

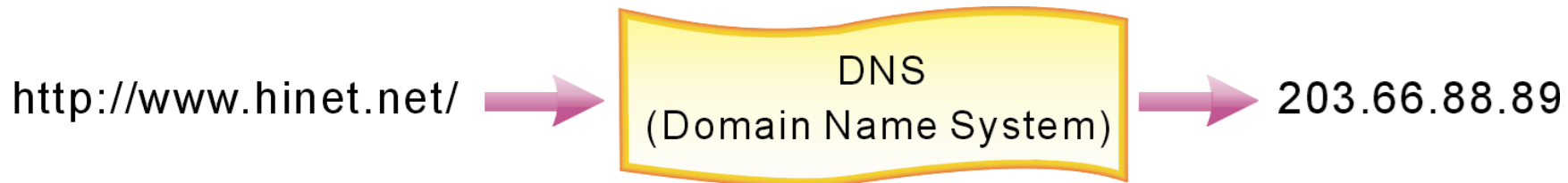


第18章 網路程式設計

網路基本觀念

- 網路(**Network**)是指「資訊交流的管道」，例如通訊系統、郵件系統都可以看見網路的影子。底下列出幾個常見的名詞：
 - **(1)IP(Internet Protocol)**：IP位址就好比是電腦的身分證字號，每一個網域名稱(**Domain Name**)都會對應一個IP位址(**IP Address**)，IP位址是由一串的數字所組成。例如中華電信IP位址：**203.66.88.89**。
 - **(2)TCP(Transmission Control Protocol)**：TCP提供了一套協定，能夠將電腦之間使用的資料透過網路相互傳送，同時也提供一套機制來確保資料傳送的準確性和連續性。

- **(3)UDP(User Datagram Protocol) : UDP**是一個非連線式(**Connectionless**)的非可靠傳輸協定，並不會運用確認機制來保證資料是否正確的被接收、或重傳遺失的資料。
- **(4)DNS(Domain Name System) : DNS**主要的功能是幫忙解析主機名稱與找出對應的IP位址。



- (1)處理IP(Internet Protocol)位址與網域名稱、網路主機：
 - **InetAddress**：處理主機名稱及IP Address。
 - **Inet4Address**
 - **Inet6Address**
- (2)關於URL(Uniform Resource Locator)通訊協定問題：
 - **URL**：處理URL(Uniform Resource Locator)並下載URL的相關資料。
 - **URLConnection**

- (3)關於**TCP(Transmission Control Protocol)**通訊協定問題：
 - **Socket**：處理TCP(Transmission Control Protocol)通訊協定。
- (4)關於**UDP(User Datagram Protocol)**通訊協定問題：
 - **DatagramSocket**：處理UDP(User Datagram Protocol)通訊協定。
- (5)伺服器的使用問題：
 - **ServerSocket**：提供伺服器端使用。

- 關於java.net套件中所有類別與介面，列表如下：

Authenticator	InetSocketAddress	SocketAddress	JarURLConnection
SocketImpl	MulticastSocket	SocketPermission	NetPermosion
ContentHandler	URI	URL	Networkinterface
DatagramPacket	DatagramSocket	PasswordAuthentic ation	URLClassLoader
URLConnection	DatagramSocketImp l	URLDecoder	HttpURLConnection
URLEncoder	InetAddress	Inet4Address	Inet6Address
Socket	ServerSocket	URLStreamHandler	

J2SE 5 新增部份

CacheRequest	CacheResponse	Proxy	ProxySelector
ResponseCache	SecureResponseCache		
ContentHandlerFactory	SocketImplFactory	DatagramSocketFactory	URLStreamHandlerFactory
FileNameMap	SocketOptions		

IP Address的介紹



- IP位址是一個長度為32位元(Bits)的二進位數值，由0與1來組成，對一般人而言，實在不容易明瞭。
- 為了方便使用，以8位元為一個區段，分為四個區段，各個區段是0~255的數字，區段與區段之間必須以小數點(dot)來隔開；例如192.18.97.36。

InetAddress類別(1)

- InetAddress類別方法

方法	說明
<code>static InetAddress getLocalHost()</code>	用來取得主機名稱。
<code>static InetAddress getByName(String host)</code>	依照網域來建立一個主機名稱。
<code>static InetAddress[] getAllByName (String host)</code>	用來取得主機名稱，以陣列方式傳回所有IP位址。
<code>static InetAddress getByAddress (byte[] addr)</code>	以InetAddress物件傳回IP位址陣列。
<code>static InetAddress getByAddress (String host, byte[] addr)</code>	依照網域和位址陣列來建立一個InetAddress物件。
<code>String getHostAddress()</code>	以字串方式取得IP位址。
<code>String getHostName()</code>	以輸入的IP位址來取得網址。

