Java講義

JDK 下載網址

<http://java.sun.com>

<http://java.oracle.com>

Java官方講義

<http://docs.oracle.com/javase/tutorial/java/>

Eclipse 下載網址

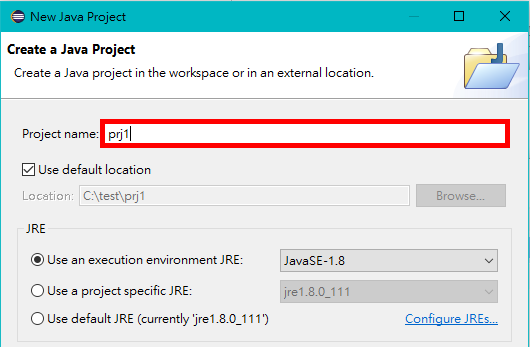
<https://www.eclipse.org/>

# Java程式語言初步

## Eclipse操作

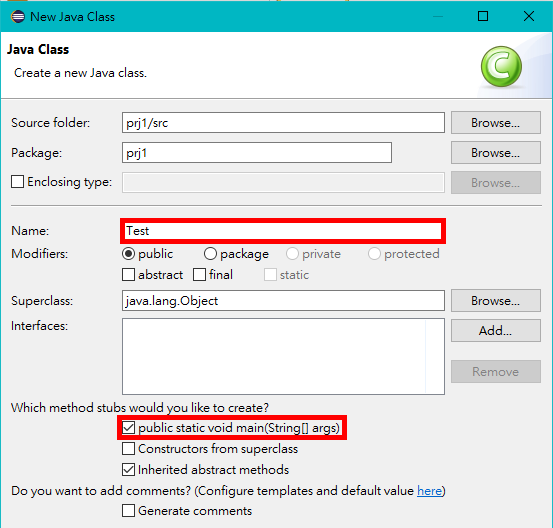
### 新增專案

開啟Eclipse，接著執行File/New Java Project新增專案，在下圖Project name輸入專案名稱如prj1。



### 新增類別

在Eclipse執行File/Class新增類別，在下圖Name輸入類別名稱如Test(類別名稱的第一個字母大寫是Java用字習慣)。勾選public static void main(String[] args)，使這個類別成為主類別。



### 第一個程式

public class Test {

public Test() {

}

public static void main(String[] args) {

System.out.println("test");

}

}

檔名：

原始檔案主檔名必須與類別名稱相同，副檔名為java；類別檔案的主檔名為類別名稱，副檔名為class。

問題：如何調整Eclipse程式碼字型大小?

答：利用功能表Window/Preferences/General/Appearance/Colors and Fonts/Java/Java Editor Text Font/Edit...

練習：

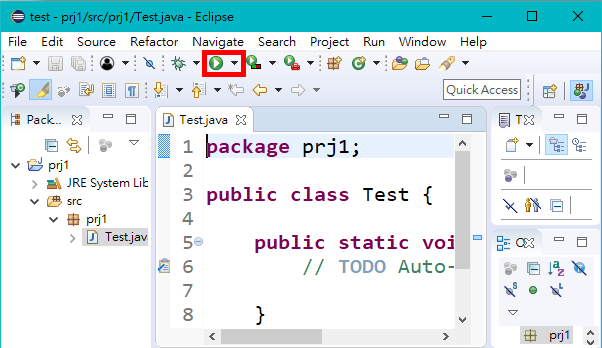
( ) 一個Java原始程式檔案只能有一個public class類別。

( ) Java的原始程式要經過編譯，產生副檔名為exe的執行檔，才可以執行。

( ) 一個Java原始程式檔案只能有一個類別。

### 執行程式

在Eclipse內執行Java程式，Eclipse會自動先編譯程式，如果沒有錯誤，就會執行程式。執行程式的方法是按功能表Run/Run 或按下圖的工具列圖示。這個程式沒有輸出任何資料，如果加上System.out.print()，就可以在視窗的下方看到主控台輸出結果。



## Java用字習慣

### 類別用字習慣

單字的第一個字母大寫，例如

System、Test、MyPhone、String、Thread、InputStream、OverChineseUniversity

### 常數用字習慣

每一個字元都是大寫，例如

Math.PI、Math.E

### 變數與方法用字習慣

第一個單字的第一個字母小寫，第二個單字以後的第一個字母大寫、其餘字母小寫

Math.sin、Math.sqrt、Math.copySign

練習：

1. 依照Java的用字習慣，下列何者可能是一個類別? ➊MyPhone ➋myPhone ➌MYPHOME ➍myphone (1)
2. 依照Java的用字習慣，下列何者可能是一個常數? ➊MyPhone ➋myPhone ➌MYPHOME ➍myphone (3)
3. 依照Java的用字習慣，下列何者可能是一個變數或方法? ➊MyPhone ➋myPhone ➌MYPHOME ➍Myphone (2)

## 變數

單行註解：

//這是註解

///這是註解

public class Test { //這是註解

多行註解：

/\* 這是註解

這是註解

\*/

練習：

●下列何者是錯誤的註解?

//這是註解

///這是註解

int a; ///這是註解//

int a; /\*int b;\*/

int a; /\*int b; //

資料型態：

|  |  |  |
| --- | --- | --- |
| 資料型態 | 長度 | 說明 |
| byte | 1 | 整數 |
| short | 2 | 整數 |
| int | 4 | 整數 |
| long | 8 | 整數 |
| float | 4 | 浮點數 |
| double | 8 | 浮點數 |
| boolean | 1 bit | 布林值 |
| char | 2 bytes | 字元Unicode |

數值表示方法

|  |  |  |
| --- | --- | --- |
|  | 範例 | 大小 |
| 2進位 | 0b10 | 2 |
| 8進位 | 070 | 56 |
| 10進位 | 10 | 10 |
| 16進位 | 0x10 | 16 |
| 浮點數 | 1.23f  1.23 | float  double |

變數宣告：

int a;

int b=5, c;

char d='a'; //有初始值

boolean e=true;

float f=1.3f;

String g="abc"; //參考型別資料

常用跳脫字元

|  |  |
| --- | --- |
| \n | 換行 |
| \t | 定位點 |
| \' | 單引號 |
| \" | 雙引號 |

範例：

System.out.println("我是\"王大明\"");//結果印出「我是"王大明"」

練習：

●寫出下列程式的執行結果?

int a=1, b=2, c=3;

a = b;

b = c;

c = a;

System.out.println(a);//

System.out.println(b);//

System.out.println(c);//

( ) 1.23f是屬於float型態的資料，1.23是屬於double型態的資料。

( ) 1.23d是屬於float型態的資料，1.23f是屬於double型態的資料。

( ) 1.23是屬於float型態的資料，1.23f是屬於double型態的資料。

# 運算子

## 算術運算子：

|  |  |
| --- | --- |
| + | 加或字串連接, 如"abc"+"de"結果為"abcde" |
| - | 減或負號 |
| \* | 乘 |
| / | 除 |
| % | 取餘數, 如5%3結果為2 |
| ++ | 加1  a++ 先做再加, System.out.println(a++); 先印出a再加1  ++a 先加再做, System.out.println(++a); 先將a加1再印出 |
| -- | 減1  a-- 先做再減  --a 先減再做 |

練習：

下列程式印出結果為何?

int a=5;

System.out.println(a+++3);

練習：

下列程式印出結果為何?

int a=5;

System.out.println(++a);

練習：

下列程式印出結果為何?

int a=5;

System.out.println(a--);

## 關係運算子

|  |  |
| --- | --- |
| == | 等於, 如(2==3); //false |
| > | 大於 |
| < | 小於 |
| >= | 大於等於 |
| <= | 小於等於 |
| != | 不等於 |

## 邏輯運算子

|  |  |
| --- | --- |
| || | 或, 一個為true就為true, 如true || false 結果為true |
| && | 且, 二個都要true才是true, 如true && false 結果為false |
| ^ | 互斥, 一個true另一個false才是true |

位元運算子：

|  |  |
| --- | --- |
| | | 或, 只要有一個位元是1就是1, 如3|5結果為7 |
| & | 且, 二個位元都是1才是1, 如3&5結果為1 |
| ^ | 互斥, 位元相同為0, 如3^5結果為6 |
| ~ | 補數, 1的補數, 0變1,1變0, 如~3結果為-4 |

練習：

●滿18歲的男生要服兵役，下列if敘述要判斷某人是否需要服兵役，在空格中填入適當的字。

int age; //年齡

boolean gender; //性別

//...

if (age gender )

System.out.print("要服兵役");

練習：

●某遊樂園滿65歲和未滿7歲的遊客可購買優待票，下列if敘述要判斷某人是否可購買優待票，在空格中填入適當的字。

int age; //年齡

//...

if (age age )

System.out.print("可購買優待票");

練習：

●TCP/IP網路遮罩用來判斷二個IP位址是否在同一個網路，遮罩有4個bytes，例如255.255.255.0，IP位址也有4個bytes，Java的int變數也有4個bytes，假設程式中有int ip1, ip2 2個IP，請在程式中自訂一個遮罩並給2個ip初值，在程式中判斷這二個IP是否在同一個網路，並用System.out.print印出判斷結果。

位元移位運算子：

|  |  |
| --- | --- |
| << | 位元左移,右邊補0 |
| >> | 位元右移,正數左邊補0,負數左邊補1,(正者恆正,負者恆負) |
| >>> | 位元右移,左邊補0 |

範例：

byte b = (byte)127; //0111 1111

b = (byte)(b << 1); //結果b等於-2

byte b = (byte)-64; //1100 0000

b = (byte)(b << 2); //結果b等於0

byte b=(byte)-3; //1111 1101

b = (byte)(b >> 2); //結果b等於-1

byte b=(byte)-1; //1111 1111

b = (byte)(b >>> 2); //結果b等於-1,因為會先轉成int再shift

//上述shift之後再強制轉成byte,不強制轉型編譯會有錯誤

int i=-1; //ffff ffff

i = (i >>> 2); //結果i等於0x3fffffff

int i=-1; //ffff ffff

i = (i >> 2); //結果i等於-1

運算後指定內容運算子

|  |  |
| --- | --- |
| += | 如a+=2; 與a=a+2;結果相同 |
| -= |  |
| \*= |  |
| /= |  |
| |= |  |
| &= |  |

## 三元運算子

(條件式 ? 條件式為真時傳回此結果 : 條件式為假時傳回此結果)

如 a = (3>2 ? 3 : 2); 結果a等於3

運算子優先順序

|  |  |
| --- | --- |
| 一元運算子 | v1++ v1--  ++v1 --v1 + - ! (type) |
| 算術運算子 | \* / %  + - |
| 關係運算子 | > >= < <=  == != |
| 邏輯運算子 | &&  || |
| 三元運算子 | ? : |
| 指定運算子 | = \*= /= += -= %= |

一二三算關邏

練習：

下列敘述的輸出結果為何?

## 資料轉型

自動轉型：資料型態相容且轉換後容量大於轉換前，運算式中如有不同型態資料，會先轉換成容量最大的資料型態再做運算。

int a = 1234;

short b = 1234;

byte c = 1234; //超出範圍, -128<= byte <=127, 編譯會有錯誤

a = b; //a的容量比b大, 沒有問題

b = a; //容量超出範圍, 編譯會有錯誤

float f;

f = a; //f的容量比a大, 沒有問題

強迫轉型，強迫將資料轉成指定的型態，若資料遺失，使用者須自行負責。語法如下：

(資料型態)數值或變數

例如

int a = 1234;

short b = 123;

byte c = (byte)129; //容量超出範圍, 強迫轉型,編譯沒問題, 結果c等於-127

b = (short)a; //容量超出範圍, 強迫轉型,編譯沒問題, 結果b等於1234

主控台輸出：

System.out.print(要印出的資料); //輸出後不換行

System.out.println(要印出的資料); //輸出後換行

System.out.printf("輸出資料轉換格式", 資料1, 資料2, ...);

System.out.format("輸出資料轉換格式", 資料1, 資料2, ...);

以上printf與format結果相同，可以互相替換。

範例：

int a=20;

System.out.print(a); //輸出後不換行

System.out.println(a); //輸出後換行

System.out.printf("姓名：%s 年齡：%d", "王大明", a);

double pi = Math.PI;

System.out.format("%f%n", pi); // --> "3.141593"

System.out.format("%.3f%n", pi); // --> "3.142"

System.out.format("%10.3f%n", pi); // --> " 3.142"

常用轉換格式：

|  |  |
| --- | --- |
| %s | 字串 |
| %d | 十進位整數 |
| %n | 換行 |
| %f | 浮點數 |

練習：

下列何者輸出資料後游標會換行?

