

程式用途

1. 應用程式 (程式 + 資料庫)→每一台電腦都要安裝
2. 動態網頁 (程式 + 資料庫 + HTML)→ ASP · ASPX · JSP · PHP→後端執行→安裝在 Web Server
3. 動態網頁效果 (程式 + HTML)→JAVASCRIPT · JQUERY→前端執行→HTML5 + javascript + CSS3
4. 手機程式

```
<script language=javascript>
<!--
function newwin(width,height)
{
msgWindow=window.open(
}
// -->
</script>
```

```
<%
if session("user_name")<>" " t
If session("user_rights")<>3
Response.write "<a href=
else
Response.write "<a href=
end if
Response.write "<a href=bo
Response.write "<a
else
%>
```

```
<td width="5%" nowrap bgcolor="Black">
<div align="center"><font color="White" >
</td>
<td width="5%" nowrap bgcolor="Black">
<div align="center"><font color="White" >
</td>
<td width="5%" nowrap bgcolor="Black">
<div align="center"><font color="White" >
</td>
```

Java 程式的種類

1. Java 平台標準版(J2SE · Java2 Standard Edition)→可用來開發一般的應用程式
2. Java 平台企業版(J2EE · Java2 Enterprise Edition)→用來開發伺服器端的應用程式，例如 Servlet、JSP..等
3. Java 平台微小版(J2ME · Java2 Micro Edition)→讓 Java 在極小的記憶體上運作，用來開發手機上的遊戲程式

JAVA SE

Java 語言

工具程式

1. Javac.exe (Compiler)
2. java .exe (Run)

JDK

(Java Development Kits)
(工具程式)

Java API ((Java Application Interface)

1. 開發技術
2. 使用者介面工具箱 (AWT · Swing)
3. 整合類別庫 (JDBC)
4. 基礎類別庫 (Math · Collection)

JRE

(Java Runtime Environment)
(執行環境)

java 虛擬機器

(JAVA 唯一認識的作業系統)

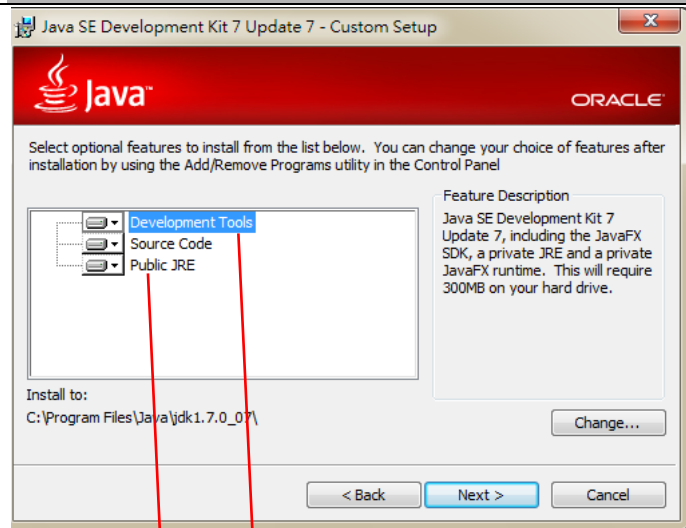
JVM

(Java virtual Machine)
(作業系統)

Windows

Linux

java JDK 安裝



本機 > 本機磁碟 (C:) > Program Files

名稱
Java

本機 > 本機磁碟 (C:) > Program Files > Java

名稱
jdk1.8.0_45
jre1.8.0_91

本機 > 本機磁碟 (C:) > Program Files > Java > jdk1.8.0_45

名稱	修改日期	類型	大小
bin	2017/7/6 上午 08...	檔案資料夾	
db	2017/7/6 上午 08...	檔案資料夾	
include	2017/7/6 上午 08...	檔案資料夾	
jre	2017/7/6 上午 08...	檔案資料夾	
lib	2017/7/6 上午 08...	檔案資料夾	
COPYRIGHT	2015/4/30 下午 0...	檔案	4 KB
javafx-src.zip	2017/7/6 上午 08...	WinRAR ZIP archi...	5,052 KB
LICENSE	2017/7/6 上午 08...	檔案	1 KB
README.html	2017/7/6 上午 08...	HTML 文件	1 KB
release	2017/7/6 上午 08...	檔案	1 KB
src.zip	2015/4/30 下午 0...	WinRAR ZIP archi...	20,746 KB
THIRDPARTYLICENSEREADME.txt	2017/7/6 上午 08...	文字文件	175 KB
THIRDPARTYLICENSEREADME-JAVAF...	2017/7/6 上午 08...	文字文件	108 KB

下載軟體的地方：

<http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html>

Menu

ORACLE

Search
Sign In
Country/Region
Contact

Oracle Technology Network / Java / Java SE / Downloads

Java SE

Java EE

Java ME

Java SE Subscription

Java Embedded

Java Card

Java TV

Community

Java Magazine

Overview

Downloads

Documentation

Community

Technologies

Training

Java SDKs and Tools

[Java SE](#)
[Java EE and Glassfish](#)
[Java ME](#)
[Java Card](#)
[NetBeans IDE](#)
[Java Mission Control](#)

Java Resources

[Java APIs](#)

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Important Oracle JDK License Update

The Oracle JDK License has changed for releases starting April 16, 2019.

Java SE 11.0.3 (LTS)

Java SE 11.0.3 is the latest release for the Java SE 11 Platform

[Learn more](#)

- Installation Instructions
- Release Notes
- Oracle JDK License
- Java SE Licensing Information User Manual
 - Includes Third Party Licenses
- Certified System Configurations
- Readme

Oracle JDK

DOWNLOAD

Java SE Development Kit 8u211

You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

☒ Accept License Agreement
☐ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.86 MB	jdk-8u211-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.76 MB	jdk-8u211-linux-arm64-vfp-hflt.tar.gz
Linux x86	174.11 MB	jdk-8u211-linux-i586.rpm
Linux x86	188.92 MB	jdk-8u211-linux-i586.tar.gz
Linux x64	171.13 MB	jdk-8u211-linux-x64.rpm
Linux x64	185.96 MB	jdk-8u211-linux-x64.tar.gz
Mac OS X x64	252.23 MB	jdk-8u211-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	132.98 MB	jdk-8u211-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.18 MB	jdk-8u211-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.57 MB	jdk-8u211-solaris-x64.tar.Z
Solaris x64	91.93 MB	jdk-8u211-solaris-x64.tar.gz
Windows x86	202.62 MB	jdk-8u211-windows-i586.exe
Windows x64	215.29 MB	jdk-8u211-windows-x64.exe

Pre-Apache NetBeans versions

- <https://netbeans.org/downloads/8.2/>
- <https://netbeans.org/downloads/8.1/>
- <https://netbeans.org/downloads/8.0/>
- <https://netbeans.org/downloads/7.4/>

程式組成 → 常數值 · 變數 · 運算子 · 運算式 · 程式區塊 · 程式檔案 · 專案

常數值	變數的命名	變數的型態	變數的宣告
運算子	運算式	程式區塊 · 副程式 · 程序 · 函數 · 結構化程式 · sub · function	
程式檔案		專案	

電腦五大單元	兩數相加	程式執行過程 → 記憶體變化
--------	------	----------------

程式分類

結構化程式	傳統程式	商用 (資料庫)	機器語言	直譯
非結構化程式	物件導向程式	控制硬體	低階	編譯
			中階	編譯且解譯 (跨平台)
			高階	

1. 結構化 與 非結構化

非結構化程式

```

public class 非結構化程式 {

    public static void main(String[] args) {
        開車_非結構化程式();
    }

    public static void 開車_非結構化程式() {

        System.out.println("甲");
        System.out.println("乙"); //直行
        System.out.println("丙");

        System.out.println("X");
        System.out.println("Y"); //左轉
        System.out.println("Z");

        System.out.println("甲");
        System.out.println("乙"); //直行
        System.out.println("丙");

        System.out.println("A");
        System.out.println("B"); //右轉
        System.out.println("C");

        System.out.println("甲");
        System.out.println("乙"); //直行
        System.out.println("丙");

        System.out.println("a");
        System.out.println("b"); //停止
        System.out.println("c");
    }
}

```

結構化程式 → 程序導向 (方法的名稱在前面)

```

public class 結構化程式 {

    public static void main(String[] args) {
        開車_結構化程式();
    }

    public static void 開車_結構化程式() {
        直行();
        左轉();
        直行();
        右轉();
        直行();
        左轉();
        直行();
        停止();
    }

    public static void 直行() {
        System.out.println("甲");
        System.out.println("乙");
        System.out.println("丙");
    }

    public static void 左轉() {
        System.out.println("X");
        System.out.println("Y");
        System.out.println("Z");
    }

    public static void 右轉() {
        System.out.println("A");
        System.out.println("B");
        System.out.println("C");
    }

    public static void 停止() {
        System.out.println("a");
        System.out.println("b");
        System.out.println("c");
    }
}

```

2. 傳統程式 (程序導向) 與 物件導向程式 (元件 與 物件)

程序導向程式 → 方法名稱在前面

物件導向程式 → (物件的名稱在前面)

```

public class 程序導向程式 {

    public static void main(String[] args) {
        開車_程序導向程式();
    }
    public static void 開車_程序導向程式() {
        直行();
        左轉();
        直行();
        右轉();
        直行();
        左轉();
        直行();
        停止();
    }
    //副程式的定義
    public static void 直行() {
        System.out.println("甲");
        System.out.println("乙"); //直行
        System.out.println("丙");
    }
    public static void 左轉() {
        System.out.println("X");
        System.out.println("Y"); //左轉
        System.out.println("Z");
    }
    public static void 右轉() {
        System.out.println("A");
        System.out.println("B"); //右轉
        System.out.println("C");
    }
    public static void 停止() {
        System.out.println("a");
        System.out.println("b"); //停止
        System.out.println("c");
    }
}

```

```

public class 物件導向程式 {

    public static void main(String[] args) {
        開車_物件導向();
    }
    public static void 開車_物件導向() {
        Car 車子1 = new Car(); //定義物件

        車子1.直行();
        車子1.左轉();
        車子1.直行();
        車子1.右轉();
        車子1.直行();
        車子1.左轉();
        車子1.直行();
        車子1.停止();
    }
}

//汽車元件
class Car {

    public void 直行() {
        System.out.println("甲");
        System.out.println("乙");
        System.out.println("丙");
    }
    public void 左轉() {
        System.out.println("X");
        System.out.println("Y");
        System.out.println("Z");
    }
    public void 右轉() {
        System.out.println("A");
        System.out.println("B");
        System.out.println("C");
    }
    public void 停止() {
        System.out.println("a");
        System.out.println("b");
        System.out.println("c");
    }
}

```

3. 商用(資料庫)・控制硬體(對硬體直接存取)
4. 低階(組合語言)・中階(C & C++)・高階程式語言(其他)
5. 直譯・編譯・編譯且解譯

<1>.直譯→直譯式的語言是利用直譯器 (Interpreter) 對原始程式碼一邊讀解，一邊執行，一邊除錯。主要的優點是對於初學者較易於使用。直譯式的語言如現在網際網路上使用的 JavaScript 是屬於直譯式

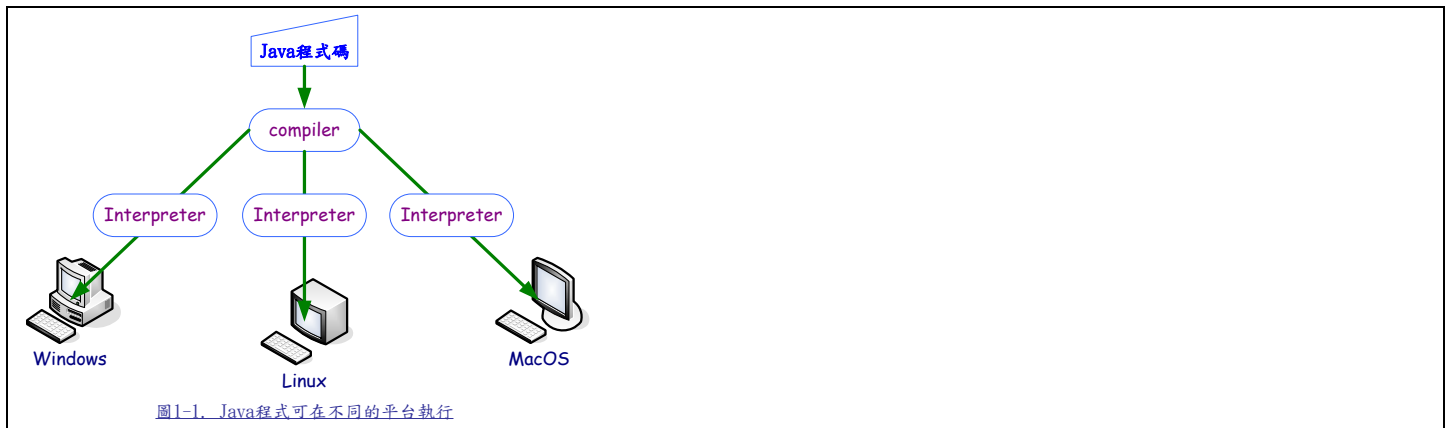
<2>.編譯→編譯式的語言是將原始程式碼透過編譯器 (Compiler) 轉成機械碼，再直接執行機械碼。主要的優點是速度快，並可一次找出程式中不合文法的部份。編譯式的語言如 C・C++ 等

<3>.編譯且解譯→Java 與 Visual Studio.NET 則採用半編譯半直譯的方式

(1).java 將一個程式轉換成 Java bytecodes(.class 檔)它是平台獨立的執行碼，將由 Java 平台的解譯器來解譯它。

編譯的行為只發生一次 (一次撰寫・到處執行)

(2).bytecode 可在任何 JavaVM(Virtual Machine)(Java 虛擬機器)上面執行



早期		中期		晚期(現在)	
BASIC	傳統・高階・商用→直譯	VB6	物件・高階・商用→編譯	VS2003 VS2005 VS2008 VS2010 VS2015	物件・高階・商用→編譯且解譯 VB.NET (商用) C#.NET (商用) C++ (控制硬體) (Visual Studio .NET)
C	傳統・中階・控制硬體→編譯	C C++	傳統・中階・控制硬體→編譯 物件・中階・控制硬體→編譯	C C++	傳統・中階・控制硬體→編譯 物件・中階・控制硬體→編譯
				JAVA	物件・高階・商用→編譯且解譯

Java 程式語言的特性

- | | |
|--|--|
| <ol style="list-style-type: none"> 1. 簡單 2. 架構獨立 3. 物件導向 4. 高可攜性 5. 分散式 | <ol style="list-style-type: none"> 6. 高效能的執行 7. 解譯的 8. 多執行緒 9. 穩建的 10. 動態的 11. 安全的 |
|--|--|

Java 平台

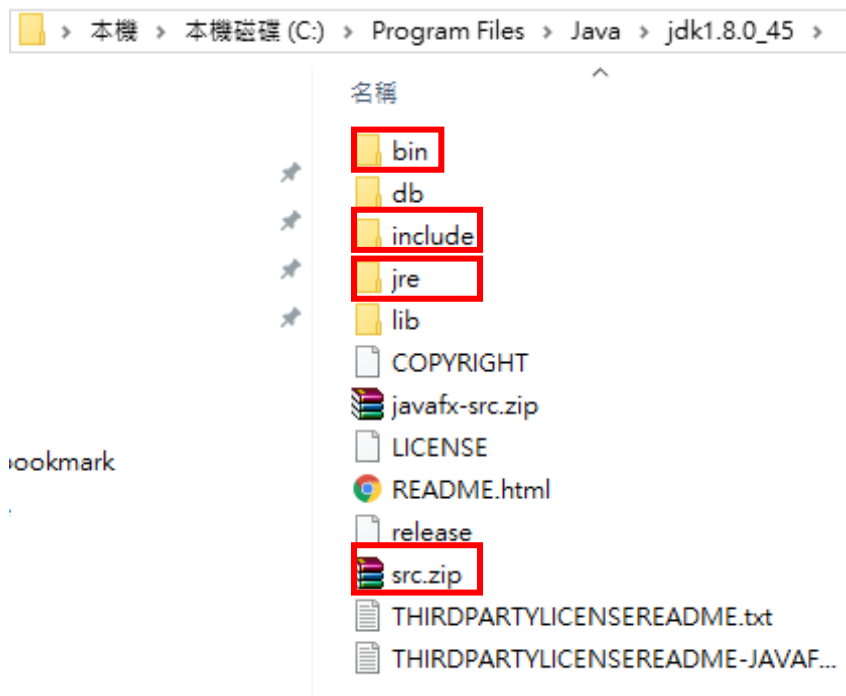
1. 大多數的平台可以描述為作業系統和硬體的結合體
2. Java 平台和其它的平台不同，因為它是單純的軟體平台
3. Java 是在其它以硬體為基底之平台如 Linux・Windows2000 上方執行
4. Java 軟體平台 (JRE) 有兩個要素 (不包含工具 javac 與 java)

