第十一章
抽象類別與介面

- **本章學習目標**
  - 認識抽象類別
  - 學習介面的使用
  - 認識多重繼承與介面的延伸

1

11-2

11.1 抽象類別

抽象類別的目的是要依據它的格式來修改並建立新的類別。

11.1.1定義抽象類別

定義抽象類別的語法如下：

```
abstract class 類別名稱   // 定義抽象類別
{
  宣告資料成員;

  傳回值的資料型態 method名稱(引數...)
  {
    ...
  }
  修飾子 abstract 傳回值資料型態 method名稱(引數...);
}
```

定義一般method

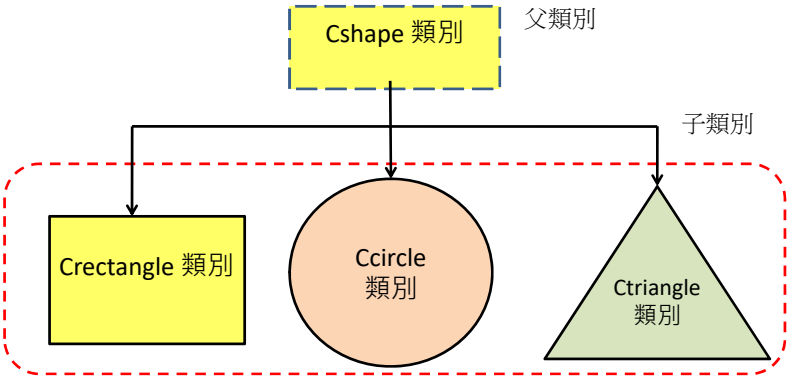
定義抽象method。注意於抽象method裡，沒有定義處理的方式

2

11-3

11.1.2 抽象類別的實作

- 例如，想設計一個父類別CShape，依據此類別可用來衍生出圓形、長方形與三角形等幾何形狀的類別：



3

11-4

- 下面的父類別程式碼，即是抽象類別CShape的內容：

```
01  // 定義抽象類別CShape
02  abstract class CShape          // 定義抽象類別CShape
03  {
04    protected String color;      // 資料成員
05    public void setColor(String str)  // 一般method,用來設定何形狀的顏色
06    {
07      color=str;
08    }
09    public abstract void show();  // 抽象函數，在此沒有定義處理方式
10  }
```

4

11-5

下面的程式碼是以子類別CCircle為例來撰寫的：

```
01  // 定義由抽象類別CShape而衍生出的子類別CCircle
02  class CCircle extends CShape  //定義子類別CCircle
03  {
04    protected double radius;     // 資料成員
05    public CCircle(double r)     // 建構元
06    {
07      radius=r;
08    }
09    public void show()
10    {
11      System.out.print("color="+color+",  ");
12      System.out.println("area="+3.14*radius*radius);
13    }
14  }
```

在此處明確定義show()的處理方式

5

app11_1是抽象類別實作的完整範例。

11-6

```
01  // app11_1, 抽象類別的實例
02  abstract class CShape   // 定義抽象類別CShape
03  {
04    protected String color;                    // 資料成員
05    public void setColor(String str)  // 一般的函數
06    { color=str;  }
09    public abstract void show();// 抽象函數,只有宣告,但沒有定義處理方式
10  }
11  class CRectangle extends CShape   // 定義子類別CRectangle
12  {
13    protected int width,height;
14    public CRectangle(int w,int h)
15    {  width=w;
17      height=h;  }
19    public void show()  // 明確定義繼承自抽象類別的Show()method
20    {
21      System.out.print("color="+color+",  ");
22      System.out.println("area="+width*height);
23    }
24  }
25  class CCircle extends CShape  // 定義子類別CCircle
26  {
27    protected double radius;
28    public CCircle(double r)
29      { radius=r;  }
32    public void show()    // 明確定義繼承自抽象類別的Show()metho
33     {
34      System.out.print("color="+color+",  ");
35      System.out.println("area="+3.14*radius*radius);
36     }      }
```

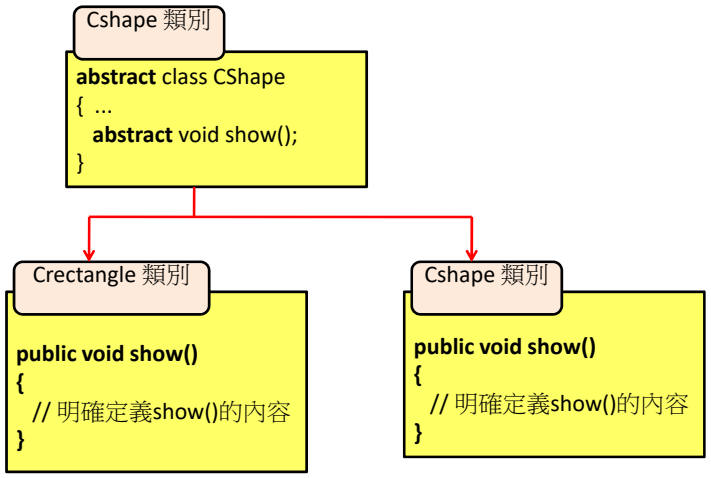
```
/* app11_1 OUTPUT---
color=Yellow,  area=50
color=Green,  area=12.56
----------*/
```

```
38  public class app11_1
39  {
40    public static void main(String args[])
41    {
42      CRectangle rect=new CRectangle(5,10);
43      rect.setColor("Yellow");
44      rect.show();
46      CCircle cir=new CCircle(2.0);
47      cir.setColor("Green");
48      cir.show();
49    }
50  }
```