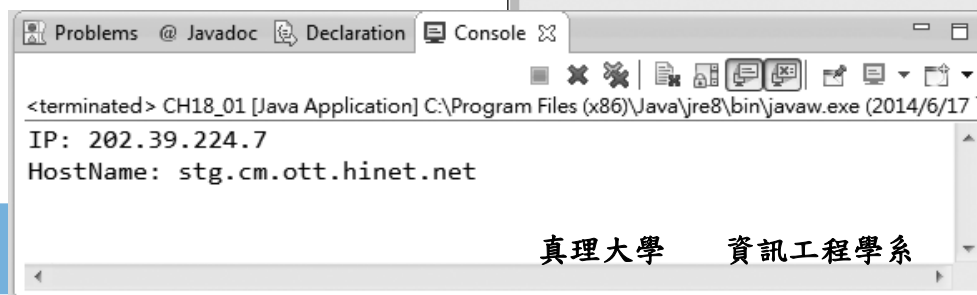
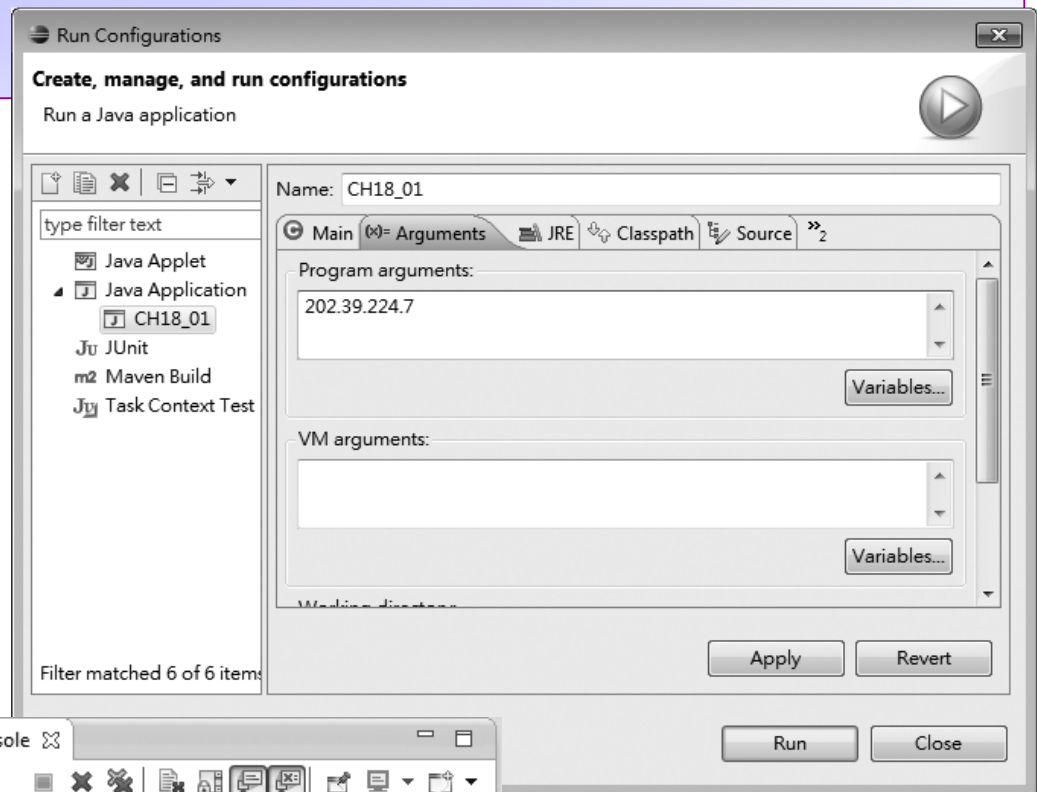
InetAddress類別(2)

- 範例程式：CH18_01】使用者輸入網址，並傳回IP位址

```
01 /* 程式：CH18_01.java
02 * 說明：使用者輸入網址，並傳回IP位址
03 */
04
05 import java.net.*; //匯入java.net
06 public class CH18_01{
07     public static void main(String args[]){
08         if(args.length == 0){
09             System.out.println("請輸入IP位址或網址");
10             System.exit(1);
11         }
12         String host = args[0];
13         try{
14             InetAddress inet = InetAddress.getByName(host);
15             System.out.println("IP: " + inet.getHostAddress());
16             System.out.println("HostName: " +
17                 inet.getHostName());
18         }
```

```
18 catch(UnknownHostException e){ //使用者輸入一個未被支援的網路連結
19 System.out.println("Could not find: ' " + host + "'");
20 }
21 }
22 }
```

• 程式編譯結果



• 程式解析

第08~11 行：建立一個判斷式，用來判斷使用者是否有輸入字串，如果使用者沒有輸入IP 位址或網址時，會顯示第9 行的提示訊息。

第13~20 行：進行UnknownHostException 例外處理，如果使用者輸入一個錯誤的網址或IP 位址，會顯示第19 行的錯誤訊息。

第14 行：取得輸入的主機名稱。

第15 行：取得輸入的IP 位址。

第16 行：如果是輸入IP 位址，轉換為網址。

InetAddress類別中靜態(static)的方法(1)

