
25     {
26        System.out.println("CCoin() constructor called");
27     }
28     public CCoin(double r, int v)   // 子類別裡有兩個引數的建構元
29     {
30        super(r);  // 呼叫父類別裡，有引數的建構元，即第 11 行所定義的建構元
31        value=v;
32        System.out.println("CCoin(double r, int v) constructor called");
33     }
34  }
35  public class app10_2
36  {
37     public static void main(String args[])
38     {
39        CCoin coin1=new CCoin();      // 建立物件，並呼叫第 24 行的建構元
40        CCoin coin2=new CCoin(2.5,10);// 建立物件，並呼叫第 28 行的建構元
41        coin1.show();
42        coin2.show();
43     }
44  }
```

呼叫父類別建構元的 super() 必須寫在子類別建構元裡的第一個敘述

即使省略此敘述，父類別中沒有引數的建構元仍會被呼叫

執行第39行所得的結果

```
/* app10_2 OUTPUT------------------------
CCircle() constructor called
CCoin() constructor called
CCircle(double r) constructor called
CCoin(double r, int v) constructor called
area=0.0
area=19.625
-------------------------------------------*/
```

執行第40行所得的結果

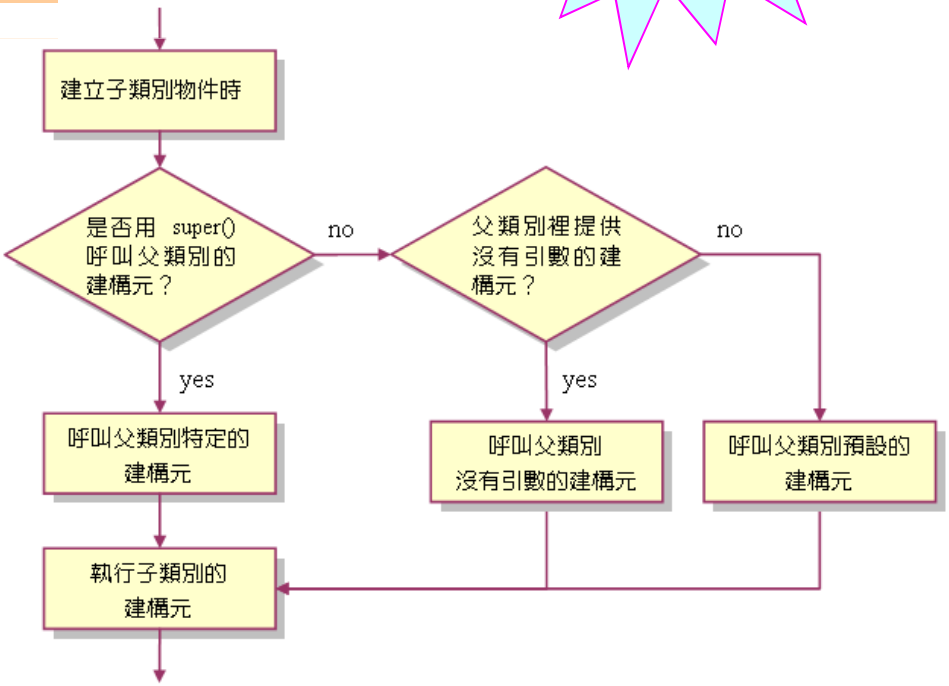
super() 可以多載

錯誤的使用建構元 (1/2)

- 下面是錯誤的建構元使用範例：

```
01  // app10_3, 建構元錯誤的範例
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)     // 有引數的建構元
08     {
09        radius=r;
10     }
11     public void setRadius(double r)
12     {
13        radius=r;
14        System.out.println("radius="+radius);
15     }
16  }
17
```

編譯時的錯誤訊息



錯誤的使用建構元 (2/2)

```
18  class CCoin extends CCircle // 定義 CCoin 類別，繼承自 CCircle 類別
19  {
20     private int value;
21
22     public CCoin(double r, int v)    // CCoin()有兩個引數的建構元
23     {
24        setRadius(r);          // 透過 setRadius() method 來設定 radius 成員
25        value=v;               // 設定 value 成員
26     }
27  }
28  public class app10_3
29  {
30     public static void main(String args[])
31     {
32        CCoin coin1=new CCoin(2.5,10); // 建立物件，並呼叫有兩個引數的建構元
33     }
34  }
```

編譯時的錯誤訊息

更正使用建構元的錯誤

- 更正app10_3的錯誤：