6

11-8

CShape、CCircle與CRectangle的show() method之間的關係圖：



7

11-9

11.1.3 用抽象類別型態的變數來建立物件

利用父類別的變數來存取子類別物件的成員

app11_2是以抽象類別型態的變數建立物件的範例：

```
01  // app11_2, 用抽象類別型態的變數來建立物件
02  // 將app11_1的CShape類別的定義放在這兒
03  // 將app11_1的CRectangle類別的定義放在這兒
04  // 將app11_1的CCircle類別的定義放在這兒
05  public class app11_2
06  {
07    public static void main(String args[])
08    {
09      CShape shape1=new CRectangle(5,10);
10      shape1.setColor("Yellow");
11      shape1.show();
12
13      CShape shape2=new CCircle(2.0);
14      shape2.setColor("Green");
15      shape2.show();
16    }
17  }
```

父 子

以類別CShape建立物件shape1，並以它來存取子類別CRectangle的成員

父 子

以類別CShape建立物件shape2，並以它來存取子類別CCircle的成員

```
/* app11_2 OUTPUT-----
color=Yellow,  area=50
color=Green,  area=12.56
----------------------*/
```

下一頁有完整程式碼

8

```
// app11_2, 用抽象類別型態的變數來建立物件
abstract class CShape   // 定義抽象類別CShape
{
  protected String color;    // 資料成員
  public void setColor(String str)  // 一般的函數
  {
    color=str;
  }
  public abstract void show(); // 抽象函數，只有宣告，但沒有定義處理方式
}
```

```
class CRectangle extends CShape  // 定義子類別CRectangle
{
  protected int width,height;
  public CRectangle(int w,int h)
  {
    width=w;
    height=h;
  }
  public void show()  // 明確定義繼承自抽象類別的Show() method
  {
    System.out.print("color="+color+", ");
    System.out.println("area="+width*height);
  }
}
```

```
class CCircle extends CShape  // 定義子類別CCircle
{
  protected double radius;
  public CCircle(double r)
  {
    radius=r;
  }
  public void show() // 明確定義繼承自抽象類別的Show()method
  {
    System.out.print("color="+color+", ");
    System.out.println("area="+3.14*radius*radius);
  }
}
```

```
public class app11_2
{
  public static void main(String args[])
  {
    CShape shape1=new CRectangle(5,10);
    shape1.setColor("Yellow");
    shape1.show();

    CShape shape2=new CCircle(2.0);
    shape2.setColor("Green");
    shape2.show();
  }
}
```

```
/* app11_2 OUTPUT-----
color=Yellow, area=50
color=Green, area=12.56
----------------------*/
```

9

11-10

利用父類別的陣列變數來存取子類別物件的成員

當所建立的物件較多時，可利用父類別的陣列變數來存取子類別物件的成員：

① 先建立父類別的陣列變數

② 利用陣列元素建立子類別的物件，並以它來存取子類別的內容

10

11-11

app11_3改寫了app11_2：

```
01  // app11_3, 利用父類別的陣列變數來存取子類別的內容
02  // 將app11_1的CShape類別的定義放在這兒
03  // 將app11_1的CRectangle類別的定義放在這兒
04  // 將app11_1的CCircle類別的定義放在這兒
05  public class app11_3
06  {
07    public static void main(String args[])
08    {
09      CShape shape[];           // 宣告CShape型態的陣列變數
10      shape=new CShape[2];      // 產生兩個CShape抽象類別型態的變
11
12      shape[0]=new CRectangle(5,10);
13      shape[0].setColor("Yellow");
14      shape[0].show();
15
16      shape[1]=new CCircle(2.0);
17      shape[1].setColor("Green");
18      shape[1].show();
19    }
20  }
```

利用陣列變數shape[0]建立物件，並存取子類別的成員

利用陣列變數shape[1]建立物件，並存取子類別的成員

```
/* app11_2 OUTPUT-----
color=Yellow,  area=50
color=Green,  area=12.56
----------------------*/
```

下一頁有完整程式碼

11

```
// app11_3, 利用父類別的陣列變數來存取子類別的內容
abstract class CShape   // 定義抽象類別CShape
{
  protected String color; // 資料成員
  public void setColor(String str) // 一般的函數
  {
    color=str;
  }
  public abstract void show(); // 抽象函數，只有宣告但沒有定義處理方式
}

class CRectangle extends CShape   // 定義子類別CRectangle
{
  protected int width,height;
  public CRectangle(int w,int h)
  {
    width=w;
    height=h;
  }
  public void show() // 明確定義繼承自抽象類別的Show() method
  {
    System.out.print("color="+color+",  ");
    System.out.println("area="+width*height);
  }
}

class CCircle extends CShape   // 定義子類別CCircle
{
  protected double radius;
  public CCircle(double r)
  {
    radius=r;
  }
  public void show() // 明確定義繼承自抽象類別的Show() method
  {
    System.out.print("color="+color+",  ");
    System.out.println("area="+3.14*radius*radius);
  }
}

public class app11_3
{
  public static void main(String args[])
  {
    CShape shape[]; // 宣告CShape型態的陣列變數
    shape=new CShape[2];  // 產生兩個CShape抽象類別型態的變數

    shape[0]=new CRectangle(5,10);
    shape[0].setColor("Yellow");
    shape[0].show();

    shape[1]=new CCircle(2.0);
    shape[1].setColor("Green");
    shape[1].show();
  }
}
```

```
/* app11_2 OUTPUT-----
color=Yellow,  area=50
color=Green,  area=12.56
----------------------*/
```

12

11.1.4使用抽象類別的注意事項

11-12

- ✓抽象類別不能用來直接產生物件。
- ✓在抽象類別內可以定義建構元，以供子類別的建構元呼叫。
- ✓定義在抽象類別裡的抽象函數，在子類別裡一定要被「改寫」。
- ✓如果子類別裡沒有「改寫」抽象函數，則子類別也要宣告成**abstract**。

