第十章
類別的繼承

本章學習目標

- 學習繼承的基本概念
- 了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

10.1 繼承的基本概念

繼承：根據既有類別為基礎，進而衍生出另一類別。

- ✓ 既有的類別稱為父類別（super class）或基礎類別（basis class）。
- ✓ 衍生出的類別稱為子類別（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）。
- ✓ 一個父類別可以擁有一個以上的子類別。

類別成員的繼承關係可用下圖來表示：

類別的繼承的語法：

```
class 父類別名稱
{
    //  父類別裡的成員。
}

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

父類別

子類別

格式 10.1.1
類別繼承的格式

10.1.1 簡單的繼承範例

app10_1 包含了 原有的 CCircle 類別，以及繼承而來的 CCoin 類別：

```
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  }
```

```
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);     // 呼叫自父類別繼承而來的 setRadius()
41        coin.show();             // 呼叫自父類別繼承而來的 show()
42        coin.setValue(5);        // 呼叫子類別的 setValue()
43     }
44  }
```

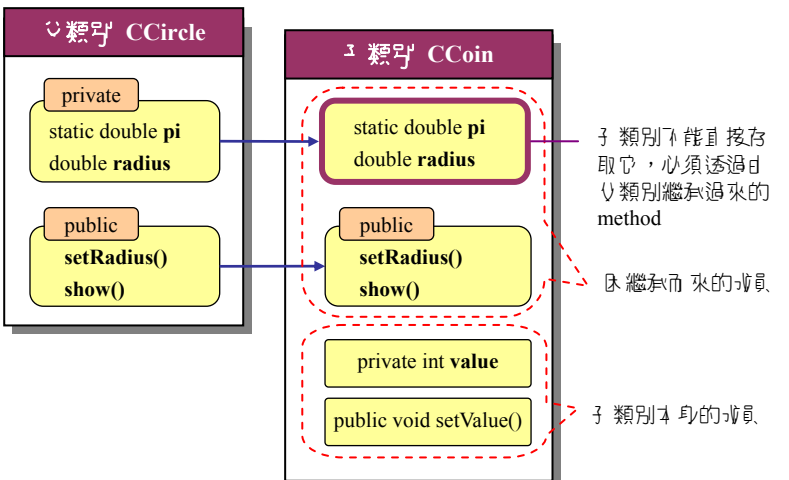
```
/* app10_1 OUTPUT-------------------------------------------------
CCircle() constructor called
CCoin() constructor called
radius=2.0
area=12.56
value=5
----------------------------------------------------------------*/
```

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

呼叫自父類別繼承而來的 method 所得的結果

呼叫 類別的 method 所得的結果

本例的繼承關係圖繪製如下：



下列幾點重要的觀念：

1. 透過 extends 關鍵字，可將父類別的成員（包含資料成員與 method）繼承給子類別。
2. Java 在執行子類別的建構元之前，會先呼叫父類別的建構元。

10.1.2 建構元的呼叫

下面的範例是透過 super() 來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  }
```

```
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  }
```

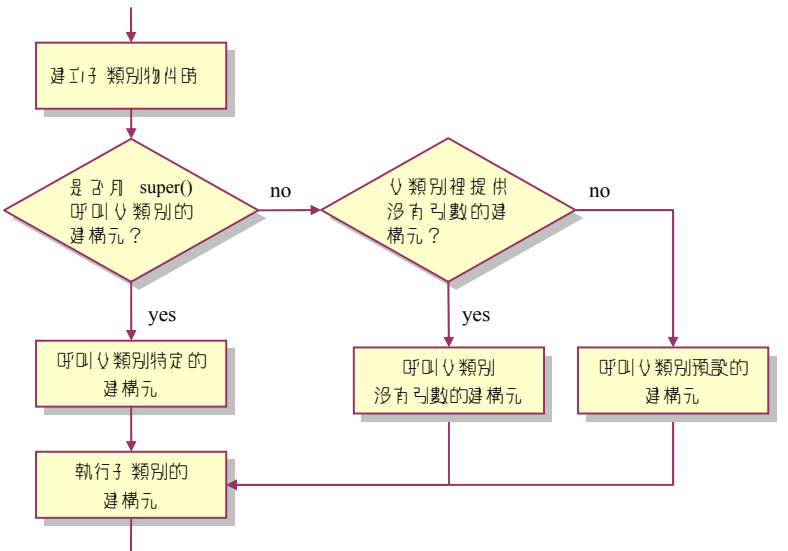
```
/* app10_2 OUTPUT-------------------------
CCircle() constructor called
CCoin() constructor called
CCircle(double r) constructor called
CCoin(double r, int v) constructor called
area=0.0
area=19.625
----------------------------------------*/
```

