

電腦網路與程式課程

JAVA 物件導向程式設計

JAVA 程式組教材

授課人員：黃怡傑

目錄

Ch1. 學習 JAVA 的第一堂課 - 程式的邏輯

- 1.1 coding 的習慣
- 1.2 開始學習 JAVA 前的準備
- 1.3 第一個 JAVA 程式
- 1.4 資料型別
- 1.5 運算子
- 1.6 迴圈
- 1.7 條件判斷
- 1.8 使用者輸入

Ch2. 學習 JAVA 的第二堂課 - 物件導向的程式

- 2.1 類別與方法 (一)
- 2.2 類別與方法 (一)
- 2.3 類別與方法 (一)
- 2.4 變數
- 2.5 作業與範例

Ch3. 學習 JAVA 的第三堂課 - 基礎的網路程式

- 3.1 基礎網路程式 (一)
- 3.2 基礎網路程式 (一)
- 2.3 作業與範例

Ch1. 學習 JAVA 的第一堂課 - 程式的邏輯

1.1 coding 的習慣

寫程式的共識

常常寫程式，或是接觸程式碼(或是各種語言的編碼)的人很容易能了解那種：看到一整篇程式碼，卻沒辦法馬上理解在做什麼。甚至有時候連編寫的人自己都有閱讀問題的情況。一個好的程式(碼)，除了方便自己編輯、使用外，同時也要能讓別人容易看懂他在做什麼以下列舉一些在編寫 JAVA(或是各種程式語言)時的良好習慣

1. 良好的命名習慣：

上自資料夾，下至小小的變數，都要有好好命名的習慣，以檔名或是資料夾為例子：資料夾由外到裡面的命名可以所屬單位、所屬專案、所屬程式等等命名。檔名的命名則盡量 遵守以下習慣：

- a. 以英文命名 - 這除了方便整理等等功能，同時也可以排除某些程式讀取上的錯誤。
- b. 不含多餘符號 - 以 java 的變數為例，JVM 本身只允許 "_"(底線)、"\$"(錢號)，兩種形式，通常 除非需求，基本上檔名的命名盡量不使用"底線"以外的符號，底線使用的時機 在於連結，如：ncu_me.java 這類的命名形式。
- c. 命名的開頭為英文 - 不要以數字或是符號當作名稱的開頭，也盡量避免單純流水號形式的命名， 如以 a1、a2、a3 命名。

2. 標好註解：

迴圈、旗標、判斷、不同類別等等都要好好標上註解。

3. 括號的習慣：

善用 Tab 鍵來區分括號內與括號外的內容，即使可以省略括號的部份也盡量加上括號，此外每打完一段程式碼就自己確認該段內容，確認括號的完整，以及結尾記號(通常是";"分號) 是否有加上。

1.2 開始學習 java 前的準備

開始寫 java 前的準備工作

java 從 1995 年正式公開以後，迄今已經出到第六版（1.4 版完成後的下一個版本 1.5 就是我們平常聽到的第五版本），本次的網管教學第一階段以 java 1.4 版本為平台介紹，詳細的版本差異度可自行從網站上尋找，這裡教大家如何完成開始寫 java 前的準備工作。

Step 1 :

→ 首先先至 <http://java.sun.com/j2se/1.4.2/download.html> 處下載編譯 java 語言用的檔案。

→ 點選中間的：Download J2SE SDK

→ **Required** 的部份勾選 **Accept**

→ 選擇 **Windows Offline Installation**

Step 2 :

→ **更改環境變數**

(環境變數的設定位置：我的電腦(右鍵) \ 內容 \ 進階 \ 環境變數)

→ 在系統變數的地方找出 **Path** 變數，選擇編輯，在最後面加上
“ ;C:\j2sdk1.4.2_14\bin ”
(不含引號，分號後的內容為步驟一安裝的位置加上\bin)

→ 在系統變數的地方新增一個名為"CLASSPATH"(大寫)的變數
變數內容為"."(只有一個點)

Step over :

完成以上的設定後就可以搭配不同的編輯器做基本的 java coding 練習
下次上課會使用 JAVA 的 IDE (編譯器) eclipse 進行編輯。屆時也會提供新的設定教學給大家。