➊ System.out.print("hello");  
➋ System.out.println("hello");  
➌ System.out.printf("hello");  
➍ System.out.format("hello");

下列敘述輸出的結果為何? 如果程式有錯就寫錯誤

System.out.print(3/2);

System.out.print(3/2.0);

System.out.print(3/(float)2);

下列敘述輸出的結果為何? 如果程式有錯就寫錯誤

byte b;

b = (byte)129;

System.out.print(b);

b = 129;

System.out.print(b);

練習：

寫出下列程式片段的執行結果，如果沒有輸出結果請寫「無」

char c='A';

int a=5, i=5, f=2;

System.out.print(c); (A)

System.out.print(i + "+" + i + "=" + (i+i)); (5+5=10)

System.out.print(f/i); (0)

System.out.print(3%2); (1)

System.out.print(a++); (5)

a+=2;

System.out.print(--a); (7)

System.out.print(\*\*a); (編譯錯誤)

System.out.print(//a); (編譯錯誤)

System.out.print(a>3 ? "A" : "B"); (A)

System.out.print((7>0) && (6<5)); (false)

System.out.print((7>0) || (6<5)); (true)

System.out.print(-3 >> 2); (-1)

System.out.print(3 >>> 2); (0)

System.out.print(3 >> 2); (0)

System.out.print(3 << 2); (12)

System.out.print(5 & 2); (0)

System.out.print(5 | 2); (7)

a=5;

while (a>5);

System.out.print(a--); (5)

a=7;

while (a>5)

System.out.print(--a); (65)

練習：

|  |
| --- |
| 下列程式片段在螢幕上印出的結果為何?  System.out.println(3+2);  答： |

練習：

|  |
| --- |
| 下列程式片段在螢幕上印出的結果為何?  System.out.println("3+2");  答： |

練習：

|  |
| --- |
| 下列程式片段在螢幕上印出的結果為何?  System.out.println(3+"2");  答： |

# 邏輯控制與輸入

## if-else

|  |  |
| --- | --- |
| if (條件式)  敘述; | 只有一行敘述可不加大括號 {}  敘述只有條件式為真時才執行 |
| if (條件式) {  敘述s;  } | 超過一行敘述需要加大括號 {}  else可有可無 |
| if (條件式) {  敘述1;  } else {  敘述2;  } | 條件式為真時執行敘述1否則執行敘述2 |
| if (條件式1) {  敘述1;  } else if (條件式2){  敘述2;  } | 多重條件 |
| if (條件式1) {  if (條件式2) {  敘述1;  } else {  敘述2;  }敘述3;  } else if (條件式3){  敘述4;  } | 巢狀if，if裡面還有if |

練習：

●寫出下列程式的執行結果

int i=3;

if (i>3)

System.out.print ('A');

else

System.out.print ('B');

答：

●寫出下列程式的執行結果

int liter=2;

if (liter >= 10) {

System.out.println("A");

}

else if (liter < 10) {

System.out.println("B");

}

else if (liter == 2) {

System.out.println("C");

}

答：

●寫出下列程式的執行結果

int i=2, j=6, k=4;

if ((i+j) > k)

if ((i+k) > j)

if ((j+k) > i)

System.out.println("A");

else

System.out.println("B");

else

System.out.println("C");

else

System.out.println("D");

答：

●試用if-else改寫下列敘述：

max=(a>b)?a:b;

●寫出下列程式片段的輸出結果，若無輸出結果則寫「無」?

|  |
| --- |
| int i=5;  if (i>3)  System.out.println("A");  答： |
| int i=5;  if (i<3)  System.out.println("A");  else  System.out.println("B");  答： |
| int i=5;  if (i<3)  System.out.println("A");  else  ;  System.out.println("B");  答： |
| int i=5;  if (i<3)  ;  System.out.println("A");  答： |
| int i=5;  if (i>3)  System.out.println("A");  else  System.out.println("B");  答： |
| int i=5;  if (i>3)  System.out.println("A");  else  ;  System.out.println("B");  答： |

## switch

switch(變數或運算式) {

case 值1:

敘述1;

break;

case 值2:

敘述2;

break;

...

default:

敘述n;

}

執行過程：

1.將switch括弧內的變數或運算式結果由上往下與case的值比較，如果相同就做case下面的敘述直到遇到break;

2.如果case下面沒有break;就會繼續執行下一個case的敘述。

3.如果所有的case都沒有符合，就執行default下面的敘述。

說明：

1.值1、值2的資料型態必須與switch括弧內的變數或運算式相同。

2.switch括弧內的變數或運算式必須是可轉換成整數的型態(如byte、short、int、char)、String、或列舉型態(enum)的資料，不可以是long、float、boolean等型態。例如

float a=3.14f;

long b =20;

String s = "abc";

switch (a) {//錯誤

}

switch (b) {//錯誤

}

switch (s) {//正確

}

範例：

int a = 3;

switch(a) {

case 1:

System.out.println(1);

break;

case 3:

System.out.println(3);

break;

default:

System.out.println(-1);

}

練習：

●寫出下列程式的執行結果

int num = 4;

switch(num){

case 1:

System.out.print("A");

break;

case 2:

System.out.print("B");

case 3:

System.out.print("C");

break;

default:

System.out.print("D");

}

答：

## Scanner

java提供Scanner類別以取得使用者從鍵盤或檔案輸入的資料，Scanner會依據空白字元、空白鍵、Tab，傳回下一個輸入，例如：

Scanner s=new Scanner (System.in);

System.out.print("請輸入一個句子:");//Good morning

System.out.println(s.nextLine());

System.out.print("請輸入一個整數:");//123

System.out.println(s.nextInt ());

System.out.print("請輸入二個整數，中間加空白:");//456 789

System.out.println(s.nextInt ());

System.out.println(s.nextInt ());

System.out.print("請輸入一單字(不含空白或Tab):");//abc

System.out.println(s.next ());

System.out.println("程式結束");

以下的例子可以從c:\test\readme.txt檔案讀入二行字，並輸出到主控台。

Scanner s;

try {

s = new Scanner(new File("c:\\test\\readme.txt"));

System.out.println(s.nextLine());

System.out.println(s.nextLine());

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

註：先以記事本編輯c:\test\readme.txt，參考內容

I like java.

I like ocu.

# 迴圈

## for迴圈

語法：

for (初始運算式; 條件式; 增量運算式) {

敘述;

}

1.初始運算式執行for迴圈前執行一次，條件式如果為真就會執行迴圈內的敘述，條件運算式通常不可省略，如果省略條件運算式且迴圈敘述又無判斷是否break; 就會變成無窮迴圈。

2.迴圈內可加上continue;或break; 敘述決定是否繼續或退出迴圈

for 迴圈流程圖：

條件式

初始運算式

敘述

增量運算式

Y

N

範例：

int i;

for (i=0; i<3; i++)

System.out.print(i); //輸出012

int sum=0;

for (i=0; i<3; i++)

for (int j=0; j<i; j++)

sum += j;

System.out.println(sum);//輸出1

for (int i=0; i<9; i++) {

if (i<5)

continue;

else {

System.out.println(i);

break;

}

}

練習：

●下列程式片段會依序印出5,4,3,2,1共5個整數，在空格中填入適當的字?

int i;

for( ; ; )

System.out.print(i);

●下列程式片段會依序印出1,2,3,4,5共5個整數，在空格中填入適當的字?

int i;

for( ; ; )

System.out.print(i);

●在下列右邊新迴圈用for迴圈改寫下列左邊原迴圈，限制新迴圈for括弧()內除了2個分號外，沒有任何程式碼

|  |  |
| --- | --- |
| 原迴圈 | 新迴圈 |
| int i;  for (i=0; i<3; i++)  System.out.println(i); | int i;  for (;;) {  } |

●寫一個程式計算1到100有幾個奇數可以被3整除?

●寫一個程式計算下列Fibonacci Sequence (0, 1, 1, 2, 3, 5, 8, ...)第20項是多少? 註：0是第1項、3是第5項。

int an=0, an\_1=1, an\_2=0, n=20;

for (int i=3; i<=n; i++) {

an = an\_1 + an\_2;

System.out.println(i + " " + an + " " +an\_1 + " "+ an\_2);

an\_2=an\_1;

an\_1=an;

}

使用遞迴會比較簡單，但遞迴需要使用較多堆疊記憶體，詳細請參考遞迴一節。

●寫一個程式計算下列Fibonacci Sequence (0, 1, 1, 2, 3, 5, 8, ...)前20項總和是多少? 註：0是第1項、3是第5項。

## while迴圈

條件成立時才執行迴圈內的敘述。

語法：

while (條件式) {

敘述;

}

●在下列右邊新迴圈用while迴圈改寫下列左邊原迴圈

|  |  |
| --- | --- |
| 原迴圈 | 新迴圈 |
| int i;  for (i=1; i<3; i++)  System.out.println(i); | int i;  while ( ) {  } |

練習：

●寫出下列程式的執行結果

int i=5;

while (i<8) {

System.out.println(i);

i++;

}

答：

●寫一個程式用迴圈檢查登入密碼，密碼正確才離開迴圈，輸入密碼可以是用Scanner類別，比較字串可以使用passwd.equals("123")，passwd是從鍵盤輸入的密碼，123是正確的密碼。

## do while迴圈

先執行迴圈一次再判斷條件式是否成立

語法：

do {

敘述;

} while (條件式); //注意最後要有一個分號

練習：

●寫出下列程式的執行結果

int i=5;

do {

System.out.println(i++);

} while (i<8);

答：

●寫出下列程式的執行結果

int i=5;

do {

System.out.println(i++);

} while (i<4);

答：

●若int i=5;下列哪一個迴圈至少會執行一次?

➊for ( ; i>6; i--) { } ➋for (; i>5; i++) { }➌while (i>6) { } ➍do { } while (i>6);

## break; 與continue;

break; 一次只能跳出一個迴圈，如果要跳出二層迴圈，break後面必須加上標籤，註明要跳出哪一個迴圈，例如下列程式：

int i, j, sum=0;

Label\_1: //標籤

for (i=0; i<5; i++)

for (j=0; j<5; j++)

if (i>2)

break Label\_1;

System.out.print(i); //輸出3

練習：

●寫出下列程式的執行結果

int i=5;

while (i<8) {

System.out.println(i++);

if (i==6)

continue;

System.out.println("test");

}

答：

●寫出下列程式的執行結果

int i=5;

while (i<8) {

System.out.println(i++);

if (i==6)

break;

System.out.println("test");

}

答：

●寫出下列程式片段的輸出結果，若無輸出結果則寫「無」?

|  |  |
| --- | --- |
| int i=5;  while (i<8) {  System.out.println(i);  i++;  }  答： | int i=5;  while (i<8) {  i++;  System.out.println(i);  }  答： |
| int i=5;  while (i<8) {  System.out.println(i++);  if (i==6)  continue;  }  答： | int i=5;  while (i<8) {  System.out.println(i++);  if (i==6)  break;  }  答： |
| int i=5;  while (i<8) {  System.out.println(i++);  }  答： | int i=5;  do {  System.out.println(i++);  } while (i<8);  答： |
| int i=5;  while (i<8) {  System.out.println(i++);  }  System.out.println(i++);  答： | int i=5;  do {  System.out.println(i++);  } while (i<8);  System.out.println(i++);  答： |

●寫出下列程式的輸出結果?

int i, j, sum=0;

Label\_1: //標籤

for (i=0; i<5; i++)

for (j=0; j<5; j++)

if (i>2)

break Label\_1;

System.out.print(i);

●寫出下列程式的輸出結果?

int i, j, sum=0;

for (i=0; i<5; i++)

for (j=0; j<5; j++)

if (i>2)

break;

System.out.print(i);