13

下面的程式碼是一個簡單數學四則運算的範例。我們在抽象類別 Math 裡已定義好一個 show()，以及4 個abstract method。請在Compute 類別裡撰寫add()、sub()、mul() 與div()這4個method 的定義，使得我們可以利用Compute 類別來做兩個整數的四則運算。例如，在第22 行建立Compute 類別的物件cmp 後，便可利用它來進行23~24 行的運算。

ex11_1_1.java

```
02 abstract class Math
03 {
04 protected int ans;
05 public void show()
06 {
07 System.out.println("ans="+ans);
08 }
09 public abstract void add(int a, int b); // 計算 a+b
10 public abstract void sub(int a, int b); // 計算 a-b
11 public abstract void mul(int a, int b); // 計算 a*b
12 public abstract void div(int a, int b); // 計算 a/b
13 }
14 class Compute extends Math
15 {
16 // 請完成這個部分的程式碼
17 }
18 public class ex11_1_1
19 {
20 public static void main(String args[])
21 {
22 Compute cmp=new Compute();
23 cmp.mul(3,5); // 計算3*5
24 cmp.show(); // 此行會回應 "ans=15" 字串
25 }}
```

```
/* output----
ans=15
----------*/
```

14

- 在習題1 中，如果在main() method 裡加入這麼一行敘述：Math mth=new Math();編譯時您將得到下面的錯誤訊息：Math is abstract; cannot be instantiated 請解釋這行錯誤訊息的大意，並指明是錯在哪兒。

- Q: 在習題1 中，抽象類別Math 裡的~~show()~~ mul(int a, int b) method 是否可以宣告成static？

A: mul(int a, int b) method 不能宣告成static，若宣告成static，再Math.mul(3,5)，則會產生程式錯誤。

抽象方法的目的是被改寫(override)，但若宣告成static的抽象方法，則無法被繼承的子類別改寫此抽象方法。

15

Can an abstract class have static methods?

- Yes, abstract class can have static Methods.
  - The reason for this is static methods do not work on the instance of the class, they are directly associated with the class itself.
  - ex show()
- But can public abstract void add(int a, int b) set into static method ?
  - Abstract methods require an instance, but static methods do not have an instance. So, you can have a static method in an abstract class, it just cannot be static abstract (or abstract static)
  - An abstract method is defined only, so that it can be overridden in a subclass. However, static methods can not be overridden.

16

下面的程式碼是抽象類別裡建構元的撰寫練習，請先閱讀它，再回答接續的問題：

ex11_1_2.java

```
02 abstract class Caaa
03 {
04 protected int num;
05 // 請在此處撰寫類別Caaa 的建構元
06 public abstract void show();
07 }
08 class Cbbb extends Caaa
09 {
10 // 請在此處撰寫類別Cbbb 的建構元
11 // 請在此處撰寫show() method
12 }
13 public class ex11_1_2
14 {
15 public static void main(String args[])
16 {
17 Cbbb bb=new Cbbb(2); // 此行可設定num 成員的值為2
18 bb.show(); // 此行可印出 "num=2" 字串
19 }
20 }
```

```
/* output---
num=2
----------*/
```

(a) 請撰寫抽象類別Caaa 的建構元，它可接收一個整數n，並把num 成員設為n。
(b) 試撰寫子類別Cbbb 的建構元，它可用來呼叫父類別的建構元，並設定num 的值。
(c) 試在Cbbb 類別裡定義抽象類別中的show() method，使得它可印出num 的值。

17

11.1 回家作業 抽象類別

hw11_1_1.java

請在抽象類別Area裡定義好3個abstract method (square(int a)、triangle(int a,int b)、circle(int a))，宣告一型態為double的 ans變數。而後在Compute類別裡撰寫square()、triangle()、circle()這3個method的實作，使得我們可以利用Compute 類別來做各種面積的運算及列印。

*square() 求正方形面積

*triangle() 求三角形面積

*circle() 求圓形面積

```
/* output--------------
正方形邊長為5，面積為25.0
三角形底為5，高為6，面積為30.0
圓形半徑為5，面積為78.5
---------------------*/
```

18

11.1 回家作業 抽象類別

hw11_1_2.java

(a)請設計一抽象類別Caaa，抽象類別中有變數x與y，並定義好set_num()method可以設定a,b的值。並有兩個抽象方法show()及forloop()

(b)請Cbbb類別裡撰寫show() 及forloop()的定義如下：

show() method，可用來顯示a與b的值

forloop() method，可用for迴圈將a的偶數累加至b。

```
/* output--------------
x=2,y=18
2+4+6+...+18=90
--------------------*/
```

19

11-13

11.2 介面的使用

介面與抽象類別有下列兩點不同：

① 介面的資料成員必須初始化(即設定初值)

② 介面裡的method必須**全部都宣告成abstract**。

20

11-14

介面定義的語法如下：

```
interface 介面名稱      // 定義介面
{
  public static final 資料型態 成員名稱=常數;   // 資料成員必須設定初值

  public abstract 傳回值資料型態 method名稱(引數...);
}
```

格式11.2.1
介面的定義格式

宣告抽象method。注意於抽象method裡，沒有定義處理的方式

可以是void or return type

三個關鍵字在介面中的存在原因：
public: 介面可以被其他interface繼承, 也可以被class實現, class與interface, interface與interface可能會形成多層級關係, 採用public可以滿足變數的訪問範圍；
static：如果變數不是static的，那麼介面必須建立物件才可以使用自己的變數，介面不能被建立物件，故需要設定成static的變數方能存取；
final：如果變數不是final的，而方法是abstract的，因此介面中的方法又不可以修改變數值，雖然可以直接修改靜態成員變數.

21

Interface variables are static and final by default in Java, Why?