<1>.Java 虛擬機器 → JVM

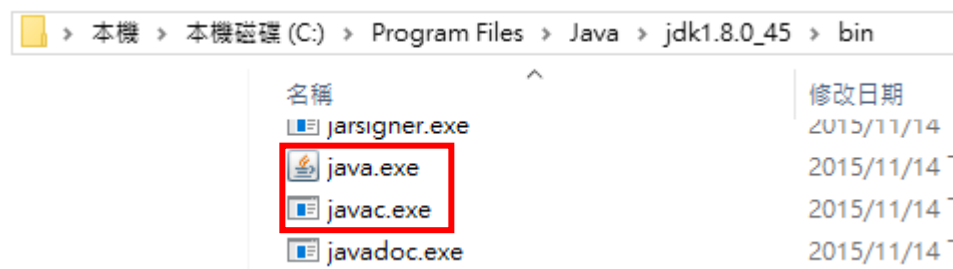
<2>.Java 應用程式介面 (Java API)(Java Application Interface)

JDK 內容

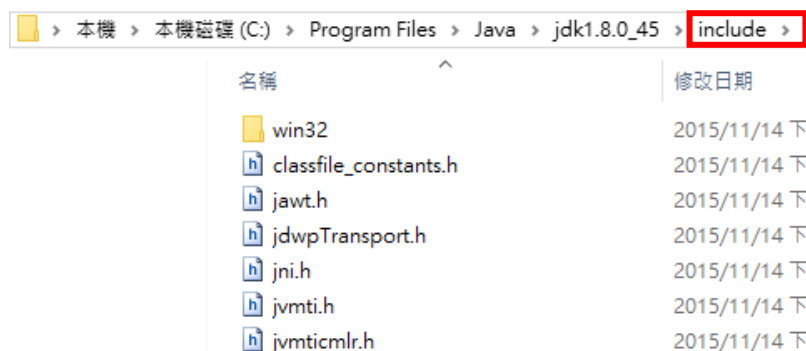
1. bin 目錄→開發・執行・除錯→bin 目錄下有兩個重要的程式 javac.exe (Java 的編譯程式)・java.exe(Java 的執行程式)
2. include→用在 Java Native Interface (JNI) 的 C 語言標頭檔
3. jre→Java 的執行環境・JVM 和 API
4. lib→開發工具所需的類別庫與檔案→Jre/Lib
5. src.zip(Java API 的原始檔)→Java API 的原始程式檔



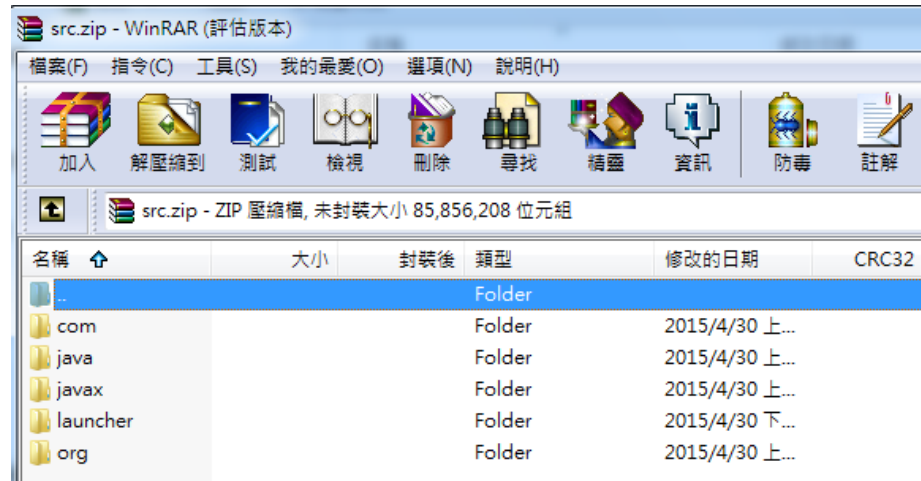
[bin]



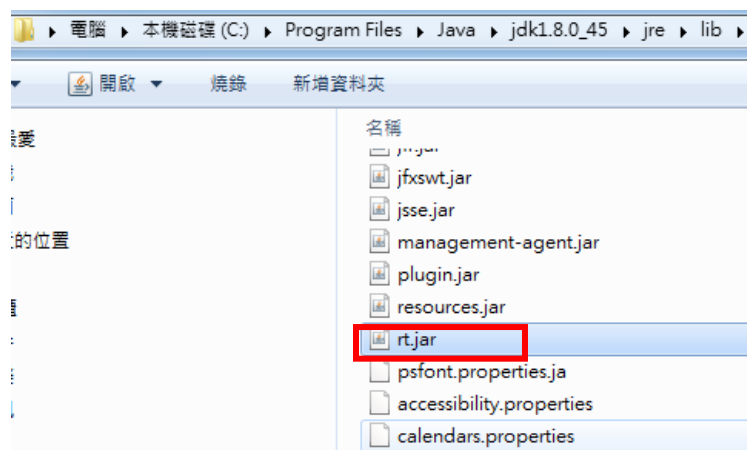
[include]→java 的 api 很多都是由 java 的原始程式(c++所寫) 所以需要有 include 標頭檔



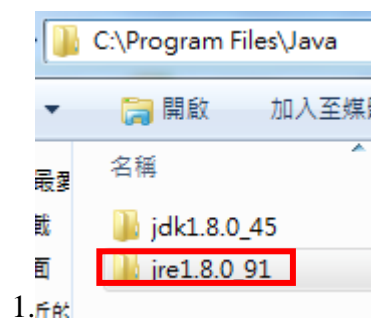
java api 的 src 檔



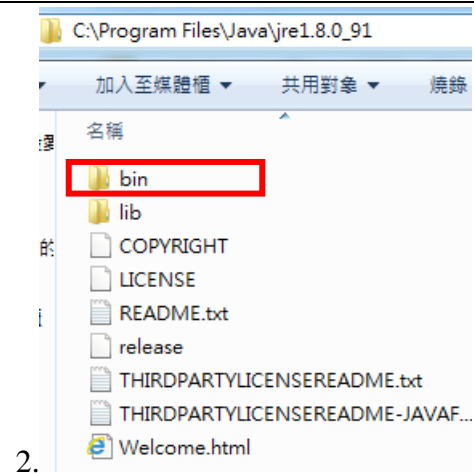
java api 的 jar 檔



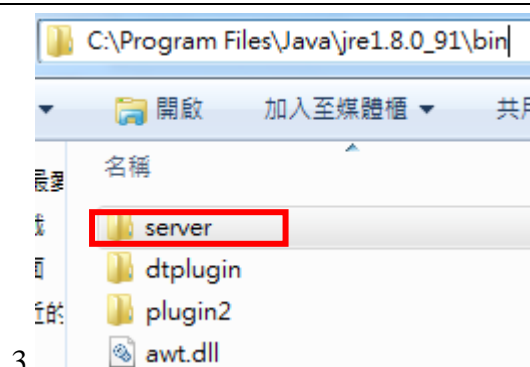
java 的 jvm → public JRE 裡面



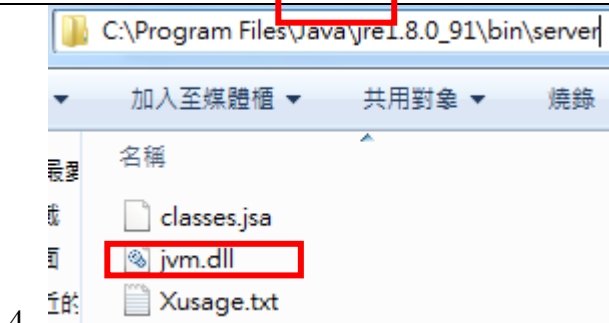
1. 所示



2.

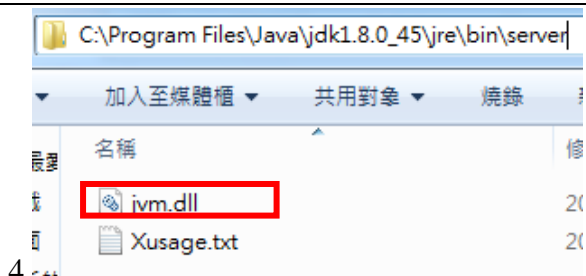
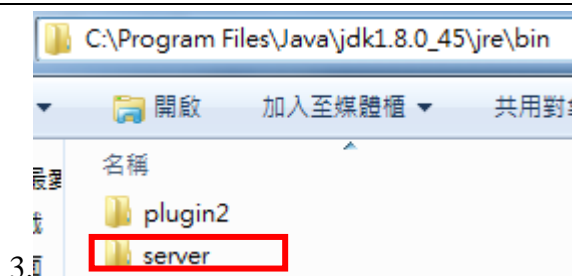
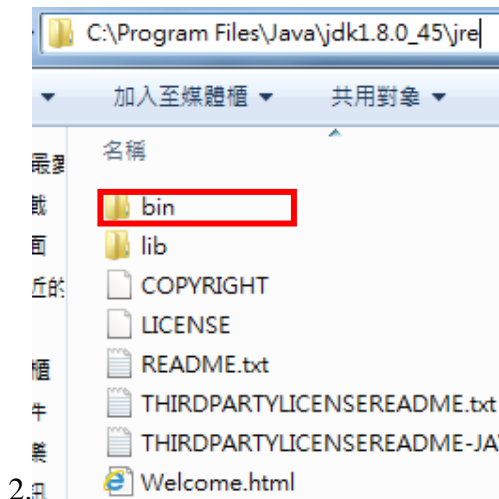
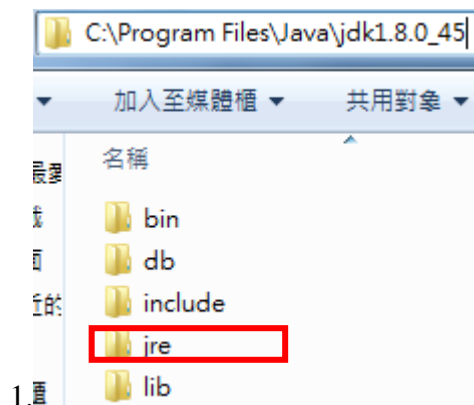


3.



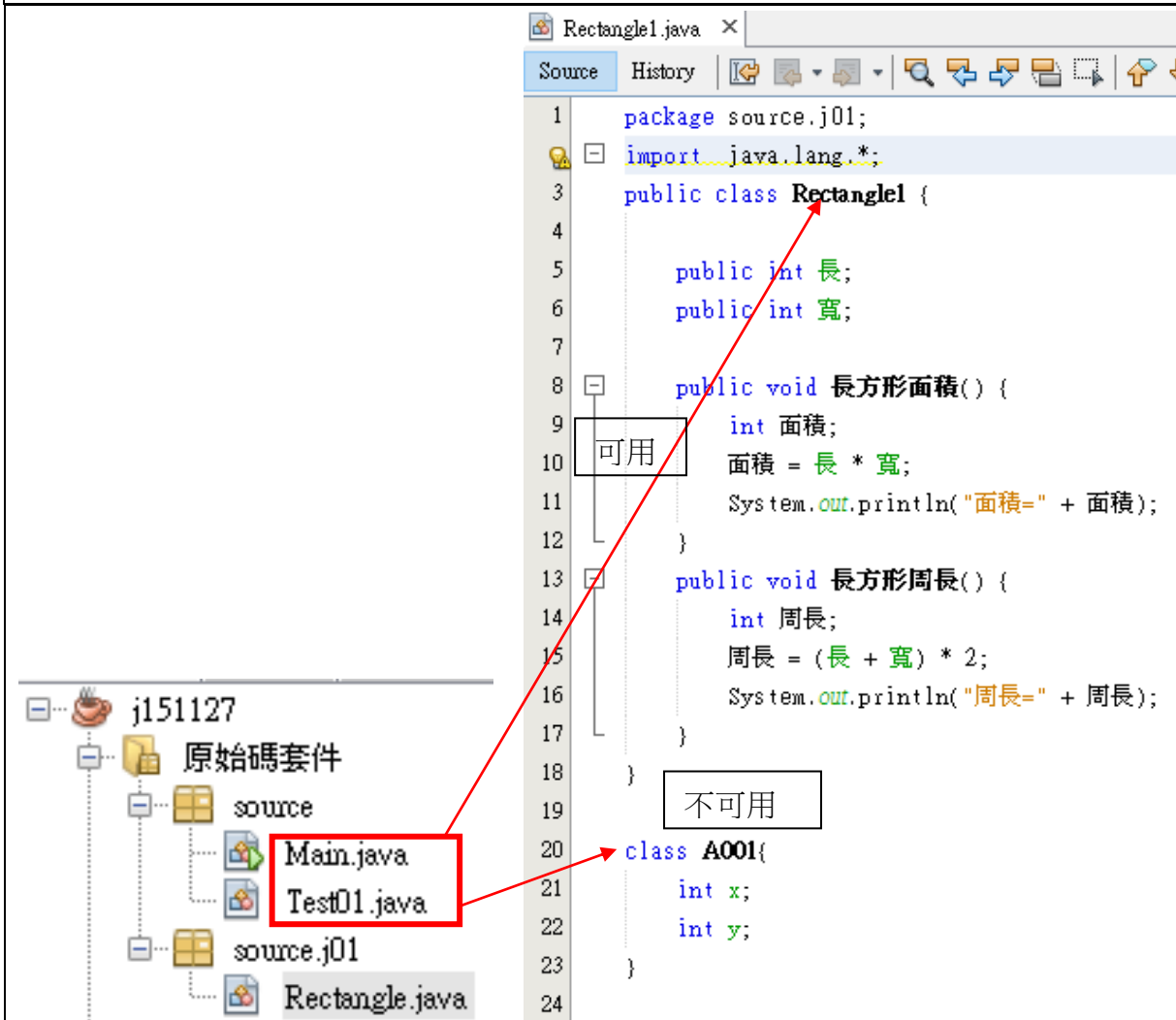
4.