# 陣列與字串

## 一維陣列

陣列是將型態相同的資料放在一塊連續的記憶體，再用索引存取資料。

陣列在宣告時不可以指定元素個數，如下畫X的部分是錯誤的寫法。如果有給初始值，元素個數由初始值的個數決定。

int a[];

a=new int[3];//(O)

int b[3]; //(X)

int c[3]={1,2,3}; //(X)

int d[]={1,2,3}; //(O)

int e[]=new int[]{1,2,3}; //(O)

int e1[]=new int [3]{1,2,3}; //(X)

int[] f; //(O)

int []g; //(O)

int[] h=new int[5]; //(O)

陣列的索引從0開始，且是連續的，如a[0]表示第一個元素。

陣列通常會搭配迴圈來操作，例如

int a[]={1,2,3};

for (int i=0; i<a.length; i++)

System.out.println(a[i]);

以下程式使用for-each迴圈操作陣列。

int a[]={1,2,3};

for (int i:a)//i是a的一個元素

System.out.println(i);

陣列int a[]=new int[3]; 的示意圖如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a |  | a[0] | a[1] | a[2] |

陣列元素交換常常用在排序，正確的陣列元素交換：

int a[]={1,2,3}, tmp;

tmp = a[1];

a[1]=a[2];

a[2]=tmp;

錯誤的陣列元素交換：

int a[]={1,2,3};

a[1]=a[2];

a[2]= a[1];

java提供Arrays類別簡化陣列的操作，例如常用的排序、搜尋等，都可以用Arrays類別輕易完成，更多Arrays的方法可以參考java有關Arrays的說明文件。以下程式使用Arrays.sort(陣列);對陣列做遞增排序：

int a[]={1,5,2,4,3};

Arrays.sort(a);

for (int i=0; i<a.length; i++) {

System.out.println(a[i]);

}

使用Arrays類別必須在第一個類別的前面加上下列匯入套件的敘述，

import java.util.Arrays;

或者是使用下列完整套件路徑呼叫Arrays的方法，否則編譯會有錯誤。

java.util.Arrays.sort(a);

以下程式使用Arrays.equals(陣列1, 陣列2);比較二個陣列是否相等，即元素個數與每一個元素是否都相等：

int a1[]= {1,2,3};

int a2[]= {1,3,2};

if (Arrays.equals(a1, a2))

System.out.println("陣列相等");

else

System.out.println("陣列不相等");

練習：

●寫出下列程式的執行結果

int a[]={5,6,7,9};

System.out.println(a.length); //答：

System.out.println(a[0]); //答：

a = new int [3];

System.out.println(a.length); //答：

System.out.println(a[0]); //答：

●下列程式片段執行後，a[0]等於多少? c[0]等於多少? b[0]等於多少?

int a[]={4,5,6},b[]={1,2,3}, c[]={7,8,9};

c=a;

a=b;

b=c;

●寫一個程式找出下列score陣列中，印出不及格的最高分為幾分、這個分數是在陣列中的哪一個元素(印出陣列索引)?

int score[]={80,70,50,55,60,90};

## 二維陣列

二維陣列有二個索引，但不一定是一個矩形的方陣，例如

int a[][]={{1,2,3},{4,5}};//有5個元素,非矩形方陣

int b[][]=new int[2][3]; //有6個元素,是矩形方陣

int c[][]=new int[3][];

c[0] = new int[2];

c[1] = new int[3];

c[2] = new int[4]; //c共有9個元素

int []d[]={{1,2,3},{4,5}};//(O)

int [][]e={{1,2,3},{4,5}};//(O)

上述程式中

c.length等於3

c[0].length等於2

c[1].length等於3

c[2].length等於4

上述陣列c 的示意圖如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| c |  | c[0] |  | c[0][0] | c[0][1] |  |  |
|  |  | c[1] |  | c[1][0] | c[1][1] | c[1][2] |  |
|  |  | c[2] |  | c[2][0] | c[2][1] | c[2][2] | c[2][3] |

操作陣列常常會用到二層迴圈，例如

for (int i=0; i<a.length; i++)

for (int j=0; j<a[i].length; j++)

System.out.println(a[i][j]);

一維陣列可以使用for-each迴圈，二維陣列不可以使用for-each迴圈。

int a[][]={{1,2,3},{4,5}};

for (int i:a[0]) {//(O)

System.out.println(i);

}

for (int i:a[1]) {//(O)

System.out.println(i);

}

for (int i:a) {//(X)

System.out.println(i);

}

練習：

●寫出下列程式的執行結果

int a[][]={{1,2,3},{4,5}};

System.out.println(a.length); //答：

System.out.println(a[0].length); //答：

System.out.println(a[0][0]); //答：

a = new int[6][7];

System.out.println(a.length); //答：

System.out.println(a[1].length); //答：

System.out.println(a[0][0]); //答：

a[0] = new int [8];

System.out.println(a.length); //答：

System.out.println(a[1].length); //答：

System.out.println(a[0][0]); //答：

●以下是資訊系同學的成績，寫一個程式計算每個同學的平均分數，並印出每位同學的平均分數。

|  |  |  |  |
| --- | --- | --- | --- |
| 座號 | 國文 | 英文 | 數學 |
| 1 | 75 | 85 | 90 |
| 2 | 60 | 80 | 75 |
| 3 | 80 | 90 | 60 |

●寫一個程式計算下列二維陣列二個對角線的元素總和?

●寫一個程式計算下列二維陣列對角線右上方三角形區域(不含對角線元素)的元素總和?

int a[][]=

{{1,2,3,2,1},

{2,3,4,3,2},

{3,4,5,4,3},

{4,5,6,5,4},

{5,6,7,6,5}};

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 2 | 1 |
| 2 | 3 | 4 | 3 | 2 |
| 3 | 4 | 5 | 4 | 3 |
| 4 | 5 | 6 | 5 | 4 |
| 5 | 6 | 7 | 6 | 5 |

練習：

( )二維陣列一定是一個矩形的方陣，即每一個橫列的元素個數相同，每一個直欄的元素個數也相同。

## 字串

String是java常用處理字串的類別，字串的宣告如下：

String str="hello";

String fruit[]={"Apple", "banana", "cherry"};

String類別提供很多字串操作所需要的方法，例如字串長度、字串比較、字串轉數字等方法。

以下的程式可以計算以Unicode為單位的字串長度，計算字串長度時中文、英數字不分全形半形，一個字都算一個單位。

String str1="abc";

String str2="你好嗎";

System.out.println(str1.length()); //印出3

System.out.println(str2.length());//印出3

System.out.println("很好!".length());//印出3

以下的程式可以比較二個字串是否完全相等。

String str1="abc";

String str2="Abc";

System.out.println(str1.equals("abc"));//印出true

System.out.println(str1.equals(str2));//印出false

System.out.println(str1.equals("abcd"));//印出false

String fruit[]={"Apple", "banana", "cherry"};

System.out.print(fruit.length); //元素個數

System.out.print(fruit[0].length()); //字串長度

練習：

( )String s="abc"; s.length等於多少? ➊0 ➋1 ➌3 ➍錯誤

( )String s="abc"; s.length()等於多少? ➊0 ➋1 ➌3 ➍錯誤

# 方法

## 方法概論

Java是百分之百物件導向程式語言，方法與資料都是類別成員，每一個方法都要寫在類別裡面，不可以寫在類別外面。

方法範例：

void f1 () { //方法不可以寫在類別外面

}

public class Test {

void f2 () {

}

}

方法是類別中的一段程式，每一個方法都有一個方法名稱，方法名稱是一個自定字，定義方法的語法如下：

可見範圍 是否static 傳回值型態 方法名稱 (參數) throws 例外名稱{

敘述;

return 值;

}

以main方法為例

public static void main(String[] args) {

...

}

其中

|  |  |
| --- | --- |
| public | 是可見範圍，不同套件的類別也可以看到 |
| static | 表示靜態方法，不須建立物件，可直接使用類別名稱呼叫；若省略static，必須建立物件，才能呼叫 |
| void | 表示沒又傳回值，若不是void，在方法中要用return 值; 將值傳回呼叫者 |
| String[] args | 表示參數是一個字串陣列 |
| throws 例外名稱 | 這個方法可能會丟出例外，main方法沒有丟出例外 |

方法結束的時機：

1.程式已執行到右大括弧}

2.沒有傳回值的方法執行到return;

3.有傳回值的方法執行到return 值;

如何呼叫方法：

1.被呼叫的方法前面有加static，則可以用類別名稱直接呼叫，不須建立物件，例如開根號

Math.sqrt(4);

2.被呼叫的方法前面沒有加static，則必須先建立物件再用物件呼叫

public class Test {

public static void main(String[] args) {

//不可在static方法中呼叫非static方法

System.out.println(sum(10)); //錯誤

}

int sum(int n) {

if (n == 1)

return 1;

else

return n + sum(n - 1);

}

}

public class Test {

public static void main(String[] args) {

//在static方法中可直接呼叫同樣是static方法

System.out.println(sum(10)); //正確

}

static int sum(int n) {

if (n == 1)

return 1;

else

return n + sum(n - 1);

}

}

public class Test {

public static void main(String[] args) {

Test test = new Test (); //建立物件

System.out.println(test.sum(10)); //正確, 用物件呼叫

}

int sum(int n) {

if (n == 1)

return 1;

else

return n + sum(n - 1);

}

}

3.呼叫同一個類別的方法

在static方法內呼叫非static方法，必須先建立物件再用物件呼叫，否則直接呼叫方法即可。例如下列程式，f2是static方法，可以直接呼叫，f1不是static方法，必須建立物件後再用物件呼叫。

public class Test {

public static void main(String[] args) {

Test t = new Test (); //建立物件

t.f1 (); //使用物件呼叫

f2 (); //f2是static方法，可直接呼叫

}

void f1 () {

}

static void f2 () {

}

}

public class Test {

public static void main(String[] args) {

}

void f1 () {

f2 (); //非static 呼叫非static

return; //返回，不傳回返回值

}

int f2 () {

return 0; //返回並傳回返回值

}

}

可見範圍請參考資料/方法可見範圍一節

練習：

寫一個方法ctof將攝氏溫度轉為華氏溫度，在main()呼叫這個方法印出傳回的華氏溫度。

練習：

寫一個static方法ctof將攝氏溫度轉為華氏溫度，在main()呼叫這個方法印出傳回的華氏溫度。

再寫一個non-static方法ftoc將ctof求得的華氏溫度轉為攝氏溫度，在main()呼叫這個方法印出傳回的攝氏溫度。

## 方法丟出例外

對於可能丟出例外(Exception)的方法，在方法名稱的右邊要加上throws關鍵字，提醒呼叫這個方法的程式要處理例外，有關丟出例外的介紹，請參考Exception例外一節。

## 方法多載

Java允許在同一個類別中有多個名稱相同的方法，只要這些方法的參數個數不同，或參數個數相同，但參數的型態不同，這種技術稱為多載(Overloading)，有關方法多載的介紹請參考，「Overloading多載」一節。

## 遞迴

遞迴(Recursive)是在一個方法中有條件的呼叫自己，例如要計算1+2+3+...+50，可以用迴圈也可以用遞迴，以下的例子是用遞迴計算1+2+3+...+n。

int sum(int n) {

if (n == 1)

return 1;

else

return n + sum(n - 1);

}

在方法中呼叫自己必須是有條件的，當條件成立時才呼叫自己，否則方法就要return，如果無條件的呼叫自己會造成堆疊overflow，因為每次呼叫方法時，都要把返回位址推入堆疊，以便將來返回正確的位址。

遞迴可以解決很多問題，讓程式簡化，例如著名的漢諾瓦塔(Hanoi Tower)與費氏(Fibonacci)級數，都是可以用遞迴輕易解決的問題。

練習：

●費氏(Fibonacci)級數每一項都是前二項的和，即F(n)=F(n-1)+F(n-2)，假設F(0)=0、F(1)=1，寫一個程式印出F(8)等於多少?

註：使用遞迴計算費式級數時，每呼叫一次F(n)就會呼叫1次F(n-1)和1次F(n-2)，如下圖所示，呼叫函數的次數約等於2n，時間複雜度為指數級O(2n)，因此n不可以太大，當n等於34時，呼叫次數就會趨近於232≅4G，有可能造成記憶體overflow。

...