1.3 第一個 JAVA 程式

就像所有程式範例或是課程的第一個程式一樣，Hello~World~

java 語言的舞台，main 方法：

無論如何都得從此開始！

Main() 的架構：

→

```
public class HelloWorld {  
    public static void main (String[] argv){  
        modifier    modifier  ReturnType    identifier  Argument    identifier  
    }  
}
```

→ 在程式之中只能有一個 public class (或是 interface)

→ 完整的方法宣告：

```
<modifier>_<Return Type>_<identifier>_<Argument>{  
}
```

→ main 方法的架構為固定架構，不可更改

→ 程式在方法中執行

JAVA 線上 API 查詢：

<http://java.sun.com/j2se/1.4.2/docs/api/>

練習：

請寫出一個可以 print 出 "Hello world" 字串的 java 程式

提示：print 在 java 中的語法為 System.out.print(" ");

1.4 資料型別

基本型別：

整數

資料型別	最小值	最大值	佔用空間
byte	-128	127	1 個位元組
short	-32768	32767	2 個位元組
int	-2147483648	2147483647	4 個位元組
long	-9223372036854775808	9223372036854775807	8 個位元組
char	0	65535	2 個位元組

注意 char 代表的是 Unicode characters，例如輸入 98 跑出來的結果會是'b'，此外也要注意進位問題，16 進位的表示方法為 0X00 也就是在數字前面加上"0x"、8 進位的表示方法為 000 也就是在數字前面加上 0。因此當 java 中出現 int i = 077 這樣的數字時，代表的是 8 進位，也就是 10 進位的 63。

小數：

資料型別	可表示範圍	佔用空間
float	$+3.4028237 \times 10^{E38} \sim +1.30239846 \times 10^{-45}$	4 個位元組
double	$+1.76769313486231570 \times 10^{+308} \sim 4.94065645841246544 \times 10^{-324}$	8 個位元組

Boolean（布林值）：

true、false

參考型別：

String

字串，可用字串的功能 s.length() //用於算字串的長度

"\n" 換行

"\t" 跳格（類似 Tab 鍵的功能）

變數的宣告：

變數的宣告基本上是以下兩種形式

int i; // 先宣告變數名稱，之後再應用

int i = 100; or int i = j + k // 直接宣告變數的內容

變數的命名需要遵守以下規則

1. 變數的命名開頭必須要是英文

- 變數只能含有 "英文字母"、"數字"、"_"底線、"\$"錢號
- 變數不可以使用保留字(Reserved Word)

Reserved Word 包含

- keyword

abstract	boolean	break	byte	case
catch	char	class	const	continue
default	do	double	else	extends
final	finally	float	for	goto
if	implements	import	instanceof	int
interface	long	native	new	package
private	protected	public	return	short
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while		

- boolean 值 : true false

- null

常數的宣告:

ex : final int i = 100 ;

在宣告前面加上 final，之後就不能在修改。

1.5 運算子

運算式：

與一般程式語言沒有差別，需要注意的是，java 有所謂的資料型態，因此彼此之間不可以隨便轉型 比如，int 對 int 的除法，若沒辦法整除，他的結果就只會是商數，而如果要得到餘數，則需要使用%來得到，此外，在計算時的計算順序為等號右邊的先計算，計算完成後再放回左邊的變數名稱。 例如：

d = d + z ;

遞增與遞減運算：(ex_2) var++ 在本次變數使用完後遞增 ++var 在本行遞增 var-- --var	運算式讀取順序：由左到右 (ex_3) ex : int k = i ++ + j ; 複合運算子 i += 5 ; i = i + 5 ;
--	--

條件運算子 (ex_4)

判斷式 ? opr1 : opr2 ;

ex i = j % 2 == 1 ? 100 : 0 ;

