Eclipse - 整合開發工具

基礎篇

Jacky Lee

2005/03/01

目錄

0.環境說明

- 作業系統
  - Microsoft Windows XP Professional
  - Service Pack 2

- Eclipse 版本
  - Version：Eclipse 3.0.1 SDK (Release)
  - Build id：200409161125
  - File Name：eclipse-SDK-3.0.1-win32.zip

- 參考資料
  - Eclipse's Help
  - O'REILLY Eclipse 整合開發工具
  - 博碩文化 Eclipse 實作手冊-活用 Java 整合開發環境

1.Eclipse 簡介

Eclipse 就像軟體開發者的『打鐵鋪』，它一開始備有火爐、鐵鉆與鐵鎚。就像鐵匠會用現有的工具打造新的工具，也能用 Eclipse 打造新工具來開發軟體-這些新工具可擴充 Eclipse 的功能。(Eclipse 其中一個賣點就是它的擴充性)

1.1 歷史背景

Eclipse 這樣功能完整且成熟的開發環境，是由藍色巨人 IBM 所釋出。IBM 花了 4 千萬美金來開發這個 IDE(Integrated Development Environment)。第一版 1.0 在 2001 年 11 月釋出，隨後逐漸受到歡迎。

Eclipse已經成為開放原始碼計劃(Open Source Project)，大部分的開發扔然掌握在IBM手中，但是有一部份由eclipse.org的軟體聯盟主導。(http://www.eclipse.org)

Eclipse 專案由 Project Management Committee(PMC)所管理，它綜觀專案全局，Eclipse 專案分成 3 個子專案：

- 平台-Platform
- 開發工具箱-Java Development Toolkit(JDT)
- 外掛開發環境-Plug-in Development Environment(PDE)

這些子專案又細分成更多子專案。例如 Platform 子專案包含數各元件，如 Compare、Help 與 Search。JDT 子專案包括三各元件：User Interface(UI)、核心(Core)及除錯(Debug)。PDE 子專案包含兩各元件：UI 與 Core。

1.2 開發原始碼軟體

Eclipse 是開放原始碼，結果很多人在使用的時候都不注重合法權的問題。開放原始碼軟體讓使用者能夠取得軟體的原始碼，有權去修改和散佈這個軟體。如果想修改軟體，這件事的另一面就是，除非其他人對修改後的軟體也有相同的權力，否則是不能散佈修改後的軟體，這種權利和著作權(copyright)相反，開放原始碼專案中有時稱之為著作義(copyleft)。

有些開放原始碼許可書，堅持要求任何和其它開發原始碼合組成的軟體也必須是開放原始碼。然而，Eclipse 使用的開放原始碼許可書：公共公眾許可書-Common Public License(CPL)作為授權方式，設計上是可以容許商業利益的。CPL 可以容許 Eclipse 和其他開放原始碼軟體合組時，能夠以更嚴謹的許可書散佈軟體，以求用於商業途徑。CPL 經過 Open Software Initiative(OSI)認證，其內容符合開放原始碼授權的需求。

1.3 Eclipse 版本介紹

可以從eclipse.org網站(http://www.eclipse.org/downloads)下載，可以發現『最新』與『最好』的版本，這兩種版本通常不一樣，基本上有四種版本-或建置(build)可供下載：

- 釋出版(Release builds)

  由 Eclipse 開發團隊所宣稱的主要穩定版本。Release builds 經過完整測試，並具有一致性、定義清楚的功能。它的定位就跟上市的商業軟體一樣。

- 穩定版(Stable builds)

  比 Release build 新一級的版本，經由 Eclipse 開發團隊測

試，並認定它相當穩定。新功能通常會在此過渡版本出現。它的定位就跟商業軟體的 beta 版一樣。

- 整合版(Integration builds)
  此版本的各個獨立的元件已經過 Eclipse 開發團隊認定具穩定度，但不保證兜在一起沒問題。若兜在一起夠穩定，它就有可能晉級成 Stable build。
- 當日最新版(Nightly builds)
  此版本顯然是從最新的原始碼產生出來的。可想而知，此版本當然不保證它跑起來沒問題，搞不好還有嚴重的 bug。

1.4 跨語言、跨平台

多數人認為 Eclipse 是 Java IDE，不過，當下載 Eclipse 之後，除了有 Java IDE(就是 JDT)，還有 PDE。然而 Eclipse 是萬用工具平台。JDT 實際上是 Eclipse 的添加品，也就是外掛程式。Eclipse 本身實際上是指 Eclipse 平台(Eclipse Platform)，除了下載時能取得 Java 工具集以外，還提供各種工具的支援，所以平台本身只是相當小的一組軟體。

如果想開發 Java 程式，用的是 Eclipse 隨附的 JDT 外掛程式。如果想開發其它語言的程式，就需要拿到其他外掛程式，諸如 CDT(C Development Toolkit)就可以開發 C/C++程式。

Eclipse 跨電腦語言，也跨人類的語言。相同的外掛機制可用來增加對不同語言的支援，這裡使用一種特殊的外掛，叫做外掛程式片斷(plug-in fragment)。IBM 以捐出一個語言套件，支援中文(繁體與簡體)、法文、德文、義大利文、日文、韓文、葡萄牙文(巴西)與西班牙文。

照理說 Eclipse 以 Java 寫成，應該可以在任何的平台執行。但嚴

格來說 Eclipse 不是跨平台的，因為它使用作業平台的原生圖形來建置。因此要等 SWT(Standard Widget Toolkit)移植到該平台，Eclipse 才能在那個平台執行。但就現實而言到不是什麼大問題，因為 SWT 已經被移植到數個常見平台上了，包括 Windows、Linux/Motif、Linux/GTK2、Solaris、QNX、AIX、HP-UX 與 Mac OS X。

2. Eclipse Platform

Eclipse 平台的目的，是提供多種軟體開發工具的整合機制，這些工具會實作成 Eclipse 外掛程式，平台必須用外掛程式加以擴充才有用處。Eclipse 設計美妙之處，在於所有東西都是外掛，除了底層的核心以外。這種外掛設計讓 Eclipse 具備強大擴充性，但更重要的是，此平台提供一個定義明確的機制，讓各種外掛程式共通合作(透過延伸點 extension points)與貢獻(contributions))，因此新功能可以輕易且無縫地加入平台。

2.1 概觀

第一次執行 Eclipse 時，會在 Eclipse 目錄下建一個 workspace 的目錄，根據預設，所有的工作都會存在此目錄。若要備份工作目錄，只要備份這個目錄就行了。若要升級至新版的 Eclipse，只要將這個目錄拷貝過去即可。

用新版時得看看 release notes，確保它支援前一版的 workspace；若不支援，只要將舊的 workspace 子目錄拷貝到新的 Eclipse 目錄下即可。所有的喜好設定都會保留。

2.2 架構

Eclipse 平台由數種元件組成：平台核心(platform kernel)、工作台(workbench)、工作區(workspace)、團隊元件(tram component)以及說明元件(help)。

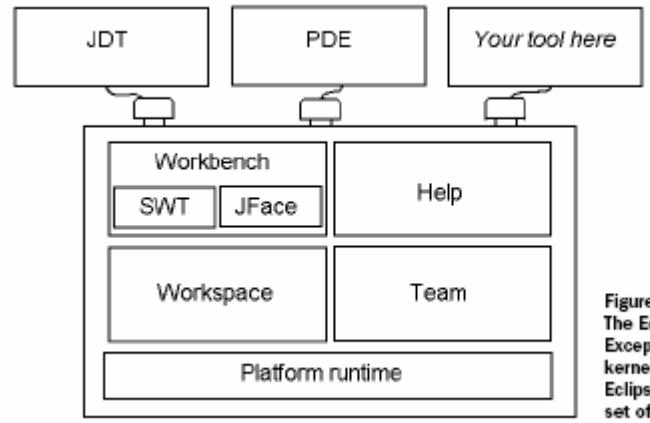


Figure 1.1
The Eclipse architecture. Except for a small runtime kernel, everything in Eclipse is a plug-in or a set of related plug-ins.