→private JRE 裡面

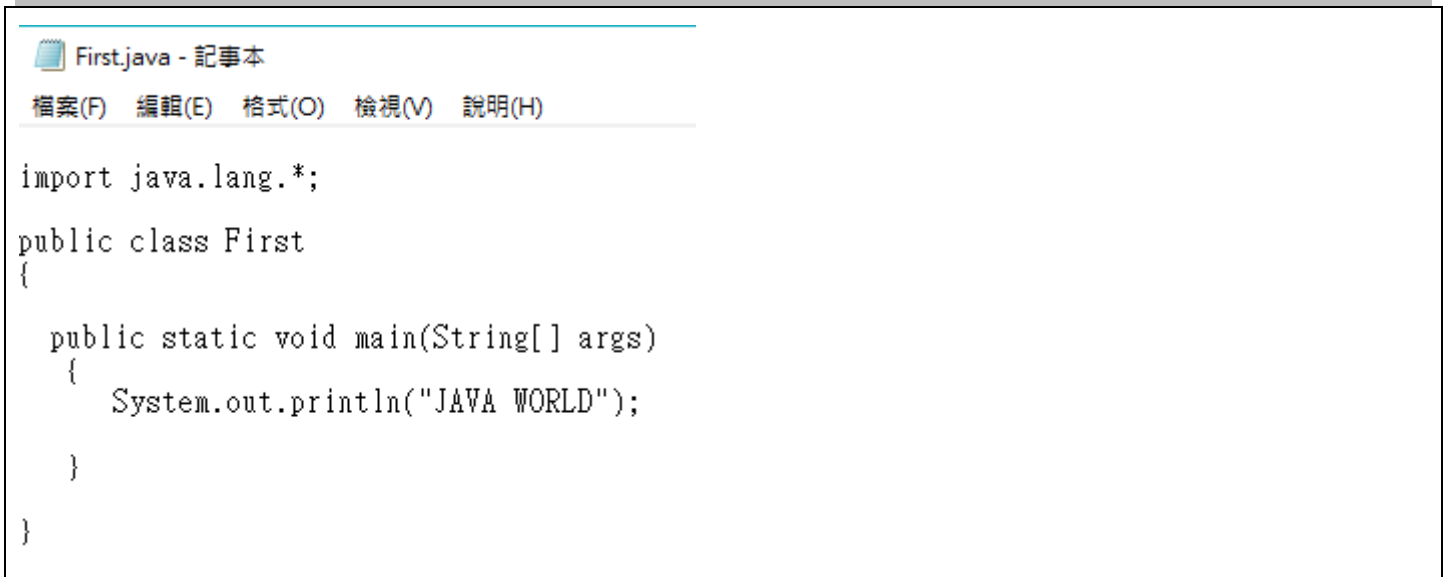


java 的程式架構(物件導向的程式架構)

專案→package→程式檔案→類別(class)→[全域變數]·[副程式·主程式 main]→[區域變數]·[運算式]→[常數值]·[變數]·[運算子]



第一支程式→先建專案目錄→放在 c:\javahome (以此為專案的根)



DOS 的常用指令

到 dos 模式 開始/執行/cmd 或 開始/ 附屬應用程式/命令列提示字元

cd\ →到 C:\

cd.. →到上一層目錄

c:\>cd javahome 到 c:\>javahome>的目錄 或 任何目錄下 cd c:\javahome 到 c:\>javahome>

↓ ↑ 翻越前後的指令

Compiler 程式注意事項

1. 編譯這個 java，編譯器的指令是 **javac**。若這是電腦中第一次裝完 JDK，在沒有進行任何設定下，會出現以下的訊息：
這是作業系統在跟你說，它找不到 **javac** 放在哪邊！

```
C:\Users\joyes>javac c:\javahome\First.java
'javac' 不是內部或外部命令、可執行的程式或批次檔。
```

2. 當你直接鍵入一個指令而沒有指定路徑資訊時，作業系統會依照 PATH 環境變數中所設定的路徑順序，依序尋找各路徑下是否有這個指令。你可以執行 `echo %PATH%` 來看看目前系統的 PATH 環境變數中包括哪些路徑資訊：

```
C:\Users\joyes>echo %path%
C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Intel\iCLS Client;C:\Program Files\Intel\iCLS Client;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Microsoft\Web Platform Installer;C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Pages\v1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files (x86)\Windows Kits\10\Windows Performance Toolkit\;C:\Program Files\Microsoft SQL Server\120\Tools\Binn\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Skype\Phone\;C:\WINDOWS\System32\OpenSSH\;C:\Users\joyes\AppData\Local\Microsoft\WindowsApps;
```

3. 依 上面的 PATH 資訊，如果你鍵入 **javac** 指令，作業系統會依序找
 - <1>.目前目錄 C:\Users\joyes>→有無 **javac.exe**
 - <2>.C:\ProgramData\Oracle\Java\javapath→C:\Program Files (x86)\Common Files...→C:\Program Files (x86)\Intel\iCLS Client... 中有無 **javac.exe** 指令，當所有路徑都找不到時，就會出現剛剛所看到的錯誤訊息，因為作業系統不知道你把 **javac** 放在哪裡了。

DOS 環境手動設定

1. **javac** 是放在 JDK 安裝目錄的 bin 中，所以鍵入以下的指令，就可以執行：

```
C:\Users\joyes>cd C:\Program Files\Java\jdk1.8.0_45\bin
C:\Program Files\Java\jdk1.8.0_45\bin>javac c:\javahome\First.java
```

然而，若每次執行都得輸入指令所在位置，實在太累了

2. 在 PATH 中設定指令的路徑資訊，使用 SET 指令來設定，設定方式為

```
SET PATH= C:\Program Files\Java\jdk1.8.0_45\bin
```

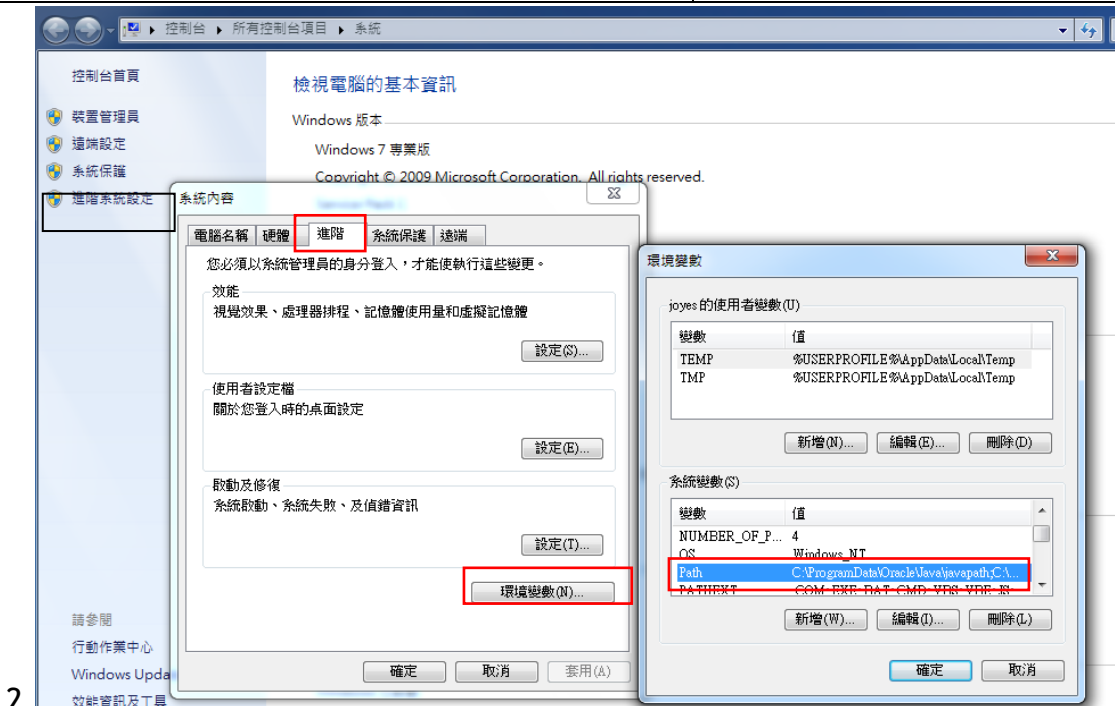
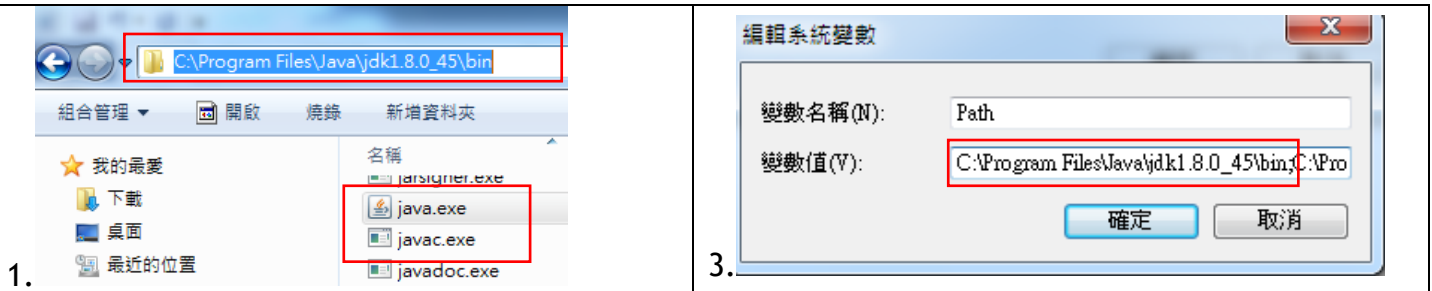
設定時，若有多個路徑，會使用分號 (;) 作區隔，通常會將原有的 PATH 附加在設定值的後面，如此需要尋找其它指令時，才可以利用保留下來的原有 PATH 資訊，不過！**關掉這個 Dos 模式後，下次要設定時又要重新設定**

```
C:\Users\joyes>set path=C:\Program Files\Java\jdk1.8.0_45\bin;%path%
C:\Users\joyes>echo %path%
C:\Program Files\Java\jdk1.8.0_45\bin;C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Intel\iCLS Client;C:\Program Files\Intel\iCLS Client;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Microsoft\Web Platform Installer;C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Pages\v1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files (x86)\Windows Kits\10\Windows Performance Toolkit\;C:\Program Files\Microsoft SQL Server\120\Tools\Binn\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Skype\Phone\;C:\WINDOWS\System32\OpenSSH\;C:\Users\joyes\AppData\Local\Microsoft\WindowsApps;
```

3. 我的電腦設定 系統 環境變數

<1>.右鍵 / 「我的電腦」 / 內容 / 進階系統設定 / 『進階』標籤 →【環境變數】

<2>.設定「系統變數」, 將安裝完成的 Java 執行路徑 (javac 與 javac 的路徑) "C:\Program Files\Java\jdk1.8.0_45\bin" 加入「系統變數」中的『path』→放在第一順位



4. 檢查環境變數是否有設定成功

```
C:\Users\joyes>echo %path%
C:\Program Files\Java\jdk1.8.0_45\bin;C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\bin;
```

環境變數的使用順序

在一個可以允許多人共用的系統中，系統環境變數的設定，會套用至每個登入的使用者，使用者環境變數則只影響個別使用者。當你開啟一個 Dos 模式時，所獲得的環境變數，

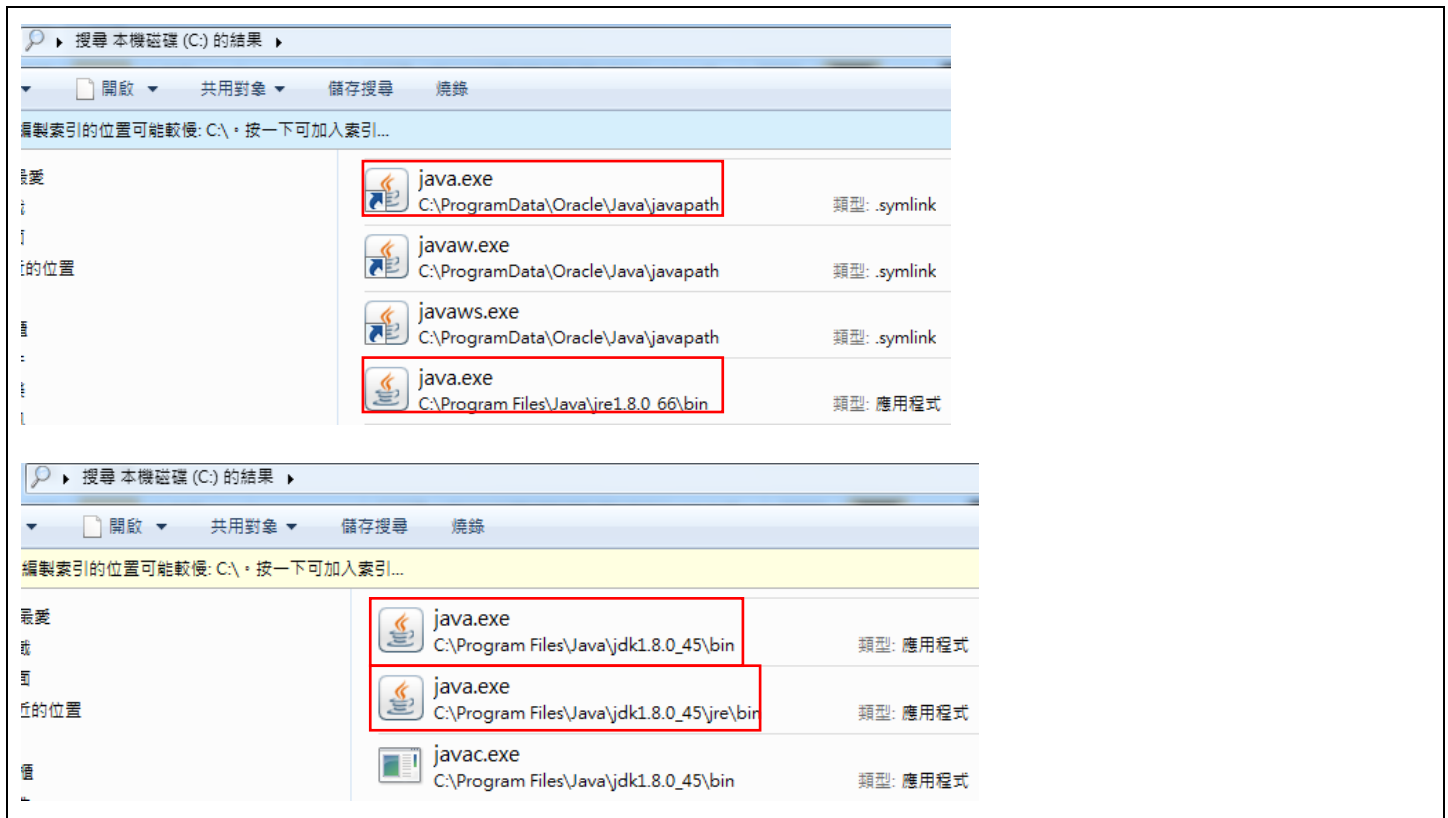
1. SET 所設定的結果再「附加」上
2. 系統環境變數再「附加」上
3. 使用者環境變數。

開始 Compiler 程式

```
C:\Users\joyes>javac c:\javahome\First.java
```

Run 程式注意事項→你執行的是哪個 JRE ?

- 因為種種的原因，你的電腦中可能不只存在一套 JRE ! 你可以試著搜尋電腦中的檔案，**你可以將一個 java.exe 視作就是有一套 JRE !**
在裝好 JDK 後，如果有選擇一併安裝 Public JRE，則至少會有兩套 JRE 存在你的電腦中，一個是 JDK 本身附的 **Private JRE**，一個是你所選擇安裝的 **Public JRE**。



- 既然在電腦中有可能同時存在多套 JRE，那麼你到底執行哪一套 JRE ?
- 在安裝好 JDK 後 還沒設定 path 之前的環境變數

```
C:\Users\joyes>echo %path%
C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\I
```

- 在安裝好 JDK 後設定好 path 的環境變數

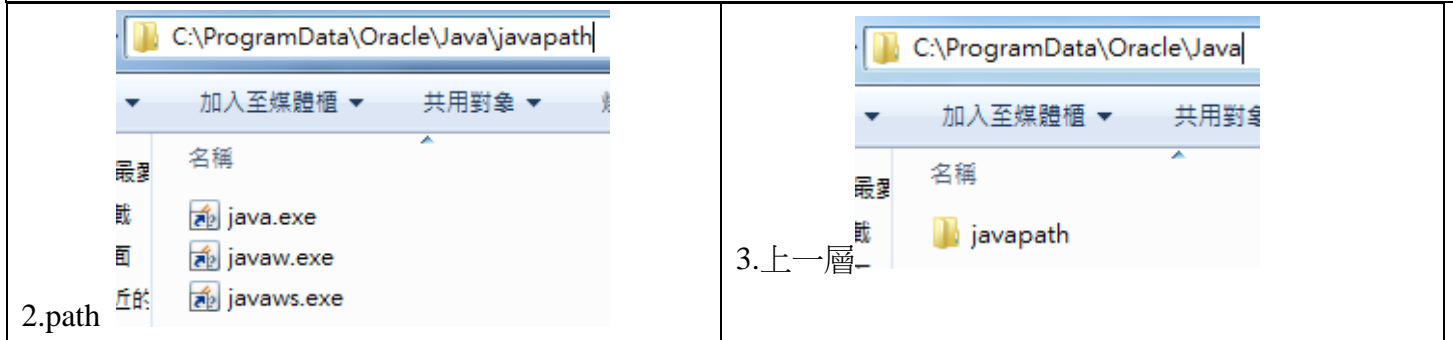
```
C:\Users\joyes>echo %path%
C:\Program Files\Java\jdk1.8.0_45\bin;C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files\Intel\iCLS Client\;C:\Program Files\Intel\iCLS Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\I
```

會依照以下的規則來尋找可用的 JRE :

- <1>.可否在 java 可執行檔目錄下找到相關的**原生程式庫** (.dll，例如 java.dll)
- <2>.可否在上一層目錄中找到 jre 目錄
- <3>.查看登錄檔 (regedit)