1.6 迴圈

迴圈：

→for

```
for (初始 ; 條件 ; 控制 ) {  
    // 迴圈條件滿足時候的執行內容  
}
```

初始表示在迴圈開始前第一個執行的內容，只會執行一次 條件為判斷式，回傳的為 boolean 值 控制則為執行完該次迴圈的動做

→while

```
while (條件 ) {  
    // 迴圈條件滿足時候的執行內容  
}
```

注意 while 的作法是：先 check 迴圈"條件"的內容，然後才決定要不要執行該次迴圈，與下面的 do while 不同。

→do..while

```
do {  
    迴圈開始前先執行的內容  
}  
while (條件 )
```

do..while 的作法是先執行 do{}的工作，然後在判斷條件，若條件符合則繼續 do 的運作

練習：

請以 do while 完成九九乘法表

1.7 條件判斷

條件分支：

→if

```
if(條件 ) {  
    // 當條件為 true 時的執行內容  
}
```

→if..elseif..else..

```
if(條件 1) {  
    // 當條件 1 為 true 時的執行內容  
}else if (條件 3){  
    // 當條件 2 為 true 時的執行內容  
}  
else if (條件 2){  
    // 當條件 2 為 true 時的執行內容  
}else{  
    // 當以上條件皆不符合時執行內容  
}
```

→switch

```
switch (條件式) {  
    case 1 :  
        // case 1 成立時要執行的內容  
        break ;  
    case 2 :  
        // case 2 成立時要執行的內容  
        break ;  
    case 3 :  
        // case 3 成立時要執行的內容  
        break ;  
    default :  
}
```

case 後面的內容可以為 char、short、byte、int，若沒有 break，則會連下一個程式一起執行。不一定要有 default，若有 default 則不需要 break

練習：

判斷選擇何種幾何形狀,而後要求數值計算面積
要求以下執行功能:

請選擇幾何形狀:1.直角三角形 2.矩形 3.梯形 (使用者以數字選擇)

請輸入邊長(或者要求輸入底.高.寬.等等)

本幾何圖形之面積為:

(另外選擇各種形狀,若輸入數值不符合該形狀時要回報錯誤)

1.8 使用者輸入

本次上課中有許多範例皆有提供使用者輸入，在此簡介使用者輸入並將其轉為 int 的方法

使用者輸入：

→基本型別

```
import java.io.*; //在 public class 前面先打上這行

public class area {
    public static void main (String[] argv)
        throws IOException{ // 在 main 方法的括號前打上這段(throws IOException)
        //表示繼承 IOException 類別的功能

        // 可以在讓使用者輸入前先打上希望他們接收的資訊

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        // 加上上面這行

        String str = br.readLine(); //這行的用處是讀取使用者剛輸入的字串
        int name = Integer.parseInt(str); //這行的用處是將使用者剛輸入的字串轉為 int

        //進行其他 coding

        str = br.readLine(); //可以不只一次的要求輸入
        int name2 = Integer.parseInt(str);

        //其他 code

    }
}
```

Ch2. 學習 JAVA 的第二堂課 - 物件導向的程式

2.1 類別與方法 (一)

宣告類別：

(下面的範例皆以施威銘研究室 - JAVA2 程式語言 中之範例用來介紹，內文則為自撰文章)

```
class 類別名稱 {  
    // 描述類別的內容  
}
```

類別的宣告基本上如上所示，而在 class 前面也可以加上一些 modifier 例如之前提過的 public 等等，用來設定他的存取等級之類的特性。

以本次作業之部份為例子如下

```
class bird { //鳥的類別  
    int x, y; //設定鳥類別的特性  
    //這裡是設定鳥的座標為(x,y)  
}  
public class Shoot1 {  
    static public void main(String[] argv) {  
        //main()方法  
    }  
}
```

上面可以看出，一個 java 程式裡面不限定只有一個類別，同時可以有許多個類別存在，但是 public class 只能有一個

宣告方法：

方法可以說是類別的行為，像是...武器的類別就可以擁有攻擊之類的方法，而由類別產生出來的物件也都可以使用同樣的方法。通常在討論方法時都會在方法名稱後面加上()小括號來避免混淆

```
回傳值型別 方法名稱 ( 參數 ){  
    // 描述方法的內容  
}
```

上述即為方法的宣告格式，其中：

1. 回傳值型別 如果標上 void 就表示這個方法不會有運算結果。
2. 方法名稱 可自行定義，注意若和類別名稱相同則稱為建構方法。

3. 參數 方法輸入的時候可以設定參數，則在使用方法的時後於括弧中加入參數即可，

例如：計算面積(長，寬)