```
01  // app10_4, 修正 app10_3 的錯誤
02  class CCircle                    // 定義類別 CCircle
03  {
04     private double pi=3.14;
05     private double radius;
06
07     public CCircle()              // 沒有引數的建構元
08     {
09     }
10     public CCircle(double r)      // 有一個引數的建構元
11     {
12        radius=r;
13     }
14     public void setRadius(double r)
15     {
16        radius=r;
17        System.out.println("radius="+radius);
18     }
19  }
20  // 將 app10_3 中，類別 CCoin 的定義置於此處
21  // 將 app10_3 中，類別 app10_3 的定義置於此處
```

```
/* app10_4 OUTPUT---
radius=2.5
--------------------*/
```

this() 與 super() 的比較

- this() 是在**同一類別**內呼叫其它的建構元
- super() 是從**子類別**的建構元呼叫其**父類別**的建構元
- this()與super() 相似之處：
  1. this() 與super() 均**可多載**
  2. this() 與super() **均必須撰寫在建構元內的第一行**，因此this() 與super() 無法同時存在同一個建構元內

保護成員的概念 (1/2)

- 在子類別內直接存取private的資料成員，編譯時將出現錯誤
  - 例如把app10_2中的28~33行改寫成如下的敘述：

```
28      public CCoin(double r, int v)
29      {
30         radius=r; // 錯誤，radius 為private 成員，無法在 CCircle 類別外部存取
31         value=v;
32         System.out.println("CCoin(double r, int v) constructor called");
33      }
```

編譯時將出現下列的錯誤訊息：

```
radius has private access in CCircle
    radius=r;
```

是因爲radius在CCircle類別內宣告爲private

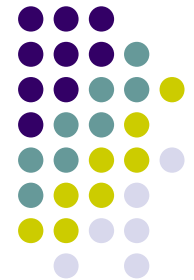
保護成員的概念 (2/2)

- 如何能使子類別存取到父類別的資料成員？
  - 做法是把資料成員宣告成**protected**（保護成員）
- 若在CCircle類別裡把radius與pi這兩個成員宣告成：

```
protected static double pi=3.14;
protected double radius;
```

  - radius與pi不僅可以在CCircle類別裡直接取用，
  - 同時也可以在繼承CCircle而來的CCoin類別裡存取

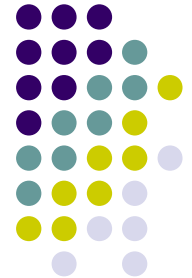
保護成員的範例



```
01  // app10_5, protected 成員的使用
02  class CCircle
03  {
04     protected static double pi=3.14;   // 將 pi 宣告成 protected
05     protected double radius;           // 將 radius 宣告成 protected
06
07     public void show()
08     {
09        System.out.println("area="+pi*radius*radius);
10     }
11  }
12  class CCoin extends CCircle   // 定義 CCoin 類別，繼承自 CCircle 類別
13  {
14     private int value;
15
16     public CCoin(double r, int v)
17     {
18        radius=r;        // 在子類別裡可直接取用父類別裡的 protected 成員
19        value=v;
20        System.out.println("radius="+radius+", value="+value);
21     }
22  }
23  public class app10_5
24  {
25     public static void main(String args[])
26     {
27        CCoin coin=new CCoin(2.5,10);
28        coin.show();
29     }
30  }
```

此範例是app10_2的精簡版

```
app10_5 OUTPUT-------
dius=2.5, value=10
ea=19.625
----------------------*/
```



改寫父類別的method (1/2)

- 下面是改寫父類別之method的範例：

```
01  // app10_6, method 的「改寫」範例
02  class CCircle                    // 父類別 CCircle
03  {
04     protected static double pi=3.14;
05     protected double radius;
06
07     public CCircle(double r)
08     {
09        radius=r;
10     }
11     public void show()            // 父類別裡的 show() method
12     {
13        System.out.println("radius="+radius);
14     }
15  }
16
```

```
/* app10_6 OUTPUT------

radius=2.0, value=5
----------------------*/
```

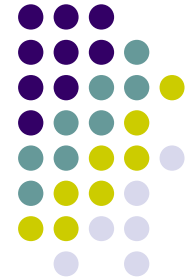
改寫父類別的method (2/2)