- 因為InetAddress類別沒有提供建構子(constructor)，所以要使用InetAddress類別，需透過類別內所提供的方法(method)，直接呼叫，進而建立InetAddress類別物件。

靜態(static)的方法名稱	使用說明
static InetAddress[] getAllByName (String host)	根據所給定的主機名稱(host name)，找出所有主機的 IP位址。
static InetAddress getByName (String host)	根據所給定的主機名稱(host name)，找出主機的 IP位址。
static InetAddress[] getLocalHost ()	找出使用端電腦的主機名稱(host name)和 IP位址。

InetAddress類別中靜態(static)的方法(2)

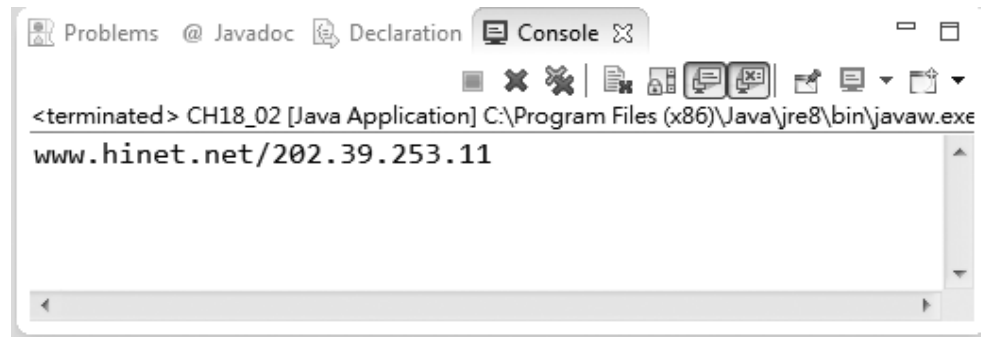


- 範例程式：CH18_02

```
01 /* CH18_02：實作靜態（static）方法
02 */
03 import java.net.*;
04 class CH18_02{
05 public static void main (String args[]){
06 try{
07 InetAddress address = InetAddress.getByName("www.hinet.net");
08 System.out.println(address);
09 }catch (UnknownHostException e){
10 System.out.println("找不到 www.hinet.net");
11 }
12 }
13 }
```

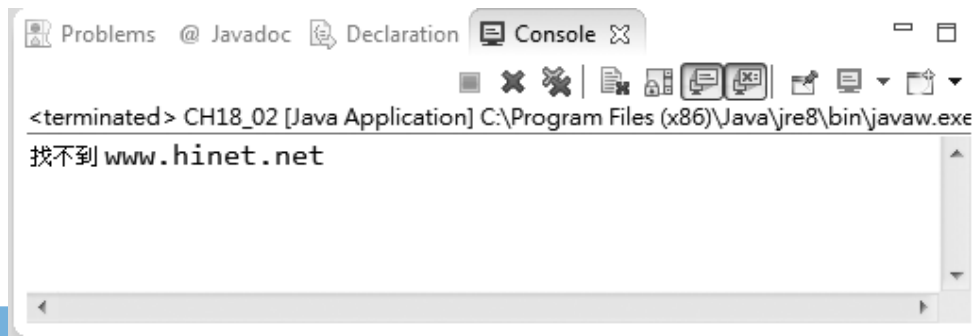
- 程式編譯結果

- (1)找到網址：「www.hinet.net」



```
<terminated> CH18_02 [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe
www.hinet.net/202.39.253.11
```

- (2)未找到網址：「www.hinet.net」



```
<terminated> CH18_02 [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe
找不到 www.hinet.net
```

- 程式解析

第7 行：建立InetAddress 物件，使用靜態方法中的「getByName」，希望輸出的結果是「找出主機的IP 位址」。

第9 行：丟出一例外訊息

「UnknownHostException」，如果網路連線情形是「斷線」或者「主機根本不存在」，則會出現「找不到 www.hinet.net」的錯誤訊息。

InetAddress類別中非靜態的方法(1)