F(n-1)

F(n-2)

F(n-2)

F(n-3)

F(n-3)

F(n-4)

F(n-2)

F(n-3)

F(n-2)

F(n-3)

F(n-4)

F(n-5)

F(n-5)

F(n-6)

F(n)

# 類別

## 概述

1.以class關鍵字定義的區塊稱為類別，類別中可能有資料(或稱屬性)與方法。

2.類別是一段尚未載入記憶體的程式，建立物件後才會將程式載入記憶體，同時配置記憶體給類別裡面的資料，並產生物件在記憶體的參照，以後就可以用這個物件的參照操作物件。

3.類別是一個模型，物件是用類別建立出來的實體，擁有自己的方法與屬性。

4.透過繼承(extends)可以讓子類別使用擴充、改寫父類別的方法，讓程式碼重複使用。

5.每一個java原始檔案只能有一個public class，且class name must be the same with file name(一個java原始檔案不能有二個public class)，主檔名與類別名稱的大小寫都要相同。

6.public class可以被不同package的class所參考

7.沒有加public的class只能被同一個package的class參考

8.public class classA 與 class classB在同一個檔案，public class classC在另一個檔案，如果這二個檔案都不指定package，classA和classC都可以繼承classB或new classB();

9.續上，如果在classA的檔案不指定package，在classC的檔案指定package C，在classA可以參考classB，在classC不可以參考classB(因為classB與classC屬於不同的package，且classB沒有使用public modifier)

10.續上，但classC可以參考classA(因為classA有使用public modifier)

11.class的modifier不可以使用private或protected

12.private class classD { ... }是錯誤的

13.protected class classD { ... }是錯誤的

14.Java沒有多重繼承，一次只能繼承一個類別，class classD extends classE, classF { ... }是錯誤的

練習：

●下列類別語法是否正確? (每個儲存格的程式都是在同一個檔案中)

|  |  |
| --- | --- |
| class A { } | protected class A { } |
| private class A { } | public class A { } |
| class A { }  class B extend A { } | class A { }  class B extends A { } |
| class A { }  class B { }  class C extends A, B { } | class A { }  class B extends A{ }  class C extends B { } |
| public class A { }  public class B { } | class { } |

●下列程式存在A.java的檔案中是否正確?

public class A { }

●下列程式存在A.java的檔案中是否正確?

class A { }

class B { }

●下列程式存在C.java的檔案中是否正確?

class A { }

class B { }

## 物件

假設Test是一個類別，下列程式可以建立一個物件，這個物件有自己的方法與屬性，t是物件的參照(reference)。

Test t=new Test();

物件可以存取的方法與屬性要看方法與屬性的可見範圍，在資料/方法可見範圍中有詳細的說明。

## 常用類別

### Math

Math類別包含許多數字運算的靜態(static)方法，如三角函數、平方根、絕對值等方法。靜態方法不需要建立物件，可用類別直接呼叫。Math類別前面有加上final(如下)，因此無法被繼承，有關final關鍵字的說明請參見"final類別與資料"一節。

public final class Math extends Object {

...

}

常用方法如下；

|  |  |  |
| --- | --- | --- |
| 常用方法 | 範例 | 說明 |
| Math.PI | Math.PI | 圓周率，大約等於3.14159 |
| Math.E | Math.E | 自然對數的底*e*，大約等於2.71828 |
| Math.abs | Math.abs(-5) | 傳回絕對值，本例為5 |
| Math.floor | Math.floor(3.2) | 傳回小於參數的最大整數，本例等於3 |
| Math.sin | Math.sin(Math.PI/2) | 傳回sin三角函數，參數為弧度，不是度數°，本例Math.sin(π/2)=1.0 |
| Math.log | Math.log(3) | 以e為底的對數，本例大約等於1.0986 |
| Math.log10 | Math.log10(10) | 以10為底的對數，本例等於10 |
| Math.max | Math.max(5,3) | 傳回2數較大者，本例等於5 |
| Math.pow | Math.pow(5,3) | 傳回指數，本例等於53=125 |
| Math.random | Math.random() | 傳回一個介於0.0到1.0的亂數 |
| Math.round | Math.round(3.5) | 傳回四捨五入到整數，本例等於4 |

練習：

●依照Java的用字習慣，Math.PI可能是下列哪一個的名稱? ➊類別 ➋方法 ➌常數 ➍變數

●依照Java的用字習慣，Math.random可能是下列哪一個的名稱? ➊類別 ➋方法 ➌常數 ➍變數

●依照Java的用字習慣，Math可能是下列哪一個的名稱? ➊類別 ➋方法 ➌常數 ➍變數

●樂透開獎每次會產生6個1到42不重複的數字，請設計一個可以用亂數產生樂透開獎的程式，印出這6個不重複的數字。

參考解答：

public class Test {

static int n=10;

static int index; //位置

static int random; //抽到的號碼

static int a[]=new int [n];

public static void main(String[] args) {

// TODO Auto-generated method stub

for (int i=0; i<n; i++)

a[i] = i+1;

for (int i=n; i>n-6; i--) {

double f = Math.random();

index = (int)(f\*i + 1);

System.out.print(f+"\t");

random=a[index-1];

a[index-1] = a[i-1];

printOut();

}

}

static void printOut() {

System.out.print(index + " " +random+"\t");

for (int i=0; i<n; i++)

System.out.print(a[i]+" ");

System.out.print("\n");

}

}

### String

String是java常用處理字串的類別，String類別前面有加上final(如下)，因此無法被繼承。

public final class String extends Object {

...

}

常見的字串操作如下：

System.out.println("abc");

String cde = "cde";

System.out.println("abc" + cde); //字串連接

String c = "abc".substring(2,3); //傳回子字串

String d = cde.substring(1, 2); //傳回子字串

字串常用方法如下；

|  |  |  |
| --- | --- | --- |
| 常用方法 | 範例 | 說明 |
| charAt | "ABC".charAt(1) | 傳回字串指定位置的字元，本例為B |
| substring | "abc".substring(2,3) | 傳回子字串，本例傳回從2到3-1=2位置的子字串c |
| equals | String str = "abc"  str.equals("xyz") | 傳回2個字串是否相等 |
| toLowerCase | "aBD". toLowerCase() | 傳回小寫字母的字串 |
| length | "abc".length() | 傳回字串長度，本例為3 |
|  | String str = "aBD"  str.compareTo("abc") | 比較二個字串，相等傳回0，參數字串較大者傳回負數，參數字串較小者傳回正數， |

練習：

假設String str = "abc"，設計一個程式讓使用者從鍵盤輸入一個字串，如果使用者輸入的字串是"abc"，就顯示輸入正確的訊息，並結束程式，否則再次提示使用者輸入字串。

## 繼承

類別繼承可以讓子類別使用父類別的方法或屬性，子類別不必從無到有重新撰寫方法或屬性，讓父類別的方法可以再使用(reuse)，類別繼承的寫法如下列程式，其中A是子類別，B是父類別。在這個例子中，子類別A可以使用父類別B的屬性age和方法f()，如果A覺得B的f()方法不好，A可以在自己的類別中再寫一個方法f()取代B的f()方法，這叫做改寫或覆寫(Override)，稍後有一個單元會詳細介紹。A如果有些方法或屬性不想讓B繼承，可以在這些方法或屬性前面加上private，這樣子類別就無法使用這些方法或屬性，如下列例子中的secret屬性。

class B {

int age;

private String secret;

void f () { }

}

class A extends B {

}

練習：

●下列程式是否正確?

class B {

private String secret;

}

class A extends B {

void a () { secret = "abc"; }

}

●下列程式是否正確?

class B {

int age;

}

class A extends B {

void a () { age = 20; }

}

●下列程式是否正確?

class B {

void f () { }

}

class A extends B {

void f () { int age = 20; }

}

子類別一次只能繼承一個父類別，不可以一次同時繼承多個父類別(多重繼承)，例如：

class C { }

class B { }

class A extends B, C { } //錯誤，不可多重繼承

class B extends C { }

class A extends B { } //正確，依序繼承

練習：

BMI(Body Mass Index)可以計算一個人的身體質量指數，公式是

BMI = w/(h\*h)

其中w是體重、h是身高，體重和身高的單位分別是公斤和公尺。

寫一個類別MyMath如下，類別中有一個bmi()方法可以計算BMI。

class MyMath {

//在此撰寫計算BMI的方法

}

將上述類別存在第一個檔案中。

再寫一個類別MyNewMath如下，類別中有一個ftoc的方法可以計算華氏轉攝氏溫度，ftoc()方法的參數是華氏溫度，傳回值是攝氏溫度，還有一個ctof()方法可以計算攝氏轉華氏溫度，ctof()方法的參數是攝氏溫度，傳回值是華氏溫度，MyNewMath繼承MyMath。

class MyNewMath extends MyMath {

/\*在此撰寫ftoc()方法計算華氏轉攝氏溫度\*/

/\*在此撰寫ctof()方法計算攝氏轉華氏溫度\*/

}

將上述二個類別存在第二個檔案中。

在第三個檔案中寫一個主類別，類別中有main()方法，在main方法中分別呼叫ctof()、ftoc()、bmi()三個方法，並印出執行結果，身高、體重、溫度請自行設定。本作業限制在第三個檔案中只能使用MyNewMath類別，不可以在程式中出現MyMath類別，完成後將三個程式檔以附檔方式上傳。

注意觀察類別繼承的作用

## super與this

super是指父類別，this是指本身類別，子類別繼承父類別，子類別和父類別如果有名稱相同的資料或方法時，子類別在資料或方法前面加上super.就可以指定存取父類別的資料或方法，而存取子類別自己的資料或方法時可以加上this.或不加也可以。

public class Test extends A {

int age = 18;

public Test() {

}

public static void main(String[] args) {

Test t = new Test ();

t.super\_this ();

}

void super\_this () {

System.out.println(age); //印出18

System.out.println(this.age); //印出18

System.out.println(super.age); //印出50

}

}

class A {

int age = 50;

}

子類別可以用super (); 呼叫父類別沒有參數的建構元，或用super(參數); 呼叫父類別有參數的建構元，有關建構元的詳細說明請參考建構元一節，

練習：

●子類別透過哪一個關鍵字可以呼叫父類別的建構元(Constructor)? ➊super ➋final ➌static ➍this

●子類別繼承父類別的關鍵字為何? ➊super ➋extends ➌inherit ➍override

●寫出下列程式的執行結果(注意：每一個println都會執行二次)：

public class Test extends A {

int age = 18;

public static void main(String[] args) {

Test t = new Test ();

Test t2 = new Test ();

t.age = 19;

t.super\_this ();

t2.super\_this ();

}

void super\_this () {

System.out.println(age); //答： ,

System.out.println(this.age); //答： ,

System.out.println(super.age); //答： ,

super.age = 60;

}

}

class A {

int age = 50;

}

## package與import的意義

package是指定類別所屬的套件，相當於資料夾，package必須位於程式第一行

import是匯入程式中用到的套件或類別，即告訴JDK到哪裡去找要用到的套件，二個人可以寫名稱相同的套件，因為他們放的資料夾不同，所以不會造成混淆。

package與import二者都是寫在類別外部。

程式中如果寫

package A;

類別檔就會輸出到資料夾A

程式中如果寫

package A.B;

類別檔就會輸出到資料夾A\B

範例

package prj1;

import javax.swing.JButton; //匯入JButton類別

public class Test {

public static void main(String[]arg) {

JButton jb1 = new JButton("按一下");

}

}

練習：

●下列程式是否正確?

class A { }

package Hello;

●下列程式是否正確?

import javax.swing.JButton;

package Hello;

class A { }

# Constructor建構子

1.Constructor是與類別名稱相同的method

2.建立物件時，會自動呼叫物件的Constructor，通常作為物件的初始化

3.Constructor可以overloading

4.可以在類別中某一個Constructor使用this(引數);在建構元中呼叫另一個Constructor，不可以直接以建構元名稱呼叫，且this要寫在Constructor的第一行

4.1子類別可以在建構子中以super()或super(參數)分別呼叫父類別無參數或有參數的建構子，但super()或super(參數)應寫在子類別建構子的第一行。

5.可以是public, private或無modifier，private Constructor只能在類別內以this(引數); 呼叫，無法在類別外被呼叫

6.Constructor可以省略，java會自動使用預設的Constructor(沒有引數也不作任何事，馬上return)

7.沒有傳回值

public ClassA () { ... } //最正確寫法

ClassA() { … } //正確

public void ClassA () { ... } //錯誤

public int ClassA () { ... return 5;} //錯誤

private ClassA () { ... } //只限內別內部呼叫，如this();

int a=new ClassA(); //錯誤

ClassA a=new ClassA(); //正確

8.以子類別建立物件時，會呼叫父類別哪一個Constructor?