- An interface defines a protocol of behavior and not how we should be implemented. A class that implements an interface adheres to the protocol defined by that interface.
  - Interface variables are static because java interfaces cannot be instantiated on their own. The value of the variable must be assigned in a static context in which no instance exists.
  - The final modifier ensures the value assigned to the interface variable is a true constant that cannot be re-assigned. In other words, interfaces can declare only constants, not instance variables.

22

Interface的好處

1.簡單、規範性：

如果一個專案比較龐大，那麼**先定義一些主要的介面，這些介面不僅告訴開發人員你需要實現那些業務，而且也將命名規範限制住**了。

2.維護、拓展性：

比如要做一個程式，其中裡面有一個類，滿足不了所需要的功能，又需要要重新設計這個類，如果這個類被其他地方引用，修改起來很麻煩。假如一開始就定義了一個介面，通過類實現這個介面，這樣修改的時候只不過是引用另一個類而已，就達到維護、拓展的方便性。

3.安全、嚴密性：

介面是實現軟體鬆耦合的重要手段，它描敘了系統對外的所有服務，而不涉及任何具體的實現細節。這樣就比較安全、嚴密一些。

4.在大專案中，介面是需要提前做好架構的。

一個完善的，輕巧，方便，直觀，容易理解的介面架構，可以方便各個崗位的工程師同時開工最後整合。

5.方便了後期優化，實現了真正意義的封裝

給別人看的都是最簡單的最直接的，把複雜的具體的難理解的東西，都封裝到實現類裡。

23

abstract class和interface 的差別與使用時機

- 何謂abstract class?

  提供一種多個class一起合作工作的方式, 將多個class中相同的元素pull up method到public class中, 再以繼承的方式來使用它, 目的是為了實現"多型"精神

  Ex: 數學計算包括求各種形狀的面積、體積 、數字的立方、平方等, 就可以把類似的方法集中到同一個public class中

```
abstract class Shape{
    String name;
    double length, width, heigh;
    double radius;

    abstract double area();          // 求面積
    abstract double perimeter();    // 求周長

    // abstract class中也可以有建構方法, 但必須在子類別中被呼叫
    // 四邊形的建構方法
    public Shape(double length, double width, String name){
        this.length = length;
        this.width = width;
        this.name = name;
    }

    // 三角形的建構方法
    public Shape(String name, double width, double height){
        this.height = height;
        this.width = width;
        this.name = name;
    }
}
```

24

abstract class和interface 的差別與使用時機

- 何謂interface?

  即spec., 完全不需要定義實作, 只需要函式原型
  若要實作interface, 就必須follow它的spec.

  [public] [abstract] interface 介面名稱{
  　權限設定　傳回型態　method(parameters);
  　權限設定　傳回型態　method(parameters);
  }
  [public] [abstract]是預設, 所以可省略, 因為interface本身就是抽象的

  (註)一個介面可以同時繼承多個介面, 即同時繼承了多個介面的abstract method和常數
  => interface A **extends** 介面1, 介面2, 介面3, ...

  一個class可以同時實作多個interface
  => class B **implement**s 介面1, 介面2, 介面3, ...

25

舉例說明

- 本田(Honda)有兩款車， Civic和Accord
  - 這兩款車的輪胎大小，煞車系統，安全氣囊等裝置都一樣，只有引擎的啟動方式不一樣
  - 這時可以定義一個抽象類別HondaCar ，並將**引擎啟動的方法定義為abstract**，強迫繼承的類別實作這個方法。
  - 為什麼要定義成抽象類別呢?
    - 1)因為Civic和Accord的特徵幾乎都一樣，只有引擎的啟動方式不同
    - 2)如果不把相同的程式碼定義在上層，這樣一來兩個類別都要寫重覆的程式碼
    - 3)HondaCar宣告成抽象，是因為它不是一台具體的車，宣告成抽象可以避免這個類別被實體化。
  - 假設現在政府規定，所有汽車都要增設功能，為每1000公里時都要將目前廢氣排放的情況回報給環保署的主機。
  - 為什麼現在要用介面呢?
    - 1)介面像是一個規範，所以實作的類別都要遵守這個規範，而不論繼承的類別實作方式是什麼。所以實作CarReporter的車子都一定要有回報的機制，至於怎麼回報，由各汽車廠自己決定。
    - 2)Civic和Accord，它們的本質是汽車，且Civic是一台有回報排氣裝置的汽車。

26

11-15

下面的範例定義一介面iShape2D：

```
01  // 定義iShape2D介面
02  interface iShape2D
03  {
04     public static final double pi=3.14;  // 資料成員一定要初值化
05     public abstract void area();  // 抽象函數，不需要定義處理方式
06  }
```

程式碼可改寫如下：

```
01  // 定義iShape2D介面，省略final與abstract關鍵字
02  interface iShape2D
03  {
04     double pi=3.14;        // 省略final關鍵字
05     void area();           // 省略abstract關鍵字
06  }
```

27

11-16

利用介面A打造新的類別B的過程，稱之為以類別B實作介面A，或簡稱介面的實作（implementation）。

介面實作的語法：

```
class 類別名稱 implements 介面名稱   // 介面的實作
{
   ... ...
}
```

格式11.2.2
以類別實作介面的語法

28

11-17

下面是以CCircle類別實作iShape2D介面的範例：

```
01  // 介面的實作
02  class CCircle implements iShape2D  // 以CCircle類別實作iShape2D介面
03  {
04    double radius;
05    public CCircle(double r)      // 建構元
06    {
07      radius=r;
08    }
09    public void area()    // 定義area()的處理方式
10    {
11      System.out.println("area="+pi*radius*radius);
12    }
13  }
```