圖 2.0

2.3 專案與資料夾

若想要手動操作檔案、拷貝或看檔案大小，就得知道檔案放哪裡。但原生檔案系統會隨作業系統而變，這對在各個作業系統均需運作一致的程式會發生問題。為了解決此問題，Eclipse 在檔案系統之上提供了一個抽象層級。換句話說，它不使用內含檔案的階層式目錄/子目錄結構，反之，Eclipse 在最高層級使用『專案』，並在專案之下使用資料夾。

根據預設，『專案』對應到 workspace 目錄下的子目錄，而『資料夾』對應到專案目錄下的子目錄。在 Eclipse 專案內的所有東西均是以獨立與平台無關的方式存在。

2.4 平台核心

核心的任務是讓每樣東西動起來，並載入所需之外掛程式。當啟動 Eclipse 時，先執行的就是這個元件，再由這個元件載入其他外掛程式。

2.5 工作區(workspace)

工作區負責管理使用者的資源，這些資源會被組織成一個(或多個)專案，擺在最上層。每個專案對應到 Eclipse 工作區目錄下的一個子目錄。每個專案可包含多個檔案和資料夾；通常每個資料夾對應到一個在專案目錄下的子目錄,但資料夾也可連到檔案系統中的任意目錄。

每個工作區維護一個低階的歷史紀錄，記錄每個資源的改變。如此便可以立刻復原改變，回到前一個儲存的狀態，可能是前一天或是幾天前，取決於使用者對歷史紀錄的設定。此歷史紀錄可將資源喪失的風險減到最少。

工作區也負責通知相關工具有關工作區資源的改變。工具可為專案標記一個專案性質(project nature)，譬如標記為一個“Java 專案”，並可在必要時提供配置專案資源的程式碼。

2.6 工作台(workbench)

Eclipse工作台(workbench)就如 圖 2.1 的畫面，這是操作Eclipse時會碰到的基本圖型介面，工作台是Eclipse之中僅次於平台核心最基本的元件，啟動Eclipse後出現的主要視窗就是這個，workbench的工作很簡單：讓操作專案。它不懂得如何編輯、執行、除錯，它只懂得如何找到專案與資源(如檔案與資料夾)。若有它不能做的工作，它就丟給其他元件，例如JDT。

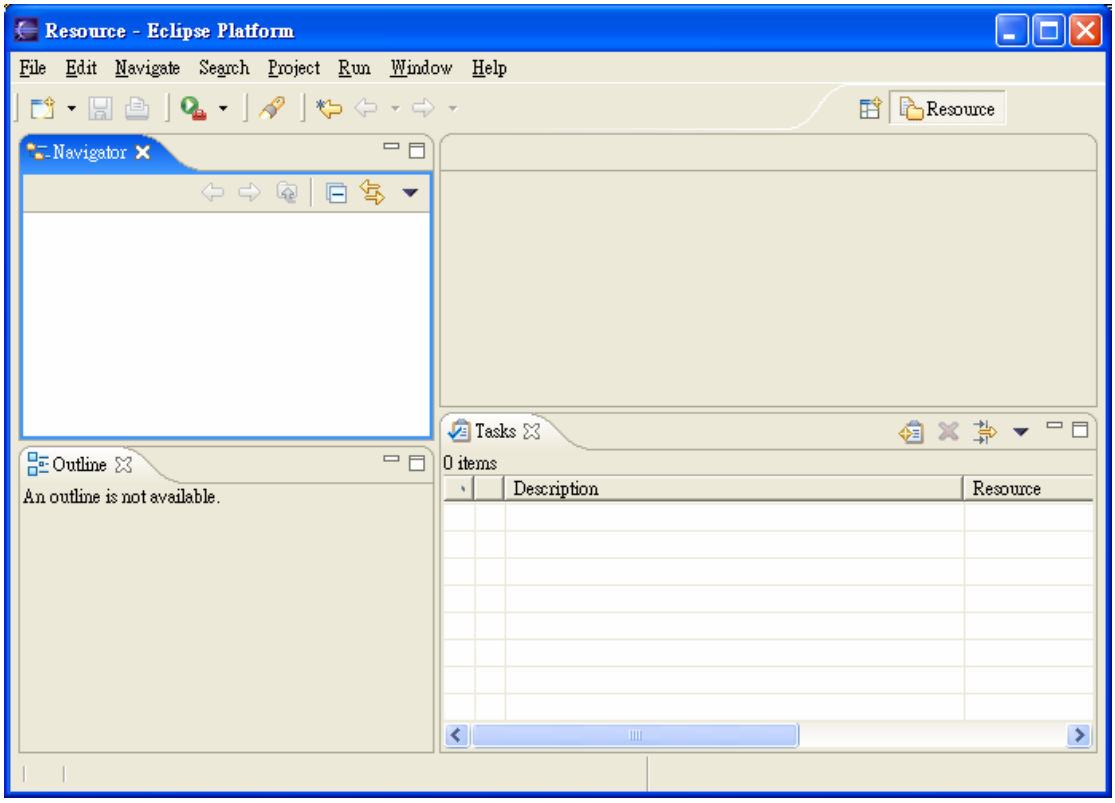


圖 2.1

工作台看起來像是作業系統內建的應用程式，可以說是 Eclipse 的特點，同時也是爭議點。工作台本身可以說是 Eclipse 的圖形操作介面，它是用 Eclipse 自己的標準圖形工具箱(Standard Widget Toolkit-SWT)和 JFace(建立在 SWT 之上)的架構。SWT 會使用作業系統的圖形支援技術，使得程式的外觀感覺(look-and-feel)隨作業系統而定。這一點和過去多數 Java 程式的做法很不同，即使是用 Swing，也沒有這樣過。

2.6.1 視圖(View)

工作台會有許多不同種類的內部視窗，稱之為視圖(view)，以及一個特別的視窗-編輯器(editor)。之所以稱為視圖，是因為這些是視窗以不同的視野來看整各專案，例如 圖 2.1，Outline的視圖可以看專

案中Java類別的概略狀況，而Navigator的視圖可以導覽整各專案。

視圖支援編輯器，且可提供工作台中之資訊的替代呈現或導覽方式。比方說:「書籤」視圖會顯示工作台中的所有書籤且會附帶書籤所關聯的檔案名稱。「Navigator」視圖會顯示專案和其他資源。在已附加標籤的筆記本中，視圖可獨自呈現，也可以與其他視圖形成堆疊。

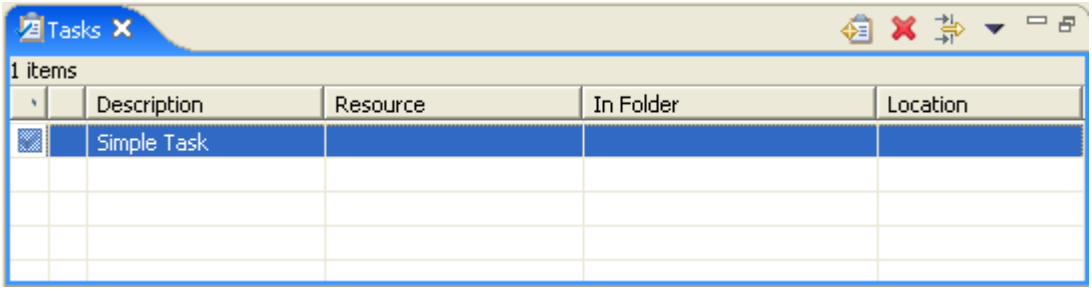


圖 2.2

如果要啟動在附加標籤的筆記本中的視圖，只要按一下標籤就行了。工作台會提供了許多又快又簡單的方式供配置環境，其中包括標籤在筆記本的底端或頂端。

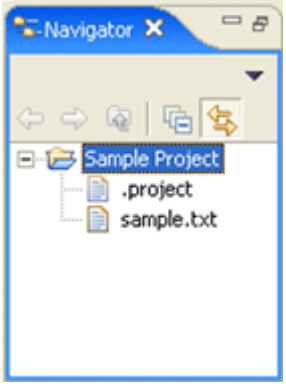


圖 2.3

視圖有兩個功能表，第一個是用滑鼠右鍵按一下視圖標籤來存取的功能表，它可以利用類似工作台視窗相關功能表的相同方式來操作視圖。

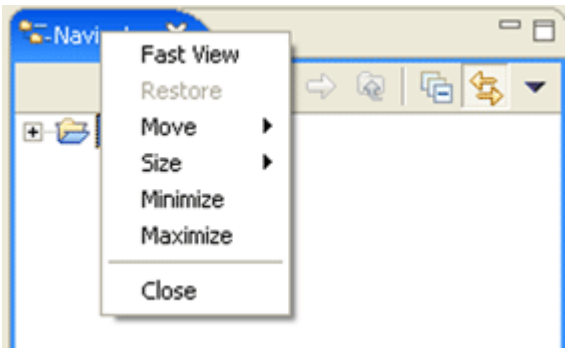


圖 2.4

第二個功能表稱為「視圖下拉功能表」，存取方式是按一下向下箭頭 。視圖下拉功能表所包含的作業通常會套用到視圖的全部內容，而不是套用到視圖中所顯示的特定項目。排序和過濾作業通常可在檢視下拉功能表中找到。

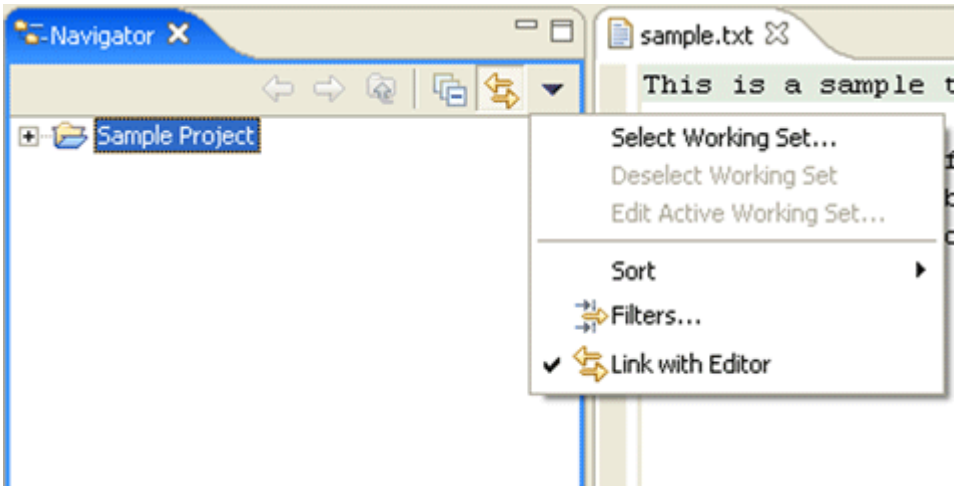


圖 2.5

自訂工作台是使用「Window」→「Reset Perspective」功能表作業的好時機。重設作業會將佈置還原成程式狀態。

可以從「Window」→「Show View」功能表中選取一個視圖來顯示它。視景決定了哪些視圖是必要的，它會將這些視圖顯示在「Show View」子功能表中。選擇「Show View」子功能表底端的「Other...」時，就可以使用其他的視圖。這只是可用來建立自訂工作環境的許多功能之一。