執行第 39 行所得的結果

執行第 40 行所得的結果

10.1.3 使用建構元常見的錯誤

下圖是建構元呼叫的流程圖：



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
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   }
```

執行時會得到如下的錯誤息：

```
cannot find symbol
symbol  : constructor CCircle ()
```

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() 則是從子類別的建構元呼叫其父類別的建構元。

this()與 super()還是有其相似之處：

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。

10.2 保護成員（protected members）

如果在子類別內直接存取 private 的資料成員，則在編譯時將出現錯誤。

若把 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;
```

若在 CCircle 類別裡把 radius 與 pi 這兩個成員宣告成 protected：

```
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。

下面的範例將 radius 與 pi 這兩個成員設為 protected：

```
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;   // 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  }

/* app10_5 OUTPUT-----
radius=2.5, value=10
area=19.625
------------------------*/
```

10.3 改寫

改寫（overriding）與多載，均是多型（polymorphism）的技術之一。

10.3.1 改寫父類別的 method

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
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，它是在相同類別內，定義名稱相同，但引數個數或型態不同的 method，Java 依據引數的個數或型態，呼叫相對應的 method。

「改寫」：overriding，它是在子類別當中，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。

10.3.2 以父類別的變數存取子類別物件的成員

下面的程式碼，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)
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  class CCoin extends CCircle   // 子類別 CCircle
17  {
18     private int value;
19
20     public CCoin(double r,int v)
```

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

```
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   }
```

10.4 再談 super 與 this

super 後面也可加上資料成員或 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
```

下面的範例說明了 super 關鍵字的用法：

```
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  }

/* app10_8 OUTPUT--

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

內 函 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 僅限於內 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
--------------------*/
```

10.5 設定終止繼承

如果想設定終止繼承，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()    // 此類別的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，改寫此類別的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  }
```

於 app10_10，在程式編譯時將產生如下的錯誤訊息：

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

final 的其它用法：

```
static final double PI=3.14;   // 設定 PI 值不能再被修改
final class CCircle  // 設定 CCircle 類別不能被其它類別繼承
```

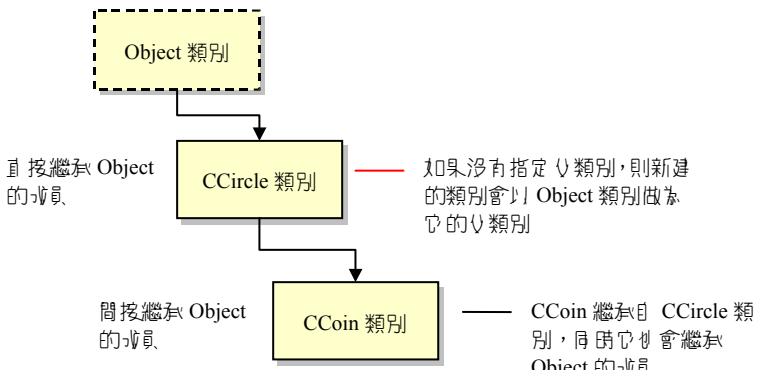
10.6 類別之源--Object 類別

如果沒指定父類別，則 Java 會自動設定 Object 這個類別為它的父類別。

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

當沒有指定父類別時，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。

Object 類別可說是 類別之源，所有的類別均直 接或間接繼承它 ：



下表列舉了 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)	與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

getClass() method 的使用

想知道某個物件 obj 是屬於哪個類別時，可用：

```
obj.getClass()
```

的語法來查詢。

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

equals() method 可用來比較兩個類別變數是否指向同一個物件：

```
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

toString() 的功用是將物件的內容轉換成字串，如下面的範例：

```
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@5d87b2
----------------------*/
```

改寫 toString() method 的方法如下面的範例：

```
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  }
```

```
/* app10_14 OUTPUT-----

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

-The End-