

# 第十章

## 類別的繼承

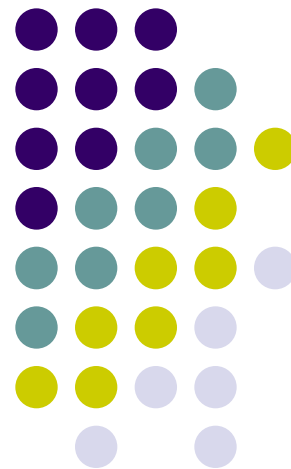
學習繼承的基本概念

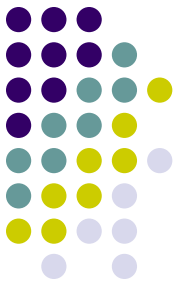
瞭解子類別與父類別之間的關係

認識函數的改寫

區分`super()` 與`this()` 的用法

認識Object類別





# 認識繼承 (1/2)

- 繼承：根據既有類別衍生出另一類別

父類別 ( super class ) 或  
基底類別 ( basis class )

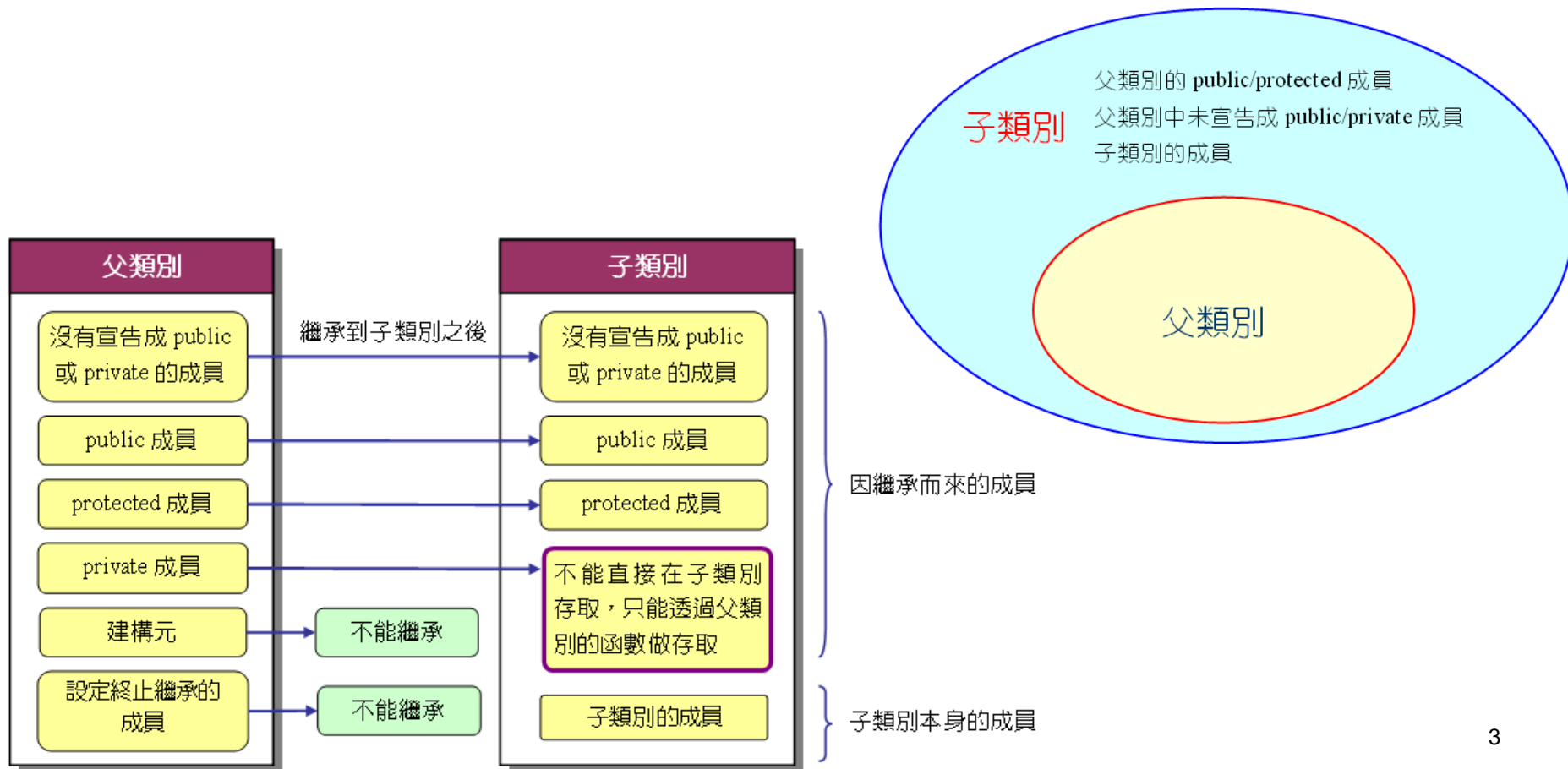
子類別 ( sub class ) 或  
衍生類別 ( derived class )