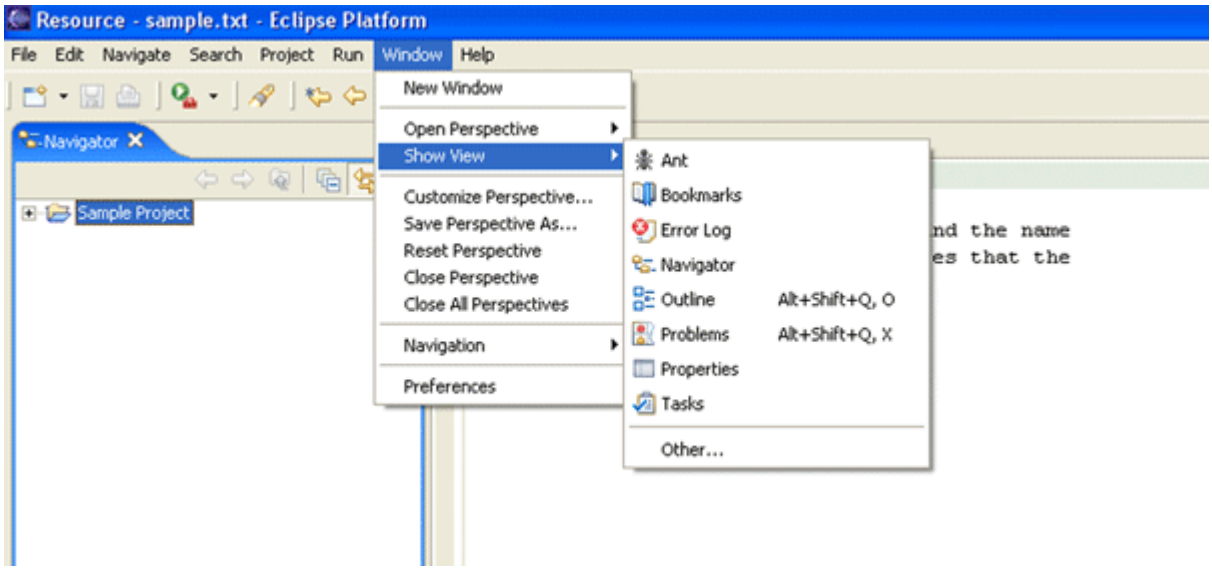


圖 2.6

2.6.2 編輯器(Editor)

編輯器是很特殊的視窗，會出現在工作台的中央。當打開文件、程式碼或其他資源時，Eclipse會選擇最適當的編輯器打開文件。若是純文字檔，Eclipse就用內建的文字編輯器打開(例如 圖 2.7)；若是Java程式碼，就用JDT的Java編輯器打開(例如 圖 2.8)；若是Word文件，就用Word打開(例如 圖 2.9)。此Word視窗會利用Object Linking and Embedding-OLE，內嵌在Eclipse中。

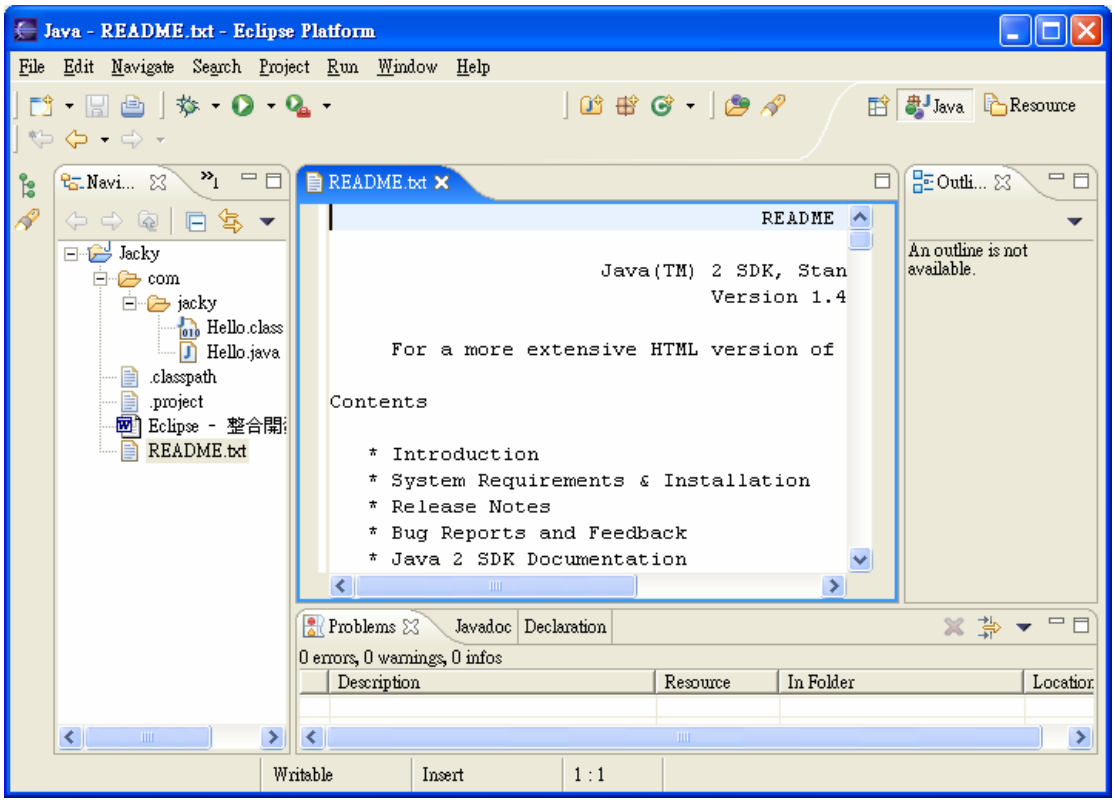

Java - README.txt - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Java Resource
Navi...
README.txt
README
Java(TM) 2 SDK, Stan
Version 1.4
For a more extensive HTML version of
Contents
* Introduction
* System Requirements & Installation
* Release Notes
* Bug Reports and Feedback
* Java 2 SDK Documentation
Outli...
An outline is not available.
Jacky
com
jacky
Hello.class
Hello.java
.classpath
.project
Eclipse - 整合開
README.txt
Problems Javadoc Declaration
0 errors, 0 warnings, 0 infos
Description Resource In Folder Location
Writable Insert 1 : 1

圖 2.7

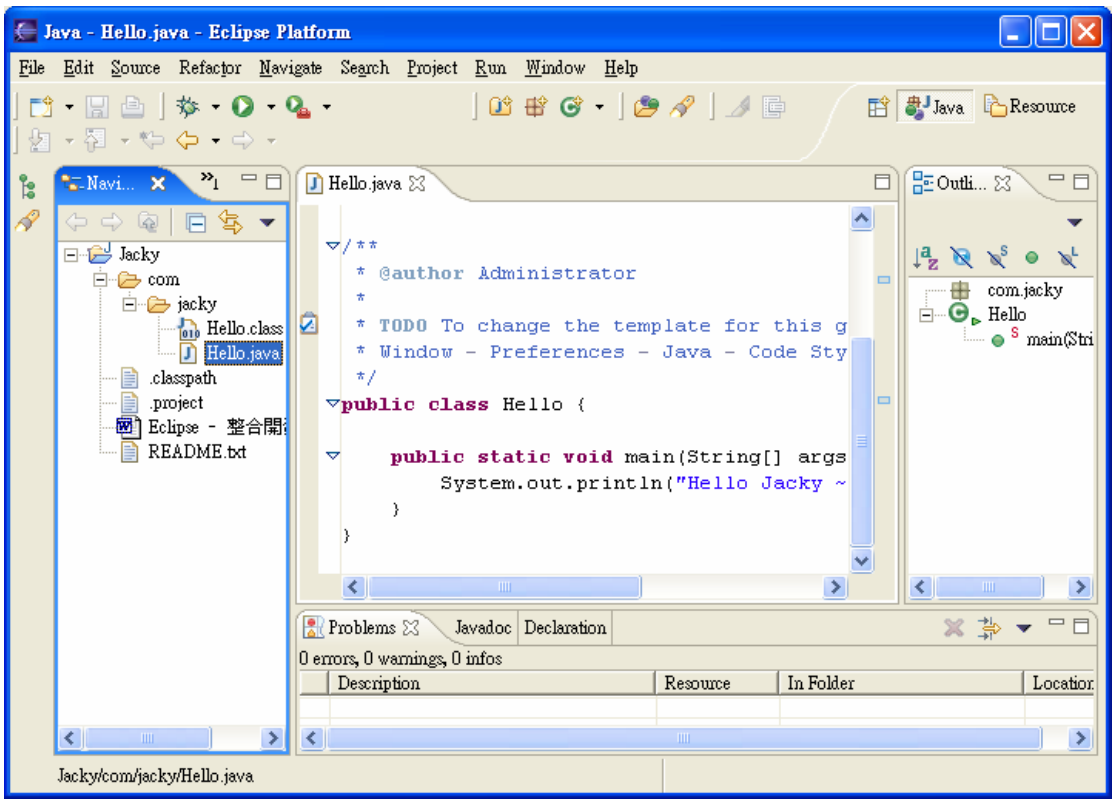

Java - Hello.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java Resource
Navi...
Hello.java
/**
* @author Administrator
*
* TODO To change the template for this g
* Window - Preferences - Java - Code Sty
*/
public class Hello {
public static void main(String[] args
System.out.println("Hello Jacky ~
}
}
Outli...
com.jacky
Hello
main(Stri
Jacky
com
jacky
Hello.class
Hello.java
.classpath
.project
Eclipse - 整合開
README.txt
Problems Javadoc Declaration
0 errors, 0 warnings, 0 infos
Description Resource In Folder Location
Jacky/com/jacky/Hello.java