- **InetAddress**類別內除了靜態的方法之外，還有提供非靜態的方法，下表列出**InetAddress**類別中非靜態的方法：

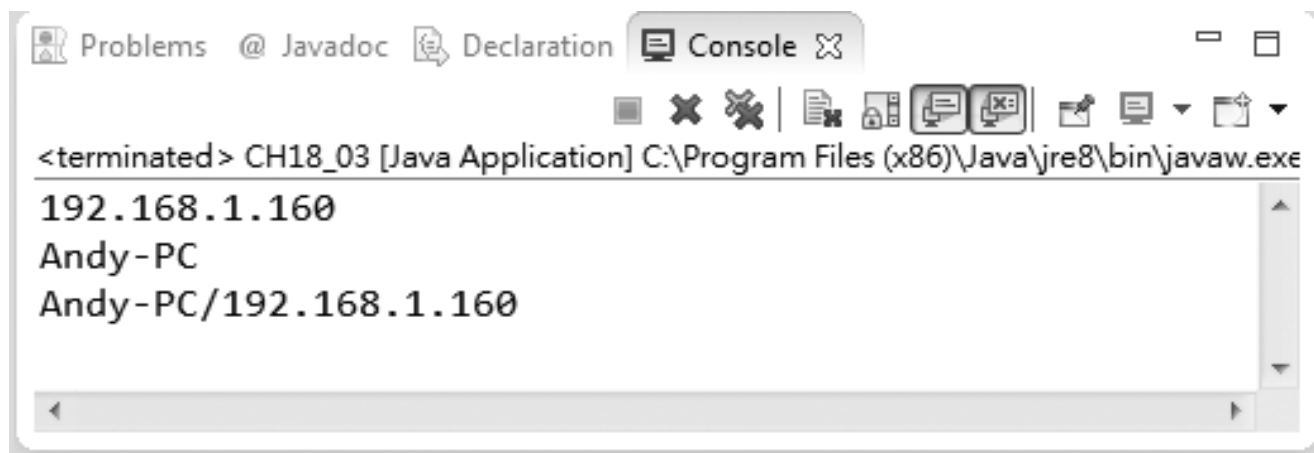
非靜態(non-static)的方法名稱	使用說明
String getHostAddress ()	回傳主機的 IP 位址。
String getHostName ()	回傳主機名稱。
String toString ()	回傳訊息字串，此字串將列出主機名稱和 IP 位址。
Boolean equal (Object other)	如果位址和 other 相同則回傳 true ，反之不相同則回傳 false 。
byte[] getAddress ()	依照網路位元組的順序，回傳所代表的網路位址

InetAddress類別中非靜態的方法(2)

- 範例程式：CH18_03

```
01 /* CH18_03：實作非靜態（non-static）方法 */
02 import java.net.*;
03 class CH18_03{
04 public static void main (String args[]){
05 try{
06 InetAddress address = InetAddress.getLocalHost();
07 System.out.println(address.getHostAddress());
08 System.out.println(address.getHostName());
09 System.out.println(address);
10 }catch (UnknownHostException e){
11 System.out.println("找不到 address");
12 }
13 }
14 }
```

- 程式編譯結果



The screenshot shows a console window with the following text:

```
<terminated> CH18_03 [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe
192.168.1.160
Andy-PC
Andy-PC/192.168.1.160
```

- 程式解析

第6行：建立InetAddress物件，使用非靜態方法「getLocalHost」，LocalHost指的是本機，因此是要取的本機的相關資訊。

第7行：取得本機（Local）的IP位址。

第8行：取得本機（Local）的電腦名稱。

第9行：取得本機（Local）的電腦名稱及IP位址。

第10行：丟出一例外訊息「UnknownHostException」，如果網路連線情形是「斷線」或者「主機根本不存在」，則會出現「找不到address」的錯誤訊息。

以Socket來建立通訊

- 透過主從架構(client-server model)的概念，得知在軟體層面的通訊管道上，一般分為「連線」與「非連線」二種方式，分述如下：
 - － 連線導向：伺服器端(Server)與客戶端(Client)必須先建立連線，才能進行資料的傳送。
 - － 非連線導向：伺服器端或客戶端只要指定接收的位址，就能將資料直接送出。



- **Stream通訊(TCP/IP通訊)：**

- TCP是以連線導向為協定，表示雙方必須先建立連線才能進行通訊。
- 它是一種保證傳送的協定，接收端於接收資料後會進行資料的確認，如果被傳送的資料在中途遺失或有毀損，會進行重送的程序；若是順序不對，也會在進行重組前，修正為正確順序。