以類別CCircle來實作介面iShape2D

```
01  // 定義iShape2D介面
02  interface iShape2D
03  {
04    final double pi=3.14;  // 資料成員一定要初值化
05    abstract void area();  // 抽象函數，不需要定義處理方式
06  }
```

29

11-18

app11_4是以iShape2D介面實作CCircle與CRectangle二個類別的範例：

```
02  interface iShape2D   // 定義介面，  app11_4, 介面的實作範例
03  { final double pi=3.14;
05    abstract void area(); }
08  class CRectangle implements iShape2D// 以CRectangle類別實作iShape2D介面
09  {  int width,height;
11    public CRectangle(int w,int h)
12    { width=w;
14      height=h; }
16    public void area()   // 定義area()的處理方式
17    {  System.out.println("area="+width*height);    }
20  }
21
22  class CCircle implements iShape2D// 以CCircle類別實作iShape2D介面
23  {
24    double radius;
25    public CCircle(double r)
26    {
27      radius=r;
28    }
29    public void area()    // 定義area()的處理方式
30    {
31      System.out.println("area="+pi*radius*radius);
32    }
33  }
34
```

```
/* app11_4 OUTPUT-----
area=50
area=12.56
----------------------*/
```

```
35  public class app11_4
36  {
37    public static void main(String args[])
38    {
39      CRectangle rect=new CRectangle(5,10);
40      rect.area(); // 呼叫CRectangle類別裡的area() method
41
42      CCircle cir=new CCircle(2.0);
43      cir.area(); // 呼叫CCircle類別裡的area() method
44    }
45  }
```

30

app11_5是利用透過介面型態的變數來存取物件的範例。 11-20

```
01  // app11_5,透過介面型態的變數來存取物件
02  // 將app11_4的iShape介面的定義放在這兒
03  // 將app11_4的CRectangle類別的定義放在這兒
04  // 將app11_4的CCircle類別的定義放在這兒
05  public class app11_5
06  {
07    public static void main(String args[])
08    {
09      iShape2D var1,var2;        // 宣告介面型態的變數
10      var1=new CRectangle(5,10); // 將介面型態的變數var1指向新建的物件
11      var1.area();     // 透過介var1呼叫show() method
12
13      var2=new CCircle(2.0);     // 將介面型態的變數var2指向新建的物件
14      var2.area();               // 透過介面var2呼叫show() method
15    }
16    }
```

```
/* app11_5 OUTPUT--
area=50
area=12.56
--------------------*/
```

下一頁有完整程式碼➔

31

```
// app11_5,透過介面型態的變數來存取物件
interface iShape2D              // 定義介面
{
  final double PI=3.14;
  abstract void area();
}
```

```
class CRectangle implements iShape2D // 以CRectangle類別實作
iShape2D介面
{
  int width,height;
  public CRectangle(int w,int h)
  {
    width=w;
    height=h;
  }
  public void area()  // 定義area()的處理方式
  {
    System.out.println("area="+width*height);
  }
}
```

```
class CCircle implements iShape2D // 以CCircle類別實作iShape2D介
面
{
  double radius;
  public CCircle(double r)
  {
    radius=r;
  }
  public void area()              // 定義area()的處理方式
  {
    System.out.println("area="+PI*radius*radius);
  }
}
```

```
/* app11_5 OUTPUT--
area=50
area=12.56
--------------------*/
```

```
public class app11_5
{
  public static void main(String args[])
  {
    iShape2D var1,var2;        // 宣告介面型態的變數
    var1=new CRectangle(5,10); // 將介面型態的變數var1指向新建的物件
    var1.area();     // 透過介面var1呼叫show() method

    var2=new CCircle(2.0);     // 將介面型態的變數var2指向新建的物件
    var2.area();               // 透過介面var2呼叫show() method
  }
}
```

下面的程式碼是修改自習題1，其中把抽象類別 Math 改以介面來宣告。請在Compute類別裡撰寫add()、sub()、mul()、div() 與show() 這五個 method 的定義，使得我們可以利用Compute 類別來做兩個整數的四則運算：

ex11_2_1.java

```
02 interface Math
03 {
04 public void show();
05 public void add(int a, int b); // 計算 a+b
06 public void sub(int a, int b); // 計算 a-b
07 public void mul(int a, int b); // 計算 a*b
08 public void div(int a, int b); // 計算 a/b
09 }
10 class Compute implements Math
11 {
12 // 請完成這個部分的程式碼
13 }
14
15 public class ex11_2_1
16 {
17 public static void main(String args[])
18 {
19 Compute cmp=new Compute();
20 cmp.mul(3,5); // 計算3*5
21 cmp.show(); // 此行會回應 "ans=15" 字串
22 }
23 }
```

```
/* output----
ans=15
----------*/
```

33

- 於ex11_1_1.java中，show() method 是詳細定義在Math 類別裡，但於ex11_2_1.java 中，show() method的定義則是放在Compute 裡。這兩者有何差別？我們可以把ex11_2_1.java中show() method的詳細定義放在Math介面裡嗎？
- 於ex11_2_1.java中，我們是否可以在4~8 行的method 之前加上abstract 關鍵字？加不加這個關鍵字有差別嗎？

34

如果某個類別C 同時繼承了類別B，又實作了介面A，則我們必須以下面的語法來撰寫類別A：
class C extends B implements A
{ // 類別C 裡的程式碼 }
下面的程式碼定義了介面iAaa 與類別Cbbb，試撰寫類別Cccc，使其繼承自類別Cbbb，時也實作介面iAaa，並於程式執行時，會於第25 行印出 "num=5" 字串。

ex11_2_2.java

```
02 interface iAaa
03 {
04 public void show();
05 }
06
07 class Cbbb
08 {
09 public int num=10;
10
11 public void set(int n)
12 {
13 num=n;
14 }
15 }
16
17 // 請於此處定義Cccc 類別
18
19 public class ex11_8
20 {
21 public static void main(String args[])
22 {
23 Cccc cc=new Cccc();
24 cc.set(5);
25 cc.show(); // 印出num=5
26 }
27 }
```

```
/* output----
num=5
----------*/
```