同樣以本次作業之部份為例子如下

```
class bird { //鳥的類別
    int x, y; //設定鳥類別的特性
    //這裡是設定鳥的座標為(x,y)

    void exploding(){ //加上告知鳥被摧毀的方法
        System.out.println("在"+x+", "+y+"座標的鳥被摧毀")
    }
}

public class Shoot1 {
    static public void main(String[] argv) {
        //main()方法
    }
}
```

上面可以看出，一個 java 程式裡面不限定只有一個類別，同時可以有許多個類別存在，但是 public class 只能有一個

類別的使用：

以本次作業為例，目標物(target)類別已經設定好了，接下來試著在 main 方法產生一個目標物之物件。

以類別產生物件的格式如下

```
類別名稱 物件名稱 = new 類別() ;
```

而要為類別產生出來的物件設定參數或是使用其方法連結的方式為 "." (點)，例如：

```
class bird { //設置鳥的類別
    int x, y; //設定鳥類別的特性
    //這裡是設定鳥的座標為(x,y)

    void exploding(){ //告知被炸的方法
        System.out.println("在"+x+", "+y+"座標的鳥被炸了")
    }
}
```

```

}
}

public class Shoot1 {
    static public void main(String[] argv) {
        bird i = new bird (); //宣告產生一個名為 i 的 bird(鳥)

        i.x = 2; //使用物件(設定 x 座標)
        i.y = 3; //使用物件(設定 y 座標)

        i.exploding(); //使用物件的方法(摧毀)
    }
}

```

以上僅示範產生一個物件，實際上設定好類別後可以產生多個物件，例如用同樣的方法設定多個目標物，再加上條件即可，而各物件之間的參數等等皆不互相抵觸。同樣的也可以以陣列來指向多個物件。

範例：

以下為本次作業極度簡化版本的範例，讓大家從中了解整個物件導向的運作

```

import java.io.*;

class bird { // 鳥
    int x,y; // 鳥位置

    void exploding() {
        System.out.println("在(" + x + "," + y +")的鳥被摧毀");
    }
}

class bullet { // 子彈
    int x,y; // 子彈位置
    int stepX,stepY; // 子彈移動速度(向量)

    void move() { // 依移動速度到下一位置
        x += stepX;
        y += stepY;
        System.out.println("子彈移到(" + x + "," + y + ")");
    }
}

class hunter { // 獵人
    int x,v; // 獵人位置

```

```

bullet fire(int stepX,int stepY) { // 產生一顆子彈
bullet c = new bullet();
c.stepX = stepX;
c.stepY = stepY;
c.x = x;
c.y = y;

return c;
}
}

class Battlefield { // 戰場
int width,height; // 戰場大小
bird i; // 在戰場中放上鳥 i
hunter f; // 在戰場中放上獵人 f
bullet c; // 在戰場中放上子彈 c

boolean hitbird(bullet c) { // 是否擊中鳥
if((c.x == i.x) && (c.y == i.y)) { // 若擊中
i.exploding(); // 鳥爆炸
i = null; // 清除鳥 i
return true; // 傳回擊中了
}
return false; //否則 傳回沒有擊中
}

boolean outOfField(bullet c) {
if(c.x < 0 || c.y < 0) // 超過左邊界或上邊界
return true;
if(c.x >= width || c.y >= height) // 超過右邊界或下邊界
return true;
return false;
}
}

public class Shooting {

public static void main(String[] argv) throws IOException{
Battlefield b = new Battlefield(); // 產生戰場
b.width = 5; // 設為 5X5 的大小
b.height = 5;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
// 用來讀取輸入資料

```

```

// 產生鳥
b.i = new bird();

// 取得使用者輸入的鳥位置
System.out.println("請輸入鳥的位置：");
System.out.print("x->");
b.i.x = java.lang.Integer.parseInt(br.readLine());
System.out.print("y->");
b.i.y = java.lang.Integer.parseInt(br.readLine());

// 產生獵人
b.f = new hunter();

// 取得使用者輸入的獵人位置
System.out.println("請輸入獵人的位置：");
System.out.print("x->");
b.f.x = java.lang.Integer.parseInt(br.readLine());
System.out.print("y->");
b.f.y = java.lang.Integer.parseInt(br.readLine());

// 取得使用者輸入的子彈移動速度
System.out.println("請輸入子彈的移動速度：");
System.out.print("x->");
int x = java.lang.Integer.parseInt(br.readLine());
System.out.print("y->");
int y = java.lang.Integer.parseInt(br.readLine());

// 發射子彈
b.c = b.f.fire(x,y);

do { // 移動子彈，並測試是否越過邊界
b.c.move();
if(b.outOfField(b.c) || b.hitbird(b.c))
b.c = null; // 清除子彈
} while (b.c != null);
System.out.println("遊戲結束！");
}
}