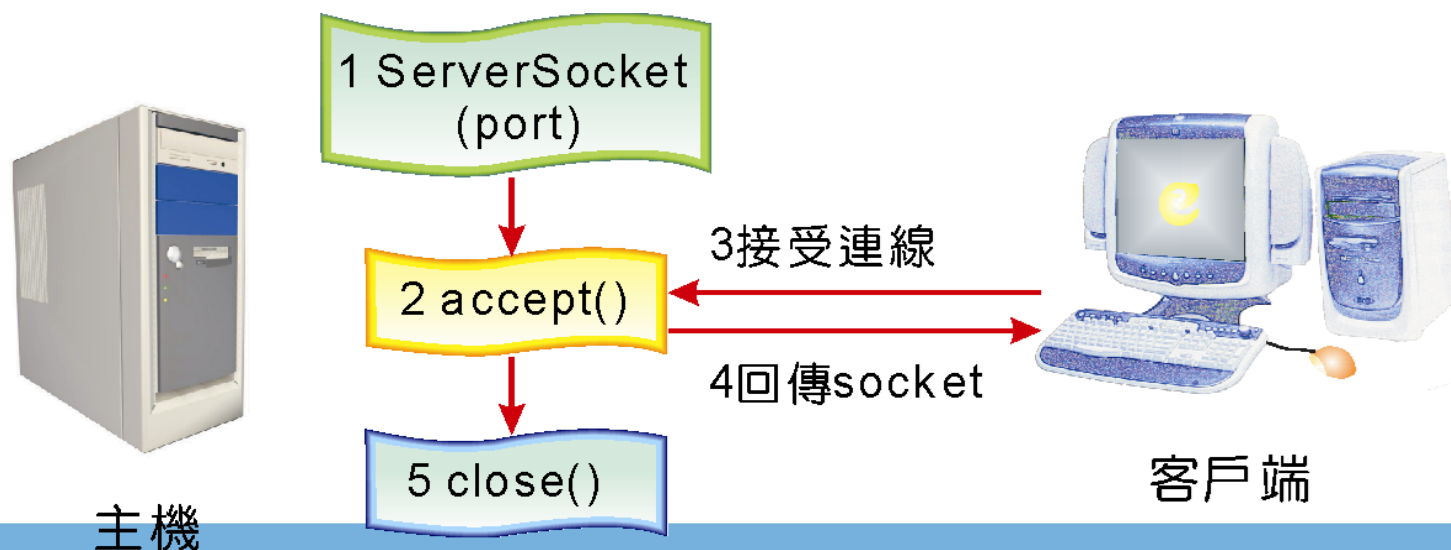
- **Datagram通訊(UDP通訊)：**

- UDP使用非連線導向的協定，表示雙方的資料是獨立的。
- 與TCP不同的是，它是一種不可靠的傳送協定，當它進行資料的傳送時，並不會保證所有的資料都會送達，所以它的傳送速度會優於TCP。

Socket應用程式



- Java的Socket應用程式若以TCP/IP通訊協定為主時，包含了伺服器端(Server)和客戶端(Client)。
- 對Java而言，ServerSocket類別是針對伺服器端來處理，這意味著伺服器端得聆聽客戶端的連結請求。伺服器端的Socket，其執行流程如下面圖例所示：



- 建立一個伺服端的Socket應用程式，執行步驟如下：
 - 先建立伺服端的ServerSocket物件，並指定聆聽的通訊埠。
 - 使用accept()方法來接受客戶端的連線請求(Connection Request)。
 - 伺服端會依照客戶端之請求，建立客戶端之Socket物件，讓伺服端與客戶端進行Socket通訊連線。
 - 處理客戶端的請求(稱為Request)，將處理的結果或錯誤訊息以Socket物件回傳。
 - 處理完畢後，關閉Socket通訊連線。

伺服器端與Socket(1)

- 伺服器端使用ServerSocket類別，我們利用下列的建構子來建立Socket物件：

建構子	說明
ServerSocket()	建立一個未連結的Socket。
ServerSocket(int port)	建立繫結時，指定一個未被使用的port。
ServerSocket(int port, int backlog)	建立繫結時，指定一個未被使用的port，並設定連入本機的連結數。
ServerSocket(int port, int backlog, InetAddress bindAddr)	建立繫結時，指定一個未被使用的port，設定連入本機的連結數和本機的IP位址。

- **backlog** : ServerSocket用來設定Client端的連線數，其預設最大值為50，也可以自訂數值來改變此參數值。
- **bindAddr** : 當我們建立ServerSocket時，會以本機(Local)的IP位址為伺服器端，作為Socket所需的IP位址；如果Local主機有一個以上的IP時，可利用bindAddr來指定其參數值。

- 用來取得伺服器端Socket的常用方法如下：

方法	說明
Socket accept()	產生一個新的Socket，用來等待Client端的連線請求。
InetAddress getInetAddress	傳回Socket連線時的主機位址。
int getLocalPort()	傳回Socket接受連線時的port。
ServerSocket getLocalSocketAddress()	傳回本機的SocketAddress物件，如果傳回null時，代表尚未進行連結。
void close()	關閉Socket。
Boolean isClosed()	用來判斷Socket是否在關閉狀態。
void setSoTimeout(int timeout)	設定accept()等待的時間。
void setTcpNoDelay(Boolean on)	以true的方式來關閉使用的buffering。
void setReceiveBufferSize(int size)	用來增加buffer的大小，以提高連線速度。
void setSendBufferSize(int size)	增加傳送buffer的大小。

伺服器與Socket(2)

- 範例程式：CH18_04建立伺服端的應用程式

```
01 /* 程式：CH18_04.java
02 * 說明：建立伺服端的應用程式
03 */
04
05 import java.net.*;
06 import java.io.*;
07
08 public class CH18_04{
09
10 public static void main(String args[]) throws Exception{
11 goServer server;
12 int port;
13 BufferedReader reader;
14 PrintWriter writer;
15 //取得通訊埠，如果沒有取得則結束程式的執行
16 if(args.length == 0){
17 System.out.println("請輸入伺服端的埠號[port]");
18 System.exit(1); //結束執行的執行
19 }
20 port = Integer.parseInt(args[0]); //將輸入的port轉換為數值
```