```
17  class CCoin extends CCircle   // 子類別 CCoin
18  {
19     private int value;
20
21     public CCoin(double r,int v)
22     {
23        super(r);
24        value=v;
25     }
26     public void show()          // 子類別裡的 show() method
27     {
28        System.out.println("radius="+radius+", value="+value);
29     }
30  }
31
32  public class app10_6
33  {
34     public static void main(String args[])
35     {
36        CCoin coin=new CCoin(2.0,5);
37        coin.show();             // 呼叫 show() method
38     }
39  }
```

```
/* app10_6 OUTPUT------
radius=2.0, value=5
----------------------*/
```

「改寫」與「多載」的比較

- 「**多載**」 overloading
  - 在相同類別內，定義**名稱相同**，但**引數個數**或**型態不同**的method，如此便可依據引數的個數或型態，呼叫相對應的method
- 「**改寫**」 overriding
  - 在子類別當中**定義名稱**、**引數個數**與**傳回值的型態**均與父類別**相同**的method，用以改寫父類別裡method的功用



父類別變數存取子類別成員 (1/4)

- app10_6的36行 與37行：

```
36   CCoin coin=new CCoin(2.0,5);  // 宣告子類別變數 coin，並將它指向新建的物件
37   coin.show();                  // 利用子類別變數 coin 呼叫 show() method
```

  - 將上面兩行程式碼改寫成如下的敘述：

```
36   CCircle cir=new CCoin(2.0,5);  // 宣告父類別變數 cir，並將它指向新建的物件
37   cir.show();                    // 利用父類別變數 cir 呼叫 show() method
```

cir.show()是透過父類別的變數cir來呼叫show() method

是定義於父類別裡的show()，還是子類別裡的show() method會被呼叫？

父類別變數存取子類別成員 (2/4)

- 下面的範例是透過父類別變數cir呼叫show() method：

```
01  // app10_7, 透過父類別變數 cir 呼叫 show() method
02  class CCircle                    // 父類別 CCircle
03  {
04     protected static double pi=3.14;
05     protected double radius;
06
07     public CCircle(double r)                          /* app10_7 OUTPUT------
08     {                                                 radius=2.0, value=5
09        radius=r;                                      ----------------------*/
10     }
11     public void show()            // 父類別裡的 show() method
12     {
13        System.out.println("radius="+radius);
14     }
15  }
16  class CCoin extends CCircle   // 子類別 CCircle
17  {
18     private int value;
19
```

父類別變數存取子類別成員 (3/4)

```
20      public CCoin(double r,int v)
21      {
22         super(r);
23         value=v;
24      }
25      public void show()       // 子類別裡的 show() method
26      {
27         System.out.println("radius="+radius+", value="+value);
28      }
29      public void showValue()  // showValue() method,此函數只存在於子類別
30      {
31         System.out.println("value="+value);
32      }
33   }
34   public class app10_7
35   {
36      public static void main(String args[])
37      {
38         CCircle cir=new CCoin(2.0,5); // 宣告父類別變數 cir，並將它指向物件
39         cir.show();                   // 利用父類別變數 cir 呼叫 show()
40         // cir.showValue();
41      }
42   }
```