```

2.2 類別與方法（二）

變數的有效範圍：

--> 區域變數

宣告在區域內的變數，僅能在該區域內生效，而注意內層的不能與外層重複宣告。以下為例

```
public class Scope {  
  
    public static void main(String[] argv){  
        int x = 3;  
  
        {  
            int y = 8;  
  
            {  
                int z = 10;  
  
                System.out.println("x = " + x); // 最外層的 x  
                System.out.println("y = " + y); // 上一層的 y  
                System.out.println("z = " + z);  
                System.out.println("");  
            }  
            int z = 8;  
  
            System.out.println("x = " + x); // 最外層的 x  
            System.out.println("y = " + y);  
            System.out.println("z = " + z);  
            System.out.println("");  
        }  
  
        int y = 3;  
        int z = 3;  
        System.out.println("x = " + x);  
        System.out.println("y = " + y);  
        System.out.println("z = " + z);  
    }  
}
```

--> 成員變數

類別中的成員也有變數，一旦類別產生出物件後，該物件即具有其成員變數，而方法可以使用其所屬類別的成員變數(即使該變數較晚宣告)。但是若方法中也以同樣名稱宣告變數的話，則會造成名稱遮蔽。防止被遮蔽的方法可以使用 `this` 變數名稱

ex :

```
class Test {
    int x = 10;
    int y = 10;

    void show(int x){           // 遮蔽成員 x
        int y = 20;           // 遮蔽成員 y
        System.out.println("x = " + x);
        System.out.println("y = " + y);
    }
}

public class Shadowing {

    public static void main(String[] argv){
        Test a = new Test();
        a.show(20);
    }
}
```

如此跑出來的結果即為 x = 20、y=20，若使用 this.如下

```
class Test {
    int x = 10;
    int y = 10;

    void show(int x){           // 遮蔽成員 x
        int y = 20;           // 遮蔽成員 y
        System.out.println("x = " + this.x);
        System.out.println("y = " + this.y);
    }
}

public class UsingThis {
    public static void main(String[] argv){
        Test a = new Test();
        a.show(20);
    }
}
```

則跑出來的結果即為 x = 10、y=10，若使用 this.如下

--> for 迴圈中的變數

如果在 for 迴圈外就已經設置變數名稱，則 for 結束後亦可使用其結果但若是 for() 括弧內的初始值才宣告，那麼當迴圈結束後再使用就會發生錯誤（見範例檔案 ex_forright、ex_forerror）。

2.3 類別與方法（三）

方法的多重定義：

--> 以 `System.out.println` 為例

之前上課常常使用到的 `System.out.print()`，括弧內的參數不只可以使用一種，可以是字串也可以是整數，這種方法擁有多重定義(overloading)的功能，但若自己設立的方法也需要有多重定義該怎麼辦呢？方法如下

同名方法 --> 即以同樣的名稱，不同的參數內容定義方法，以下為例

```
class Test {  
  
    // 1 號版本：使用寬與高  
    int rectangleArea ( int width , int height) {  
        return width * height;  
    }  
  
    // 2 號版本：使用座標  
    int rectangleArea ( int top , int left , int bottom , int right) {  
        return (right - left + 1) * (bottom - top + 1);  
    }  
}  
  
public class Overloading {  
    public static void main(String[] argv){  
        Test a = new Test();  
        int area;  
  
        area = a.rectangleArea(10,20);  
        System.out.println("矩形面積：" + area);  
  
        area = a.rectangleArea(1,1,10,20);  
        System.out.println("矩形面積：" + area);  
    }  
}
```