35

11.2 回家作業 介面的使用

hw11_2_1.java

1. 宣告一個介面 Data，
   void best()是判斷那一科成績較高或一樣高，
   void failed()是判斷那一科成績低於60分或同時不及格或同時及格。
2. 再宣告另一個介面Test繼承它，
   void showData()顯示學生的資料及平均成績，
   double average()是計算數學和英文的平均成績。
3. 由類別CStu實作Test，並於類別中宣告姓名、數學成績及英文成績等變數。
4. 請在CStu類別裡撰寫所有method的定義，再於CStu類別中加入一個show() method，用來呼叫best()、failed()、showData()與average()等method。

```
/* output--------------
姓名:Amy
數學成績:90
英文成績:51
平均成績:70.5
Amy的數學成績比較高
Amy的英文低於60分
--------------------*/
```

36

11.2 回家作業 介面的使用

hw11_2_2.java

1. 試著定義Cry介面，該介面將定義crying()方法。
2. 增加一個Eat介面，該介面將定義eating()方法，繼承Cry介面。
3. 並用Dog、Cat及Monkey三個類別實作Eat介面

甲、實作crying()方法，將輸出不同的叫聲，如：Dog類別將輸出”汪汪”，Cat類別將輸出”喵喵”，Monkey類別將輸出”吱吱”。

乙、實作eating()方法，將輸出喜歡吃的食物，如：Dog類別將輸出為”喜歡吃骨頭”，Cat類別將輸出為”喜歡吃魚肉”，Monkey類別將輸出為”喜歡吃香蕉”。

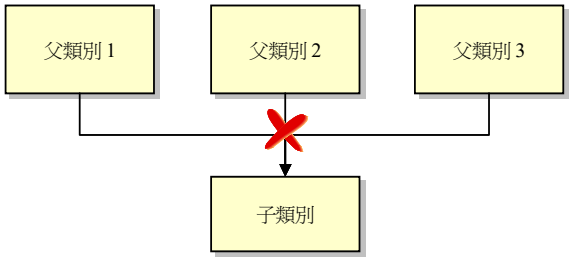
```
/* output----------
狗狗：汪汪，喜歡吃骨頭
貓咪：喵喵，喜歡吃魚肉
猴子：吱吱，喜歡吃香蕉
----------*/
```

37

11-21

11.3 多重繼承

Java並不允許多個父類別的繼承：



用類別來實作兩個以上的介面，即可實現多重繼承。

38

interface重點整理

interface的特性

(1) 完全的抽象類別:裡面的方法都要宣告成abstract,
而abstract class中,則允許有不是 abstract的method

(2) 變數必為(final)一定要設初值; 方法必為 (abstract)

(3) 只有類別的架構和概念

(4) 使用implements來宣告及建置

implements (實作介面)

(1)讓類別具有interface的特性

(2) 彌補單一繼承的限制

(3) 以,區隔多個interface

39

11-22

將類別和兩個以上的介面實作在一起的語法如下：

```
class 類別名稱 implements 介面1,介面2,...
{
    ... ...
}
```

格式11.3.1
實作兩個以上的介面

40

app11_6是以兩個介面iShape2D和iColor來實作CCirlce類別的範例。

11-23

```
01  // app11_6, 用CCircle類別實作兩個以上的介面
02  interface iShape2D   // 定義iShape2D介面
03  { final double pi=3.14;
05    abstract void area(); }
07
08  interface iColor   // 定義iColor介面
09  { abstract void setColor(String str); }
13  class CCircle implements iShape2D,iColor // 實作iShape2D與iColor介面
14  {
15    double radius;
16    String color;
17    public CCircle(double r)
18    {
19      radius=r;
20    }
21    public void setColor(String str)  // 定義iColor介面裡的setColor()
22    {
23      color=str;
24      System.out.println("color="+color);
25    }
26    public void area()    // 定義iShape2D介面裡的area() method
27    {
28      System.out.println("area="+pi*radius*radius);
29    } }
```

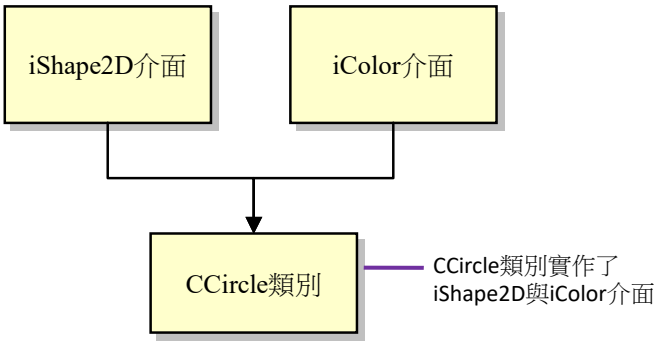
```
/* app11_6
OUTPUT---
color=Blue
area=12.56
----------------------*/
```

```
31  public class app11_6
32  {
33    public static void main(String args[])
34    {
35      CCircle cir;
36      cir=new CCircle(2.0);
37      cir.setColor("Blue");
                  // 呼叫setColor() method
38      cir.area(); // 呼叫show() method
39    }
40  }
```