答：父類別沒有參數的建構元

9.constructor不能被子類別繼承

10.如果子類別沒有指定父類別的建構子，父類別必須定義沒有參數的建構元，或完全不定義任何建構元，以下的程式是錯誤的

class Test2 {

public Test2(int i){ //錯誤，缺少無參數的建構元

System.out.println("test2-2");

}

}

class Test3 extends Test2 {

public Test3( int i ){

System.out.println(i);

}

}

以下程式是正確的

class Test2 {

public Test2(int i){ //正確，子類別有指定父類別的建構元

System.out.println("test2-2");

}

}

class Test3 extends Test2 {

public Test3( int i ){

super(i);

System.out.println(i);

}

}

以下程式是正確的

class Test2 {

Test2 () { }

public Test2(int i){

System.out.println("test2-2");

}

}

class Test3 extends Test2 {

public Test3( int i ){

System.out.println(i);

}

}

以下程式是正確的

class Test2 {

//沒有建構元

}

class Test3 extends Test2 {

public Test3( int i ){

System.out.println(i);

}

}

練習：

●下列程式是否正確?

class B {

int age;

B() { age = 20; }

B(int myAge) { age = myAge;}

}

class Test extends B {

Test() {

age = 22;

super(18);

}

}

●寫出下列程式的執行結果

class B {

int age;

B() { age = 20; }

B(int myAge) { age = myAge;}

}

class Test extends B {

public static void main(String[] arg) {

Test a = new Test();

System.out.print(a.age);

}

Test() {

super(18);

age = 22;

}

}

●寫出下列程式的執行結果(注意列印資料的先後順序)：

public class Test extends Test2{

public static void main(String[] args) {

Test2 a = new Test2(3); //呼叫參數為int的建構子

Test2 b = new Test2();//呼叫無參數的建構子

}

}

class Test2 {

public Test2(int i) {

System.out.print(i-1);

}

public Test2() {

System.out.print(4);

}

}

答：

●寫出下列程式的執行結果(注意列印資料的先後順序)：

public class Test extends Test2{

public static void main(String[] args) {

Test2 a = new Test2(3);

Test2 b = new Test2();

}

}

class Test2 {

public Test2(int i) {

System.out.print(i-1);

}

public Test2() {

this(5); //呼叫自己有參數的建構子

System.out.print(4);

}

}

答：

●寫出下列程式的執行結果(注意列印資料的先後順序)：

public class Test extends Test2{

public static void main(String[] args) {

Test2 a = new Test2(3);

}

}

class Test2 {

public Test2(int i) {

this();

System.out.print(i-1);

}

public Test2() {

System.out.print(4);

}

}

答：

●寫出下列程式的執行結果(注意列印資料的先後順序)：

public class Test extends Test2{

public static void main(String[] args) {

Test2 a = new Test2();

}

}

class Test2 extends Test3 {

public Test2(int i) {

this();

System.out.print(i-1);

}

public Test2() {

super();

System.out.print(4);

}

}

class Test3 {

public Test3 (int a) {

System.out.print(5);

}

public Test3 () {

System.out.print(6);

}

}

答：

●寫出下列程式的執行結果(注意列印資料的先後順序)：

public class Test {

public static void main(String[] args) {

Test2 a = new Test2();

}

}

class Test2 extends Test3 {

public Test2(int i) {

this();

System.out.print(i-1);

}

public Test2() {

System.out.print(4);

}

}

class Test3 {

public Test3 (int a) {

System.out.print(5);

}

public Test3 () {

System.out.print(6);

}

}

答：

# 類別進階

## abstract class抽象類別

●抽象類別前面必須加上abstract。

●抽象類別可以含有抽象方法，一般類別不可以有抽象方法。

●一般類別A繼承抽象類別B必須實作抽象類別B中的抽象方法。

●抽象類別C繼承抽象類別B，抽象類別C可不實作抽象方法。

●抽象類別不一定要有抽象方法，但這種抽象類別沒有意義。

●抽象類別和interface一樣可提供子類別與父類別溝通的管道。本節最後有一個例子說明子類別如何與抽象類別的父類別溝通。

abstract class可以有一般method和abstract method

abstract class只能被繼承，不可以用abstract class 建立物件

class classB

{

void funb(){

classA c=new classA (); //錯誤，不可以new抽象類別

classA c=new classA; //錯誤

classA c[]=new classA[2]; //正確

c[0]=new 子類別(); //正確,少用

}

}

abstract class classA

{

abstract void funa();

}

abstract class的子類別必須改寫(override)父類別的abstract method

普通類別不可以有abstract method, abstract class才可以有abstract method

abstract class classA //正確

{

abstract void funa();

}

class classA

{

abstract void funa(); //錯誤，因為不可有abstract method

}

abstract method不可以有body

abstract class classA

{

abstract void funa()

{

int i=5; //錯誤，abstract method不可以有body, {}要刪除

}

}

abstract method之前不可加private(可以加public 或 protected)

abstract class classA

{

private abstract void funa(); //錯誤，子類別會看不到

}

abstract class之前可加public modifier

意義和一般public class相同(但file name must be the same with class name)

public abstract class test2 { //正確, public 與abstract順序don't care

...

}

abstract public class test3 { //正確, abstract 與 public 順序可以對調

...

}

abstract class 可以繼承其他類別(如下test2可以是abstract class或普通class)

class test2 {

...

}

abstract class test3 extends test2 {

...

}

Abstract的用途：

在ClassB改寫error()可以提供ClassA與ClassB溝通的管道(注意ClassB改寫error()後，在ClassA呼叫error()會執行在ClassB的error()，即印出error2)，如果ClassB可以不改寫error()，ClassA就無法與ClassB溝通；但如果ClassA將error宣告為abstract method，ClassB就一定要改寫error()，否則編譯會有錯誤。

abstract class ClassA {

public void t1 (int i) {

if(i<0)

error ();

}

void error (){

System.out.println("error1");

}

}

class ClassB extends ClassA {

public static void main (String arg[]){

ClassB b=new ClassB();

b.t1(-1);

}

void error () {

System.out.println("error2");

}

}

練習：

●若abstract class A { ... }，則class B extends A{ ...} 是正確的。(O)

●若abstract class A { ... }，則A a=new A(); 是正確的。(X)

●下列何者是抽象類別? ➊abstract class A{…} ➋class A{…} ➌public class A{…} ➍final class A{…}

●下列何者是抽象類別? ➊class A { } ➋abstract class A { } ➌abstract class {void f();} ➍class A { abstract void f(); }

●下列何者是抽象方法? ➊void f( ); ➋int test (); ➌abstract int test (); ➍public void test (){ };

●下列何者是抽象方法? ➊void f(); ➋void f(){ } ➌abstract void f(); ➍abstract void f() { }

●抽象方法只能出現在何處?(選2個) ➊抽象類別 ➋介面 ➌final類別 ➍public類別

●抽象方法的用途為何? ➊強迫子類別實作抽象方法 ➋簡化子類別 ➌簡化父類別 ➍節省記憶體

●下列類別語法是否正確?

|  |  |
| --- | --- |
| class B {  abstract void f();  } | class B {  void f();  } |
| abstract class B {  abstract void f();  } | abstract class B {  void f();  } |
| abstract class B  {  public abstract void f ();  } | abstract class B {  void f(){ }  } |
| abstract class B  {  private abstract void f ();  } | abstract class B {  abstract void f(){ }  } |
| abstract class B {  abstract void f();  }  abstract class A extends B { } | abstract class B {  abstract void f(){ }  }  class A extends B { } |
| abstract class B {  abstract void f();  }  class A extends B {  void f() { }  } | abstract class B {  void f1(){ }  abstract void f2();  } |
| abstract class A {  abstract void f ();  }  class B {  void test ( ) {  A a = new A ();  }  } |  |

## abstract method抽象方法

●方法前面加上abstract且沒有實作的方法稱為抽象方法，也就是類別中的方法沒有大括弧也沒有大括弧內的程式。

●只有抽象類別或介面(Interface)可以有抽象方法，其他類別不可以有抽象方法。

●在介面的抽象方法前可不加abstract，因為在介面裡的方法都是抽象方法。

●抽象方法通常是要留給子類別實作。

●抽象方法與一般方法的比較如下列程式：

抽象方法：

abstract public void methoda (int a); //抽象方法

一般方法：

public void methoda (int a) { //只要有大括弧，就不是抽象方法

}

## final類別與資料

final的類別不可以被繼承，final 的方法不可以被改寫，final的資料不能被修改，宣告為final的資料通常是作為常數用，所以不希望被修改。

public class Test extends A { //錯誤，不可以繼承final 類別

final float PI=3.14f;

public static void main(String[]args) {

}

void setPI (){

PI = 3.14159f; //錯誤，不可改變final的資料

}

}

final class A {

int a;

}

final int i=5;

1.final 的變數不可以改變

class Test4

{

final int i=5;

void funa()

{

i=6; //錯誤，i不可以改變

}

}

2.final的類別不可以被繼承

final class Test {

}

class Test2 extends Test { //錯誤，test類別不可以被繼承

}

3.final 的方法不可以被改寫

class Testa {

public final void t2 (){

}

}

class Test2 extends Testa {

public final void t2 (){

//錯誤，t2 method不可以被子類別改寫(overriding)

}

}

練習：

●下列哪一個類別無法被子類別繼承?   
➊ ➋ ➌ ➍

public class ClassA{

//…

}

class ClassA{

//…

}

final class ClassA{

//…

}

abstract class ClassA{

//…

}

●下列哪一個方法無法被子類別改寫?   
➊ ➋ ➌ ➍

public void f ( ){

//…

}

void f ( ){

//…

}

final void f ( ){

//…

}

static void f ( ){

//…

}

●如果一個類別不希望被其他類別繼承，在宣告類別時可以加上哪一個修飾子?   
➊final ➋this ➌super ➍static

常用的Math與String就是一個典型的final類別，Java不希望使用者繼承他們，他們的定義如下。

public final class Math extends Object

public final class String extends Object

implements Serializable, Comparable<String>, CharSequence

## inner class內部類別

(內部類別是進階的內容)

定義在類別內部的類別稱為內部類別。

內部類別前面可以加上static，一個內部類別如果有static成員，這個內部類別前面就要加上static，但static內部類別的成員不一定要加static。

內部類別如果有成員要給內部類別的物件共用，這些成員與內部類別的前面都要加static。

|  |  |
| --- | --- |
| **class** A {  **class** A1 {  **void** f () { }  }  } | 正確 |
| **class** A {  **class** A1 {  **static** **void** f () { }  }  } | 錯誤，內部類別如果有static成員，這個內部類別前面就要加上static |
| **class** A {  **static** **class** A1 {  **static** **void** f () { }  }  } | 正確 |
| **class** A {  **static** **class** A1 {  **void** f () { }  }  } | 正確，static內部類別的成員不一定要加static |

最上層的類別(外部類別)前不可以加上static。

|  |  |
| --- | --- |
| **static** **class** A {  **class** A1 {  **void** f () { }  }  } | 錯誤 |
| **static** **class** A {  **static** **class** A1 {  **void** f () { }  }  } | 錯誤 |
| **static** **class** A {  **static** **class** A1 {  **static** **void** f () { }  }  } | 錯誤 |

含有內部類別的程式範例

class A {

static class A1 {// 內部static類別

static int a\_static;

int a\_nonstatic;

static void f() { }

void g() { }

}

class A2 {// 內部類別

// static void f () { } //錯誤

void g() { }

}

void h() {

A1.f();

// A1.g (); //錯誤

// A2.g (); //錯誤

}

static void h1() {

A1.f();

// A1.g (); //錯誤

}

}

public class Test {

public static void main(String[] args) {

A.A1.f();

// A.A1.g (); //錯誤

A.A2 a2 = new A().new A2();

//a2可改用下列方法建立物件

A a = new A();

A.A2 a3 = a.new A2();

//A1是A的內部靜態類別

A.A1 a1 = new A.A1();

//A1.f(); //錯誤

a1.g();

a2.g();

A.A1.a\_static = 3;