- 每一個類別只能有一個父類別 ( 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：

## 類別繼承的語法

```
class 父類別名稱{  
    // 父類別裡的成員  
}
```

} 父類別

```
class 子類別名稱 extends 父類別名稱{  
    // 子類別裡的成員  
}
```

} 子類別

extends是「延伸」之意，  
表示子類別是用來延  
伸父類別的功能

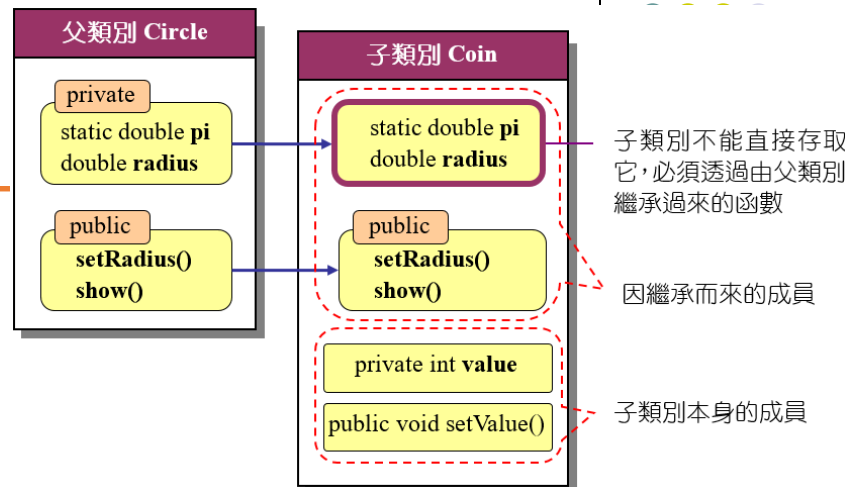
# 簡單的繼承範例 (1/2)

## 10.1 繼承的基本概念



- 繼承的使用範例：

```
01 // Ch10_1, 簡單的繼承範例
02 class Circle{           // 父類別 Circle
03     private static double pi=3.14;
04     private double radius;
05
06     public Circle(){      // Circle()建構子
07         System.out.println("Circle() constructor called ");
08     }
09     public void setRadius(double r){
10         radius=r;
11         System.out.println("radius="+radius);
12     }
13     public void show(){
14         System.out.printf("area=%6.2f\n",pi*radius*radius);
15     }
16 }
```



執行結果：

```
Circle() constructor called
Coin() constructor called
radius=2.0
area= 12.56
value=5
```

先呼叫父類別的建構子，再呼叫子類別的建構子  
呼叫由父類別繼承而來的函數所得的結果  
呼叫子類別的函數所得的結果

接續下一頁



# 簡單的繼承範例 (2/2)

```

17 class Coin extends Circle{           // 子類別 Coin，繼承自 Circle 類別
18     private int value;                 // 子類別的資料成員
19
20     public Coin(){                     // 子類別的建構子
21         System.out.println("Coin() constructor called ");
22     }
23     public void setValue(int t){ // 子類別的 setValue() 函數
24         value=t;
25         System.out.println("value="+value);
26     }
27 }
28 public class Ch10_1{
29     public static void main(String[] args){
30         Coin coin=new Coin(); // 建立 coin 物件
31         coin.setRadius(2.0); // 呼叫由父類別繼承而來的 setRadius()
32         coin.show();          // 呼叫由父類別繼承而來的 show()
33         coin.setValue(5);      // 呼叫子類別的 setValue()
34     }
35 }