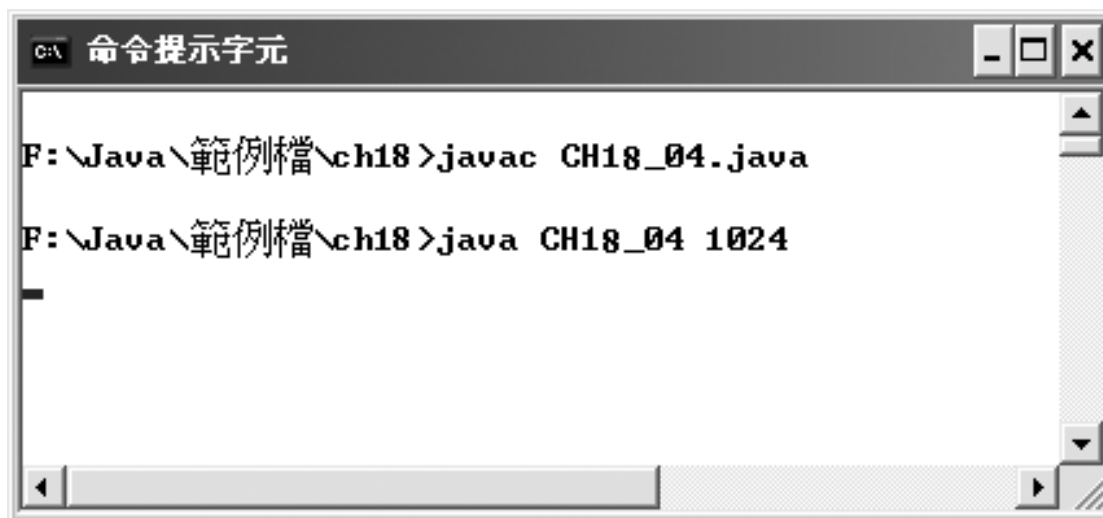
```
21 server = new goServer(port); //建立server物件
22 reader = new BufferedReader(new InputStreamReader(server.in));
23 writer = new PrintWriter(new
24 OutputStreamWriter(server.out), true);
25 }
26 }
27
28 //建立取得port的類別
29 class goServer {
30 ServerSocket server; //建立ServerSocket物件變數
31 Socket client; //建立Socket物件變數
32 InputStream in;
33 OutputStream out;
34 public goServer(int port) {
35 try{
36 server = new ServerSocket(port); //建立ServerSocket物件
37 while(true){ //判斷是否有客戶端的連線請求
38 client = server.accept(); //以accept()方法來接受客戶端的請求
39 //取得客戶端的主機位址
40 System.out.println("連線來自於：" +
41 client.getInetAddress().getHostAddress());
42 //以資料流方式取得客戶端的資料
```

```

43 in = client.getInputStream();
44 out = client.getOutputStream();
45 //在顯示訊息中加入換行
46 String SepLine = System.getProperty("line.separator");
47 InetAddress addr = server.getInetAddress().getLocalHost();
48 String outData = "Server information: " + SepLine +
49 "Local Host : " +
50 server.getInetAddress().getLocalHost() + SepLine +
51 "Port : " + server.getLocalPort();
52 byte[] outByte = outData.getBytes();
53 out.write(outByte, 0, outByte.length);
54 }
55 }
56 catch(IOException ioe){
57 System.err.println(ioe);
58 }
59 }
60 }

```

- 執行方式
 - (1) 將範例CH18_04.java 進行編譯。
 - (2) 以本端電腦為Server 端，執行範例程式CH18_04，執行指令為：「javaCH18_04 1024」，其中的「1024」為通訊埠。