要注意的是，多重定義方法時，不同版本的同名方法參數要不同，至少要有一個參數，無論是型別、個數之中要不一樣，才能讓程式判斷。

建構方法(Constructor)：

前面提到在類別中方法與類別同名會被視為建構方法，建構方法即是在類別產生物件後自動被呼叫，建構物件的初始狀態。將前面的作業範例加入建構子即變成如下，範例為幫 `bullet` 類別加上建構子，另外在讓其可以自動判斷是否擊中目標。

```

class Bullet { // 子彈
    int x,y; // 子彈位置
    int stepX,stepY; // 子彈移動速度及方向
    Battlefield b; // 戰場

    Bullet(int x,int y,int stepX,int stepY,Battlefield b) {
        this.x = x;
        this.y = y;
        this.stepX = stepX;
        this.stepY = stepY;
        this.b = b;
    }

    Bullet move() { // 依據移動速度換到下一位置
        x += stepX;
        y += stepY;
        System.out.println("子彈移到(" + x + "," + y + ")");
        return (b.outOfField(this) || b.hitBird(this)) ? null : this;
        //判斷是否擊中或是超出戰場
    }
}

//剩餘部份見附檔 Shooting2

```

2.4 變數

變數宣告的方法：

--> 使用 java.io 套件

--> 變數實際上是用特定演算法產生出不規則的數字，並非真正的亂數

--> Math.random() 產生出來的為一個介於 0 與 1 之間(≥ 0 , < 1) 的浮點數

因此若要產生整數亂數則需要在前面加上 int，並在後面乘上範圍

以下為例

```

import java.io.*; //使用 java.io 套件下的類別

public class randtest {

    public static void main(String[] argv) throws IOException{ //繼承 IOException

        int a =(int)(Math.random()*4); // 限定範圍為 0~3，並且將浮點數型別轉成 int 型別

        System.out.println(a);
    }
}

```

2.5 作業與範例

本次作業

自己打自己的無聊小遊戲

(請應用 類別、方法、建構方法等等內容來寫)

要求:

1. 可以亂數設定目標(鳥)座標(整數座標) (範圍: $x=0\sim 10$ $y=0\sim 10$)
2. 戰場座標 $x=0\sim 10$ $y=0\sim 10$
3. 設定自己的座標
4. 設定發射子彈的角度(用向量)
5. 設定子彈彈若擊中小鳥則回傳擊中目標的訊息,若沒有擊中小鳥則讓使用者繼續發射子彈、或是更改自己的位置後發射子彈。若移動自己位置到小鳥所在位置則回傳自爆訊息。
6. 程式內容要判斷子彈已經超過目標物範圍

Ch3. 學習 JAVA 的第三堂課 - 基礎的網路程式

3.1 基礎網路程式 (一)

server+client

基礎的網路程式：

網路程式單是就運作方式等等來劃分就可以分為許多種類，在本次網管上課只跟大家介紹最簡單的一種--> 分別設立 Server 端以及 Client 端來進行連結。

第一個範例的基本訴求如下：

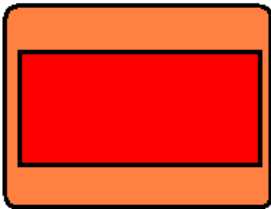
Server 端：由指令參數輸入執行埠(port)，以執行埠建立伺服器平台，等待 Client 端連結，並且讀取其網址(ip)。

Client 端：由指令參數輸入 server 之網址(IP)及執行埠(port)，以 server 之網址(IP)及執行埠(port)建立網路串流連接平台 連結 server 端

Java 系統本身內建有網路伺服平台的 class --> ServerSocket，使用此類別就可以簡單的設定出一個網路伺服器。驅動的語法如下：

<pre>SS = new ServerSocket(port); //SS 為伺服網站的名稱(自訂) //port 為設定的連接埠(int、自訂)</pre>	
---	---

設定完成後便可以設定網路連接平台，語法如下：

<pre>socket = SS.accept(); //socket 為網路連接平台的名稱(自訂) //SS 為伺服網站的名稱(自訂)</pre>	
---	---

完成上述設定後，Server 端的程式大致上就算設定完成了，記得在與 Client 端的連接結束後使用 `socket.close()` 方法來關閉網路平台。而 Client 不需要建立伺服器網站，因此在程式中建立與 Server 端連結的網路平台即可，程式碼如下：

<pre>socket=new Socket(InetAddress.getByName(servername),port); //以前面取得的 servername 以及 port 建立連線</pre>	
---	---