```
/* app10_7 OUTPUT------
radius=2.0, value=5
-----------------------*/
```

父類別變數存取子類別成員 (4/4)

- 子類別的變數**不能**把它**指派**給**父類別**的物件
- 以app10_7爲例，不能撰寫如下的程式碼：

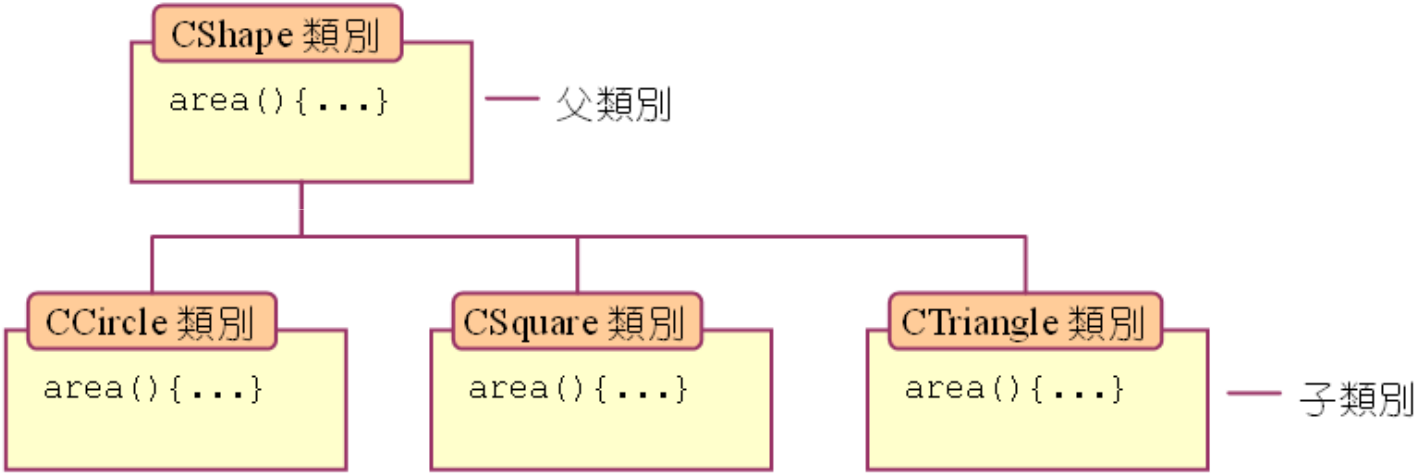
```
CCoin coin=new CCircle();          // 錯誤，子類別的變數不能指派給父類別的物件
```

  - 如果把40行的註解符號拿掉，編譯時將出現錯誤訊息：

```
cannot find symbol
symbol  : method showValue()
location: class CCircle
     cir.showValue();
        ^
```

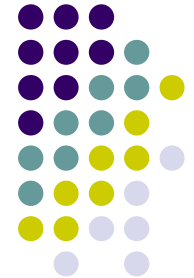
父類別陣列變數存取子類別成員

- 以父類別的陣列變數呼叫子類別的area() method：




```
CShape shp[]=new CShape[5];          // 宣告 CShape 的陣列變數
shp[0]=new CCircle(12);              // 建立圓形物件(一)
shp[1]=new CCircle(21);              // 建立圓形物件(二)
shp[2]=new CSquare(15);              // 建立正方形物件
shp[3]=new CTriangle(12,7);          // 建立三角形物件(一)
shp[4]=new CTriangle(3,18);          // 建立三角形物件(二)
CShape.largest(shp);                 // 比較物件面積的大小，並傳回最大者
```

用super存取父類別



```
01  // app10_8, 透過 super 關鍵字來存取父類別的變數
02  class Caaa
03  {
04     protected int num;                    // 父類別的資料成員 num
05
06     public void show()
07     {
08        System.out.println("Caaa_num="+num);
09     }
10  }
11  class Cbbb extends Caaa
12  {
13     int num=10;                           // 子類別的資料成員 num
14
15     public void show()
16     {
17        super.num=20;                      // 設定父類別的資料成員 num 為 20
18        System.out.println("Cbbb_num="+num);
19        super.show();                      // 呼叫父類別的 show() method
20     }
21  }
22
23  public class app10_8
24  {
25     public static void main(String args[])
26     {
27        Cbbb b=new Cbbb();
28        b.show();
29     }
30  }
```

此範例說明super的用法

```
/* app10_8 OUTPUT---

Cbbb_num=10
Caaa_num=20
-------------------*/
```

super後面可加上資料成員或method的名稱：

```
super.資料成員名稱  // 存取父類別的資料成員
super.method名稱    // 存取父類別的method
```

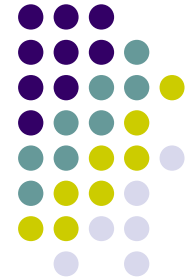
this關鍵字

- 下面的範例是利用this來呼叫實例變數：

```
01  // app10_9, 用 this 來呼叫實例變數
02  class Caaa
03  {
04     public int num=10;    // num 是實例變數
05
06     public void show()
07     {
08        int num=5;          // num 是區域變數，其有效範圍僅限於在 show()內
09        System.out.println("this.num="+this.num);  // 印出實例變數
10        System.out.println("num="+num);            // 印出區域變數
11     }
12  }
13  public class app10_9
14  {
15     public static void main(String args[])
16     {
17        Caaa a=new Caaa();
18        a.show();
19     }
20  }
```

```
/* app10_9 OUTPUT---
this.num=10
num=5
--------------------*/
```

終止繼承 (1/2)



- 設定終止繼承可利用final關鍵字，如下面的範例：