```

本例中學到的觀念：

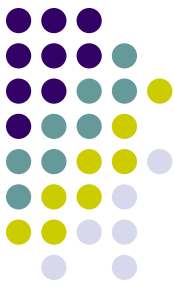
- 透過 **extends** 關鍵字，可將父類別的成員繼承給子類別
- 執行子類別的建構子前，會先呼叫父類別的建構子，目的是要幫助繼承自父類別的成員初始化

執行結果：

```

Circle() constructor called
Coin() constructor called } 先呼叫父類別的建構子，再呼叫子類別的建構子
radius=2.0
area= 12.56 } 呼叫由父類別繼承而來的函數所得的結果
value=5      — 呼叫子類別的函數所得的結果

```



# 建構子的呼叫 (1/3)

- 下面的範例是透過super() 呼叫父類別中特定的建構子：

```
01 // Ch10_2, 呼叫父類別中特定的建構子
02 class Circle{                                //
03     private static double pi=3.14;
04     private double radius;
05
06     public Circle(){                            //
07         System.out.println("Circle() constructor called");
08     }
09     public Circle(double r) {                  // 父類別裡有一個引數的建構子
10         System.out.println("Circle(double r) constructor called");
11         radius=r;
12     }
13     public void show(){
14         System.out.printf("area=%6.2f\n",pi*radius*radius);
15     }
16 }
```

執行結果：

Circle() constructor called  
Coin() constructor called } 執行第 30 行所得的

Circle(double r) constructor called  
Coin(double r, int v) constructor called } 執行第 31 行所得的  
area= 0.0  
area=19.63



# 建構子的呼叫 (2/3)

```

17 class Coin extends Circle{ // 定義子類別 Coin，繼承自 Circle 類別
18     private int value;
19     public Coin(){           // 子類別裡沒有引數的建構子
20         System.out.println("Coin() constructor called");
21     }
22     public Coin(double r, int v){ // 子類別裡有兩個引數的建構子
23         super(r);             // 呼叫父類別裡有引數的建構子，即第 9 行所定義的建構子
24         value=v;
25         System.out.println("Coin(double r, int v) constructor called");
26     }
27 }
28 public class Ch10_2{
29     public static void main(String[] args) {
30         Coin coin1=new Coin();           // 建立物件，並呼叫第 19 行的建構子
31         Coin coin2=new Coin(2.5,10);     // 建立物件，並呼叫第 22 行的建構子
32         coin1.show();
33         coin2.show();
34     }
35 }

```

呼叫父類別建構子的 `super()` 必須寫在子類別建構子裡的第一個敘述

即使省略此敘述，父類別中沒有引數的建構子仍會被呼叫

`super()` 可以多載

執行結果：

```

Circle() constructor called    } 執行第 30 行所得的
Coin() constructor called
Circle(double r) constructor called } 執行第 31 行所得的
Coin(double r, int v) constructor called
area= 0.0
area=19.63

```





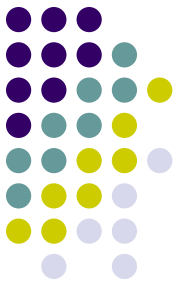
# 建構子的呼叫 (3/3)