圖 2.8

在 Windows 中，工作台會試圖啟動現有的編輯器，如 OLE(Object Linking and Embedding)文件編輯器。比方說，如果機器中安裝了 Microsoft Word，編輯 DOC 檔案會直接在工作台內開啟 Microsoft Word(例如圖 2.9)。如果沒有安裝 Microsoft Word，就會開啟 Word Pad。

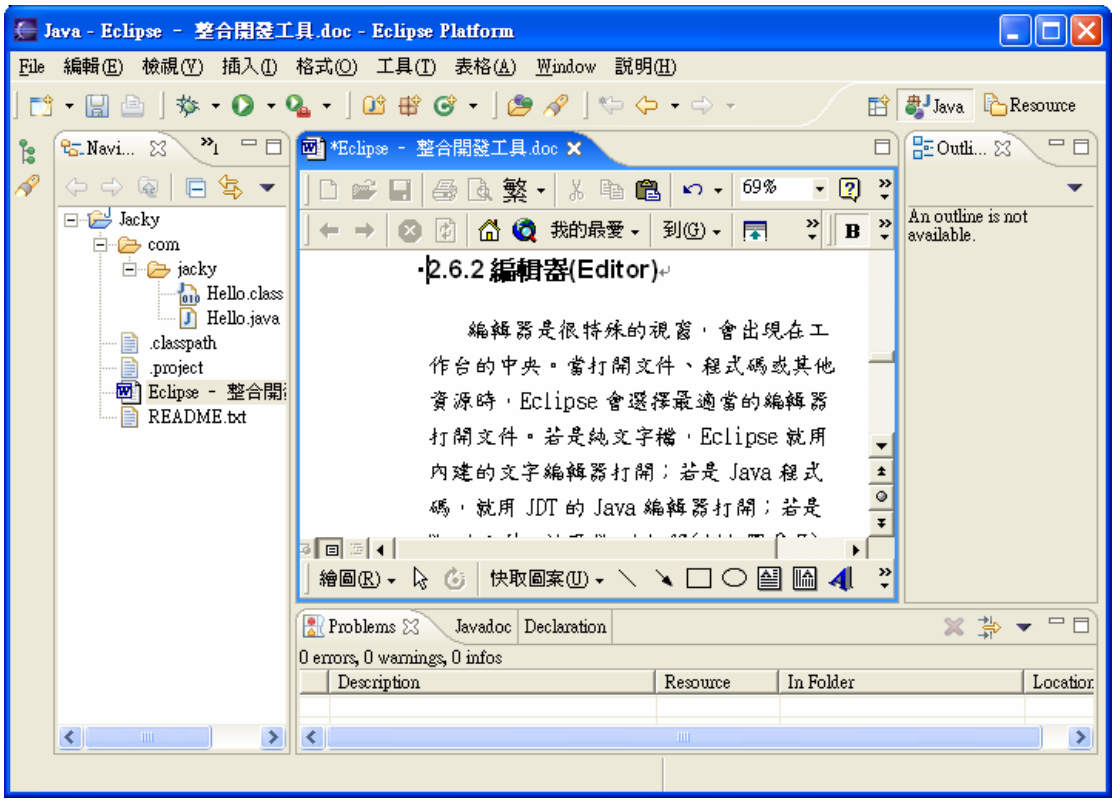


圖 2.9

如果標籤左側出現星號 (*)(例如 圖 2.9)，就表示編輯器有未儲存的變更。如果試圖關閉編輯器或結束工作台，但沒有儲存變更，就會出現儲存編輯器變更的提示。

工具列中的向後和向前箭頭按鈕，或利用 Ctrl+F6 加速鍵來切換編輯器。箭頭按鈕會移動通過先前的滑鼠選取點，可以先通過檔案中的多個點，之後才移到另一個點。 Ctrl+F6 會蹦現目前所選取的編輯

器清單，依預設，會選取在現行編輯器之前所用的編輯器。(在 Macinosh 中，加速鍵是 Command+F6。)

2.6.3 視景(Perspective)