//A.A1.a\_nonstatic = 3; //錯誤

A.A1 a4 = new A.A1 ();

A.A1 a5 = new A.A1 ();

System.out.print(a4.a\_nonstatic);//0

System.out.print(a4.a\_static);//3

a4.a\_nonstatic = 4;

a4.a\_static = 4;

System.out.print(a5.a\_nonstatic);//0

System.out.print(a5.a\_static);//4

//-->a4.a\_static = 4;和a5.a\_static是同一個

}

}

## anonymous class匿名類別

沒有類別名稱的類別稱為匿名類別，通常用來實作一個介面，只用在程式某個地方(例如處理某個按鈕的click事件)，別的地方不會用到，這種用途往往省略類別名稱以簡化程式。

1.匿名類別也算是一種內部類別。

2.匿名類別是在建立物件時才定義類別。

3.以下的例子說明分別使用匿名類別、外部類別、內部類別處理按鈕的事件，其中ActionListener是一個介面(interface)，有關interface的說明詳見interface介面一節。

**public** **class** Test {

**public** **static** **void** main(String[] args) {

MyJframe jf =**new** MyJframe ();

}

}

**class** MyJframe **extends** JFrame {

MyJframe () {

setDefaultCloseOperation (JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100,100,500,300);

setTitle("視窗標題");

JButton jb1 = **new** JButton("按一下");

jb1.setBounds(50, 50, 100, 50);//x, y, width, height

jb1.addActionListener(

**new** ActionListener() { //匿名類別

**public** **void** actionPerformed(ActionEvent e) {

JOptionPane.*showMessageDialog*(**null**, "恭喜成功了");

}

}

);//使用匿名類別

JButton jb2 = **new** JButton("按一下");

jb2.setBounds(200, 50, 100, 50);//x, y, width, height

//使用外部類別

jb2.addActionListener(**new** ButtonAction());

JButton jb3 = **new** JButton("按一下");

jb3.setBounds(350, 50, 100, 50);//x, y, width, height

//使用內部類別

jb3.addActionListener(**new** InnerButtonAction());

JPanel p = **new** JPanel ();

setContentPane (p);

p.setLayout(**null**);

p.add(jb1);

p.add(jb2);

p.add(jb3);

setVisible(**true**);//setVisible要放在最後

}

//內部類別

**class** InnerButtonAction **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e) {

JOptionPane.*showMessageDialog*(**null**, "恭喜三度成功了");

}

}

}

//外部類別

**class** ButtonAction **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e) {

JOptionPane.*showMessageDialog*(**null**, "恭喜又成功了");

}

}

匿名類別通常含有抽象方法的實作，用以提供二個類別溝通的方法。下列程式使用匿名類別與普通類別，都可以達到相同的結果，匿名類別可以讓程式簡短一些。

public class Test {

public static void main(String[] args) {

new A () { //匿名類別

void funa (String str) {

System.out.println("hi");

}

}.funa("hi");

A b = new B (); //普通類別

b.funa("hello");

}

}

abstract class A //正確

{

abstract void funa(String str);

}

class B extends A {

void funa (String str) {

System.out.println(str);

}

}

練習：

●上述程式使用外部類別與匿名類別完成印出一串字，請為上述程式加上一個內部類別完成印出一串字。

●沒有類別名稱的類別稱為 ➊內部類別 ➋外部類別 ➌匿名類別 ➍抽象類別

●下列何者含有匿名類別?

|  |
| --- |
| class { } |
| new A () {  void funa (String str) {  System.out.println("hi");  }  }.funa("hi"); |

●執行下列程式會印出hello，刪除下列程式的B類別，試在Test類別中以匿名類別改寫?

public class Test {

public static void main(String[] args) {

A b = new B (); //普通類別

b.funa("hello");

}

}

abstract class A //正確

{

abstract void funa(String str);

}

class B extends A {

void funa (String str) {

System.out.println(str);

}

}

●下列程式是否為一種匿名類別?

class { }

# 物件導向技術-多載與覆寫

## Overloading多載

1.在同一類別，名稱相同，引數個數不同，或引數個數相同但型態不同的method

(一個method有很多種用法，其用法是依據引數個數與引數型態決定)

2.名稱相同，引數個數與型態均相同，只有傳回值型態不同的method，不屬於overloading，而是一種錯誤

3.傳回值型態與overloading無關

下列程式的類別中有3個f1方法，引數的個數與型態均有差異，是overloading，當呼叫f1方法時，可以從參數的個數與型態判斷是要呼叫哪一個方法，overloading在Java是很常見的的方法。

public class Test {

public static void main(String[] args) {

Test t = new Test ();

t.f1 ();

}

void f1 () {

}

void f1 (int a) {

}

void f1 (String s) {

}

}

●下列哪一個類別含有多載(Overloading)的method?

|  |  |  |  |
| --- | --- | --- | --- |
| ➊ | ➋ | ➌ | ➍ |
| class ClassA {  int k;  void a () {  k=1;  }  void a () {  k=2;  }  } | class ClassA {  int k;  void a () {  k=1;  }  int a () {  return 0;  }  } | class ClassA {  int k;  void a () {  k=1;  }  void a (int p) {  k=p;  }  } | class ClassA {  int k;  void a () {  k=1;  }  void b () {  k=2;  }  } |

●下列哪一個類別含有多載(Overloading)的method?

|  |  |  |  |
| --- | --- | --- | --- |
| ➊ | ➋ | ➌ | ➍ |
| class ClassA {  int k;  void a () {  k=1;  }  void a () {  k=2;  }  } | class ClassA {  int k;  public void a () {  k=1;  }  void a (int p) {  k=p;  }  } | class ClassA {  int k;  void a () {  k=1;  }  int a () {  return 0;  }  } | class ClassA {  int k;  void a () {  k=1;  }  void b () {  k=2;  }  } |

●下列程式的類別中有三個f1方法，這種方法稱為 ➊覆寫(override) ➋多載(overloading) ➌多型(Polymorphism) ➍以上皆非 (2)

public class Test {

public static void main(String[] args) {

Test t = new Test ();

t.f1 ();

}

void f1 () { }

void f1 (int a) { }

void f1 (String s) { }

}

## Override覆寫

如果想重寫父類別的方法，比如toString()方法的話，在方法前面加上@Override可以讓編譯程式幫你檢查方法的正確性。若要改寫父類別方法，子類別的方法必須返回值型態、方法名稱、參數個數與型態均與父類別方法相同。以下範例編譯器會檢查是否為改寫方法。

@Override

public String toString(){//這是正確的

}

一旦寫錯，寫成這樣(tostring全部小寫)

@Override

public String tostring(){

}

編譯器可以檢測出這種寫法是錯誤的 這樣能保證你重寫的方法正確

而如果不加@Override

public String tostring(){...}

這樣編譯器是不會認為錯的，它只會認為這是你自己新加的一個方法而已。

一個完整的程式範例如下：

public class Test {

public static void main(String[] args) {

Test t = new Test ();

t.f1 ();

}

void f1 () {

}

}

class TestOverride extends Test {

@Override //編譯器可以檢查這是一個錯誤的改寫

void f1 (int a, int b) {

}

@Override

void f1 () {

}

}

子類別成員如要改寫父類別成員，子類別成員的能見度不可小於父類別成員

//正確

class Man {

void test () {}

}

class YoungMan extends Man{

public void test () {}

}

//錯誤，子類別成員的能見度(visibility)小於父類別成員

class Man {

public void test () {}

}

class YoungMan extends Man {

void test () {}

}

Override總結：

1.在子類別中，名稱、引數個數與型態均與父類別相同的method，會取代父類別的method

2.父類別private method不會被子類別override，因為子類別根本看不到父類別private的方法。

3.子類別的方法如果要override父類別的方法，其返回值型態必須與父類別相同，否則編譯會有錯誤，例如

class A { public void methoda () { } }

class B extends A { public int methoda () {return 0;}} //錯誤

4.子類別的方法如果要override父類別的方法，子類別的方法可見範圍不可以比父類別小，例如

class A { public void methoda () { } }

class B extends A { void methoda () { }} //錯誤

class B extends A { public void methoda () { }} //正確

class C { protected void methoda () { } }

class B extends C { private void methoda () { }} //錯誤

class B extends C { void methoda () { }} //錯誤

class B extends C { protected void methoda () { }} //正確

class B extends C { public void methoda () { }} //正確

5.子類別non-static method不可以override父類別static method,

子類別static method不可以override父類別non-static method

class D { static void methoda () { } }

class B extends D { void methoda () { }} //錯誤

class B extends D { static void methoda () { }} //正確

class E { void methoda () { } }

class B extends E { static void methoda () { }} //錯誤

class B extends E { void methoda () { }} //正確

●下列類別語法是否正確? (每個儲存格的程式都是在同一個檔案中)

|  |
| --- |
| class A { public void f () { } }  class B extends A { public int f () {return 0;}} |
| class A { public void f () { } }  class B extends A { public void f () {return 0;}} //錯誤 |
| class A { public void f () { } }  class B extends A { void f () { }} //錯誤 |
| class A { public void f () { } }  class B extends A { public void f () { }} //正確 |
| class C { protected void methoda () { } }  class B extends C { private void methoda () { }} //錯誤 |
| class C { protected void methoda () { } }  class B extends C { void methoda () { }} //錯誤 |
| class C { protected void methoda () { } }  class B extends C { protected void methoda () { }} //正確 |
| class C { protected void methoda () { } }  class B extends C { public void methoda () { }} //正確 |
| class D { static void methoda () { } }  class B extends D { void methoda () { }} //錯誤 |
| class D { static void methoda () { } }  class B extends D { static void methoda () { }} //正確 |
| class E { void methoda () { } }  class B extends E { static void methoda () { }} //錯誤 |
| class E { void methoda () { } }  class B extends E { void methoda () { }} //正確 |

# 物件導向技術-多型與封裝

## Polymorphism多型

多型（Polymorphism）是可使用一致介面來處理不同的資料型態，如下列程式m方法可以處理四種不同型態的參數，參數的型態會在run time才決定，這叫dynamic binding，也就是資料的繫結是延後到執行時期；Java從子類別往上找資料型態

YoungMan找不到toString，往上找到Man，在Man找到toString

Man、Person、Object都有找到toString，執行結果可印證

這樣在m方法中宣告時只要使用Object當作參數型態，呼叫m方法時可以傳Object或其子類別物件，沒有限制一定要傳Object物件，這種特性稱為多型。因為子類別也是父類別的一種。

public class Test {

public static void main(String[]args) {

m(new YoungMan());

m(new Man());

m(new Person());

m(new Object());

}

public static void m(Object obj) {

System.out.println(obj.toString());

}

}

class Person extends Object {

public String toString() {

return "This is a Person";

}

}

class Man extends Person {

public String toString() {

return "This is a Man";

}

}

class YoungMan extends Man {

}

執行結果：

This is a Man

This is a Man

This is a Person

java.lang.Object@6d06d69c

註：toString在Object的能見度為public，所以子類別的能見度不可小於public

## Encapsulation封裝

封裝(Encapsulation)就是適度的將資訊隱藏(Information Hiding)，避免外界不當使用。使用者知道如何使用一個方法但不需要也不會知道這個方法如何實作，方法的實作被封裝且被隱藏(hidden)，這樣就可以達到方法的抽象化，也就是把方法當作一個黑箱(black box)。如果要存取類別的私有(private)成員，必須透過類別的公有(public)成員，這樣私有資料成員就可以被隱藏起來，不會被使用者任意修改，例如不會把一個人的身高或體重改為負數，這就是資料封裝。

下列程式將age的可見範圍設為private，其他類別無法用修改變數的方法改變age，也看不到setAge與getAge的實作，但可透過這二個方法改變與讀取age，在setAge方法中只接受合理的age，這樣將資料可見範圍設為private，達到隱藏資料，並提供適當方法讓其他類別存取，就是資料封裝。

public class Test {

private int age=0;

public void setAge (int a){ //設定age

if (age > 0 && age < 120)

age = a;

}

public int getAge () { //傳回age

return age;

}

}

練習：

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test {

void f1 (boolean b) {

}

void f1 (int a) {

}

}

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test {

void f1 (boolean b) {

}

int f1 (int a) {

return 0;

}

}

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test {

void f1 (int b) {

}

int f1 (int a) {

return 0;

}

}

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test {

public void f1 () {

f2(3);

}

private void f2 (int a) {

}

}

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test {

private int a;

public void f1 () {

a=3;

}

}

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test {

private int a=3;

public int f1 () {

return (a);

}

}

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test extends Test2 {

void f1 () { }

}

class Test2 { void f1 () { } }

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test extends Test2 {

void f1 () { }

}

class Test2 { void f2 () { } }

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

public class Test {

public static void main(String[]args) {

m(new Person());

}

public static void m(Object obj) {

System.out.println(obj.toString());

}

}

class Person extends Object {

public String toString() {

return "This is a Person";

}

}

# Exception例外

是指執行時發生不正常的錯誤事件，如開啟檔案時找不到檔案，這些錯誤在被呼叫方法已經有定義，一旦發生這些錯誤，被呼叫方法就會丟出Exception讓呼叫者處理，呼叫者可以決定是否繼續執行。

1.常見的Exception

ArithmeticException(除以0)

ArrayIndexOutOfBoundsException(陣列索引超出範圍)

IOException(IO錯誤)

SQLException(SQL連接錯誤)

2.處理Exception

try { ... }

catch ( ... ) { ... }

finally { ...}

呼叫會丟出Exception的method時，要在前面加try、在後面加catch，catch小括弧內要指出所要捕捉的Exception，finally這一段可有可無，如果有finally，不論是否發生Exception，都會執行finally這一段。

3.Exception發生時可以用printStackTrace()將呼叫堆疊印出，或getMessage()取得錯誤訊息。

4.如何定義一個方法丟出Exception?

public void test( ) **throws IOException** {   
//只可能丟出一個Exception

...