這裡有幾點要提醒您：

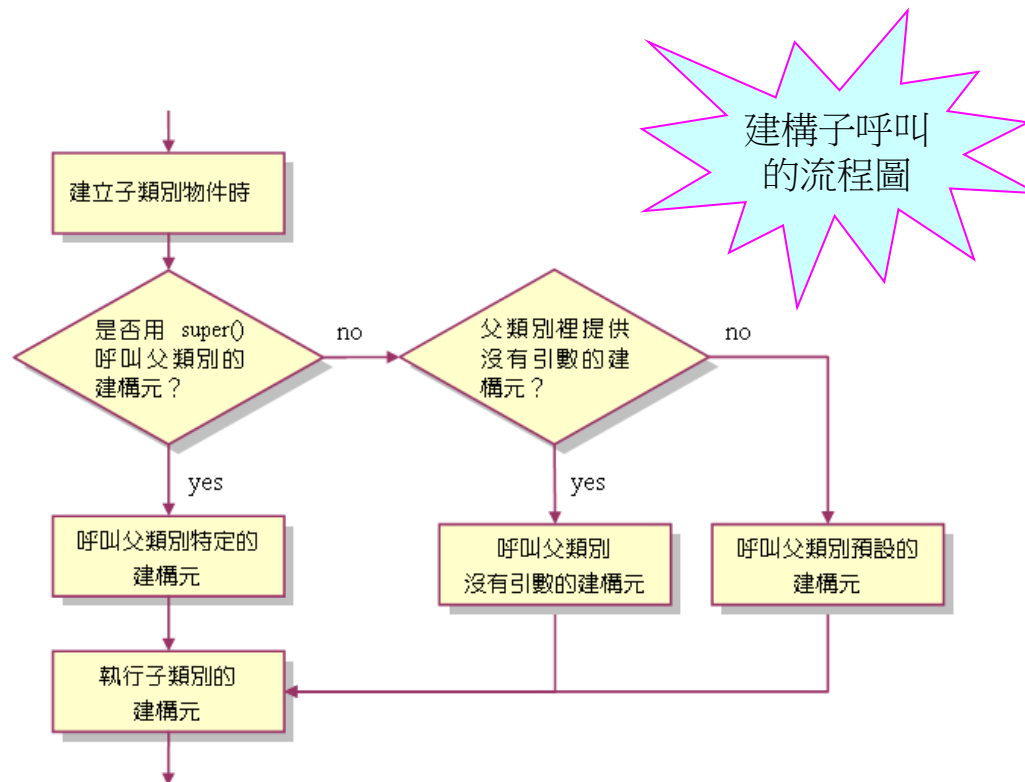
1. 如果省略第 23 行的 `super(r)` 敘述，則父類別中沒有引數的建構子還是會被呼叫，讀者可自行試試。
2. 呼叫父類別建構子的 `super()` 必須寫在子類別建構子裡的第一個敘述，不能置於它處，否則編譯時將出現錯誤訊息。
3. `super()` 會根據引數的數量及型別，執行相對應之父類別的建構子。

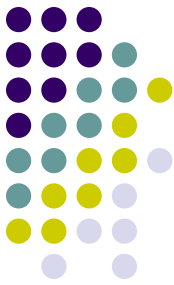
# 錯誤的使用建構子 (1/2)

## 10.1 繼承的基本概念



- 下面是建構子呼叫的流程：





# 錯誤的使用建構子 (2/2)

```
01 // Ch10_3, 建構子錯誤的範例
02 class Circle{                      // 定義類別 Circle
03     private static double pi=3.14;
04     private double radius;
05
06     public Circle(double r){        // 有引數的建構子
07         radius=r;
08     }
09     public void setRadius(double r){
10         radius=r;
11         System.out.println("radius="+radius);
12     }
13 }
14 class Coin extends Circle{
15     private int value;
16
17     public Coin(double r, int v) {    // Coin()有兩個引數的建構子
18         setRadius(r);                // 透過 setRadius() 函數來設定 radius 成員
19         value=v;                     // 設定 value 成員
20     }
21 }
22 public class Ch10_3{
23     public static void main(String[] args){
24         Coin coin1=new Coin(2.5,10); // 建立物件，並呼叫有兩個引數的建構子
25     }
26 }
```

Implicit super constructor Circle() is undefined. Must explicitly invoke another constructor

出現的錯誤訊息



# 更正使用建構子的錯誤

- 更正Ch10\_3的錯誤：

```
01 // Ch10_4, 修正 Ch10_3 的錯誤
02 class Circle{                                // 定義類別 Circle
03     private double pi=3.14;
04     private double radius;
05
06     public Circle(){                            // 沒有引數的建構子
07     }
08     public Circle(double r) {                  // 有一個引數的建構子
09         radius=r;
10     }
11     public void setRadius(double r){
12         radius=r;
13         System.out.println("radius="+radius);
14     }
15 }
16 // 將 Ch10_3 中，類別 Coin 的定義置於此處
17 // 將 Ch10_3 中，類別 Ch10_3 的定義置於此處
```

加上一個沒有  
引數的建構子

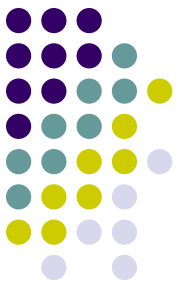
執行結果：

radius=2.5



# this() 與 super() 的比較

- this() 是在同一類別內呼叫其它的建構子
- super() 是從子類別的建構子呼叫父類別的建構子
- this()與super() 相似之處：
  1. this() 與super() 均可多載
  2. this() 與super() 均必須撰寫在建構子內的第一行，因此this() 與super() 無法同時存在同一個建構子內