Eclipse 提供數群育先選定的視圖，並已事先定義好的方式排列，稱之為視景(perspective)。所有視景的主要元件式編輯器。

每個視景的目的是執行某特定的工作，如編寫Java程式，在每個視圖以各種不同的觀點處理工作，例如 圖 2.10。

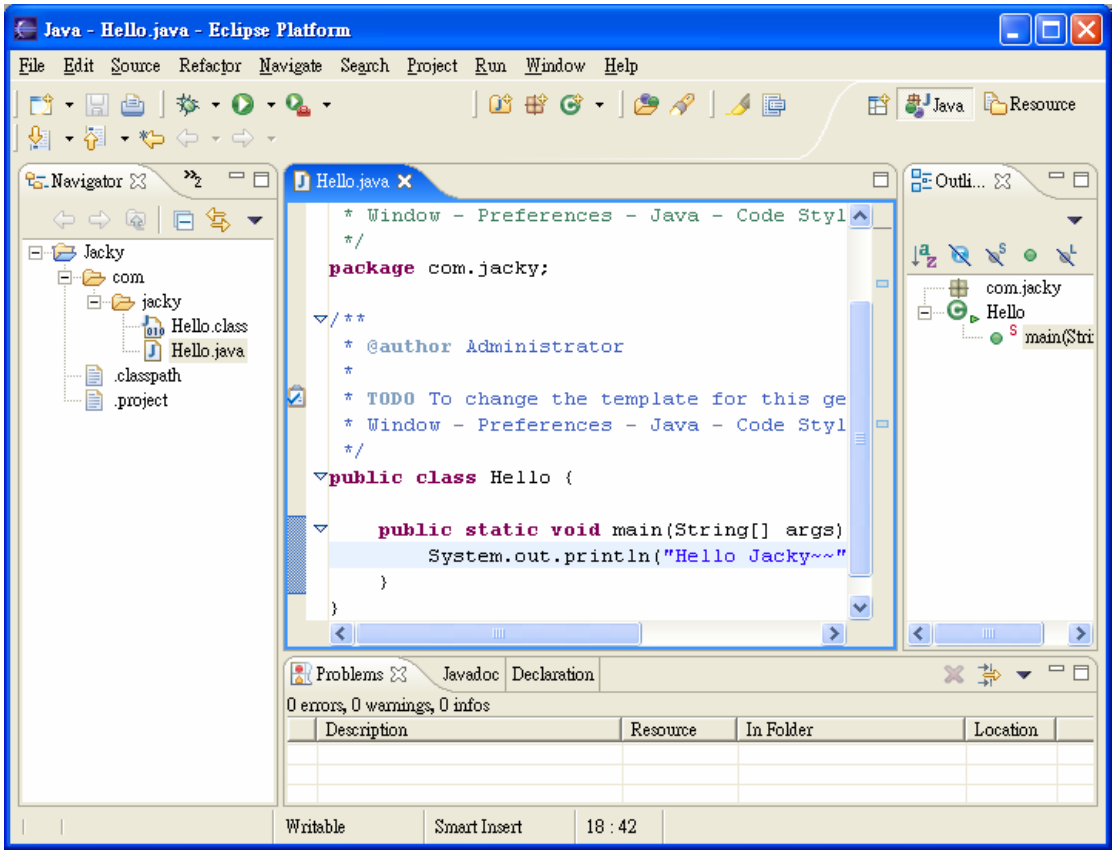


圖 2.10

若在Debug的視景中，其中一個視圖會顯示程式碼，另一個可能換顯示變數目前的值，還有一個可能會顯示程式的執行結果。例如 圖 2.11。

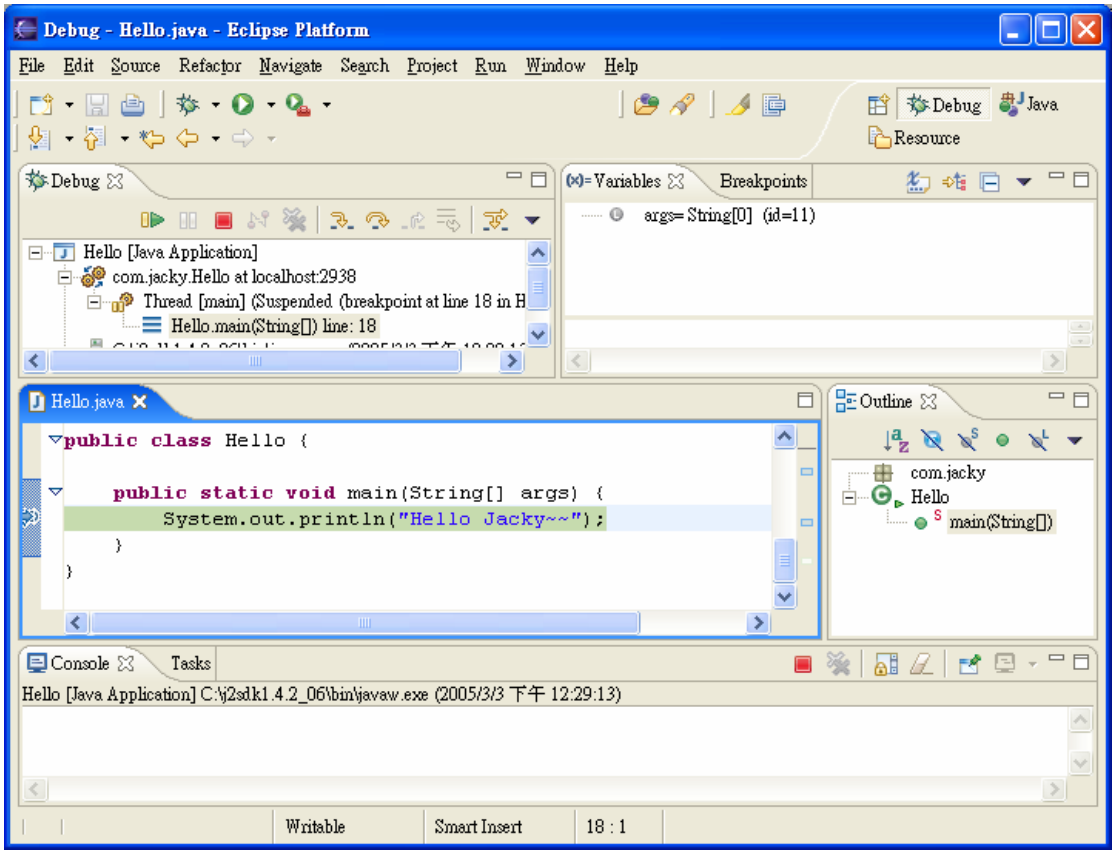


圖 2.11

2.7 重新排列視圖和編輯器

2.7.1 放置游標

放置游標表示視圖可以定置在工作台視窗的哪裡。當重新排列視圖時，可能會出現幾種不同的放置游標。

圖示	說明
	定置上方：如果在顯示定置上方游標時放開滑鼠按鈕，視圖會放在游標所在視圖的上面。
	定置下方：如果在顯示定置下方游標時放開滑鼠按鈕，視圖會放在游標所在視圖的下面。
	定置右側：如果在顯示定置右側游標時放開滑鼠按鈕，視圖會放在游標所在視圖的右側。

圖示	說明
	定置左側：如果在顯示定置左側游標時放開滑鼠按鈕，視圖會放在游標所在視圖的左側。
	堆疊：如果在顯示堆疊游標時放開滑鼠按鈕，視圖會變成與游標下面的視圖同一個窗格中的標籤。
	限制：如果在顯示限制游標時放開滑鼠按鈕，視圖不會定置在這個位置。比方說，視圖不能定置在編輯區。

2.7.2 重新排列視圖

可以變更「Navigator」視圖在工作台視窗中的位置。

I. 按一下「Navigator」視圖的標題列，並且拖曳視圖以橫跨工作台視窗。目前還不要放開滑鼠按鈕。

II. 當仍在工作台視窗的頂端拖曳視圖時，請注意，各種放置游標時會出現。這些放置游標（請參閱 上一節）表示當放開滑鼠按鈕時，視圖會關聯於游標所在的視圖或編輯區而定置在哪裡。請注意，這時會繪製用來強調顯示的矩形，以提供視圖將定置在哪裡的其他回饋。

III. 將視圖定置在工作台視窗中的任何位置，再檢視這個動作的結果。

IV. 按一下並且拖曳視圖的標題列，將視圖重新定置在工作台視窗中的其他位置。請觀察這個動作的結果。

V. 最後，將「Navigator」視圖拖曳到「Outline」視圖上面。這時會顯示一個堆疊游標。如果放開滑鼠按鈕，「Navigator」就會和「Outline」視圖一起堆放到附加標籤的筆記本中。

2.7.3 並列編輯器

工作台可以在編輯區中建立兩組或更多組編輯器。也可以調整編輯區的大小，但不能將視圖拖曳到編輯區。

I. 在「Navigator」視圖中按兩下可編輯的檔案，以在編輯器區中開啟至少兩個編輯器。

II. 按一個編輯器的標示，將它拖曳到編輯器區域之外。不要放開滑鼠按鈕。

III. 請注意，如果試圖將編輯器放到任何視圖的頂端，或放在工作台視窗之外，就會出現限制游標。

IV. 仍按住滑鼠按鈕，將編輯器拖曳到編輯器區，沿著編輯器區的四邊移動游標，以及在編輯器區中央另一開啟的編輯器上移動游標。請注意，沿著編輯器區域的邊緣會出現有方向箭頭的放置游標，編輯器區域中央會出現堆疊放置游標。

V. 將編輯器定置在有方向箭頭的放置游標上，使兩個編輯器都出現在編輯器區域中。

VI. 請注意，必要時，也可以調整各編輯器和整個編輯區的大小來容納編輯器和視圖。

VII. 請務必觀察編輯器標籤的顏色（下圖中有兩個群組，一個群組在另一群組的上面）

藍色 - 表示編輯器目前在作用中。

預設值（在 Windows XP 中呈灰色）- 表示編輯器是前次作用中的編輯器。如果有作用中的視圖，它就是作用中視圖目前在使用的編輯器。當使用會密切搭配編輯器的「Outline」和「內容」這類視圖時，這一點非常重要。

VIII. 拖曳編輯器，將它定置在編輯器區的其他位置，請注意定置各種放置游標時所產生的行為。請繼續嘗試定置編輯器和視圖及

調整其大小，直到工作台的安排符合要求為止。圖 2.12 說明將一個編輯器拖放到另一編輯器之下的佈置。

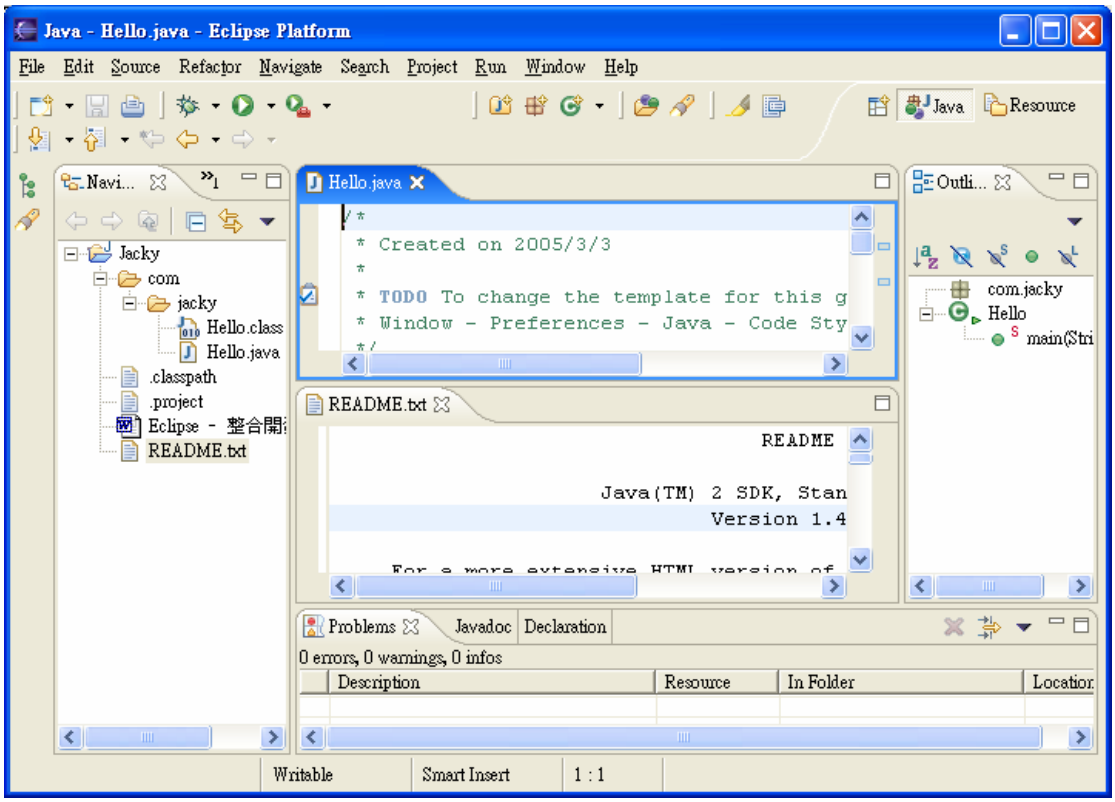


圖 2.12

2.7.4 重新排列附加標籤的視圖

除了在工作台中拖放視圖之外，也可以在附加標籤的筆記本內重新排列視圖的次序。

I. 選擇「Window」→「Reset Perspective」，將「Resource」視景重設回程式佈置。

II. 按一下「Outline」標題列，然後在「Navigator」視圖頂端加以拖曳。現在「Outline」將會堆疊在「Navigator」的頂端。

II. 按一下「Navigator」標籤，將它拖曳到「Outline」標籤的右側。

IV. 游標到了「Outline」標籤右側且變成堆疊游標之後，放開滑鼠按鈕。

請觀察「Navigator」標籤，它現在已在「Outline」標籤的右側。

2.7.5 最大化

能夠將視圖或編輯器最大化，有時非常有用。將視圖和編輯器兩者最大化很容易。

- 如果要將視圖最大化，請按兩下它的標籤，或從標籤的蹦現功能表中選取「Maximize」。
- 如果要將編輯器最小化，請按兩下編輯器標籤，或從標籤的蹦現功能表中選取「Minimize」。