- 當你安裝完 JDK 後尚未設定 path 環境變數時，鍵入 java 指令，應該就是執行

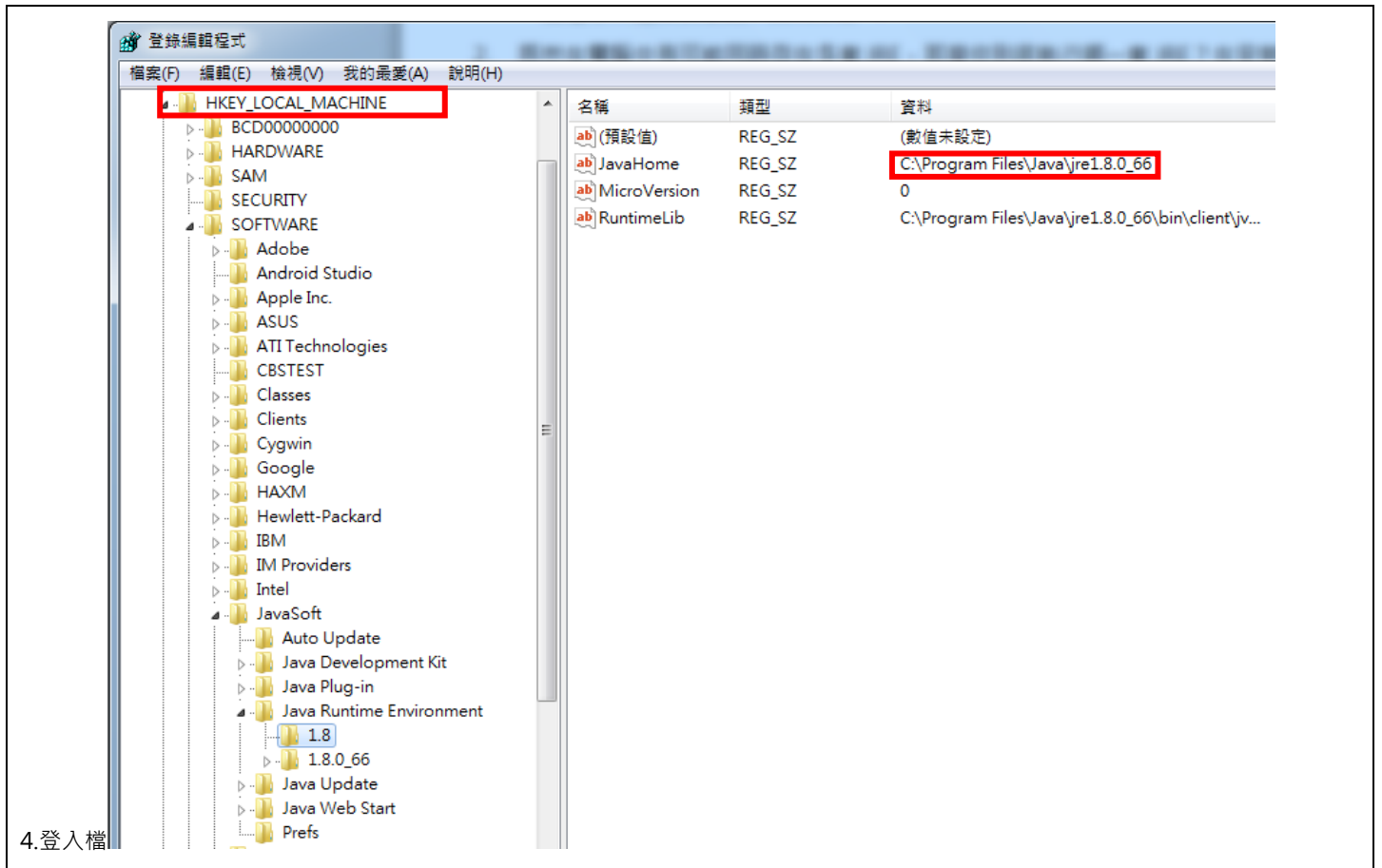
- <1>.目前目錄 C:\Users\joyes>→無 java.exe
- <2>.path 中的設定 第一順位 C:\ProgramData\Oracle\Java\javapath 中的 java.exe，但當中並沒有相關 java.dll 檔案，因此
- <3>.找尋上一層目錄，也沒有 jre 目錄



<4>.此時查看登錄檔：執行中打 regedit

HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Runtime Environment\的版本與目錄資訊

所以最後所執行的是 C:\Program Files\Java\jre1.8.0_51 的 public JRE。



4.登入檔



4.public JRE

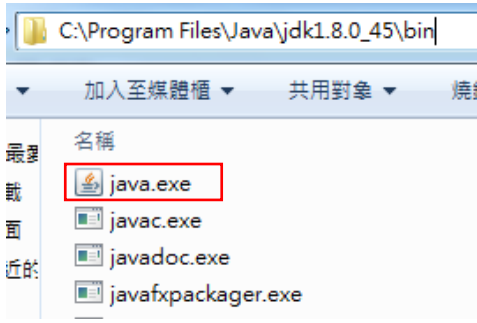
6. 如果你設定了 path 環境變數讓它指向 JDK 的 bin 目錄，則執行 java 指令時，

```
C:\Users\joyes>echo %path%
C:\Program Files\Java\jdk1.8.0_45\bin;C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Tools\;
```

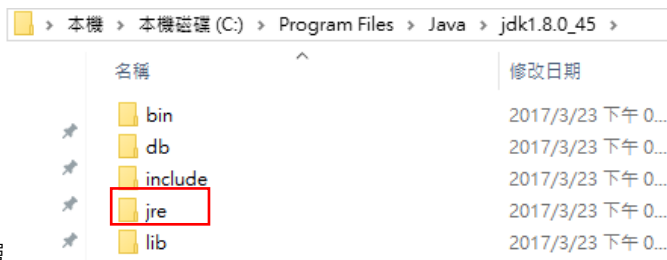
<1>.因為在 JDK 的 bin 中找不到相關的.dll 檔案

<2>.因此找上一層目錄中的 jre，於是找到的是 JDK 的 Private JRE。→必須設定在第一個順位

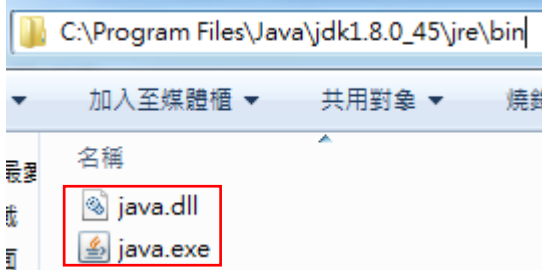
1. path



2. 上一層



3. private JRE



7. 為什麼 JDK 自己還要附帶 Private JRE 呢？這不是多此一舉嗎？直接裝 Public JRE 不就好了？答案其實在於，JDK 中多數工具程式，本身就是用 Java 寫的，執行時自然就得有 JRE，為了防止你不知道要裝 JRE，所以乾脆直接給你一個 Private JRE

開始 Run 程式

1. 不在類別的路徑上 → 執行時必須加上 -cp

```
C:\Users\joyes>javac c:\javahome\First.java
C:\Users\joyes>java -cp c:\javahome First
JAVA WORLD
```

2. 在類別的路徑上 → 直接 compiler 與 run 即可

```
C:\Users\joyes>cd c:\javahome
c:\javahome>java First
JAVA WORLD
```

將 src 放在 c:\javahome\j01 package 內

```
Second.java - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
package j01;
import java.lang.*;

public class Second
{
    public static void main(String[] args)
    {
        System.out.println("Second-JAVA WORLD");
    }
}
```


1. 不在類別的路徑上→執行時必須加上 -cp

```
c:\>javac c:\javahome\j01\Second.java

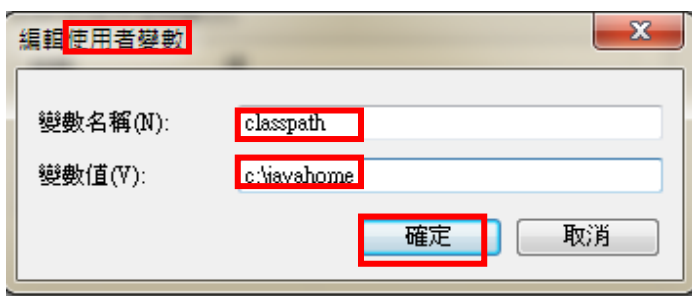
c:\>java -cp c:\javahome j01.Second
Second-JAVA WORLD
```

2. 在類別的路徑上→直接 compiler 與 run 即可

```
c:\>cd javahome

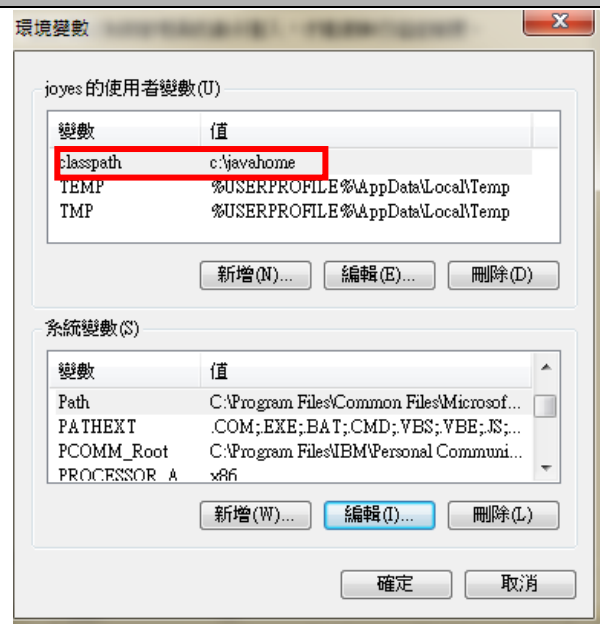
c:\javahome>java j01.Second
Second-JAVA WORLD
```

設定用者環境變數「CLASSPATH」內容為→c:\javahome;



```
C:\Users\joyes>java First
JAVA WORLD

C:\Users\joyes>java j01.Second
Second-JAVA WORLD
```



main 的參數 (String[] args)

```
Third.java - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

import java.lang.*;

public class Third
{
    public static void main(String[] args)
    {
        //System.out.println("JAVA WORLD");
        System.out.println(args[0]);
        System.out.println(args[1]);
    }
}
```

```
C:\Users\joyes>javac c:\javahome\Third.java
C:\Users\joyes>java Third
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
    at Third.main(Third.java:10)
C:\Users\joyes>java Third aa
aa
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
    at Third.main(Third.java:11)
C:\Users\joyes>java Third aa bb
aa
bb
C:\Users\joyes>java Third aa bb cc
aa
bb
C:\Users\joyes>java Third aa "bb cc"
aa
bb cc
C:\Users\joyes>
```

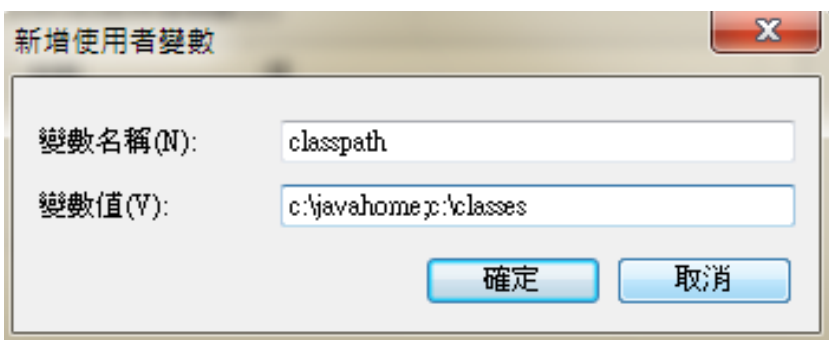
Compiler 後 class 指定放到 classes 資料夾→用 -d 這個命令

```
C:\Users\joyes>javac -d c:\classes c:\javahome\First.java
```

執行放在 classes 的程式

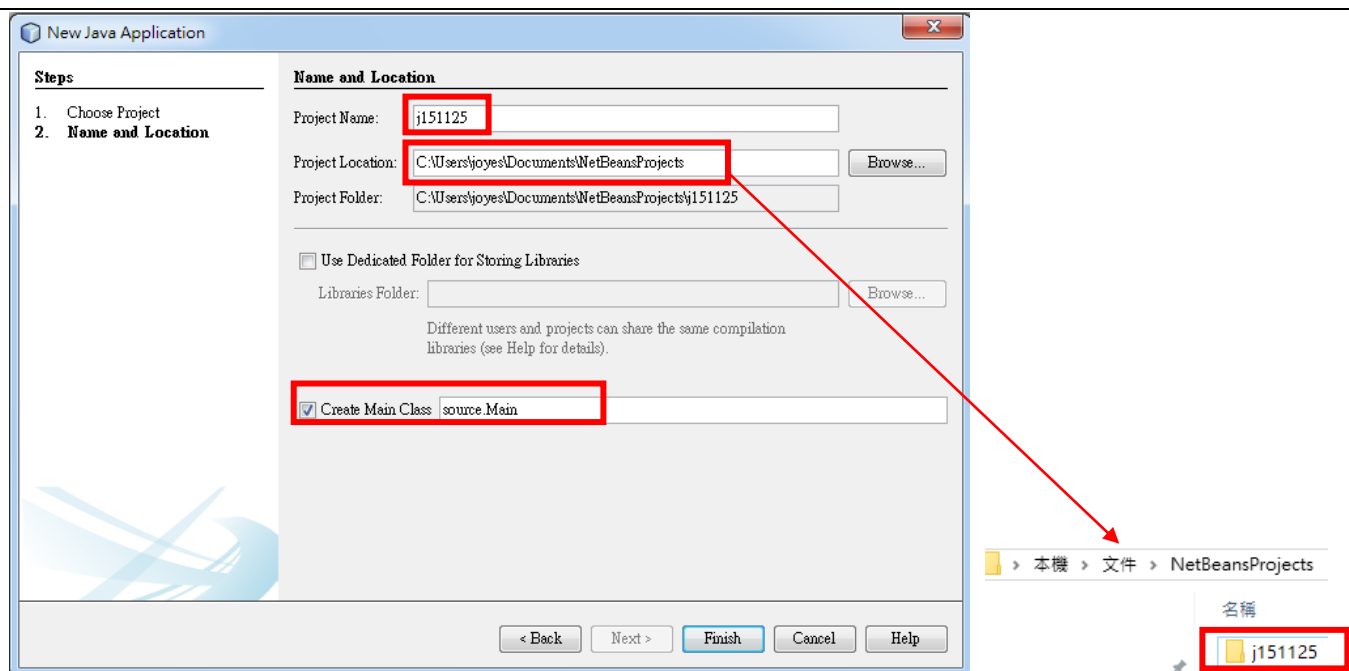
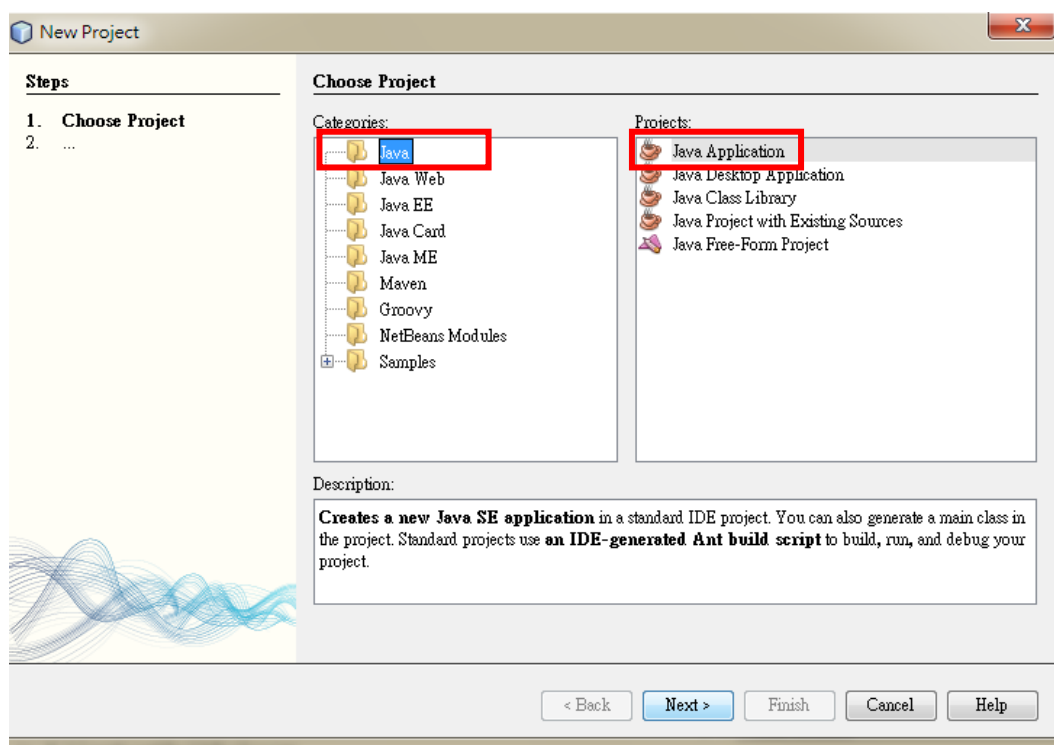
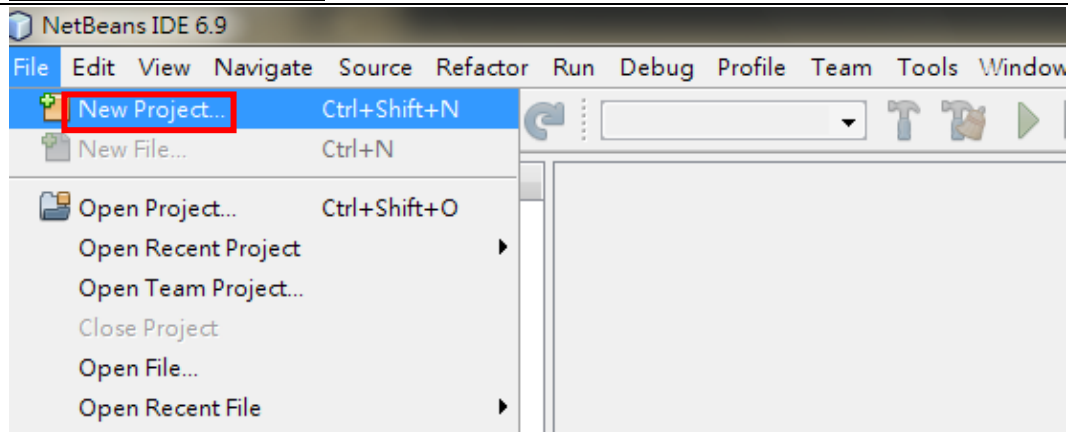
```
1. C:\Users\joyes>java -cp c:\classes First
   JAVA WORLD
```

2.