}

public void test2( ) **throws Exception1, Exception2** {   
//可能丟出二個Exception

...

}

6.呼叫上述test()需要使用下列方式，否則編譯時會有錯誤。

try {

System.out.println(“Exception test”); //測試用

test ();

}

catch (IOException e){ System.out.println(e.getMessage());}

7.如果二個方法會丟出相同的Exception，可以使用同一個  
catch(...) { ...} 捕捉Exception，如：

try {

test1 ( ); //可能丟出IOException

test2 ( ); //可能丟出IOException

}

catch (IOException e){System.out.println(e.getMessage());}

8.同時處理多種Exception：如果二個方法會丟出不同的Exception或一個方法可能丟出二個Exception，可以分別用catch ( ... ) { ... }捕捉不同的Exception。如：

try {

test1 ( ); //可能丟出Exception1,Exception2

test2 ( ); //可能丟出Exception3

}

catch (Exception1 e) { System.out.println(e.getMessage());}

catch (Exception2 e) { System.out.println(e.getMessage());}

catch (Exception3 e) { System.out.println(e.getMessage());}

9.如何丟出Exception?

public void test(int arg) **throws Exception** {   
if(arg==0 )

throw **new Exception**(“參數arg不可以是0”);

}

10.其他說明：

➊throws Exception是定義在method不是定義在class。

➋方法沒有定義throws Exception就不可以在方法中throw Exception。

➌Exception也是一個類別，丟出例外就是丟出一個Exception物件。

➍catch (...) { /\*空白\*/ }大括弧內可以空白。

➎查被呼叫方法的類別，就可以知道該方法會不會丟出Exception。

➏處理Exception的邏輯類似if-else，如果if-else太多，程式邏輯控制就會顯得複雜，改用Exception可以使邏輯控制簡化，一個catch就相當於發生一個if。

➐沒有宣告throws Exception的方法也可以用try ... catch...呼叫，如下test沒有宣告throws Exception但可用try ... catch...呼叫，但沒有意義。

class classA {//正確

void test (int i) {

if (i<0)

System.out.println("error");

else

System.out.println(i);

}

void test2 () {

try {

test(1);

}

catch(Exception e) {

System.out.println(e.getMessage());

}

}

}

11.自訂Exception類別

class MyException extends Exception {

//改寫Exception的方法

//如public String getMessage (){ return “錯誤訊息”;}

}

12.以下的類別是錯誤的

class classA { //錯誤,test沒有宣告throws Exception,在程式中不可throw Exception

void test (int i) { //方法後加上throws Exception才可丟出例外

if (i<0)

throw new Exception("error");

else

System.out.println(i);

}

}

class classA {//錯誤,test有宣告throws Exception,必須用   
try { test(2);}   
catch(Exception e) { ... } 呼叫

void test (int i) throws Exception {

if (i<0)

throw new Exception("error");

else

System.out.println(i);

}

public void test2 () {

//呼叫會丟出例外的方法必須在方法後加上throws Exception

//或在方法內使用try { } catch (...) { }自行捕捉例外

test(2);

}

}

練習：

●指出下列程式編譯時錯誤的地方，並更正之

public class Test {

static int a[]={1,2,3};

void fun(){

try

a[2]=2;

catch(Exception e)

System.out.println(e.getMessage());

}

}

●test.java檔案內容如下，編譯時是否有錯誤，如有錯誤，請說明錯誤原因

class classA {

void classA(){

test(1);

}

void test (int i) {

if (i<0)

throw new Exception("error"); //錯誤,test沒有宣告throws Exception,在test method不可以throw Exception

}

}

●test.java檔案內容如下，編譯時是否有錯誤，如有錯誤，請說明錯誤原因

class classA {

void classA(){

try {

test1(1);

test2(2);

}

catch(ArithmeticException e) {

System.out.print(e.getMessage());

}

}//錯誤,test2可能 throw Exception,需要catch(Exception f) { ... }

void test1 (int i) throws ArithmeticException {

if (i<0)

throw new ArithmeticException("error");

}

void test2 (int i) throws Exception {

if (i<0)

throw new Exception("error");

}

}

●填入適當的選項：

( )下列程式可以捕捉例外，可能發生例外的程式碼要放在➊-➐哪一處?

( )要捕捉的例外要放在➊-➐哪一處?

( )發生例外時，處理例外的程式碼要放在➊-➐哪一處?

try { ➊ } ➋

catch ( ➌ ) { ➍ }➎

finally { ➏ } ➐

●執行「int i = 3/0;」會遇到ArithmeticException的例外，若不捕捉例外，程式會因此中斷，下列程式try { } catch { } finally { }會用捕捉例外的方式執行，發生例外時執行「System.out.print("程式中發生除以0的例外");」告訴使用者發生例外，但程式仍能繼續執行不中斷，填入適當的選項：

( )「int i = 3/0;」的程式碼要放在➊-➐哪一處?

( )「ArithmeticException e」要放在➊-➐哪一處?

( )上述告訴使用者發生例外的程式碼應該放在➊-➐哪一處?

try { ➊ } ➋

catch ( ➌ ) { ➍ }➎

finally { ➏ } ➐

●下列三個區塊try、catch、finally的程式常用在捕捉例外的程式碼中，哪一個區塊是可以省略的?

try { }

catch ( ) { }

finally { }

●下列程式空格中應分別填入下列一個關鍵字 (extends, implements, final, public, package, static, return, throw, try, catch, finally, throws)

class A {

void test (int i) Exception {

if (i<0)

new Exception("error");

}

}

# static member

static member可以直接以類別名稱存取，或經過繼承存取，或建立物件後，再以物件存取,只有一份,大家共用

non-static member必須先建立物件，再以物件存取，或經過繼承才能存取，每一個物件有一份,不被物件共用

class Test1

{

public static int a;

public int b;

}

class Test3

{

void t ()

{

Test1.a=5;

Test1.b=6; //錯誤, non-static variable

a=1; //錯誤

b=2; //錯誤

}

}

class Test1

{

public static int a;

public int b;

}

class Test3 extends Test1

{

void t ()

{

Test1.a=5;

Test1.b=6; //錯誤

a=1; //正確,因為繼承

b=2; //正確,因為繼承

}

}

class Test1

{

public static int a;

public int b;

}

class Test3 extends Test1

{

void t ()

{

Test1 t1=new Test1 ();

Test1.a=5;

t1.b=6; //正確, t2是物件

a=1; //正確,因為繼承

b=2; //正確,因為繼承

}

}

static member所屬的類別不論是否被建立物件，或被建立多少個物件，或是否被繼承，static member在一個java Application都只有一份，因此同類別的物件可以共用static member

class Test1

{

public static int a;

}

class Test3 extends Test1

{

public Test3 ()

{

Test1 t1=new Test1();;

Test1.a=5;

System.out.print(t1.a); //印出5

System.out.print(a); //印出5

}

}

基本資料型態的資料宣告後即可使用，因為宣告後就會配置記憶體,如

int i=5;

int k;

k=6;

物件必須經過建立或繼承才可以存取成員，只有宣告還不能存取成員

class Test1 {

public static int a;

public int b;

}

class Test3 {

void t () {

Test1 t1;

t1.a=6; //錯誤, t1只有宣告尚未建立,尚未配置記憶體,無論是否為static均無法存取

t1.b=6; //錯誤, t1只有宣告尚未建立,無論是否為static均無法存取

}

}

class Test1 {

public static int a;

public int b;

}

class Test3 {

void t () {

Test1 t1=new Test1();

t1.a=6; //正確, t1已建立,無論是否為static均可存取

t1.b=6; //正確, t1已建立,無論是否為static均可存取

}

}

Test1 t1=new Test1();也可以寫成以下二行

Test1 12;

t1= new Test1();

this關鍵字指目前的物件

static method中不可以使用this(不算錯，但會有警告)

(static method只有一份,有很多物件時，不知道是哪一個物件)

static method不可以直接呼叫non-static method, 必須透過物件呼叫

static method不可以直接存取類別內的non-static variable, 必須透過物件存取

static 不可以使用在區域變數，如

class A {

void a () { static int a; } } //錯誤 區域變數不可加static

static void b () { static int a; } } //錯誤 區域變數不可加static

}

// 程式範例

public class Test {

int age = 18;

static int height = 160;

public static void main(String[] args) {

age = 19; //錯誤

height = 165; //正確

t1 (); //錯誤

t2 ();//正確

Test t = new Test ();

t.t1 ();//正確

}

void t1 () {

}

static void t2 () {

}

}

方法f()與變數a在同類別 class T2 { }

|  |  |  |
| --- | --- | --- |
| 程式是否正確? | class T2 { static int a; } | class T2  { int a; } |
| static void f () {a=3;} | O | X |
| void f () {a=3;} | O | O |
| static void f () {  T2 t2=new T2(); t2.a=3;  } | O | O |
| void f () {  T2 t2=new T2(); t2.a=3;  } | O | O |

方法f()在類別 class FF{ }、變數a在class AA{ }

|  |  |  |
| --- | --- | --- |
| 程式是否正確? | class AA{  static int a;} | class AA{  int a; } |
| static void f () {a=3;} | X | X |
| void f () {a=3;} | X | X |
| static void f () {  AA aa=new AA(); aa.a=3;  } | O | O |
| static void f () {  AA.a=3;  } | O | X |
| void f () {  AA.a=3;  } | O | X |
| void f () {  AA aa=new AA(); aa.a=3;  } | O | O |

●寫出下列程式的執行結果?

public class test {

static int x;

int y;

public static void main(String[] args) {

test a = new test ();

test b = new test ();

a.x=1;

a.y=2;

b.x=3;

b.y=4;

System.out.println(a.x); //答：

System.out.println(a.y); //答：

}

}

●下列程式如果正確，請在空格中寫出執行結果；如果錯誤，請在空格中寫出錯誤的地方，並說明錯誤原因與如何更正?

public class T2 {

static int a=2;

public static void main(String[] args) {

f();

}

static void f () {

a = 3;

System.out.print(a);//答:

}

}

●下列程式如果正確，請在空格中寫出執行結果；如果錯誤，請在空格中寫出錯誤的地方，並說明錯誤原因與如何更正?

public class T2 {

static int a=2;

public static void main(String[] args) {

f();

}

void f () {

a = 3;

System.out.print(a);//答:

}

}

●下列程式如果正確，請在空格中寫出執行結果；如果錯誤，請在空格中寫出錯誤的地方，並說明錯誤原因與如何更正?

public class T2 {

int a=2;

public static void main(String[] args) {

a = 3;

System.out.print(a);//答:

}

}

●下列程式如果正確，請在空格中寫出執行結果；如果錯誤，請在空格中寫出錯誤的地方，並說明錯誤原因與如何更正?