A screenshot of a Windows Command Prompt window titled "命令提示字元" (Command Prompt). The window shows the following commands and their execution paths:

```
C:\ 命令提示字元

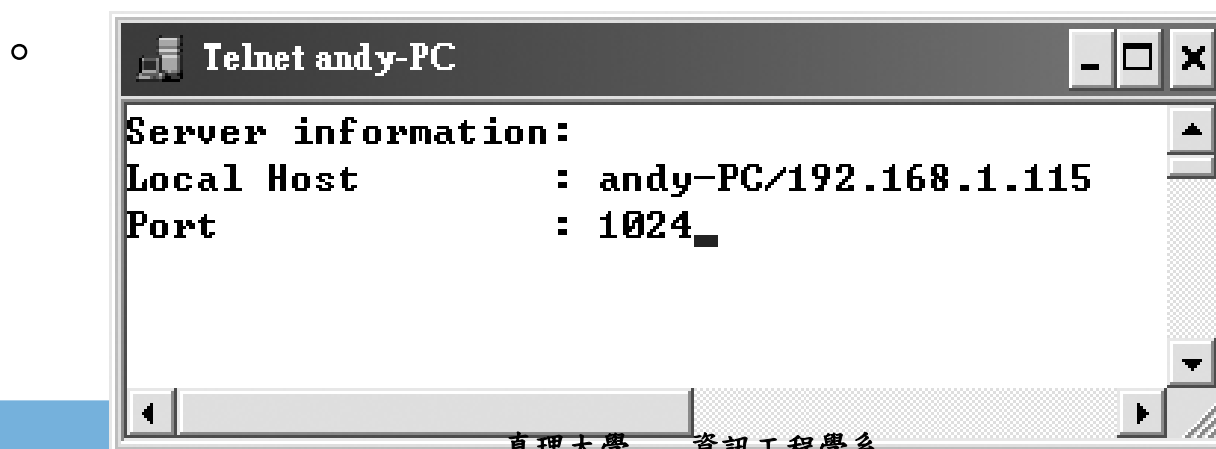
F:\Java\範例檔\ch18>javac CH18_04.java

F:\Java\範例檔\ch18>java CH18_04 1024
```

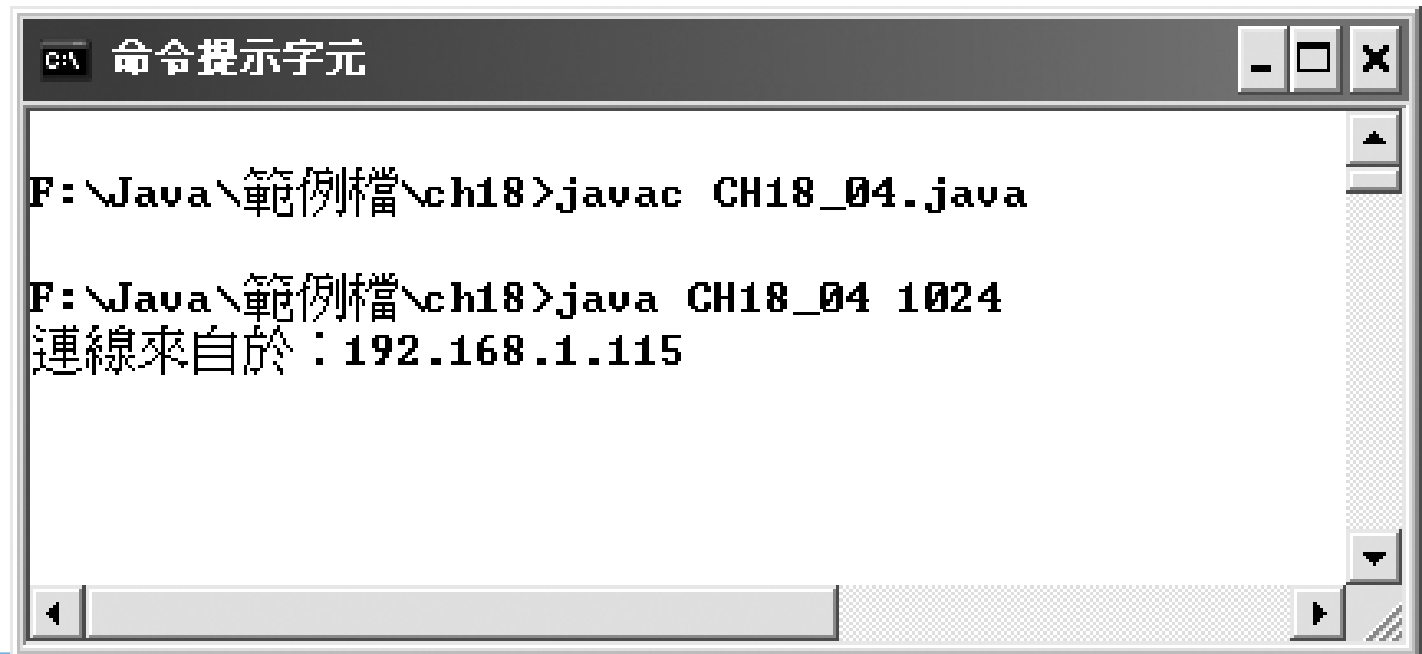

- (3) 從客戶端進行連線：執行指令：「開始」功能表→「執行」指令。
- (4) 開啟「執行」對話方塊，輸入指令：「telnet andy-PC 1024」。「telnet」是
- Windows Telnet，使用TCP/IP 通訊協定，執行此指令時，必須透過網路才能連結遠端的電腦。「andy-PC」代表遠端的主機名稱。「1024」則為通訊埠；因為Server 端與Client 端都設定相同的通訊埠，才能進行溝通。