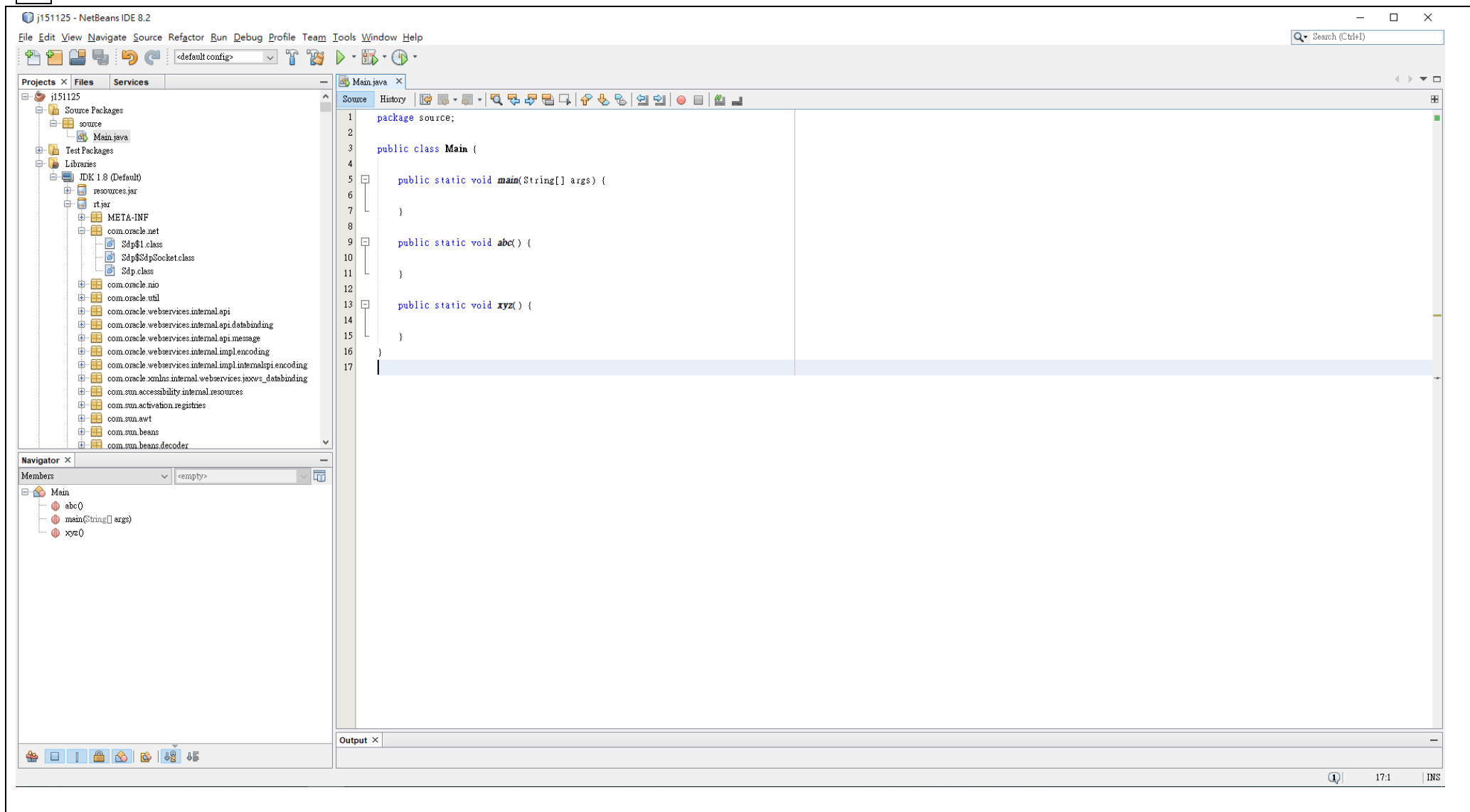


netbeans 的基本操作→開新專案，結束專案，開啟專案，重設視窗配置，重要資料夾，註解顏色，字型，開檔案，改檔名，刪檔案，改範本，自動秀出指令，執行點，備份

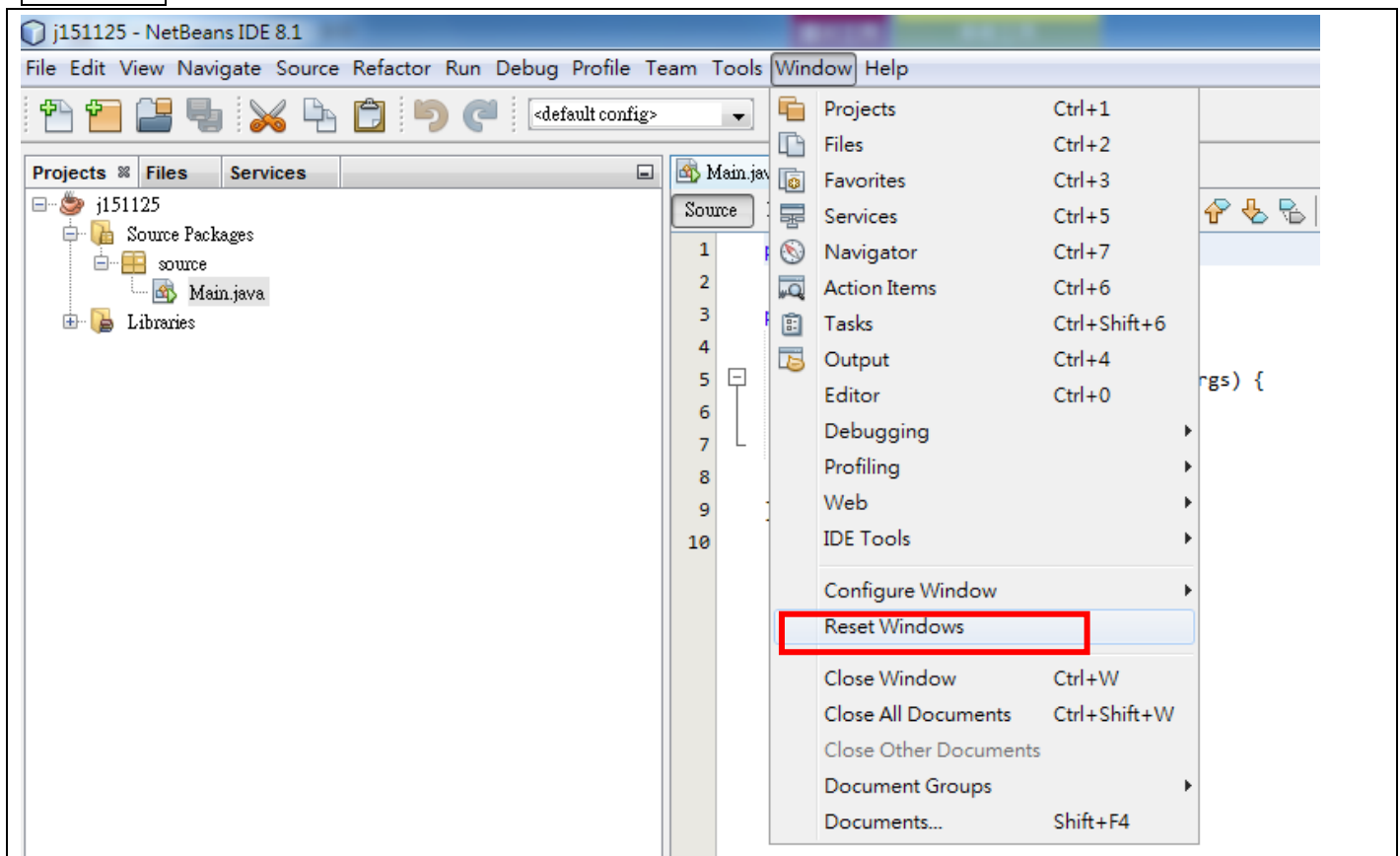
開專案，結束專案，開啟專案



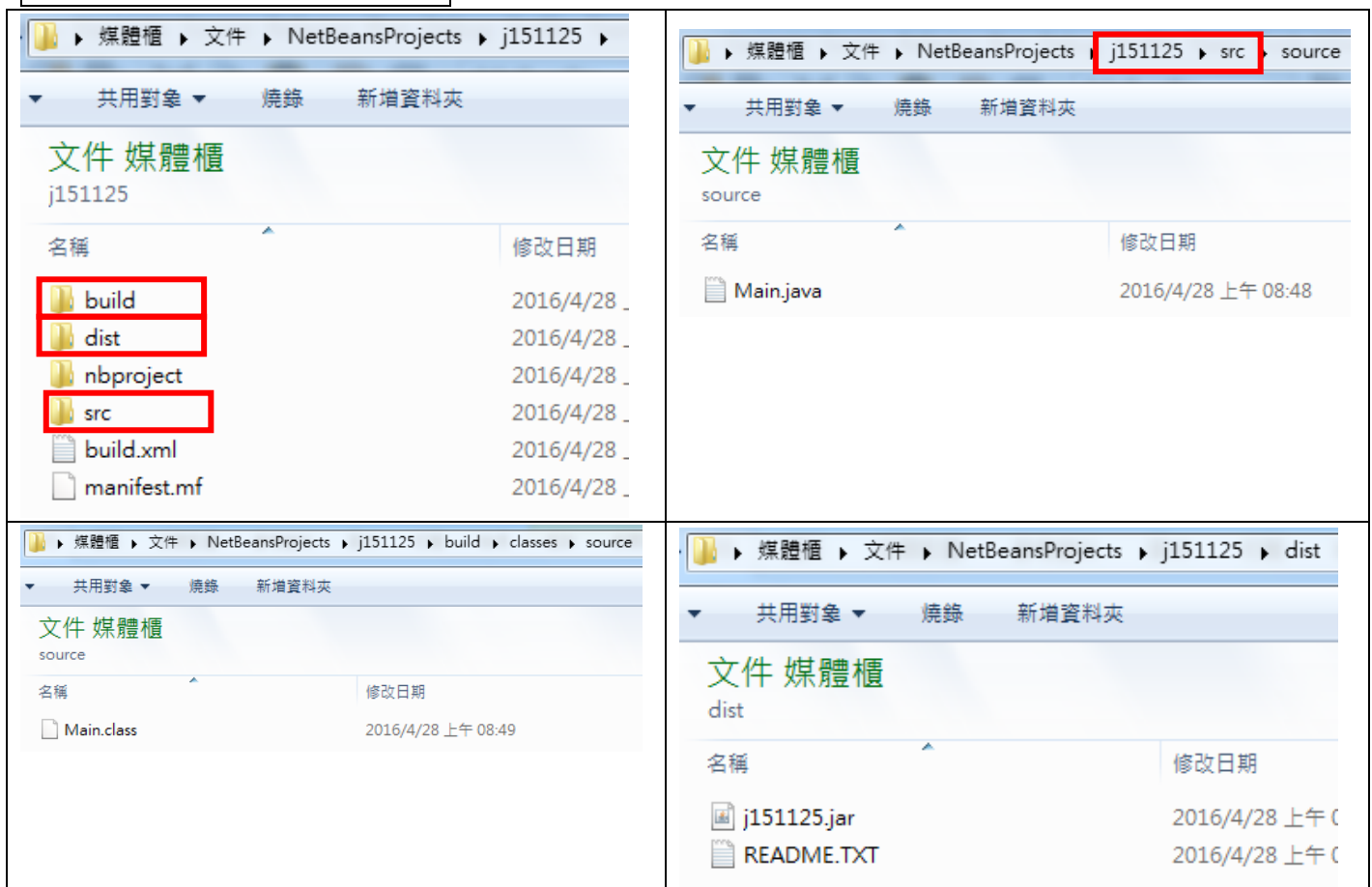
畫面



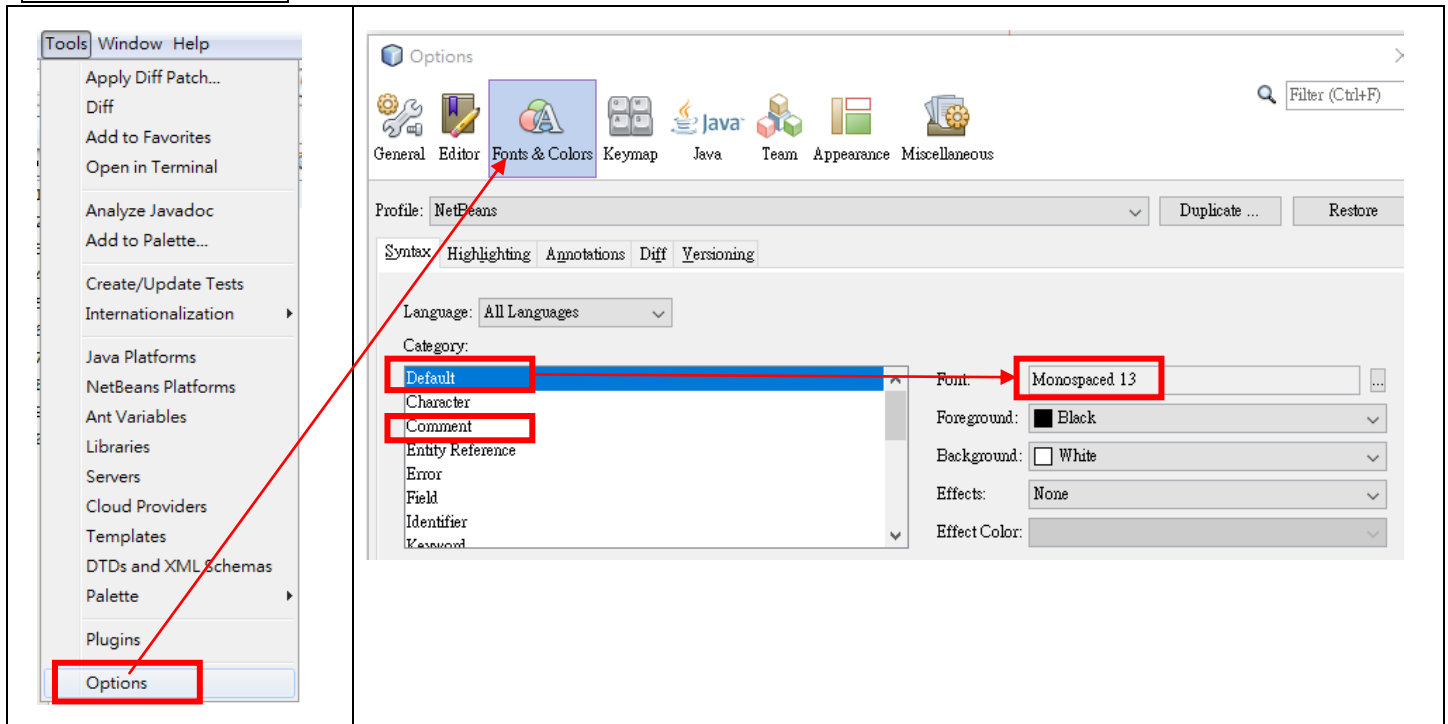
重整視窗配置



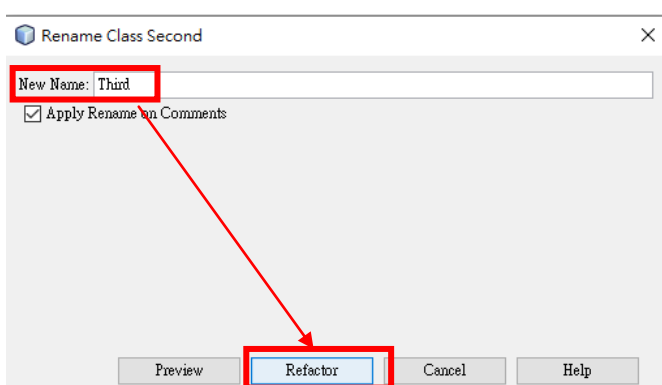
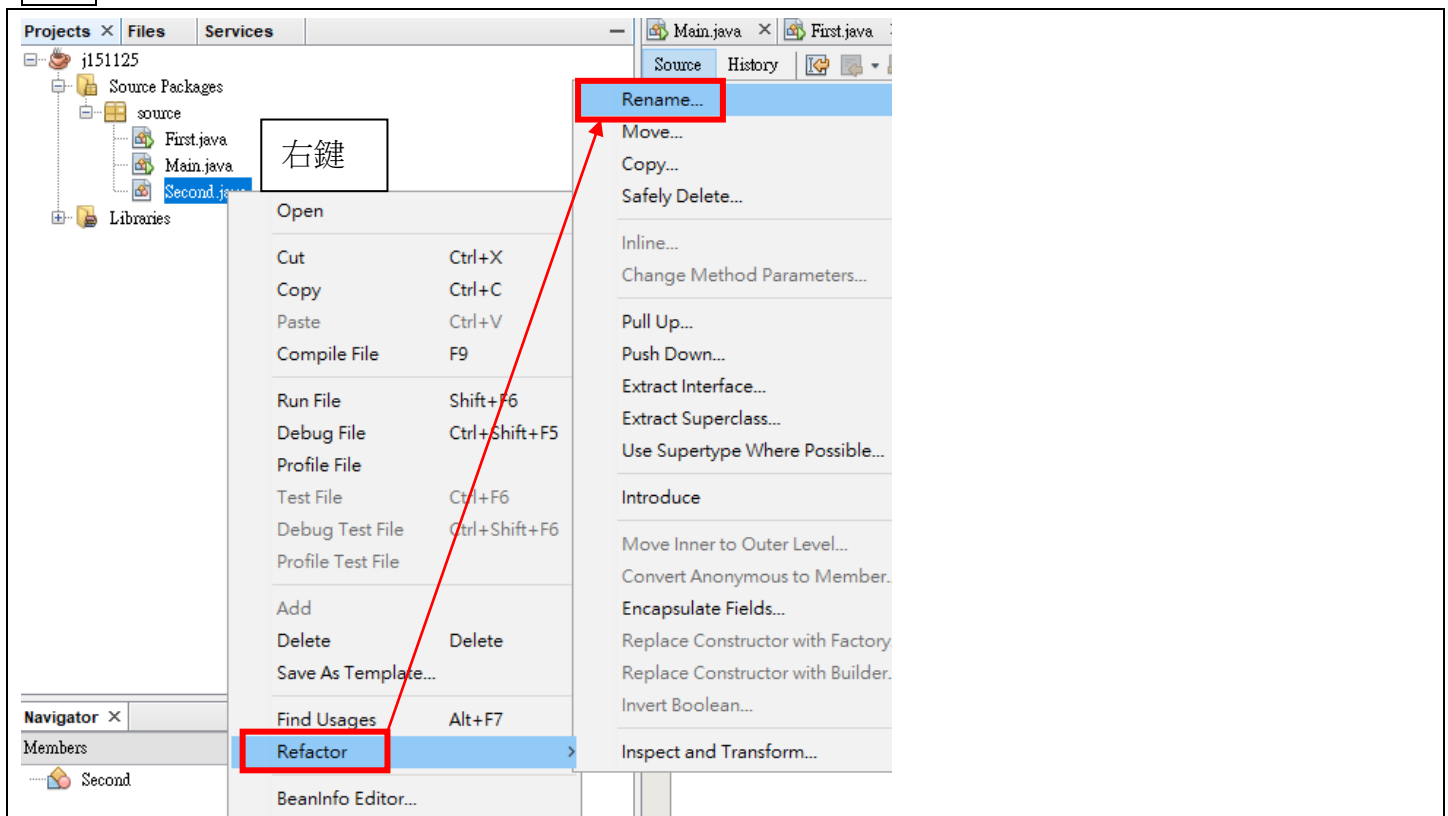
專案重要的資料夾→[src] · [build] · [dist]



改變註解與字型顏色大小



改檔名



刪除範本中的註解

The screenshot illustrates the steps to delete comments from Java templates in NetBeans:

- Tools Menu:** The 'Templates' option is highlighted.
- Template Manager:** The 'Java Class' and 'Java Main Class' templates are listed. The 'Open in Editor' button is visible.
- Java Class Template:**

```

1 <#assign licenseFirst = "/"*>
2 <#assign licensePrefix = " * ">
3 <#assign licenseLast = " */">
4 <#include "${project.licensePath}">
5
6 <#if package?? && package != "">