public class T2 {

int a = 2;

public static void main(String[] args) {

f();

}

static void f() {

T2 t2 = new T2();

t2.a = 3;

System.out.print(t2.a); //答:

}

}

# 資料/方法可見範圍

classA類別的資料成員在classB類別的可見範圍(前提classB可以參考classA,ex.public classA或both classes are in the same package)

public 同一類別、同一個package的類別、子類別、其他類別

protected 同一類別、同一個package的類別、子類別

沒有modifier 同一類別、同一個package的類別

private 同一類別

(寫在同一個檔案的class視為同一個package)

p1和p1.p2是不同的package

public int k; //所有類別

protected int k; //子類別

int k; //同一package

private int k;

//同一class

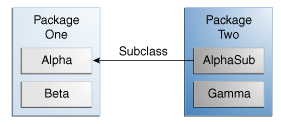
記法：

public int a; //公共的，可以被任何類別存取，可見範圍最廣

protected int b; //保護的，可以被子類別和同一個套件存取(含同類別)

int c; //不指定，只能被同一個套件的類別存取(含同類別)

private int d; //私有的，只有自己類別可以存取，可見範圍最小

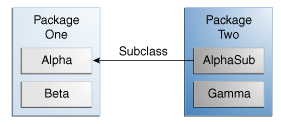


Visibility of the members of Alpha:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modifier | Alpha | Beta | AlphaSub | Gamma |
| public | Y | Y | Y | Y |
| protected | Y | Y | Y | N |
| no modifier | Y | Y | N | N |
| private | Y | N | N | N |

source <http://java.sun.com/docs/books/tutorial/java/javaOO/accesscontrol.html>

練習：



●Alpha、Beta、AlphaSub、Gamma四個類別的關係如上圖，Alpha裡面有四個資料如下，a可以被哪些類別存取? b可以被哪些類別存取? c可以被哪些類別存取? d可以被哪些類別存取?

//資料成員在同一類別的可見範圍

package One;

public class Alpha {

public int a;

protected int b;

int c;

private int d;

void f1 () {

a = 0;

b = 0;

c = 0;

d = 0;

}

}

//資料成員在同一個package的可見範圍

package One;

public class Beta {

void f1 () {

Alpha alpha = new Alpha ();

alpha.a = 0;

alpha.b = 0;

alpha.c = 0;

alpha.d = 0; //錯誤

}

}

//資料成員在子類別的可見範圍

package Two;

public class AlphaSub extends One.Alpha { //繼承

void f1 () {

a = 0;

b = 0;

c = 0; //錯誤

d = 0; //錯誤

}

}

//資料成員在其他類別的可見範圍

package Two;

public class Gamma {

void f1 () {

One.Alpha alpha = new One.Alpha();

alpha.a = 0;

alpha.b = 0; //錯誤

alpha.c = 0; //錯誤

alpha.d = 0; //錯誤

}

}

練習：

Alpha類別如下，下列Gamma類別哪列程式是錯誤的?

|  |  |
| --- | --- |
| package One;  public class Alpha {  public int a;  protected int b;  int c;  private int d;  void f1 () {  a = 0;  b = 0;  c = 0;  d = 0;  }  } | package Two;  public class Gamma {  void f1 () {  One.Alpha alpha = new One.Alpha();  alpha.a = 0; //  alpha.b = 0; //  alpha.c = 0; //  alpha.d = 0; //  }  } |

## class modifier

1.同一個java程式檔案內最多只能有一個public類別，且檔案名稱必須與public類別名稱相同。

2.class不論是否使用public modifier，只要有main method均可以在命令列使用c:\>java classname的方式執行。

3.public class可以被其他package的class 繼承或new

4.不是public class只能讓同一個package裡的類別存取

public class classA //可以被其他package的class 繼承或new

{

...

}

class classB //只能讓同一個package裡的類別存取

{ //

...

}

private class classC //錯誤的寫法，class不可以使用private修飾

{

...

}

5.寫在同一個檔案的class視為同一個package

6.class與interface均不可以使用private/protected modifier

7.同一個檔案內的class無論是否使用public modifier,均可以互相參考

8.在classA程式第一行中如果沒有指定程式是package A;,即使將classA.class檔案複製到A資料夾，classA仍然不屬於package A,在其他程式import A.classA;會有錯誤。

9.類別裡面可以再有類別(稱為內部類別)

練習：

( )下列何者不可以寫在下列程式類別前方的空格中? (選2個)

class A { int height=160;}

➊public ➋final ➌private ➍static ➎空白

# Interface介面

Interface提供二個類別溝通管道

資料成員必須有初值

interface interface2

{

int w; // 錯誤,沒有設初值

int x=5; //正確, final can be ignored

final int y=6; //正確

static int i=5; //正確

private int j=6; //錯誤

protected int c=5; //錯誤

}

成員可見範圍不可以是private或protected，只能用public modifier或沒有modifier

interface interface2

{

private int j=6; //錯誤

protected int c=5; //錯誤

private void f1 ();//錯誤

protected void f2();//錯誤

void funa(); //正確

private void funb(); //錯誤

protected void func(); //錯誤

public void fund(); //正確

}

interface的method必須全部宣告成abstract

interface interface2

{

void funa(); //正確, abstract can be ignored

abstract void funb (); //正確

void f () { } //錯誤

//...

}

實作interface的method只能用public modifier，原因是abstract method提供二個類別溝通，這二個類別不一定在同一個package或有繼承的關係，因此必須加上public才可以讓另一個類別看得到。實作interface的類別前則不一定要加上public，如下程式，ClassA會傳給ClassB i2(ClassA實作interface2，因此也是一種interface2)，ClassB就會知道ClassA在哪裡，ClassA前就可以不加public。

interface interface2

{

void fund();

}

class ClassA implements interface2 {

void fund() { /\*...\*/ } //錯誤

public void fund () { /\*...\*/ } //正確

}

class ClassB {

void f (interface2 i2, int par ) { ... }

}

不可以用interface產生物件

void funa()

{

interface2 t2=new interface2(); //錯誤,不可以用interface產生物件

}

一個類別只能繼承一個類別，不可以繼承多個類別。

class A1 { }

class A2 { }

class classA extends classA1,classA2 //錯誤,不可以多重繼承

{ //....

}

一個類別只能繼承一個類別但可以implements多個interface。實作多個介面就是提供一個類別與很多類別溝通的管道，繼承abstract class只能提供一個子類別與一個父類別溝通的管道。

class A1 { }

class classA extends classA1 implements ifs1,ifs2 //正確,可實作多個介面

{ //....

}

介面不可以繼承類別。

class A1 { }

interface ifs2 extends classA //錯誤,介面不可以繼承類別

{//....

}

介面可以繼承介面，也可以繼承多個介面。

interface ifs2 {

int i=3;

}

interface ifs3 extends ifs2 //正確,介面可以繼承介面

{//...

}

interface ifs4 extends ifs2, ifs3 //正確,介面可以多重繼承介面

{//...

}

介面不可以實作介面

interface ifs2 {

int i=3;

}

interface ifs3 implements ifs2 //錯誤

{//...

}

interface提供二個類別互相溝通的管道，如下範例。

class ClassA {

void f (I i, int height) {

if (height<0)

i.error (); //ClassA與實作interface I的類別溝通

}

}

class ClassB implements I {

public static void main (String arg[]){

ClassA a=new ClassA();

ClassB b=new ClassB();

a.f(b, -1);

}

public void error () {

System.out.println("error");

}

}

interface I {

public void error ();

}

一般類別、抽象類別、介面比較：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 一般類別（Class） | 抽象類別（Abstract Class） | 介面（Interface） |
| 宣告屬性（attribute） | O | O | O 要有初值 |
| 常數（const） | O | O | O |
| 實例化（new class） | O | X | X |
| 抽象方法（abstract function） | X | O | O |
| 實作方法內容（functoin()） | O | O | X |
| 類別是否可繼承多個 | X | X | O |

練習：

●以下程式3~5行中，有一行程式是錯誤的，錯誤的程式在第幾行? 錯誤原因爲何。

interface interface2

{

int funa();

private void funb();

public void fund();

}

●下列程式中，CShape是一個介面，說明程式錯誤的地方，並更正之

interface CShape

{

protected String color="Yellow";

int type;

}

●下列程式空格中應填入下列哪一個字 (extends, implement, implements, final, public, package, static, import, return, throw, throws, try, catch, finally, default)

interface inf {

//...

}

class bclass inf {

public static void main(String arg[]) {

//...

}

//...

}

●下列程式空格中應填入下列哪一個字 (extends, implement, implements, extend, extends, final, public, package, static, import, return, throw, throws, try, catch, finally, default)

class test1 { /\*...\*/ }

interface I2 { /\*...\*/ }

class test3 test1 I2 { /\*...\*/ }

●下列程式空格中應填入下列哪一個字 (extends, implement, implements, extend, extends, final, public, package, static, import, return, throw, throws, try, catch, finally, default)

class test1 { ... }

interface I2 { ... }

interface I3 I2 { /\*...\*/ }

下列程式使用物件導向哪一個技術?

➊Encapsulation ➋Polymorphism ➌Override ➍Overloading ➎程式錯誤 ➏以上皆非

class ClassA {

void f (I i) {

//...

}

}

class ClassB implements I {

public static void main (String arg[]){

ClassA a=new ClassA();

ClassB b=new ClassB();

a.f(b);

}

public void error () {

//...

}

}

interface I {

public void error ();

}

## 資料型態轉換

int,boolean,float,double,long,char轉換成String

String s=new String();

s.valueOf(123);

s.valueOf(true);

s.valueOf(12.3f);

s.valueOf(12.3);

s.valueOf('A');

Alternatively, since valueOf is a static method which can be invoked directly prior to instantiating class. For instance,

String.valueOf(123);

,and so on.

|  |  |
| --- | --- |
| 字串轉int | int i = Integer.parseInt("123"); |
| 字串轉long | long l = Long.parseLong("123"); |
| 字串轉float | float f = Float.parseFloat("123.6"); |
| 字串轉double | double d = Double.parseDouble("123.6"); |
| 字串轉boolean | boolean d = Boolean.parseBoolean("false"); |

如何以十六進位格式印出一個整數?

可以用i=0x25;指定數值

System.out.println(Integer.toHexString(123));

或

System.out.println(Integer.toString(123,16));

或轉成大寫如下

System.out.println(Integer.toHexString(123).toUpperCase());

# 參考資料

## JVM Instruction Set

<https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-6.html>

輸入 javap -c Test.class

即可看到反组譯結果如下

Compiled from "Test.java"

public class prj1.Test {

public prj1.Test();

Code:

0: aload\_0

1: invokespecial #8 // Method java/lang/Object."<init>":()V

4: return

public static void main(java.lang.String[]);

Code:

0: ldc #16 // String aBD

2: astore\_1

3: getstatic #18 // Field java/lang/System.out:Ljava/io/PrintStream;

6: aload\_1

7: ldc #24 // String abc

9: invokevirtual #26 // Method java/lang/String.compareTo:(Ljava/lang/String;)I

12: invokevirtual #32 // Method java/io/PrintStream.println:(I)V

15: iconst\_3

16: anewarray #27 // class java/lang/String

19: dup

20: iconst\_0

21: ldc #38 // String Apple

23: aastore

24: dup

25: iconst\_1

26: ldc #40 // String banana

28: aastore

29: dup

30: iconst\_2

31: ldc #42 // String cherry

33: aastore

34: astore\_2

35: return

void sum(java.lang.String);

Code:

0: iconst\_3

1: newarray int

3: dup

4: iconst\_0

5: iconst\_1

6: iastore

7: dup

8: iconst\_1

9: iconst\_2

10: iastore

11: dup

12: iconst\_2

13: iconst\_3

14: iastore

15: astore\_2

16: aload\_2

17: dup

18: astore 6

20: arraylength

21: istore 5

23: iconst\_0

24: istore 4

26: goto 45

29: aload 6

31: iload 4

33: iaload

34: istore\_3

35: getstatic #18 // Field java/lang/System.out:Ljava/io/PrintStream;

38: iload\_3

39: invokevirtual #32 // Method java/io/PrintStream.println:(I)V

42: iinc 4, 1

45: iload 4

47: iload 5

49: if\_icmplt 29

52: return

static void f(java.lang.String);

Code:

0: return

}