將視圖還原至程式大小的方法也類似（按兩下或從功能表中選擇「Restore」）。

2.8 功能表和工具列

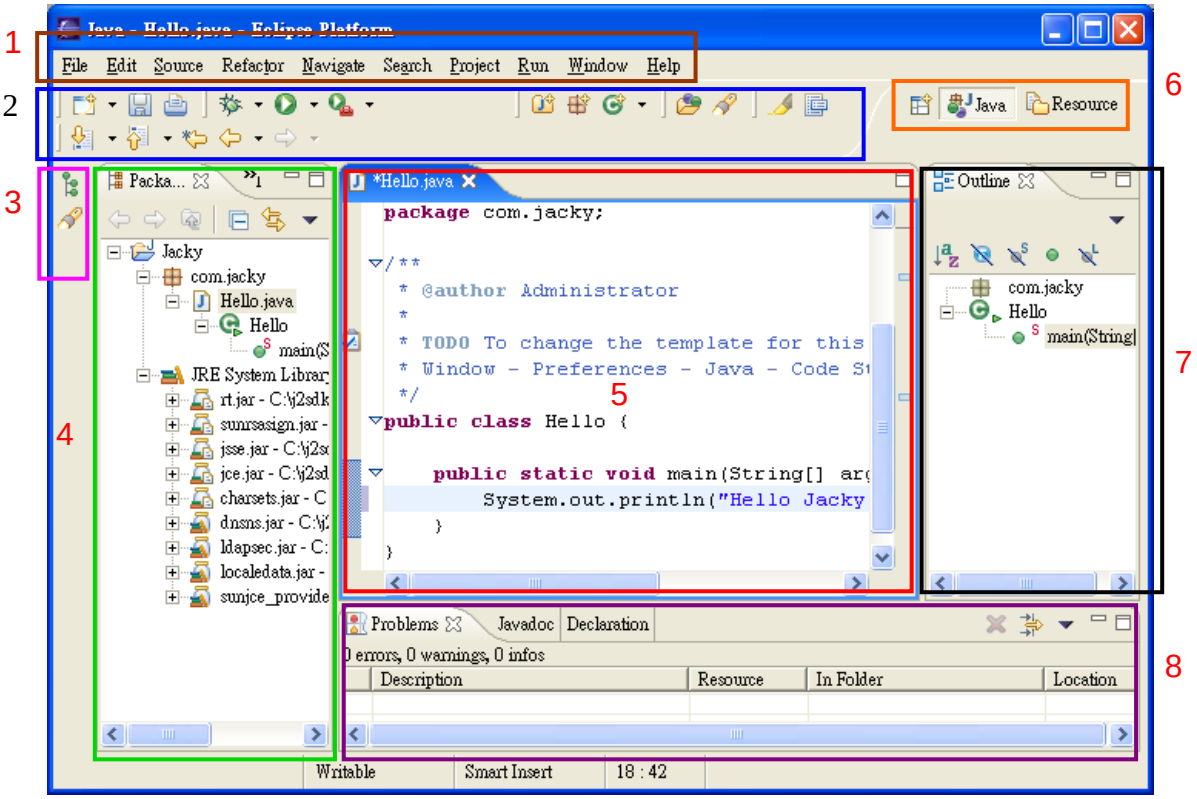


圖 2.13

1. 功能表(Menu Bar)
2. 工具列(Tool Bar)
3. 快速視圖(Fast View)
4. Package Explorer 視圖
5. Editor 視圖
6. 捷徑工具列(Shortcut Toolbar)
7. Outline 視圖
8. Tasks 視圖和 Console 視圖

2.8.1 功能表

「File」功能表

這個功能表可以建立、儲存、關閉、列印、匯入及匯出工作台資源以及結束工作台本身。

名稱	功能
New(新建)	建立 Java 元素或新資源。配置哪些元素會顯示在「Window」→「Preferences」的子功能表中。在 Java 視景中，依預設，會提供專案、套件、類別、介面、來源資料夾、即時運算簿、檔案和資料夾的建立動作。
Close(關閉)	關閉現行編輯器。如果編輯器中有資料尚未儲存，則會顯示一個儲存要求對話框。
Close All(全部關閉)	關閉所有編輯器。如果編輯器中有資料尚未儲存，則會顯示一個儲存要求對話框。
Save(儲存)	儲存現行編輯器的內容。如果編輯器中沒有未儲存的變更，則會停用。
Save As(另存新檔)	以新名稱儲存現行編輯器中的內容。
Save All(全部儲存)	儲存所有編輯器內容以及未儲存的變更。如果沒有編輯器中有未儲存的變更，則會停用。
Revert(回復)	將現行編輯器的內容回復成已儲存檔案中的內容。如果編輯器中沒有未儲存的變更，則會停用。
Move(移動)	移動資源。如果是 Java 元素則會停用。如果要移動 Java 元素，請使用「Refactor」→「Move」(如此會更新檔案的所有參照)，或使用「Edit」→「Cut/Paste」(如此不會更新參照)。
Rename(重新命名)	將資源重新命名。如果是 Java 元素則會停用。如果要重新命名 Java 元素，請使用「Refactor」→「Rename」(如此會更新檔案的所有參照)。
Refresh(重新整理)	以本端檔案系統來重新整理所選元素的內容。如果不是從特定選項啟動，這個指令會重新整理所有專案。

Print(列印)	列印現行編輯器的內容。會在編輯器成為焦點時啟用。
Switch workspace(切換工作區)	這個指令可以切換至不同的工作區這會重新啟動工作台
Open external file(開啟外部檔案)	這個指令可以在文字編輯器中開啟不在工作區中的檔案
Import(匯入)	開啟匯入精靈對話框。JDT 不會提供任何匯入精靈。
Export(匯出)	開啟匯出精靈對話框。JDT 會提供 JAR 檔匯出精靈和 Javadoc 產生精靈。
Properties(內容)	開啟所選元素的「內容」頁面。依據 Java 專案開啟 Java 建置路徑頁面，且可使用 Javadoc 位置頁面。如果是 JAR 保存檔,請在這個配置 JAR 的程式檔附加與 Javadoc 位置。
Recent file list(最近使用的檔案清單	「File 底端維護了一份最近在工作台中存取的檔案的清單只要選取檔名,就可以從「File 開啟這其中的任何檔案。
Exit(結束)	結束 Eclipse

「Edit」功能表

這個功能表可協助操作編輯器區域中的資源

名稱	功能
Undo(復原)	回復成編輯器中的前一次變更
Redo(重做)	回復已取消的變更
Cut(剪下)	將目前所選取的文字或元素複製到剪貼簿中，並移除元素。就元素而言，在貼到剪貼簿前不會移除。
Copy(複製)	將目前所選取的文字或元素複製到剪貼簿中。
Paste (貼上)	將目前的內容當成文字貼到編輯器中，或當成同層級或下層元素，貼到目前所選的元素中。

Delete(刪除)	刪除目前的文字或元素選項。
Select All(全選)	選取所有的編輯器內容。
Find / Replace(尋找/取代)	開啟「尋找/取代」對話框。限編輯器。
Find Next(尋找下一個)	尋找目前所選文字下一個搜尋結果。限編輯器。
Find Previous(尋找上一個)	尋找目前所選文字上一個搜尋結果。限編輯器。
Incremental Find Next(增量尋找下一個)	啟動增量尋找模式。在呼叫後，請按照狀態列中的指示來輸入搜尋文字。限編輯器。
Incremental Find Previous(增量尋找上一個)	啟動增量尋找模式。在呼叫後，請按照狀態列中的指示來輸入搜尋文字。限編輯器。
Add Bookmark(新增書籤)	為目前的文字選項或所選取的元素新增書籤。
Add Task(新增作業)	為目前的文字選項或所選取的元素新增使用者定義的作業。
Expand Selection to(展開選項至)	■ 含括元素：選取程式碼中的含括表示式、區塊、方法。這個動作會注意 Java 語法。如果程式碼的語法有錯，可能無法運作正常。(上移鍵) ■ 下一個元素：選取現行與下一個元素。(右移鍵) ■ 上一個元素：選取現行與上一個元素 (左移鍵) ■ 還原前次的選擇：在呼叫*展開選項至*之後，還原先前的選項。(下移鍵)
Show Tooltip Description(顯示工具提示說明)	以浮動說明方式顯示出現在現行游標位置上的值。對話框可以捲動，因而不會縮短說明。
Content Assist(內容輔助)	在現行游標位置開啟一個內容輔助對話框，以顯示 Java 程式碼的輔助提議與範本。請參閱「範本」喜好設定頁面，以取得可用的範本(「Window」→「Preferences」→「Java」→「Editor」→「Templates」)，然後移至「編輯器」喜好設定頁面 (「Window」→「Preferences」→「Java」→