- 客戶端telnet到「andy-PC」主機，並且連線成功時，會在客戶端的視窗顯示一個telnet視窗。



- 上述步驟3~5 都是在Client 端進行操作；而Server 端也會隨著Client 端的連線成功來顯示相關訊息。

A screenshot of a Windows Command Prompt window titled "命令提示字元". The window shows the execution of two Java commands. The first command is "javac CH18_04.java" and the second is "java CH18_04 1024". The output of the second command is "連線來自於：192.168.1.115". The window has a standard Windows interface with a title bar, maximize, minimize, and close buttons, and a scroll bar on the right side.

```
命令提示字元

F:\Java\範例檔\ch18>javac CH18_04.java

F:\Java\範例檔\ch18>java CH18_04 1024
連線來自於：192.168.1.115
```

• 程式解析

第10~25 行：從主程式中取得通訊埠，參數值以if 來進行判斷，如果沒有輸入通訊埠的埠號，則顯示錯誤訊息；如果有取得的埠號值則進行轉換。

第29~60 行：建立一個取得通訊埠的類別。

第34 行：利用建構子來取得傳入的通訊埠。

第36 行：將取得的通訊埠，以SocketServer 物件來聆聽。

第37~54 行：以while 迴圈來判斷客戶端是否有連線：當客戶端有連線時，以accept() 方法來接受連線請求。

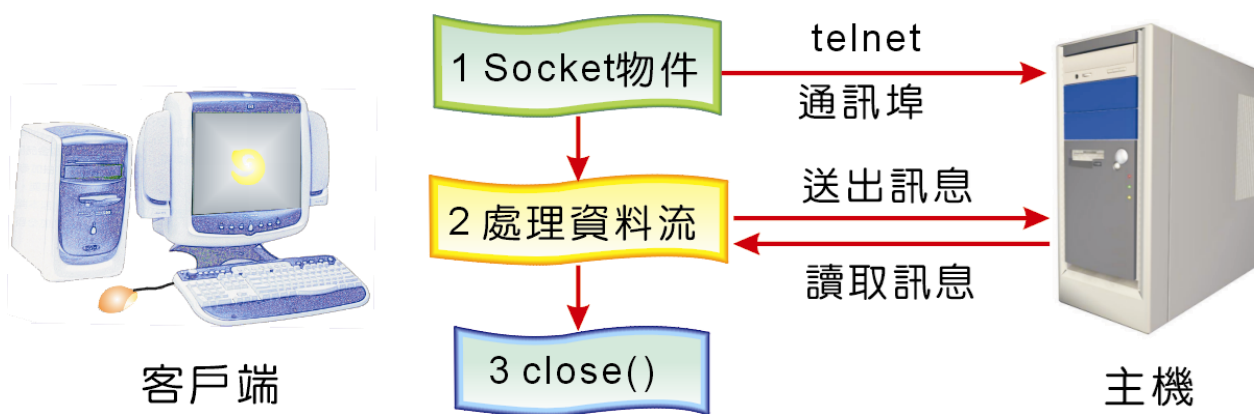
第40~44 行：以getInetAddress() 方法來取得客戶端的主機位址，並以資料流（Stream）方式來進行處理。

第46~51 行：當客戶端取得伺服端的相關訊息時，利用SepLine 字串物件做換行動作。

客戶端與Socket



- 客戶端的Socket執行流程如下面圖例所示



- 建立一個客戶端的Socket應用程式，步驟如下：
 - 先建立客戶端的Socket物件，連結到指定的主機名稱和通訊埠。
 - 利用資料流(stream)方式來處理送給伺服端的訊息或接收伺服端的訊息。

- 當客戶端的不再進行連結時，關閉Socket物件。
- 以Socket類別在網路上進行行程間 (interprocess) 的通訊，使用下列建構子來建立一個Socket物件，並將它連結到指定

建構子	說明
Socket()	建立一個未連線的Socket。
Socket(InetAddress, int port) Socket(String host, int port)	建立連線時，指定主機名稱及port。
Socket(InetAddress address, int port, InetAddress localAddr, int localPort) Socket(String host, int port, InetAddress localAddr, int localPort)	建立連線時，指定遠端主機名稱及port。
Socket(SocketImpl impl)	指定SocketImpl類別來建立一個未連線的Socket。

- 在任何時間，我們可經由下列方法來檢視socket所取得的位址或port：

方法	說明
InetAddress getInetAddress()	傳回Socket連線時的主機位址。
InetAddress getLocalAddress	取得Socket與本機連結時的位址。
int getLocalPort()	傳回Socket與本機連結時的port。
int getPort()	傳回Socket連線時遠端主機的port。
SocketAddress getLocalSocketAddress()	傳回本機的SocketAddress物件，如果傳回null時，代表尚未進行連結。
SocketAddress getRemoteSocketAddress()	傳回遠端主機的SocketAddress物件，如果傳回null時，代表尚未進行連線。

- 建立了Socket物件後，可利用下列方法來檢視它所獲得的輸入或輸出資料流(stream)：

方法	說明
InputStream getInputStream()	取得Socket的輸入資料流。
OutputStream getOutputStream()	取得Socket的輸出資料流。