# 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Ch10\_2中的22~26行改寫成如下的敘述：

```
22     public Coin(double r, int v){        // 子類別裡有兩個引數的建構子
23         radius=r;    // 錯誤，radius 為 private 成員，無法在 Circle 類別外部存取
24         value=v;
25         System.out.println("Coin(double r, int v) constructor called");
26     }
```

編譯時將出現下列的錯誤訊息：

"The field Circle.radius is not visible"

是因為radius在Circle類別內宣告為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Circle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Circle類別裡直接取用，
- 同時也可以在繼承Circle而來的Coin類別裡存取

# 保護成員的範例

## 10.2 保護成員



```
01 // Ch10_5, protected 成員的使用
02 class Circle{
03     protected static double pi=3.14; // 將 pi 宣告成 protected
04     protected double radius;         // 將 radius 宣告成 protected
05
06     public void show(){
07         System.out.printf("area=%6.2f",pi*radius*radius);
08     }
09 }
10 class Coin extends Circle{ // 定義 Coin 類別，繼承自 Circle 類別
11     private int value;
12
13     public Coin(double r, int v){
14         radius=r; // 在子類別裡可直接取用父類別裡的 protected 成員
15         value=v;
16         System.out.println("radius="+radius+", value="+value);
17     }
18 }
19 public class Ch10_5{
20     public static void main(String[] args){
21         Coin coin=new Coin(2.5,10);
22         coin.show();
23     }
24 }
```

此範例是Ch10\_2  
的精簡版

執行結果：

```
radius=2.5, value=10
area= 19.63
```





# 改寫父類別的函數 (1/2)

- 下面是改寫父類別之函數的範例：

```
01 // Ch10_6, 函數的「改寫」範例
02 class Circle{                // 父類別 Circle
03     protected static double pi=3.14;
04     protected double radius;
05
06     public Circle(double r){
07         radius=r;
08     }
09     public void show(){        // 父類別裡的 show() 函數
10         System.out.println("radius="+radius);
11     }
12 }
13 class Coin extends Circle{    // 子類別 Coin
14     private int value;
15
16     public Coin(double r,int v){
17         super(r);
18         value=v;
19     }
```



## 改寫父類別的函數 (2/2)

```
20 public void show(){ // 子類別裡的 show() 函數
21     System.out.println("radius="+radius+", value="+value);
22 }
23 }
24 public class Ch10_6{
25     public static void main(String[] args){
26         Coin coin=new Coin(2.0,5);
27         coin.show(); // 呼叫 show() 函數
28     }
29 }
```

改寫父類別的  
show()

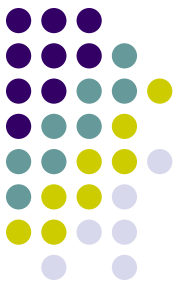
執行結果：

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函數，如此便可依據引數的個數或型態，呼叫相對應的函數
- 「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函數，用以改寫父類別裡函數的功用



# 父類別變數存取子類別成員 (1/4)

- Ch10\_6的26行 與27行：

```
26  Coin coin=new Coin(2.0,5);    // 宣告子類別變數 coin，並將它指向新建的物件
27  coin.show();                  // 利用子類別變數 coin 呼叫 show()函數
```

- 將上面兩行程式碼改寫成如下的敘述：

```
26  Circle cir=new Coin(2.0,5);    // 宣告父類別變數 cir，並將它指向新建的物件
27  cir.show();                    // 利用父類別變數 cir 呼叫 show()函數
```

cir.show()是透過父類別的變數cir來呼叫show() 函數

是定義於父類別裡的show()，還是子類別裡的show() 函數  
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()：

```
01 // Ch10_7, 透過父類別變數 cir 呼叫 show() 函數
02 class Circle{                                // 父類別 Circle
03     protected static double pi=3.14;
04     protected double radius;
05
06     public Circle(double r){
07         radius=r;
08     }
09     public void show(){                        // 父類別裡的 show() 函數
10         System.out.println("radius="+radius);
11     }
12 }
13 class Coin extends Circle{ // 子類別 Circle
14     private int value;
15
16     public Coin(double r,int v){
17         super(r);
18         value=v;
19     }
```