41

11-25

透過類別與介面的實作，可達到多重繼承的目的，如下圖所示：



42

下面的程式碼是修改自**ex11_2_1.java**，其中增加了一個AdvancedMath 介面。
AdvancedMath介面裡定義了3 個method：
(1) public void mod(int a, int b) // 計算a%b
(2) public void fac(int a); // 計算a!
(3) public void pow(int a, int b); // 計算a^b

ex11_3_1.java

請在Compute 類別裡撰寫所有method 的定義，使得我們可以利用Compute 類別來做加減乘除與mod()、fac()、pow() 等運算：

```
01 // ex11_9, 多重繼承的練習
02 interface Math
03 {
04 void show();
05 public void add(int a, int b);
06 public void sub(int a, int b);
07 public void mul(int a, int b);
08 public void div(int a, int b);
09 }
10 interface AdvancedMath
11 {
12 public void mod(int a, int b); // 計算a%b
13 public void fac(int a); // 計算a!
14 public void pow(int a, int b); // 計算ab
15 }
16 class Compute implements Math,AdvancedMath
17 {
18 // 請完成這個部分的程式碼
19 }
```

```
21 public class ex11_3_1
22 {
23 public static void main(String args[])
24 {
25 Compute cmp=new Compute();
26 cmp.mul(3,5);
27 cmp.show(); // 此行會回應 "ans=15" 字串
28 cmp.mod(14,5);
29 cmp.show(); // 此行會回應 "ans=4" 字串
30 cmp.fac(5);
31 cmp.show(); // 此行會回應 "ans=120" 字串
32 }
33 }
```

```
/* output----
ans=15
ans=4
ans=120
----------*/
```

43

- 在已經完成的ex11_3_1.java中，如果把第16行改寫成Compute 類別只實作Math 介面，而非同時實作Math與AdvancedMath介面，亦即把16行改寫成：

  16 class Compute implements Math, 那麼mod()、fac() 與pow() 這些method 是否還是可以正確執行？為什麼？

44

11.3 回家作業 多重繼承

hw11_3_1.java

假設有個CStu類別，其資料成員如下：

```
class CStu implements Data,Test {
  protected String id;   // 學號
  protected String name; // 姓名
  protected int mid;     // 期中考成績
  protected int finl;    // 期末考成績
  protected int common;  // 平時成績 }
```

```
/* output----------
學號:940001
姓名:Fiona
期中考成績:90
期末考成績:92
平時成績:85
學期成績:88.6
----------*/
```

下面的程式中，在介面Data裡已定義好一個showData()，用來顯示學生的學號及姓名。介面Test裡以定義好showScore()，用來顯示學生的各項成績；calcu() method則是江學期成績以期中、期末考佔30%，平時成績佔40%的方式計算。試完成下面的程式，使得輸出的項目，除了該生的資料之外，還要顯示學期成績。

```
interface Data
{
  public void showData();
}
interface Test
{
  public void showScore();
  public double calcu(); }
```

```
//請完成這部分的程式
public class hw11_11
{
  public static void main(String args[])
  {
    CStu stu=new CStu("940001","Fiona",90,92,85);
    stu.show();
  } }
```

45

11.3 回家作業 多重繼承

hw11_3_2.java

a. 介面**IAnimal**裡定義一個變數Name,初始值為Animal
b. 介面**Ipeople**裡定義一個變數Age,初始值為0，定義一個方法String Talk()
c. 類別NewBorn實作以上兩個介面，將Age設為1、Name設為Baby，Talk()傳回BraBra…，設計一方法Cry()可傳回其是否在哭泣
d. Main主程式內容為

```
public static void main(String[] args){
    NewBorn oTemp = new NewBorn();
    System.out.println(oTemp.Name);
    System.out.println(oTemp.Age);
    System.out.println(oTemp.Cry());
    System.out.println(oTemp.Talk());
    oTemp.Age = 2;
    oTemp.Name = "John";
    System.out.println(oTemp.Name);
    System.out.println(oTemp.Age);}
```

```
/* output----------
他的名字:Baby
今年0歲
他在哭嗎?true
他發出BraBra...的聲音
他的名字:John
今年2歲
----------*/
```

46

11-26

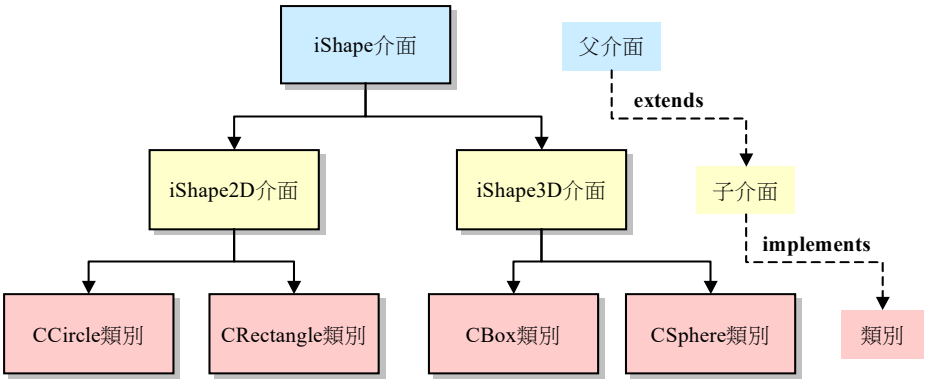
11.4 介面的延伸

- ✓介面可透過繼承的技術來衍生出新的介面。
- ✓原來的介面稱為基底介面（base interface）或父介面（super interface）。
- ✓衍生出的介面稱為衍生介面（derived interface）或子介面（sub interface）。