7   package ${package};
8
9 </#if>
10 /**
11  *
12  * @author ${user}
13  */
14 public class ${name} {
15
16 }
```

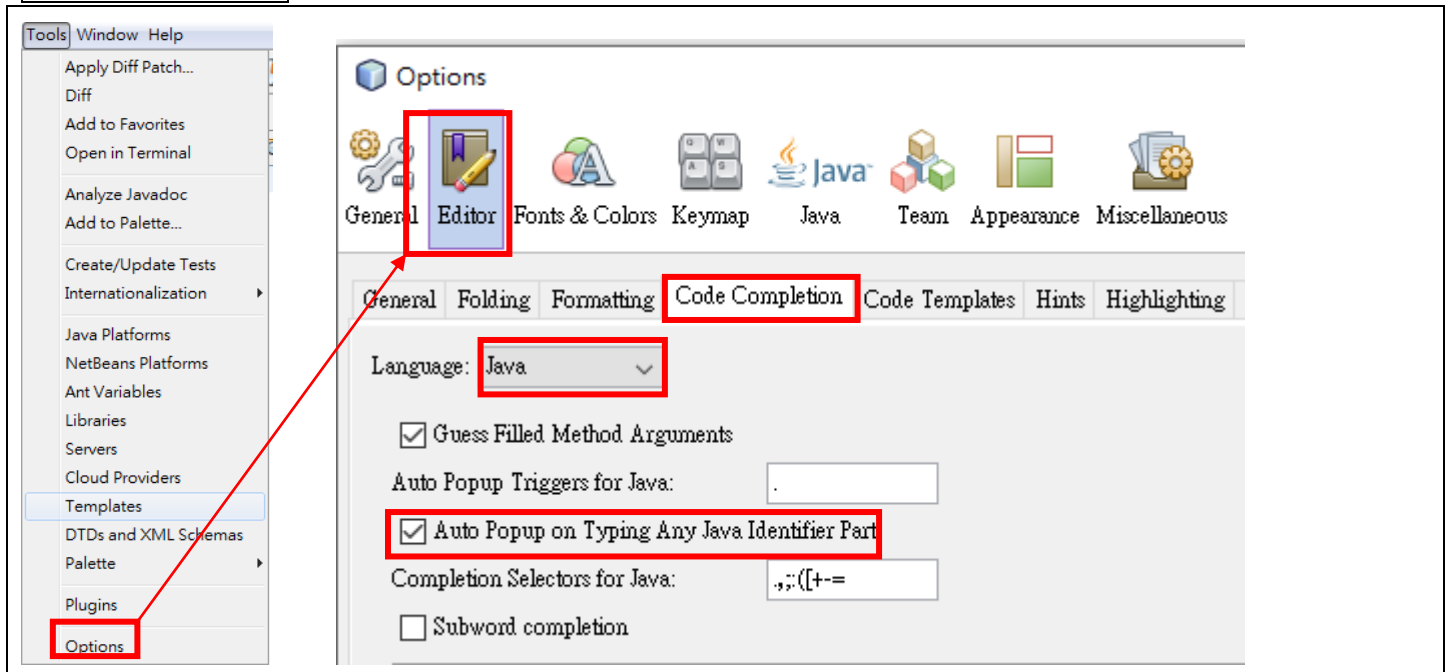
Comments 1-4 and 10-13 are highlighted with red boxes and labeled '可刪除'.
- Java Main Class Template:**

```

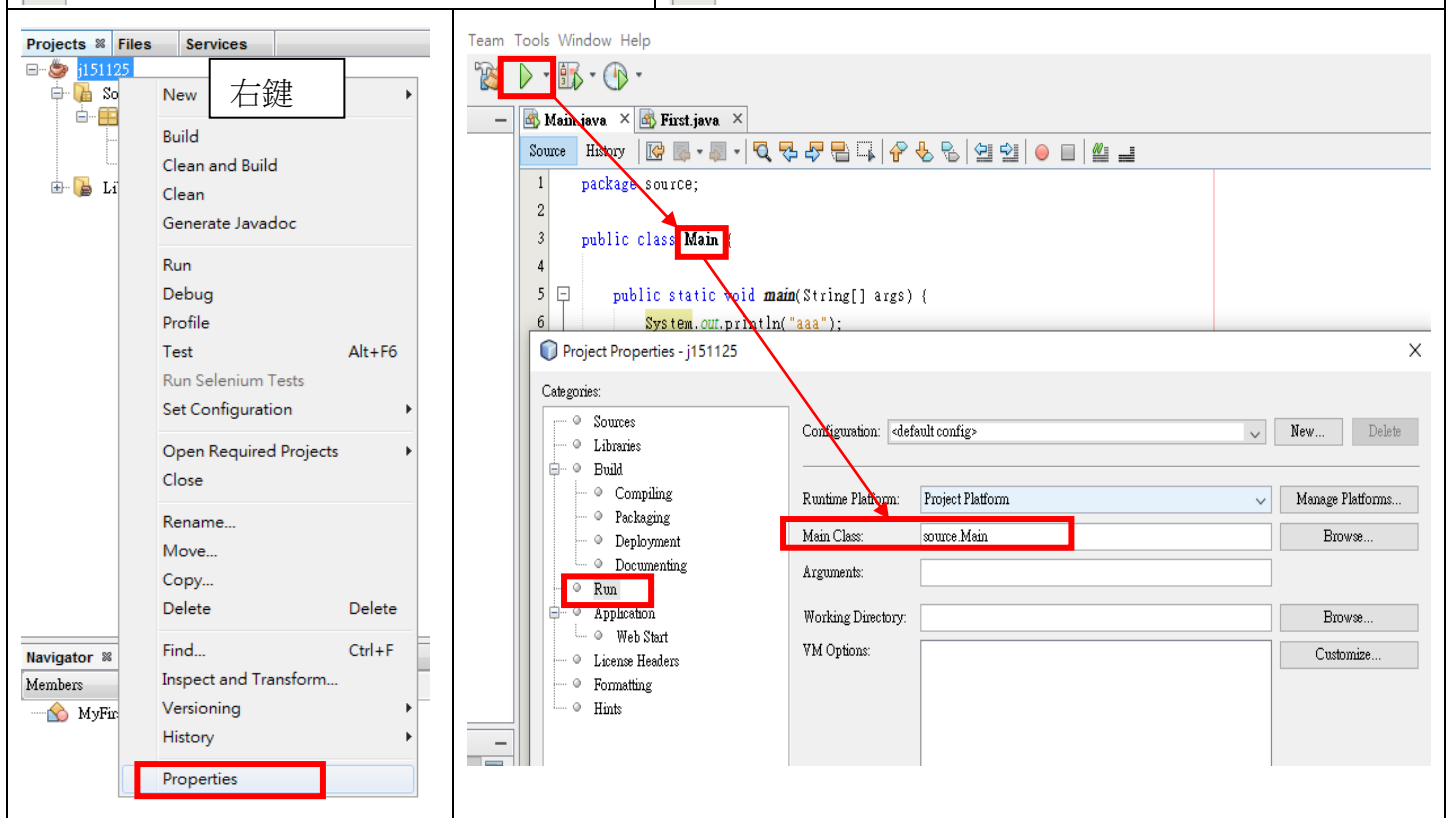
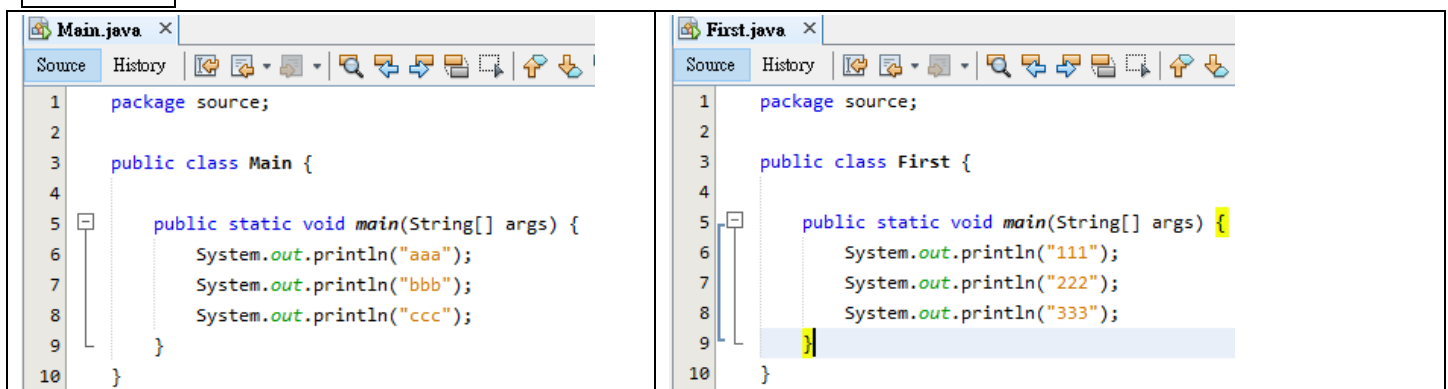
1 <#assign licenseFirst = "/"*>
2 <#assign licensePrefix = " * ">
3 <#assign licenseLast = " */">
4 <#include "${project.licensePath}">
5
6 <#if package?? && package != "">
7   package ${package};
8
9 </#if>
10 /**
11  *
12  * @author ${user}
13  */
14 public class ${name} {
15
16   /**
17    * @param args the command line arguments
18    */
19   public static void main(String[] args) {
20     // TODO code application logic here
21   }
22
23 }
```

Comments 1-4, 10-13, and 16-18 are highlighted with red boxes and labeled '可刪除'.