完成上述設定後，Client 端便已經與 server 端建立連線了，同樣在與 Server 端的連接結束後使用 `socket.close()` 方法來關閉網路平台。

上面要注意的是，大部分的語法都是使用 java 內建的語法，java 本身其實尚有許多套件可以應用在各方面，請自行鑽研。

範例程式：

Server 端程式（見範例檔案 ServerEx_1.java）

```
import java.net.*;                //加入 JAVA 內建網路套件
import java.io.*;                 //加入 java 內建 IO 套件

public class ServerEx_1 {
    ServerSocket SS;              //在 public class 下的變數 1 : SS（伺服器網站名稱）
    Socket socket;                //同上，socket(網路平台的名稱)
    static int port;              //用來設定自己 port 用的變數

    public ServerEx_1() {        //注意這是建構子
        try {                    //try...catch 語法是爲了捕捉 exception
            SS = new ServerSocket(port); //以 port 產生一個伺服器
            System.out.println("Server created."); //告知已經產生好了
            System.out.println("waiting for client to connect...");
            //告知可以給人家連接了
        }
    }
}
```

```

        socket = SS.accept();           //設定網路平台可以開始接收
        System.out.println("已經和 "+位址+"連接");
        //這裡是設定告訴自己對方的名稱
        socket.close();                 //停止接收
    }
    catch(IOException){                 //try...catch 語法是爲了捕捉 exception
        System.out.println(e.toString());
        e.printStackTrace();
        System.exit(1);                 //如果發生 exception 錯誤直接結束
    }
}

public static void main(String args[]){
    if(args.length < 1){               //首先先判斷指令參數輸入是否正確
        System.out.println("請用: java ServerEx_1 [port] 這樣的格式執行");
        System.exit(1);                 //如果輸入格式錯誤直接結束
    }
    port=Integer.parseInt(args[0]);     //讀取指令參數的值當成 port
    ServerEx_1 ServerStart=new ServerEx_1(); //以類別產生出一個網路平台物件
}
}

```

Client 端程式 (見範例檔案 ClientEx_1.java)

```

import java.io.*;
import java.net.*;

public class ClientEx_1 {
    Socket socket;           //設定變數(自己的網路平台名稱)
    static String servername; //設定變數(欲連接的網路平台名稱)
    static int port;         //設定 port

    public ClientEx_1() {    //注意這是建構子
        try{
            socket=new Socket(InetAddress.getByName(servername),port);
            //以前面取得的 servername 以及 port 建立連線
        }
    }
}

```

```

        socket.close();
    }
    catch(IOException e){
        System.out.println("IOException when connecting Server!");
    }
}

public static void main(String args[]) {
    if (args.length < 2){
        System.out.println("USAGE: java Client09 [servername] [port]");
        System.exit(1);
    }
    servername= args[0];
    port=Integer.parseInt(args[1]);
    ClientEx_1 ClientStart=new ClientEx_1();
}
}

```

3.2 基礎網路程式（二）

server+client 訊息傳遞

簡單的訊息傳遞：

前面介紹了建立網路伺服器以及網路連線平台的方法，建立好網路連線平台後，便可以以網路串流的類別來進行串流物件的傳送，這部份 server 和 client 的語法是一樣的，只需就已有的類別新增出物件，再使用即可

輸入（或者說接收串流）的語法如下：

```

instream = new DataInputStream(socket.getInputStream());
// 首先以 DataInputStream 類別新增一個物件，名為 instream
// 此物件的內容是 socket 這個連線平台接收到的串流
// 讀取某連線平台接收到之串流的方法為 getInputStream()

messagein = instream.readUTF();
// 將 instream 讀到的串流存入 messagein 這個變數中
// (變數名稱自訂，但是要注意變數的資料型別)

```


輸出 (或者說傳送串流) 的語法如下：

```
outstream = new DataOutputStream(socket.getOutputStream());
// 首先以 DataOutputStream 類別新增一個物件，名為 outstream
// 此物件的內容是 socket 這個連線平台要傳送的串流

outstream.writeUTF(messageout);
// messageout 變數的內容放入 out 串流並傳送
```