# 父類別變數存取子類別成員 (3/4)

```
20     public void show(){           // 子類別裡的 show() 函數
21         System.out.println("radius="+radius+", value="+value);
22     }
23     public void showValue(){       // showValue() 函數,此函數只存在於子類別
24         System.out.println("value="+value);
25     }
26 }
27 public class Ch10_7{
28     public static void main(String args[]){
29         Circle cir=new Coin(2.0,5); // 宣告父類別變數 cir，並將它指向物件
30         cir.show();                  // 利用父類別變數 cir 呼叫 show()
31         // cir.showValue();
32     }
33 }
```

執行結果：

radius=2.0, value=5

子類別的函數被呼叫



## 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Ch10\_7為例，不能撰寫如下的程式碼：

```
Coin coin=new Circle();
```

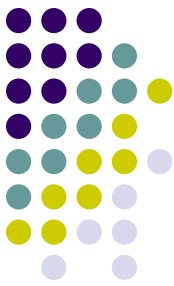
// 錯誤，子類別的變數不能指向父類別的物件

子類別變數

父類別物件

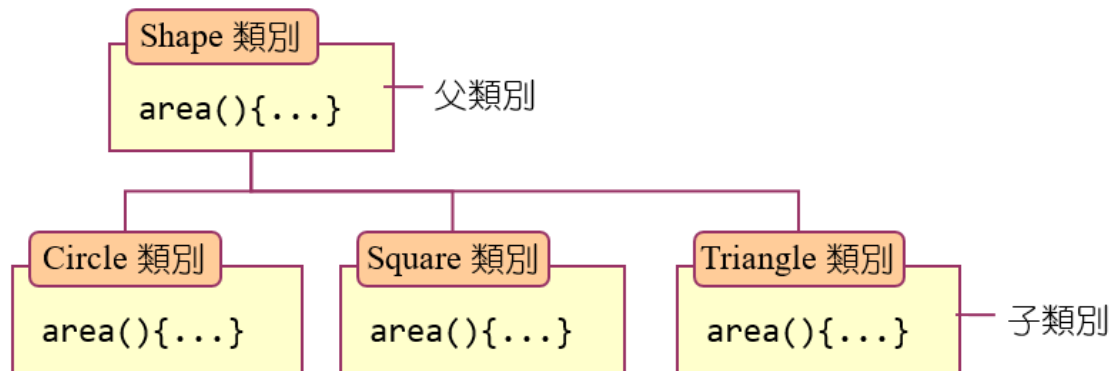
- 如果把31行的註解符號拿掉，編譯時將出現錯誤訊息：

"The method showValue() is undefined for the type Circle"



# 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() 函數：



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



# 用super存取父類別

## 10.4 再談super與this



```
01 // Ch10_8, 透過 super 關鍵字來存取父類別的變數
02 class Caaa{
03     protected int num;                // 父類別的資料成員 num
04
05     public void show(){
06         System.out.println("Caaa_num="+num);
07     }
08 }
09 class Cbbb extends Caaa{
10     int num=10;                        // 子類別的資料成員 num
11
12     public void show(){
13         super.num=20;                  // 設定父類別的資料成員 num 為 20
14         System.out.println("Cbbb_num="+num);
15         super.show();                  // 呼叫父類別的 show() 函數
16     }
17 }
18 public class Ch10_8{
19     public static void main(String[] args){
20         Cbbb b=new Cbbb();
21         b.show();
22     }
23 }
```

此範例說明  
super的用法

super後面可加上資料成員或函數的名稱：

super.資料成員名稱 // 存取父類別的資料成員  
super.函數名稱 // 存取父類別的函數

執行結果：

Cbbb\_num=10

Caaa\_num=20

「super」可以存取父類別的資料  
成員、函數成員及建構子



# this關鍵字

- 下面的範例是利用this來呼叫實例變數：

```
01 // Ch10_9, 用 this 來呼叫實例變數
02 class Caaa{
03     public int num=10;    // num 是實例變數
04
05     public void show(){
06         int num=5;        // num 是區域變數，其有效範圍僅限於在 show()內
07         System.out.println("this.num="+this.num);    // 印出實例變數
08         System.out.println("num="+num);    // 印出區域變數
09     }
10 }
11 public class Ch10_9{
12     public static void main(String[] args){
13         Caaa a=new Caaa();
14         a.show();
15     }
16 }
```