	「Editor」→「Code Assist」)，來配置程式碼輔助的行為。
Quick Fix(快速修正)	如果游標位於有出現問題指示之處，則這個動作會在現行游標處開啟一個內容輔助對話框，以提供可能的更正動作。
Parameter Hints(參數提示)	如果游標位於方法參照的參數規格處，這個動作會以浮動說明的方式顯示參數類型資訊。現行游標處的參數會以粗體字顯示。
Encoding(編碼)	切換目前所示文字內容的編碼。

「Source」功能表

名稱	功能
Toggle Comment(註解)	標註出內含現行選擇項的所有字行。
Add Block Comment(註解區塊)	標註出內含現行選擇項的區塊。
Remove Block Comment(解除註解區塊)	取消標註內含現行選擇項的區塊。
Shift Right(向右移位)	增加目前所選字行的內縮層次。只有在選擇項涵蓋多行或一整行時才會啟用。
Shift Left(向左移位)	減少目前所選字行的內縮層次。只有在選擇項涵蓋多行或一整行時才會啟用。
Format(格式)	可使用程式碼格式製作器，來設定目前文字選擇項的格式。格式設定選項是在「Code Formatter」喜好設定頁面(「Window」→「Preferences」→「Java」→Code Formatter)) 中配置
Format Element(格式成員)	格式化成員
Sort Members(排序成員)	「Window」→「Preferences」→「Java」→「Appearance」→「Members Sort Order」中指定的排序次序，來排序類型中的成員

Organize Imports(組織匯入)	組織目前開啟或所選編譯單元中的匯入宣告。會移除不必要的匯入宣告，且會按照「Organize Import」喜好設定頁面(「Window」→「Preferences」→「Java」→「Organize Import」) 中的指定，來排列必要的匯入宣告。「Organize Import」可執行於不完整的程式檔上，並且會在所參照的類型名稱無法唯一對映至現行專案中的某個類型時提示。也可以組織多個編譯單元，其做法是對某個套件呼叫動作，或選取一組編譯單元。
Add Import(新增匯入)	為目前所選的類型參照建立一項匯入宣告。如果類型參照完整，則會移除資格。如果所參照的類型名稱無法唯一對映至現行專案中的某個類型，將會提示指定正確的類型。「Add Import」會試著遵循「Organize Import」喜好設定頁面中指定的匯入順序。
Override/Implement Methods(置換/實作方法)	會開啟「Override Method」對話框，可以置換或實作現行類型中的方法。適用於類型或類型中的某個文字選擇項。
Generate Getter and Setter(產生 Getter 和 Setter)	開啟「Generate Getter and Setter」對話框，可以為現行類型中的欄位，建立 Getter 和 Setter。適用於欄位與類型或類型中的某個文字選擇項。
Generate Delegate Methods(產生委派方法)	開啟「Generate Delegate Methods」對話框，可以為現行類型中的欄位建立方法委派。可用在欄位。
Add Constructor from Superclass(新增 Super 類別中的建構子)	為目前所選的類型新增 Super 類別中所定義的建構子。適用於類型或類型中的某個文字選擇項。
Surround with try/catch(以 try/catch 包覆)	針對所選的陳述式，評估所有必須捕捉到的異常狀況。這些表示式會包覆 try catch 區塊。可以使用編輯功能表中的*展開選項至*，以取得有效的選項範圍。
Externalize Strings(將字串提出)	開啟「Externalize Strings」精靈。這個精靈可以藉由會存取內容檔的陳述式，來更換程式碼中的所有字串。
Find Strings to	會出現一個對話框，其中顯示未提出字串數目的摘要。適

Externalize(尋找要提出的字串)	用於專案、來源資料夾與套件。
Convert Line Delimiters To(將行定界字元轉換成)	在目前開啟的編輯器中，變更所有行定界字元，而採用下列作業系統中所用的行定界字元： ■ CRLF(Windows) ■ LF（Unix、MacOSX） ■ CR（傳統 MacOS） Java 編輯器容許混合使用行定界字元。不過，其他某些工具會要求使用和 OS 一致的行定界字元，或者要求至少行定界字元要一致。

「Refactor」功能表

重構指令也可以在一些視圖的快速功能表與 Java 編輯器中找到。

名稱	功能
Undo(復原)	「Undo」前次的重構作業。重構復原緩衝區，共在執行重構後程式檔未變更的狀況下有效。
Redo(重做)	重做前次復原的重構作業。重構復原/重做緩衝區的有效期，僅限於執行重構後到沒有其他程式檔變更的這段時間。
Rename(重新命名)	啟動「Rename Refactoring」對話框：重新命名所選的元素，並且（如果有啟用的話）更正元素的（以及其他檔案中的）所有參照。適用於方法、欄位、區域變數、方法參數、類型、編譯單元、套件、來源資料夾、專案，並且適用於可解析成這些元素類型之一的文字選項。
Move(移動)	啟動「Move」重構對話框：移動所選的元素，並（如果有啟用的話）更正元素的（以及其他檔案中的）所有參照。可套用至一或多個 Static 方法、Static 欄位、類型、編譯單元、套件、來源資料夾與專案，並且套用於可解析成這些元素類型之一的文字選擇項。
Change Method Signature(變更方法簽章)	啟動「Change Method Signature」重構對話框。變更參數名稱、參數類型、參數順序，並更新對應方法的所有參照。此外，可以移除或新增參數，也可以變更方法傳

	回類型及其可見性。這個重構作業可套用至方法或套用在解析成方法的文字選項。
Convert Anonymous Class to Nested(將匿名類別轉換成巢狀)	啟動「Convert Anonymous Class to Nested」重構對話框。協助將匿名內部類別轉換成成員類別。這個重構作業可套用至匿名內部類別。
Convert Nested Type to Top Level(將巢狀類型轉換成最上層)	啟動「Convert Nested Type to Top Level」重構對話框。為所選成員類型建立新的 Java 編譯單元，同時依需要更新所有參照。對於非 static 成員類型，將新增一個欄位，以容許存取先前含括的實例。這個重構作業可套用至成員類型或解析成成員類型的文字。
Push Down(下推)	啟動「Push Down」重構對話框。將類別中的一組方法和欄位移至它的子類別。這個重構作業可套用至一個或多個以相同類型宣告的方法和欄位，或套用在欄位或方法內的文字選項。
Pull Up(上拉)	啟動「Pull Up」重構精靈。將欄位或方法移至其宣告類別的 Super 類別，或（如果是方法）將方法宣告成 Super 類別中的 abstract。這個重構作業可套用至一個或多個以相同類型宣告的方法、欄位和成員類型，或套用在欄位、方法或成員類型內的文字選項。
Extract Interface(擷取介面)	啟動「Extract Interface」重構對話框。以一組方法建立新的介面，並使所選類別實作介面，同時選擇性將類別參照變更為新介面（在可能的情況下）。這個重構作業可套用至類型。
Use Supertype Where Possible(適當時使用 Super 類型)	啟動「Use Supertype Where Possible」重構對話框。在識別所有可能發生這個取代的位置後，將出現的類型換成其 Super 類型之一。這個重構作業可用在類型之上。
Inline(列入)	啟動「Inline」重構對話框。列入區域變數、方法或常數。這個重構作業可用在方法、static final 欄位，以及解析為方法、static final 欄位或區域變數的文字選項。
Extract Method(擷取方法)	啟動「Extract Method」重構對話框。會建立一個內含目前所選之陳述式或表示式的新方法，並將選擇項換成新方法的參照。可以使用編輯功能表中的*展開選項至*，

	以取得有效的選項範圍。 這項特性非常適合用來清除冗長、雜亂和太複雜的方法。
Extract Local Variable(擷取區域變數)	啟動「Extract Local Variable」重構對話框。會建立一個新變數，以指定給目前所選的表示式，並將選擇項換成新變數的參照。這個重構作業可用在解析為區域變數的文字選項。可以使用編輯功能表中的*展開選項至*，以取得有效的選項範圍。
Extract Constant(擷取常數)	啟動「Extract Constant」重構對話框。從所選表示式中建立 static final 欄位並替代欄位參照，以及選擇性地重新寫入其他出現相同表示式的位置。這個重構作業可用在 static final 欄位，以及解析為 static final 欄位的文字選項。
Convert Local Variable to Field(將區域變數轉換成欄位)	啟動「Convert Local Variable to Field」重構對話框。將區域變數轉換成欄位。如果在建立時已起始設定變數，則作業會將起始設定移至新欄位的宣告，或移至類別的建構子。這個重構作業可用在解析為區域變數的文字選項。
Encapsulate Field(封裝欄位)	啟動「Encapsulate Field」重構對話框。會將欄位的所有參照換成 getting 與 setting 方法。適用於所選的欄位或可解析成欄位的文字選擇項。