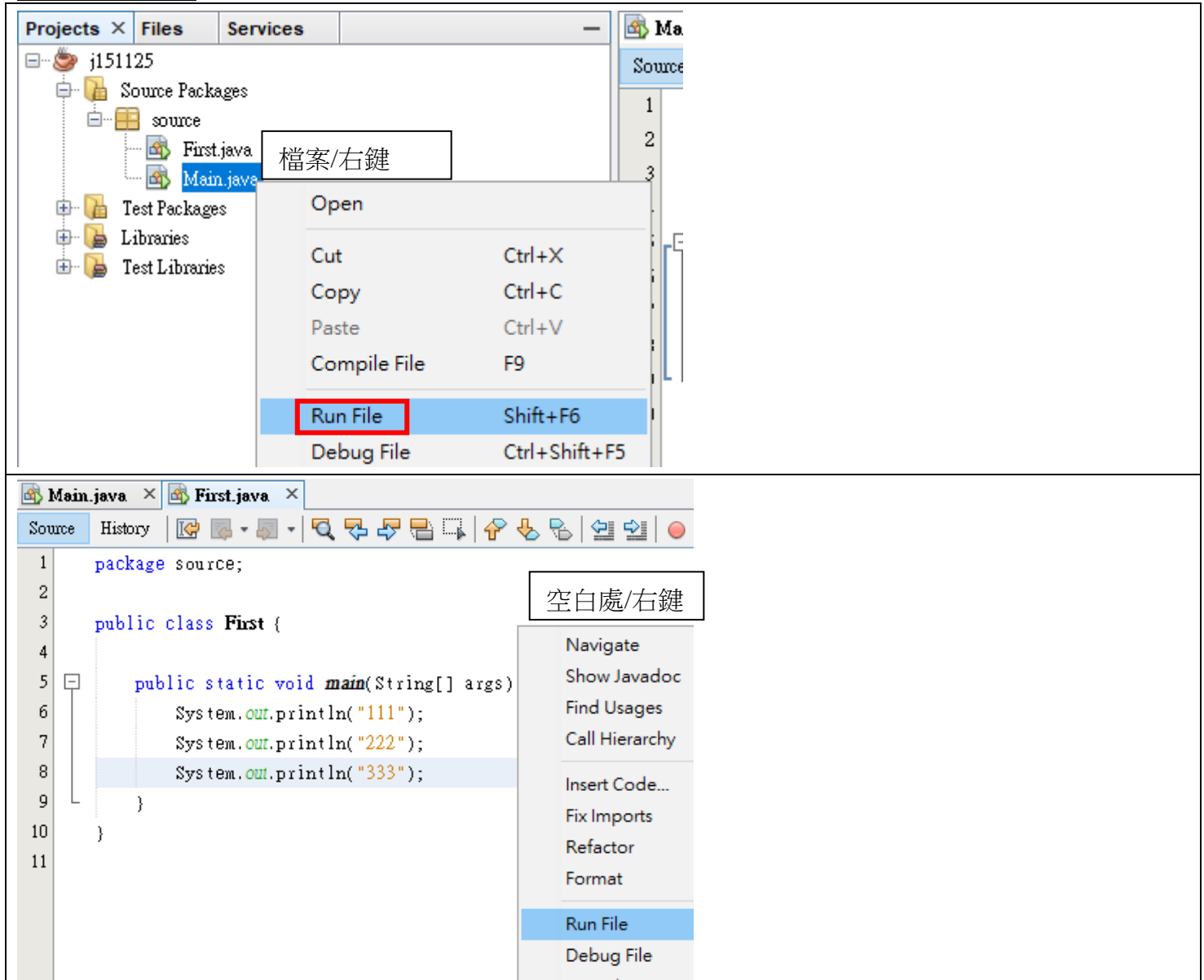
開啟自動秀出指令的功能



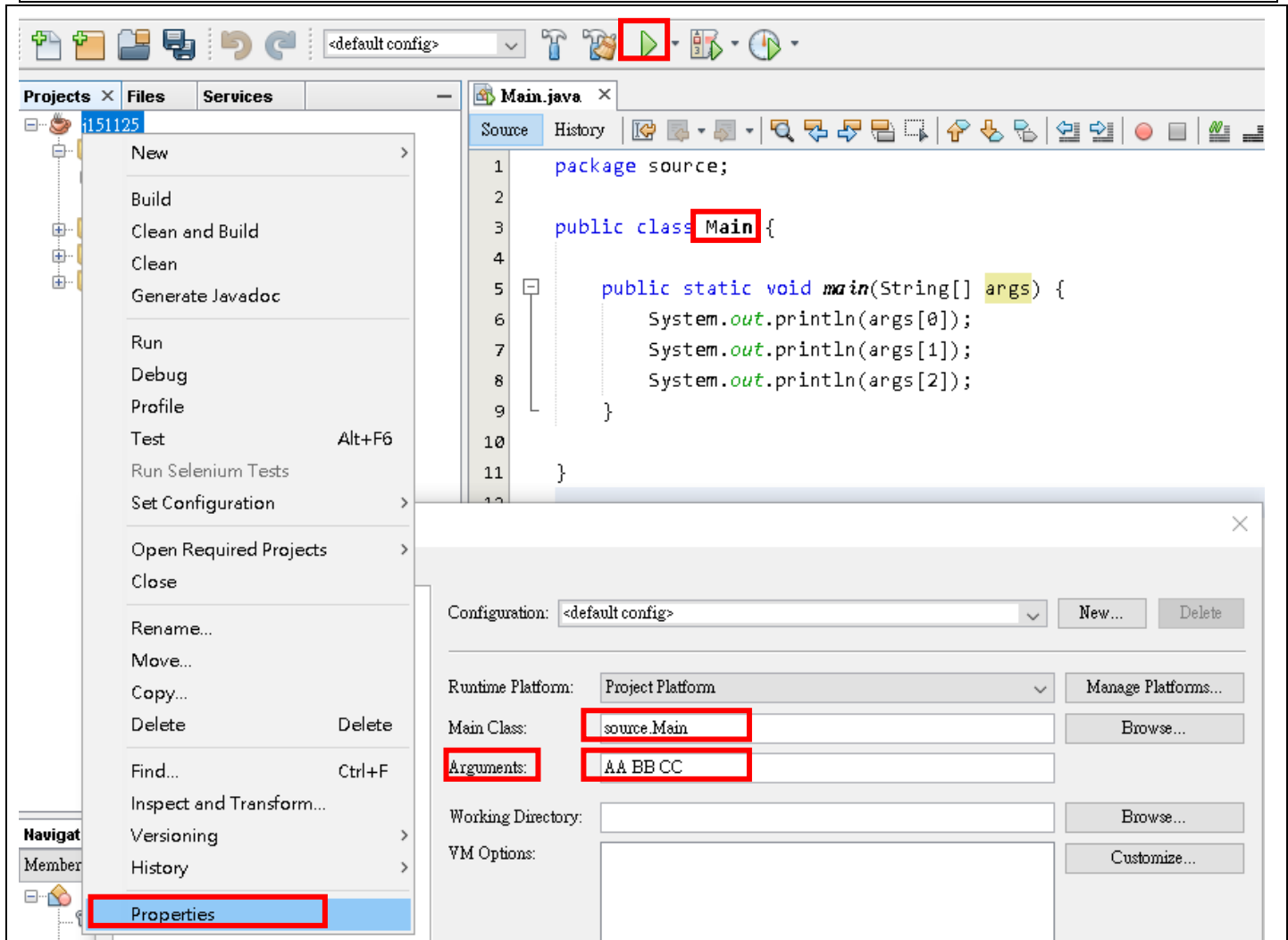
設定主程式的執行點→右鍵/專案/Properties · 或 Run/Set Project Configuration 專案每次都執行同一個主程式，因為專案的屬性已經設定死了！



不讀專案屬性的方式



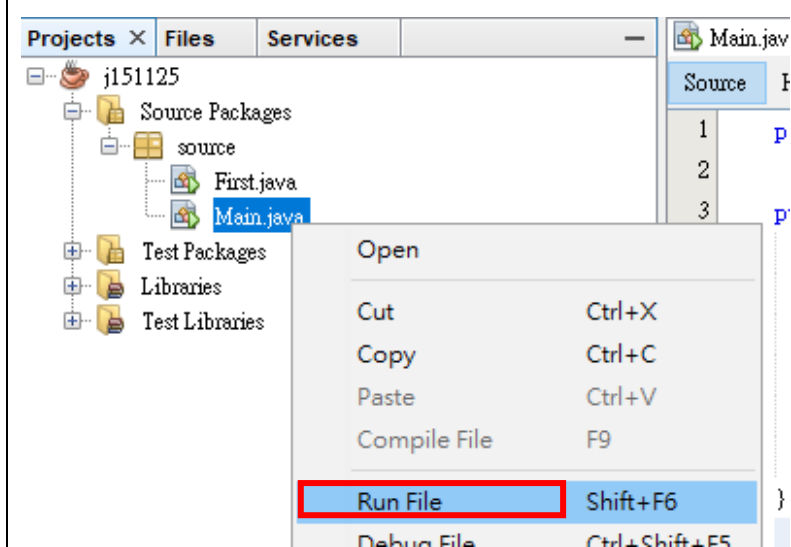
接收參數→要用 三角形 去執行 (才會執行到 專案屬性的設定) · 不可用 右鍵 / Run (因為不會執行到專案屬性的設定) · 否則會當掉



aa

bb

cc



]Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0

呼叫程式→副程式的種類→方法（無回傳值）·函數（有回傳值）

```
public class Main {

    public static void main(String[] args) {
        Main.呼叫程式();
    }

    public static void 呼叫程式() {
        //呼叫方法
        sub1();
        //呼叫函數1
        int x = fun1();
        System.out.println("x=" + x);
        //呼叫函數2
        System.out.println("x=" + fun1());
    }

    public static void sub1() {
        int x = 123;
        System.out.println(x);
    }

    public static int fun1() {
        int x = 123;
        return x;
    }
}
```

```
public class Main {

    public static void main(String[] args) {
        Test01.呼叫程式();
    }
}

public class Test01 {

    public static void 呼叫程式() {
        //呼叫方法
        sub1();
        //呼叫函數1
        int x = fun1();
        System.out.println("x=" + x);
        //呼叫函數2
        System.out.println("x=" + fun1());
    }

    public static void sub1() {
        int x = 123;
        System.out.println(x);
    }

    public static int fun1() {
        int x = 123;
        return x;
    }
}
```

參數傳遞

```

public static void 參數傳遞與回傳() {
    int x = 123;
    int y = 456;
    int z = 789;
    int sum;
    //呼叫方法
    sub2(x, y, z);
    //呼叫函數1
    sum = fun2(x, y, z);
    System.out.println("sum=" + sum);
    //呼叫函數2
    System.out.println("sum=" + fun2(x, y, z));
}

public static void sub2(int a, int b, int c) {
    System.out.println("a=" + a);
    System.out.println("b=" + b);
    System.out.println("c=" + c);
}

public static int fun2(int a, int b, int c) {
    return a + b + c;
}

```

```

public static void 副程式的種類() {
    int x;
    int y;
    int sum;
    x = 4;
    y = 5;
    //方法呼叫
    sumxy(x, y);
    //函數呼叫1
    sum = sumab(x, y);
    System.out.println("函數 sum=" + sum);
    //呼叫函數2
    System.out.println("函數 sum=" + sumab(x, y));
}

//方法 , 程序 , sub
public static void sumxy(int x, int y) {
    int sum;
    sum = x + y;
    System.out.println("方法 sum=" + sum);
}

//函數 , 函式 , function
public static int sumab(int a, int b) {
    int sum;
    sum = a + b;
    return sum;
}

```

結構化程式

```

public static void 結構化程式1() {
    System.out.println("我愛妳");
    System.out.println("我恨妳");
    System.out.println("我喜歡你");
}

```

```

public static void 結構化程式2() {
    印字1();
    印字2();
    印字3();
}

```

```

public static void 印字1() {
    System.out.println("我愛你");
}

public static void 印字2() {
    System.out.println("我恨你");
}

public static void 印字3() {
    System.out.println("我喜歡你");
}

```

```
public static void 結構化程式3() {
    印字("我愛妳");
    印字("我恨妳");
    印字("我喜歡妳");
}
```

```
public static void 結構化程式4() {
    String x = "我愛妳";
    String y = "我恨妳";
    String z = "我喜歡妳";
    印字(x);
    印字(y);
    印字(z);
}
```

```
public static void 印字(String x) {
    System.out.println(x);
}
```

程式的寫法→程序導向(傳統程式) · 物件導向

程序導向 (以方法為主)	物件導向 (以物件為主)
GetWeight(Car1) Start(Car1)	Car1.GetWeight Car1.Start()

1. 程序導向(Procedure Oriented) → 一個好的程序導向程式，結構化程式設計是必要的基本要件

<1>.由上而下分析問題→透過主程式整合

<2>.採 [模組化] 程式設計→方法 · 函式

缺點→一個撰寫完成的程序或函式雖然可重複使用但不具擴充性

→ 複製到另一支程式，修改，再加上新功能→程式會比舊程式更龐大

2. 物件導向(Object-oriented programming 簡稱 OOP) → 物件導向按照人類真實的想法來分析和解決問題

<1>.物件導向三要素

(1).物件具有屬性→外觀

(2).物件具有方法→行為

(3).物件具有事件→事件也是物件的一種方法，只不過這種方法是由“物件本身”或者“其它物件”來啟動執行的

(4).物件要能被識別→car1.Forward() · car2.Forward()→car1 與 car2 採用不同的記憶體空間來使用

<2>.物件導向的設計原理

(1).抽象化的設計 (自訂資料型態) [結構 · 類別 · 列舉] (屬性 · 方法 · 事件)

→以抽象化來處理複雜的事物→只注重物件和物件溝通的行為，與資料內部執行細節沒有關係

(2).封裝 (加上安全性) → 物件必須將私有的部份封裝在物件的內部，使用者只能藉由物件所提供的方法(介面)，來操控物件
這個特性可以有效隱藏物件複雜的內部設計

a. 第一層封裝→將所有資料結構與用來操作該資料結構的所有方法都封裝在物件的類別定義中

b. 第二層封裝→加上 封裝程度 (public · private)

(3).介面 (與外界溝通管道)

(4).繼承 (擴充設計，不用全部將舊程式 copy 到新程式 只要 extends · 再加上新功能) → 由於物件具有 繼承 的特性，使得物件導向程式設計具可 [再用] 和 [擴充性]

Car(汽車) 屬性：燃料 · 重量 · 速度 方法：加速 · 煞車	繼承 (extends)	RaceCar (跑車) 屬性：燃料 · 重量 · 速度 方法：加速 · 煞車 · Turbo
--	--------------	---

(5).多型 (應用)→物件可使用相同功能(方法)介面，來操作不同類型的物件，產生不同行為的一種機制

C=A

C.加速()

C=B

C.加速()

(6).動態繫結 (應用)→編譯階段並不將物件與方法繫結在一起，而是將物件的方法函式的位址建立成一個虛擬表格，在執行階段時，再由虛擬表格中判斷該呼叫那一個物件的方法或函式

<3>.抽象化，封裝，介面 說明

(1).物件與介面

- a. 物件是一個黑盒子，外面的人看不到它的內部構造，但是卻可以透過它所提供的介面去使用物件
販賣機物件的介面→銀兩該往那裡投，選擇按鈕在那裡，東西從那裡掉出來，從何處取回剩下的零錢
- b. 把物件看成是黑盒子的動作，包含了兩個物件的基本觀念
 1. 抽象化→在消費者眼中的販賣機好比是個“販賣機黑盒子”
提供了“投幣口”，“退幣口”，“選擇鈕”，“商品出口”等四個商品介面
消費者只需了解怎麼使用這四個介面，便可架馭販賣機
把複雜的販賣機看作是一個黑盒子與介面的過程，叫“抽象化”
 2. 封裝→販賣機製造商把實作販賣機的細節從消費者的面前隱藏起來，這個隱藏的動作稱為“封裝”
- c. 封裝所帶來的好處，可以分別從物件的使用者 與 創造者的角度來看
 1. 使用者：只需要，也只能了解使用介面，使用起來很簡單，使用者無法知道內部細節，必須依照使用介面所提供的一切使用物件
 2. 創造者：因為使用者一切按介面來，因此物件創造者可以隨時更新物件內部實作方式，只要保證修改成相同的介面即可

(2).物件與類別

- a. 物件是由類別而來，類別是物件的藍圖，就好比是人與人類之間的關係
- b. 物件是類別的實體，一種類別可以有很多的物件實體(Instance)
- c. 類別是靜態的程式碼，而物件是類別執行起來的樣子

<4>.物件導向設計步驟

1. 定義類別 人類 (元件)(類別)(模型)(自訂資料型態) 身高-----名詞(屬性) 體重-----名詞(屬性) 吃()-----動詞(方法) 跑()-----動詞(方法)	2. 定義物件 3. 由類別產生物件 4. 使用屬性與方法
程序導向→沒有元件可以用	

```

public static void 傳統程式() {
    int 長;
    int 寬;

    長 = 10;
    寬 = 5;

    長方形面積(長, 寬);
    長方形周長(長, 寬);
}

public static void 長方形面積(int 長, int 寬) {
    int 面積;
    面積 = 長 * 寬;
    System.out.println("面積=" + 面積);
}

public static void 長方形周長(int 長, int 寬) {
    int 周長;
    周長 = (長 + 寬) * 2;
    System.out.println("周長=" + 周長);
}

```

物件導向程式 → 需要設計 元件 · 加上 封裝

```

public class Rectangle1 {

    public int 長;
    public int 寬;

    public void 長方形面積() {
        int 面積;
        面積 = 長 * 寬;
        System.out.println("面積=" + 面積);
    }

    public void 長方形周長() {
        int 周長;
        周長 = (長 + 寬) * 2;
        System.out.println("周長=" + 周長);
    }
}

```

```

public static void 物件導向程式1() {
    Rectangle1 長方形1 = new Rectangle1();
    長方形1.寬 = 10;
    長方形1.長 = 20;
    長方形1.長方形面積();
    長方形1.長方形周長();

    Rectangle1 長方形2 = new Rectangle1();
    長方形2.寬 = 20;
    長方形2.長 = 30;
    長方形2.長方形面積();
    長方形2.長方形周長();
}