「this」可以存取自己本身類別的資料成員、函數成員及建構子

執行結果：

this.num=10

num=5

# this與super的特性

## 10.4 再談super 與 this



- this : 可以存取自己本身類別的資料成員、函數成員及建構子。
- super : 可以存取父類別的資料成員、函數成員及建構子。由此可知 super 必須要在有繼承關係的情況下才能使用。



# is-a與has-a的意義

- 「is-a」是繼承的觀念
  - 如果A類別繼承B類別（A是B的子類別），則我們說A "is-a" B
- 「has-a」是「擁有」的概念
  - 如果A類別裡宣告另一個B類別的物件（B是A的成員），則我們說A "has-a" B
    - ✓ Circle裡「有一個」（has-a）String成員
    - ✓ Coin「是一個」（is-a）圓形（Circle）
    - ✓ Coin裡「有一個」（has-a）String成員

```
01 class Circle{    // 父類別 Circle
02     String name;
03 }
```

Circle "has-a" String

```
04
05 class Coin extends Circle{ // 子類別 Coin
06     ...
07 }
```

Coin "is-a" Circle  
Coin "has-a" String

# 終止繼承 (1/2)

## 10.6 設定終止繼承



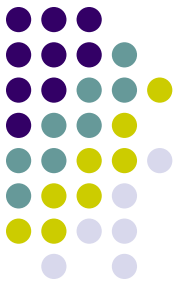
- 設定終止繼承可利用final關鍵字，如下面的範例：

```
01 // Ch10_10, 設定終止繼承
02 class Caaa{
03     public final void show(){    // 父類別的 show()已被設為終止繼承
04         System.out.println("show() 函數 in class Caaa called");
05     }
06 }
07 class Cbbb extends Caaa{
08     public void show(){          // 錯誤，改寫父類別的 show() 函數
09         System.out.println("show() 函數 in class Cbbb called");
10     }
11 }
12 public class Ch10_10{
13     public static void main(String[] args){
14         Cbbb b=new Cbbb();
15         b.show();
16     }
17 }
```

設定show不再被繼承

# 終止繼承 (2/2)

## 10.6 設定終止繼承



- Ch10\_10編譯時會產生如下的錯誤訊息：

"Cannot override the final method from Caaa"

- 此錯誤訊息是說Caaa類別裡的show() 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 Circle{                               // 設定 Circle 類別不能被其它類別繼承
    ...
}
```



# Object類別

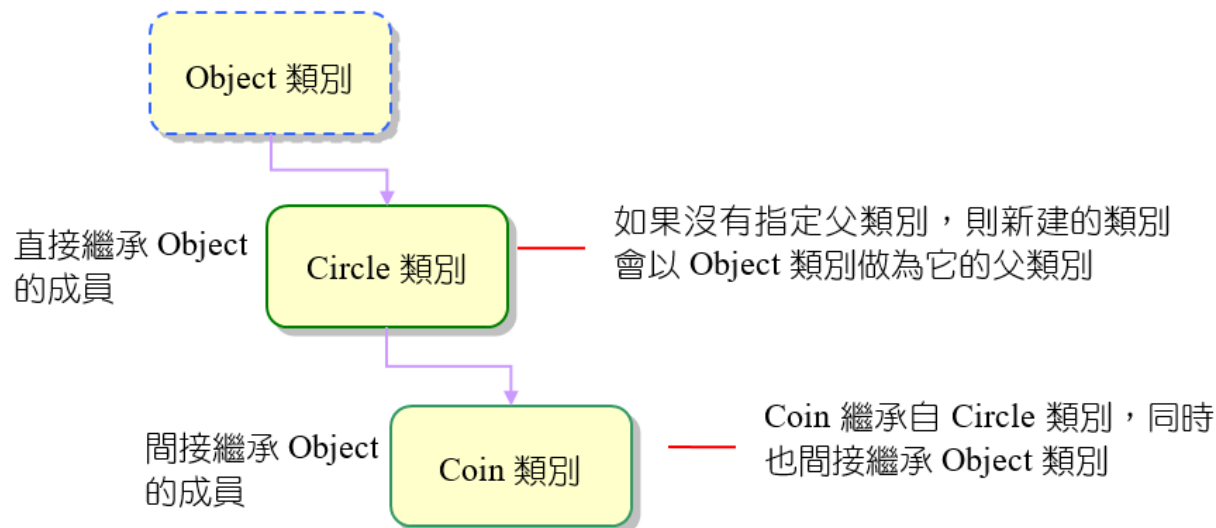
- 若沒指定父類別，則會自動設定Object類別為它的父類別

java.lang.Object 類別) :