```
01  // app10_10, 設定終止繼承
02  class Caaa
03  {
04     public final void show()    // 父類別的 show()已被設為終止繼承
05     {
06        System.out.println("show() method in class Caaa called");
07     }
08  }
09  class Cbbb extends Caaa
10  {
11     public void show()          // 錯誤，改寫父類別的 show() method
12     {
13        System.out.println("show() method in class Cbbb called");
14     }
15  }
16  public class app10_10
17  {
18     public static void main(String args[])
19     {
20        Cbbb b=new Cbbb();
21        b.show();
22     }
23  }
```

終止繼承 (2/2)

- app10_10編譯時會產生如下的錯誤訊息：

```
show() in Cbbb cannot override show() in Caaa; overridden method is final
```

  - 此錯誤訊息是說Caaa類別裡的show() method已宣告成final，無法再讓子類別改寫

- final的另一個功用是把它加在資料成員前面，就變成一個常數（constant），如：

```
protected static final double PI=3.14;   // 設定 PI 值不能再被修改
```

- 不希望某個類別被其它的類別繼承時，可以在宣告時加上final修飾子，如：

```
final class CCircle                  // 設定 CCircle 類別不能被其它類別繼承
```

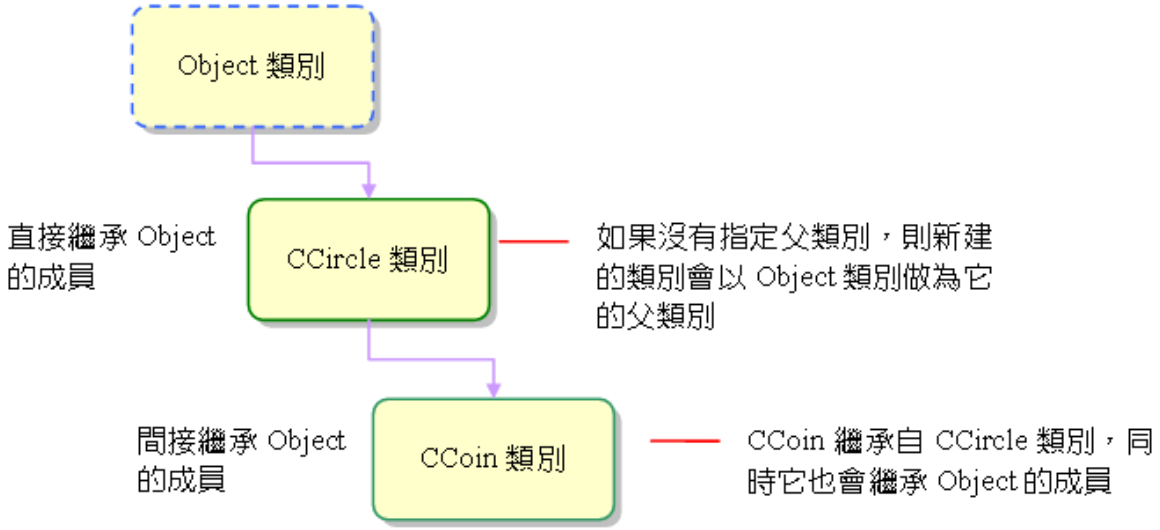
Object類別

- 若沒指定父類別，則會自動設定Object類別爲它的父類別

```
class CCircle
{
    ...
}
```

若是沒有指定父類別時，便會以 java.lang.Object 類別做為它的父類別，而自己變成它的子類別。

- 所有的類別均直接或間接繼承Object類別：



Object類別裡的method

- 下表列舉Object類別裡三個常用的method：

表 10.6.1 Object 類別裡常用的 method

Method 名稱	功能說明
`Class getClass()`	取得呼叫 **getClass()** 的物件所屬之類別
`Boolean equals(Object obj)`	兩個類別變數所指向的是否為同一個物件
`String toString()`	將呼叫 **toString()** 的物件轉成字串

- 想知道某個物件obj是屬於哪個類別時，可用：

```
obj.getClass()                    // 取得變數 obj 所指向之物件所屬的類別
```

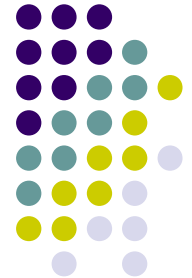
的語法查詢

getClass() method的使用

- 下面的程式是getClass() 簡單的使用範例：