48

11-27

下圖中的iShape是父介面，透過關鍵字extends衍生出iShape2D與iShape3D子介面：



49

11-28

介面延伸的語法：

interface 子介面名稱 **extends** 父介面名稱1, 父介面名稱2,...
{
    … …
}

格式11.4.1
介面延伸的語法

50

下面是介面延伸的範例：

11-29

```
01  // app11_7, 介面的延伸
02  interface iShape  // 定義iShape介面
03  {
04    final double pi=3.14;
05    abstract void setColor(String str);
06  }
07
08  interface iShape2D extends iShape  // 定義iShape2D介面繼承自iShape
09  {
10    abstract void area();
11  }
12
13  class CCircle implements iShape2D  // 實作iShape2D介面
14  {
15    double radius;
16    String color;
17
18    public CCircle(double r)
19    {
20      radius=r;
21    }
22    public void setColor(String str) // 定義iShape介面的setColor()
23    {
24      color=str;
25      System.out.println("color="+color);
26    }
27    public void area()            // 定義iShape2D介面裡的area()
28    {
29      System.out.println("area="+pi*radius*radius);
30    }
31  }
```

```
/* app11_7 OUTPUT--
color=Blue
area=12.56
----------*/
```

```
32  public class app11_7
33  {
34    public static void main(String args[])
35    {
36      CCircle cir;
37      cir=new CCircle(2.0);
38      cir.setColor("Blue"); // 呼叫setColor() method
39      cir.area(); // 呼叫area() method
40    }
41  }
```

51

下面的範例修改自ex11_3_1.java，我們先宣告一個Show_ans 介面，再宣告另一個介面Math繼承它。請在Compute 類別裡撰寫所有method 的定義，使得我們可以利用Compute類別的物件來呼叫show()、add()、sub()、mul() 與div() 等運算。

ex11_4_1.java

```
01 // ex11_11,介面的延伸
02 interface Show_ans
03 {
04 public void show();
05 }
06 interface Math extends Show_ans
07 {
08 public void add(int a, int b);
09 public void sub(int a, int b);
10 public void mul(int a, int b);
11 public void div(int a, int b);
12 }
14 class Compute implements Math
15 { // 請完成這個部分的程式碼 }
19 public class ex11_11
20 {
21 public static void main(String args[])
22 {
23 Compute cmp=new Compute();
24 cmp.mul(3,5);
25 cmp.show();
26 }
27 }
```

```
/* output----
ans=15
----------*/
```

52

ex11_4_2.java

請參考課本圖11.4.1，於app11_7 中只完成了iShape 與iShape2D 介面，以及CCircle 類別的撰寫。請試修改app11_7，並依據下面的條件設計iShape3D 介面與CRectangle、CBox、CSphere 類別，使得app11_7 能完全符合圖11.4.1（設計完後，請於main() method裡測試之，所有的參數請自訂）：

(a) iShape3D 裡有一個abstract void volume() method，可用來顯示三維物件的體積。

(b) CRectangle 類別裡的資料成員有width、height 與color 成員，其中width 與heigh的型態為int，color 的型態為String。函數成員則有setColor() 與area() method。

(c) CBox 類別裡的資料成員有length、width、height 與color 成員，其中length、width 與heigh 的型態為int，color 的型態為String。函數成員則有setColor() 與volume() method （CBox 物件的體積為length*width*height）。

(d) CSphere 類別裡的資料成員有radius 與color 成員，其中radius 的型態為double，color 的型態為String。函數成員則有setColor() 與volume() method （CSphere物件的體積4*pi*radius*radius*radius/3）。

```
/* output----------
color=Blue, area=12.56
color=Red, area=50
color=Yellow, volume=48
color=Green, volume=4.1866666666666665
----------*/
```

53

11.4 回家作業 介面的延伸

hw11_4_1.java

下面的程式中，我們先宣告一個介面iVolume，再宣告一個抽象函數CSphere實作它。

```
interface iVolume
{
  public void showData();   // 顯示球體的資料
  public double vol();      // 計算球體積
}
abstract class CSphere implements iVolume
{
  final double PI=3.14;
  protected int x;
  protected int y;
}
class CCircle extends CSphere
{
 //請完成這部分的程式碼
}
```

```
public class hw11_13
{
  public static void main(String args[])
  {
    CCircle cir=new CCircle(8,6,2);
    cir.showData();
  }
}
```

```
/* output--------------
球心:(8,6)
半徑:2
球體積:33.49333333333333
---------------------*/
```

54

11.4 回家作業 介面的延伸

- hw11_4_2.java
- 1.建立一個計算機類別CCalculator，包含double 資料變數result（用來存放運算結果）。主程式中使用Math.random()函式亂數設值。讓計算機類別實作一個IBasicCompute 介面，該介面定義中包含兩個參數的四個方法Add,Sub,Mul,Div，代表加減乘除運算。
- 2.定義一個IAdvCompute 介面， 當中記載一個自然指數欄位e， 資料值為2.71828182845905。兩個方法，LOG(double x) 與LN(double x)，用來求以10 為底的對數及以自然指數為底的對數值。並使用CCalculator 類別同時實作IBasicCompute 介面與IAdvCompute 介面。
- 【註1】：LOG(double x) 的實作中，使用Math.log10(x) 求 以10 為底的對數。
- 【註2】：LN(double x) 的實作中，使用自行設計的LOG(double x) 求得解答，亦即ln(x)=log(x)/log(e)，LN(X)=LOG(x)/LOG(e)。
- 3.定義一個IFullCompute 介面，繼承上兩題的IBasicCompute 與IAdvCompute 介面，然後以CCalculator類別實作IFullCompute 介面。

```
/* output--------------
0.6179804110532139 Add 2.021274212766735 = 2.639254623819949
0.6179804110532139 Sub 2.021274212766735 = -1.403293801713521
0.6179804110532139 Mul 2.021274212766735 = 1.2491078688568482
0.6179804110532139 Div 2.021274212766735 = 0.3057380374963166
log(0.6179804110532139) = -0.20902529110265475
ln(0.6179804110532139) = -0.4812985193517129 (亂數設值，因此結果皆不盡相同)
---------------------*/
```

55