```
class Circle{  
    ...  
}
```

—— 若是沒有指定父類別時，Circle 會以 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類別裡的函數

- 下表列舉Object類別裡三個常用的函數：

· Object 類別裡常用的函數

函數名稱	說明
<code>Class getClass()</code>	取得呼叫 <code>getClass()</code> 的物件所屬之類別
<code>Boolean equals(Object obj)</code>	兩個類別變數所指向的是否為同一個物件
<code>String toString()</code>	將呼叫 <code>toString()</code> 的物件轉成字串

- 查詢obj屬於哪個類別：

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





# getClass() 的使用

- 下面的程式是getClass() 簡單的使用範例：

```
01 // Ch10_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 public class Ch10_11{
05     public static void main(String[] args){
06         Caaa a=new Caaa();
07         System.out.println(a.getClass()); // 印出物件 a 所屬的類別
08     }
09 }
```

執行結果：

```
class Caaa
```



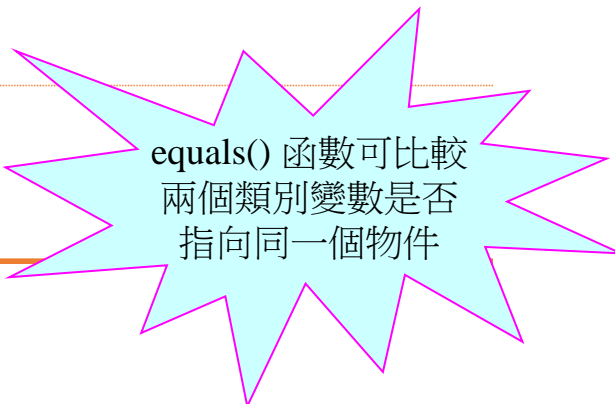
# equals() 的使用範例

```
01 // Ch10_12, 利用 equals() 判別兩個類別變數是否指向同一個物件
02 class Caaa{           // 定義 Caaa 類別
03 }
04 public class Ch10_12{
05     public static void main(String[] args){
06         Caaa a=new Caaa();
07         Caaa b=new Caaa();
08         Caaa c=a;       // 宣告類別變數 c，並讓它指向變數 a 所指向的物件
09         System.out.println("a.equals(b)="+a.equals(b));
10         System.out.println("a.equals(c)="+a.equals(c));
11     }
12 }
```

執行結果：

a.equals(b)=false

a.equals(c)=true



equals() 函數可比較  
兩個類別變數是否  
指向同一個物件



# toString() 的使用 (1/3)

- 下面的範例是toString()的使用：

```
01 // Ch10_13, Object 類別裡的 toString() 函數
02 class Caaa{
03 }
04 public class Ch10_13{
05     public static void main(String[] args){
06         Caaa a=new Caaa();
07         System.out.println(a.toString());    // 印出物件 a 的內容
08     }
09 }
```

執行結果：

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() 的使用 (2/3)

```
01 // Ch10_14, 改寫 Object 類別裡的 toString() 函數
02 class Caaa{
03     private int num;
04
05     public Caaa(int n){
06         num=n;
07     }
08     public String toString(){    // 改寫 toString() 函數
09         String str="toString() called, num="+num;
10         return str;
11     }
12 }
13 public class Ch10_14{
14     public static void main(String[] args){
15         Caaa a=new Caaa(2);
16         System.out.println(a.toString());    // 印出物件 a 的內容
17     }
18 }
```



此範例改寫  
toString()

執行結果：

toString() called, num=2



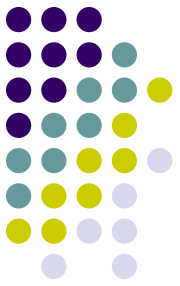
## toString() 的使用 (3/3)

- 改寫toString() 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()，再把結果當成println() 的引數印出

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



-The End-