```
01  // app10_11, 利用 getClass()取得呼叫物件所屬的類別
02  class Caaa                                  // 定義 Caaa 類別
03  {
04     private int num;
05
06     public Caaa(int n)
07     {
08        num=n;
09     }
10  }
11  public class app10_11
12  {
13     public static void main(String args[])
14     {
15        Caaa a=new Caaa(5);
16        Class ca=a.getClass();          // 用變數 a 呼叫 getClass()
17        System.out.println("Class of obj = "+ca);
18     }
19  }
```

```
/* app10_11 OUTPUT--------
Class of obj = class Caaa
-------------------------*/
```

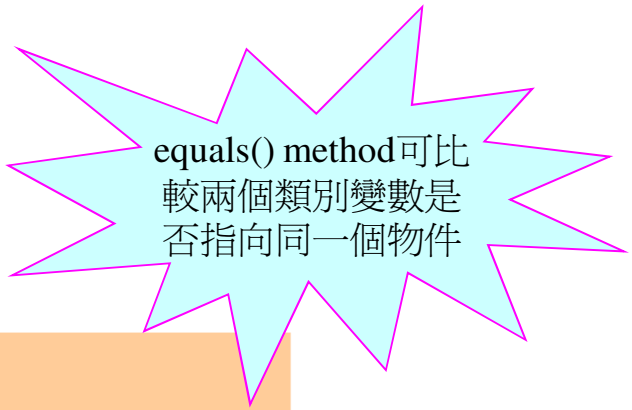


equals() method的使用範例

```
01  // app10_12, 利用 equals()來判別兩個類別變數所指向的是否為同一個物件
02  class Caaa                    // 定義 Caaa 類別
03  {
04     private int num;
05
06     public Caaa(int n)
07     {
08        num=n;
09     }
10  }
11  public class app10_12
12  {
13     public static void main(String args[])
14     {
15        Caaa a=new Caaa(5);
16        Caaa b=new Caaa(5);
17        Caaa c=a;              // 宣告類別變數 c，並讓它指向變數 a 所指向的物件
18        boolean br1=a.equals(b);      // 測試 a 與 b 是否指向同一物件
19        boolean br2=a.equals(c);      // 測試 a 與 c 是否指向同一物件
20        System.out.println("a.equals(b)="+br1);
21        System.out.println("a.equals(c)="+br2);
22     }
23  }
```

```
/* app10_12 OUTPUT---

a.equals(b)=false
a.equals(c)=true
---------------------*/
```



toString() method的使用 (1/3)

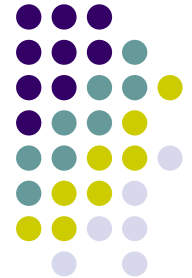
- 下面的範例是toString()的使用：

```
01  // app10_13, Object 類別裡的 toString() method
02  class Caaa
03  {
04     private int num;
05
06     public Caaa(int n)
07     {
08        num=n;
09     }
10  }
11  public class app10_13
12  {
13     public static void main(String args[])
14     {
15        Caaa a=new Caaa(2);
16        System.out.println(a.toString()); // 印出物件 a 的內容
17     }
18  }
```

```
/* app10_13 OUTPUT---
Caaa@757aef
--------------------*/
```

toString() 是將物件的內容轉換成字串，如：

```
a.toString();     // 傳回代表此物件a的字串
```



toString() method的使用 (2/3)

```
01  // app10_14, 改寫 Object 類別裡的 toString() method
02  class Caaa
03  {
04     private int num;
05
06     public Caaa(int n)
07     {
08        num=n;
09     }
10     public String toString()    // 改寫 toString() method
11     {
12        String str="toString() called, num="+num;
13        return str;
14     }
15  }
16  public class app10_14
17  {
18     public static void main(String args[])
19     {
20        Caaa a=new Caaa(2);
21        System.out.println(a.toString());   // 印出物件 a 的內容
22     }
23  }
```

此範例改寫
toString() method

```
/* app10_14 OUTPUT-------
toString() called, num=2
------------------------*/
```

toString() method的使用 (3/3)

- 改寫toString() method的好處是在於使用上的方便
  - 也可以直接把變數a當成println() 的引數印出，如下面的敘述：

```
System.out.println(a);                    // 印出物件 a 的內容
```

    此時會先呼叫toString() method，然後再把結果當成println() 的引數印出

- 各類別中的轉換函數toString() method，皆是繼承自Object類別，而它們也都是改寫Object類別裡的toString() method