範例程式：

Server 端程式 (見範例檔案 ServerEx_2.java)

```
import java.net.*;
import java.io.*;
import java.util.*;

public class ServerEx_2 {
    ServerSocket SS;
    DataOutputStream outstream;
    DataInputStream instream;
    Socket socket;
    static int port;
    static String messageout;
    static String messagein;
    Date currentDate;

    public ServerEx_2() {
        try{
            SS = new ServerSocket(port);
            currentDate = new Date();

            System.out.println("Server created.");
            System.out.println("waiting for client to connect...");
```

```

        while(true){
            socket = SS.accept();

            System.out.println("現在與 "
+socket.getInetAddress().getHostAddress()+"連線");
            System.out.println("對方的網路名稱爲:"
+socket.getInetAddress().getHostName());
            System.out.println("連線時間是:" + currentDate);

            instream = new DataInputStream(socket.getInputStream());
            messagein = instream.readUTF();
            System.out.println("對方傳送訊息: " + messagein);

            System.out.println("waiting...");

            instream = new DataInputStream(socket.getInputStream());
            messagein = instream.readUTF();
            System.out.println("message: " + messagein);

            outstream = new DataOutputStream(socket.getOutputStream());
            outstream.writeUTF(messageout);
            outstream.close();

        }
    }
    catch(IOException e){
        System.out.println(e.toString());
        e.printStackTrace();
        System.exit(1);
    }
}

public static void main(String args[]){
    if(args.length < 2){
        System.out.println("Usage: java Server11 [port] [messageout]");
    }
}

```

```

        System.exit(1);
    }
    port=Integer.parseInt(args[0]);
    messageout = args[1];
    ServerEx_2 ServerStart=new ServerEx_2();
    }
}

```

Client 端程式（見範例檔案 ClientEx_1.java）

```

import java.io.*;
import java.net.*;

public class ClientEx_2 {
    Socket socket;
    DataOutputStream outstream;
    DataInputStream instream;
    String messagein;
    static String messageout;
    static String servername;
    static int port;

    public ClientEx_2() {
        try{
            socket=new Socket(InetAddress.getByName(servername),port);

            outstream = new DataOutputStream(socket.getOutputStream());
            outstream.writeUTF(messageout);

            instream=new DataInputStream(socket.getInputStream());
            messagein = instream.readUTF();
            System.out.println("對方傳送的訊息為： " + messagein);

            socket.close();
        }
        catch(IOException e){

```

```

System.out.println("IOException when connecting Server!");
}
}
public static void main(String args[]) {
if (args.length < 3){
System.out.println("USAGE: java ClientEx_2 [servername] [port] [messageout]");
System.exit(1);
}
servername= args[0];
port=Integer.parseInt(args[1]);
messageout = args[2];
ClientEx_2 ClientStart=new ClientEx_20;
}
}

```

3.3 作業與範例

本次作業 5/4

對打的無聊小遊戲(上次作業加上網路功能)

(請應用 類別、方法、建構方法以及網路等等內容來寫)
(作業不要求必須使用上課所學，但限定使用 java 編寫)

本次要求:

1. 本次作業需上傳 2 個(含)以上的檔案，程式內容分別要有 server 與 client 端的遊戲程式 (若寫出功能更強大的可以加分!!) (若可以不用區分 server 與 client 端程式，則不受以下 2~5 限制)
2. 所撰寫的 client 端程式要可以連線到對方程式，並且進行遊戲
3. 遊戲方式為，兩人設定好自己的座標以後，將自己的座標傳送給對方並開始遊戲，先擊敗對手的勝利。
4. 本次作業分數視實際撰寫出遊戲功能之"效能"、"功能" 而評分
5. 參考檔案：[亂槍打鳥 online](#)

基本要求:

1. 可以亂數設定目標(鳥)座標(整數座標) (範圍: x=0~10 y=0~10)
2. 戰場座標 x=0~5 y=0~5
3. 設定自己的座標
4. 設定發射子彈的角度(用向量)
5. 設定子彈彈若擊中小鳥則回傳擊中目標的訊息若沒有擊中小鳥則讓使用者繼續發射子彈、或是更改自己的位置後發射子彈。若移動自己位置到小鳥所在位置則回傳自爆訊息
6. 程式內容要判斷子彈已經超過目標物範圍