```

```

public class Rectangle2 {
    private int 長; //我被鎖住了
    private int 寬; //我被鎖住了

    public void 長方形面積() {
        int 面積;
        面積 = 長 * 寬;
        System.out.println("面積=" + 面積);
    }

    public void 長方形周長() {
        int 周長;
        周長 = (長 + 寬) * 2;
        System.out.println("周長=" + 周長);
    }

    //我是跟外介溝通的管道 ( 介面 )
    public void set長寬(int h, int w) {
        長 = h;
        寬 = w;
    }
}

```

```

public static void 物件導向程式2() {
    Rectangle2 長方形1 = new Rectangle2();
    長方形1.set長寬(10, 20);
    長方形1.長方形面積();
    長方形1.長方形周長();

    Rectangle2 長方形2 = new Rectangle2();
    長方形2.set長寬(20, 30);
    長方形2.長方形面積();
    長方形2.長方形周長();
}

```

圖

```

public class Main {
    public static void main(String[] args) {
        Test01.傳統程式();
        Test01.物件導向程式1();
    }
}

public class Test01 {

    public static void 傳統程式() {
        長方形面積(長, 寬);
        長方形周長(長, 寬);
    }

    public static void 長方形面積(int 長, int 寬) {
    }

    public static void 長方形周長(int 長, int 寬) {
    }

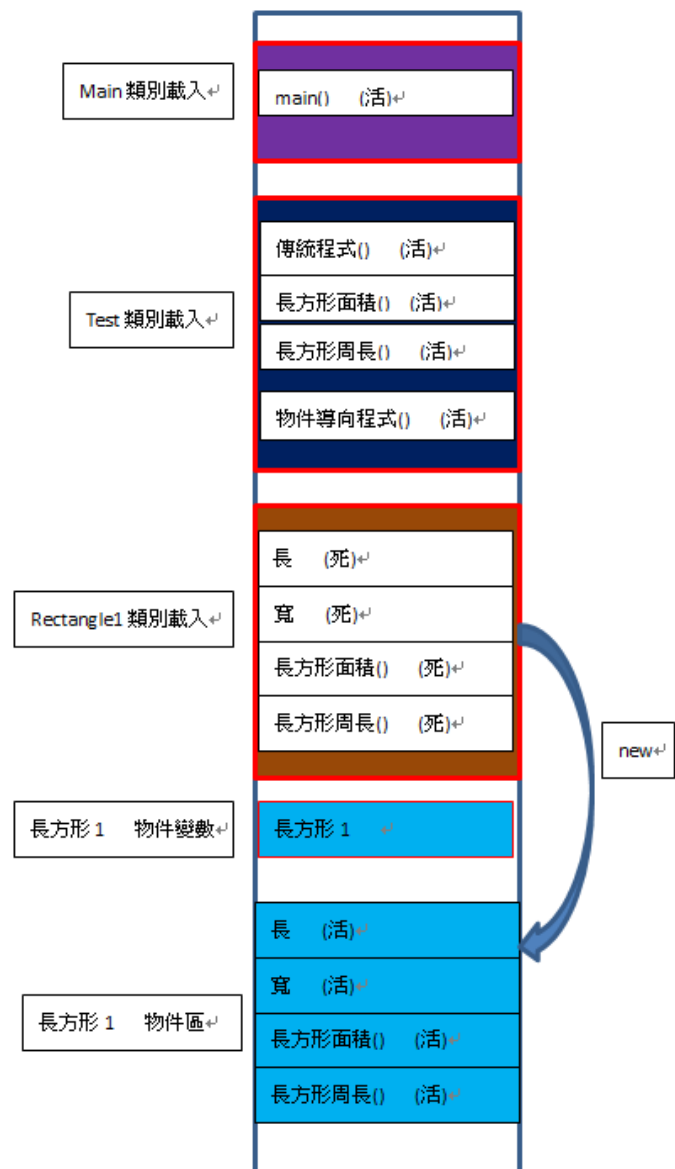
    public static void 物件導向程式1() {
        Rectangle1 長方形1 = new Rectangle1();
    }
}

public class Rectangle1 {
    public int 長;
    public int 寬;

    public void 長方形面積() {
    }

    public void 長方形周長() {
    }
}

```




```

public class Human1 {

    public String 名字;
    public int 身高;
    public int 體重;

    public void 吃(int x) {
        System.out.println("吃" + x + "碗飯");
    }

    public void 跑(int y) {
        System.out.println("跑" + y + "公里");
    }
}

```

```

public static void 由類別產生物件1() {
    Human1 人1 = new Human1();
    人1.名字 = "林青霞";
    人1.身高 = 190;
    人1.體重 = 50;
    System.out.println("人1.名字 " + 人1.名字);
    System.out.println("人1.身高 " + 人1.身高 + "公分");
    System.out.println("人1.體重 " + 人1.體重 + "公斤");
    人1.吃(5);
    人1.跑(10);

    Human1 人2 = new Human1();
    人2.名字 = "張學友";
    人2.身高 = 180;
    人2.體重 = 70;
    System.out.println("人2.名字 " + 人2.名字);
    System.out.println("人2.身高 " + 人2.身高 + "公分");
    System.out.println("人2.體重 " + 人2.體重 + "公斤");
    人2.吃(10);
    人2.跑(20);
}

```

```

public class Human2 {
    //我被鎖住了

    private String 名字;
    private int 身高;
    private int 體重;

    public void 吃(int x) {
        System.out.println("吃" + x + "碗飯");
    }

    public void 跑(int y) {
        System.out.println("跑" + y + "公里");
    }

    //我是跟外界溝通的管道 介面
    public void setdata(String n, int h, int w) {
        名字 = n;
        身高 = h;
        體重 = w;
    }

    public void getdata() {
        System.out.println("名字 " + 名字);
        System.out.println("身高 " + 身高 + "公分");
        System.out.println("體重 " + 體重 + "公斤");
    }
}

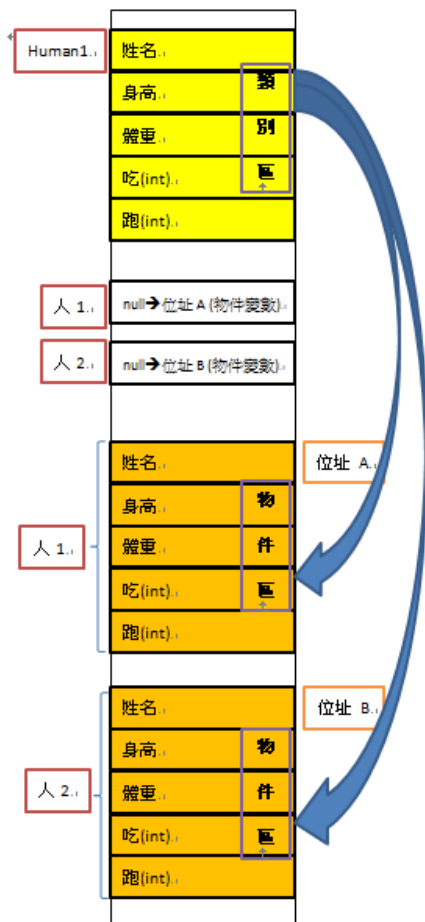
```

```

public static void 由類別產生物件2() {
    Human2 人1 = new Human2();
    人1.setdata("林青霞", 190, 50); //透過介面
    人1.getdata(); //透過介面
    人1.吃(5);
    人1.跑(10);

    Human2 人2 = new Human2();
    人2.setdata("張學友", 180, 70); //透過介面
    人2.getdata(); //透過介面
    人2.吃(10);
    人2.跑(20);
}

```



Java 的專案 → 可能由很多的類別所組成

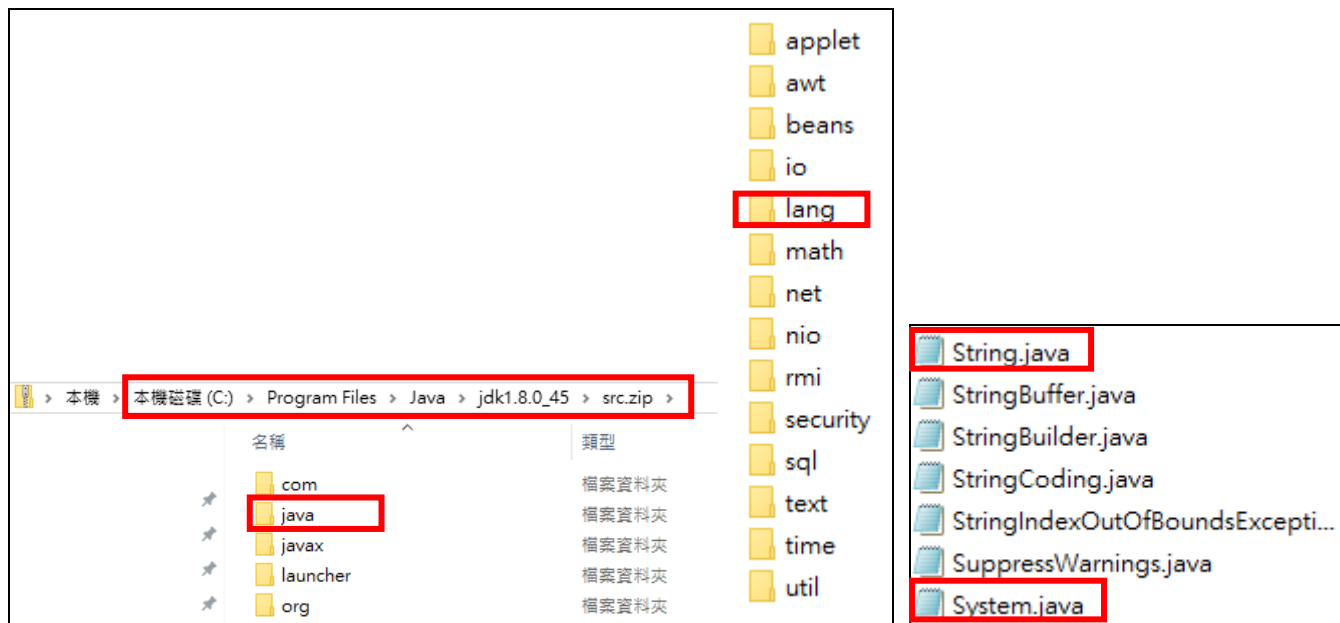
甲專案 由 A 類別 · B 類別 · C 類別 組成

1. 有幾種寫法(寫在同一個檔案裡 · 或分開寫)
2. 判斷主要類別的幾個要素 → 有 public · 檔名 與 類別名稱一樣

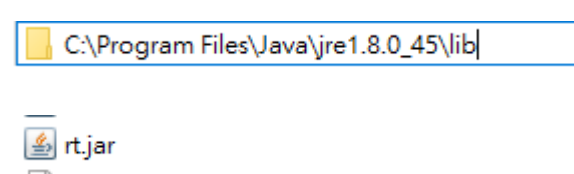
	<pre>package 甲; public class A { } class B{ } class C{ }</pre>		
	<pre>package 甲; public class A { }</pre>	<pre>package 甲; public class B { }</pre>	<pre>package 甲; public class C { }</pre>

Java API

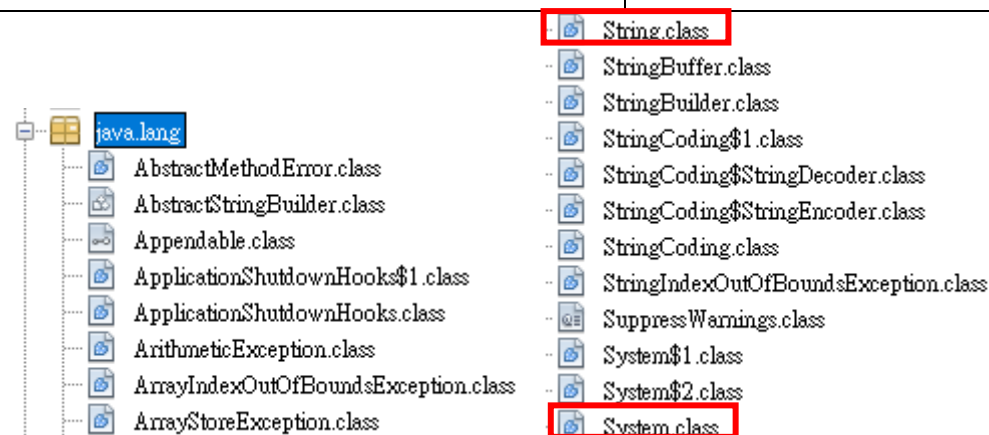
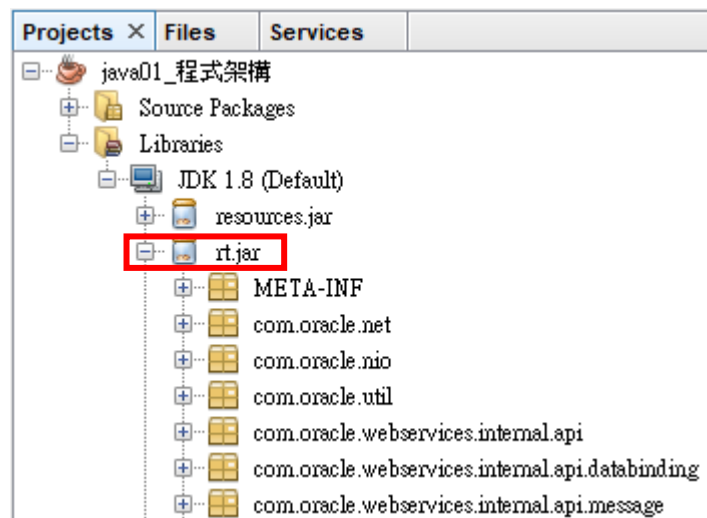
1. 原始檔



2. api 的壓縮檔 class→.jar



3. netbeans



Java SDK Documentation → <http://www.oracle.com/technetwork/java/api-141528.html>

Oracle Technology Network > Java > Java SE > Documentation

Java SE

Java EE

Java ME

Java SE Support

Java SE Advanced & Suite

Java Embedded

JavaFX

Java DB

Web Tier

Java Card

Java TV

New to Java

Community

Java Magazine

Overview

Downloads

Documentation

Community

Technologies

Training

Java SE Documentation at a Glance

At a Glance

Code

API


Tutorials



Technical Articles

White Papers

FAQs

A wealth of information is available to help you learn and use Java platform technology. In addition, some Technologies offer listings of reference material specific to that technology.

 See the Java SE Documentation

Release Notes

This page lists the update release notes summarizing changes made in all Java Platform, Standard Edition (Java SE) and the JDK. See also Java SE 7 Features and Enhancements, which includes information on features and enhancements in Java SE 7 and in JDK 7, Oracle's implementation of Java SE

Product License and Terms

- Binary Code License for Java SE Platform Products (HTML, PDF)
- Overview of the Java SE Product Editions and the Commercial Features available in each edition (HTML, PDF)

README Files

Java SDKs and Tools

Java SE

Java EE and Glassfish

Java ME

JavaFX

Java Card

NetBeans IDE

Java Mission Control

Java Resources

Java APIs

Technical Articles

Demos and Videos

Forums

Java Magazine

Java.net

Developer Training

Tutorials

Java.com

JAVA STANDARD EDITION

Java SE 8

Java SE 7

Java SE 6

J2SE 1.5.0

J2SE 1.4.2

J2SE 1.3.1

java01_程式架構.doc

36 / 38

2019/8/2

The screenshot shows the Java Platform Standard Ed. 7 API documentation for the `System` class. The left sidebar contains a navigation menu with sections: All Classes, Packages (highlighted with a red box), java.lang, Interfaces, and Classes (highlighted with a red box). The main content area displays the `System` class details, including its inheritance hierarchy, a description, and two summary tables.

Class System

`java.lang.Object`
`java.lang.System`

`public final class System`
`extends Object`

The `System` class contains several useful class fields and methods. It cannot be instantiated.

Among the facilities provided by the `System` class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

Since:
JDK1.0

Field Summary

Fields	
Modifier and Type	Field and Description
<code>static PrintStream</code>	<code>err</code> The "standard" error output stream.
<code>static InputStream</code>	<code>in</code> The "standard" input stream.
<code>static PrintStream</code>	<code>out</code> The "standard" output stream.

Method Summary

Methods	
Modifier and Type	Method and Description
<code>static void</code>	<code>arraycopy(Object src, int srcPos, Object dest, int destPos, int length)</code> Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array.
<code>static String</code>	<code>clearProperty(String key)</code> Removes the system property indicated by the specified key.
<code>static Console</code>	<code>console()</code> Returns the unique <code>Console</code> object associated with the current Java virtual machine, if any.
<code>static long</code>	<code>currentTimeMillis()</code> Returns the current time in milliseconds.
<code>static void</code>	<code>exit(int status)</code> Terminates the currently running Java Virtual Machine.