「Navigate」功能表

這個功能表可以尋找及導覽工作台中顯示的資源及其他成品。

名稱	功能
Go Into(進入)	將視圖輸入設定在目前所選的元素上。「套件瀏覽器」視圖可支援這項。
Go To(移至)	■ 上一頁：將視圖輸入設定在歷程中的上一個輸入上:必須有歷程,才能用到這項(已使用「Go Into」) ■ 下一頁：將視圖輸入設定在歷程中的下一個輸入上：必須有歷程，才能用到這項（已使用「Go Into」、「Go Into」→「Back」) ■ 往上移一層:將現行視圖的輸入設定在其輸入的母

	元素上。 ■ 參照測試:瀏覽以找出所有參照目前選取之類型的 JUnit 測試 ■ 類型：瀏覽以找出類型，並在現行視圖中顯示它。「Package Explorer」視圖支援這項。 ■ 套件：瀏覽以找出套件，並在現行視圖中顯示它。「Package Explorer」視圖支援這項。 ■ 資源：瀏覽以找出資源，並在現行視圖中顯示它。
Open(開啟)	試著解析現行程式碼選項上所參照的元素，並開啟宣告該參照的檔案。
Open Type Hierarchy(開啟類型階層)	試著解析現行程式碼選項上所參照的元素,並在「Type Hierarchy」視圖中開啟該元素。針對元素呼叫，並開啟元素的類型階層。顯示 Java 元素的 Java 編輯器與視圖中可支援這項。
Open Call Hierarchy(開啟呼叫階層)	試著開啟呼叫現行程式碼選項上所參照的元素，並在「Call Hierarchy」視圖中開啟該元素。
Open Super Implementation(開啟 super 實作)	開啟一個編輯器，以顯示目前所選方法或現行游標位置旁之方法的 super 實作。如果未選取方法，或者方法沒有 super 實作，則不會開啟編輯器。
Open External Javadoc(開啟外部 Javadoc)	開啟目前所選元素或文字選項的 Javadoc 文件。JAR 或專案的 Javadoc 位置是在專案或 JAR 的「Javadoc Location」內容頁面中指定。請注意,這個外部 Javadoc 文件可能未以現行程式碼中指定的 Javadoc 加以更新。可以使用 Javadoc 匯出精靈，在 Java 專案中為程式檔建立 Javadoc 文件。
Open Type(開啟類型)	顯示「Open Type」選擇對話框，以便在編輯器中開啟一個類型。「開啟類型」選擇對話框中顯示工作區中的所有現有類型。
Open Type In Hierarchy(在「階層」中開啟類型)	顯示「Open Type」選擇對話框，以便在編輯器與「Type Hierarchy」視圖中開啟一個類型。「Open Type」選擇對話框中顯示工作區中的所有現有類型。
Show in → Package Explorer(顯示在→套件	在「Package Explorer」視圖中顯示目前所選的元素（或現行游標位置旁的元素）。

瀏覽器)	
Quick Outline(顯示概要)	為目前選取的類型開啟小型概要器。
Quick Type Hierarchy(顯示類型階層)	為目前選取的類型開啟小型類型階層器。
Next Annotation (移至下一個問題)	選取下一個問題。Java 編輯器中支援這項。
Previous Annotation (移至上一個問題)	選取上一個問題。Java 編輯器中支援這項。
Go to Last Edit Location(移至前次編輯位置)	顯示前次發生編輯的位置。
Go to Line(移至指定行號)	開啟對話框，以輸入指示編輯器應移至的行號。限編輯器。
Back(向後)	這個指令會導覽至之前在編輯器中檢視的前一個資源。這個指令和 Web 瀏覽器的上一頁按鈕相同。
Forward(向前	這個指令會導覽並復原之前的上一頁指令所造成的效果。這個指令和 Web 瀏覽器的下一頁按鈕相同。

「Search」功能表

名稱	功能
Search...(搜尋...)	開啟搜尋對話框
File...(檔案...)	針對「檔案搜尋」頁面開啟搜尋對話框
Java...(Java...)	針對「Java 搜尋」頁面開啟搜尋對話框
References(參照)	尋找所選 Java 元素的所有參照
Declarations(宣告)	尋找所選 Java 元素的所有宣告
Implementors(實作者)	尋找所選介面的所有實作者。
Read Access(讀取權)	尋找所選欄位的所有讀取權
Write Access(寫入權)	尋找所選欄位的所有寫入權
Referring Tests...()	尋找所選 Java 元素的所有測試參照

Occurrences in File(檔案中的搜尋結果)	尋找所選 Java 元素在其檔案中的所有出現項目
Exception Occurrences(拋出例外中的搜尋結果)	尋找所選 Java 元素在其拋出例外中的所有出現項目

Search Scopes Submenu(搜尋範圍子功能表)：

範圍	可用性	說明
Workspace(工作區)	所有元素	在整個工作區中搜尋
Project(專案)	所有元素	在含括所選元素的專案中進行搜尋
Hierarchy(階層)	類型和成員	在類型的階層中搜尋
Workings Set(工作集)	所有元素	在工作集中搜尋

工作集對話框可以儲存並命名範圍。「搜尋範圍」子功能表中亦會顯示工作集的現有實例。

可在下列視圖中透過所選資源與元素的快速功能表，來執行 Java 搜尋：

- 「Package Explorer」
- 「Outline」視圖
- 「Search Result」視圖
- 「Hierarchy」視圖
- 「Browsing」視圖

Java 編輯器中亦提供「Search」快速功能表。目前所選文字必須可解析成 Java 元素，才能執行搜尋。

所選 Java 元素的類型會定義所能使用的「Search」快速功能表。Java 編輯器不會根據選項而限制可用的 Java 搜尋項清單。

「Project」功能表

「專案」功能表可以對工作台中的專案執行動作（建置或編譯）。

名稱	功能

Open Project(開啟專案)	顯示對話框，可以選取開啟已關閉的專案
Close Project(關閉專案)	關閉目前所選取的專案
Build All(全部建置)	這個指令會對工作台中的所有專案執行增量(incremental)建置。也就是說，它會建置（編譯）自從前次增量建置後,工作台中受到任何資源變更所影響的所有資源。自動建置關閉時，才可使用這個指令。
Build Project(建置專案)	這個指令會對目前選取的專案執行增量(incremental)建置。也就是說，它會建置（編譯）自從前次建置後，受到任何資源變更所影響的專案中的所有資源。自動建置關閉時，才可使用這個指令。
Build Workings Set(重新建置工作集)	這個功能表可以在工作集上執行增量(incremental)建置。也就是說，它會建置（編譯）前次建置之後，受到任何資源變更所影響之工作集中的所有資源。自動建置關閉時，才可使用這個指令。
Clean(清除)	這個指令會捨棄先前的所有建置結果。如果自動建置是開啟的，這會呼叫完整的建置。
Build Automatically(自動建置)	自動建置工作區中的所有專案。這個指令可以切換自動建置喜好設定。
Generate Javadoc...(產生 Javadoc...)	對目前選取的專案開啟「Generate Javadoc」精靈。
Properties(內容)	對目前選取的專案開啟內容頁面。

「Run」功能表

名稱	功能
Toggle Line Breakpoint(切換行岔斷點)	這個指令可以在目前於作用中 Java 編輯器中所選之行處，新增或移除 Java 行岔斷點。
Toggle Method	這個指令可以針對目前的二進位方法,新增或移除方法

名稱	功能
Breakpoint(切換方法岔斷點)	岔斷點。可在 Java 類別檔編輯器的來源中選取二進位方法，或在其他任何視圖中選取（像是「Outline」視圖）。
Toggle Watchpoint(切換監視點)	這個指令可以針對目前的 Java 欄位，新增或移除欄位監視點。可在 Java 編輯器的來源中選取欄位，或在其他任何視圖中選取（像是「Outline」視圖）。
Skip All Breakpoints(忽略所有的岔斷點)	這個指令可以忽略所有的岔斷點
Add Java Exception Breakpoint(新增 Java 異常狀況岔斷點)	這個指令可以建立一個異常狀況岔斷點。可藉由指定異常狀況岔斷點，而在擲出異常狀況時，暫停執行緒或 VM 的執行。可設為在未捕捉到或捕捉到（或兩者）異常狀況時暫停執行。
Add Class Load Breakpoint	這個指令可讓以建立一個 Class Load 岔斷點。
Run Last Launched(執行前一次的啟動作業)	這個指令可以在執行模式下迅速重複最近一次的啟動作業（如果有支援該模式的話）。
Debug Last Launched(除錯前一次的啟動作業)	這個指令可以在除錯模式下迅速重複最近一次的啟動作業（如果有支援該模式的話）。
Run History(執行歷程)	呈現在執行模式下啟動的啟動配置之最近歷程的子功能表
Run As(執行為)	呈現所登錄之執行啟動捷徑的子功能表。啟動捷徑可支援工作台或作用中編輯器選項的感應式啟動。
Run...(執行...)	這個指令會瞭解啟動配置對話框，以管理執行模式下的啟動配置。
Debug History(除錯歷程)	呈現在除錯模式下啟動的啟動配置之最近歷程的子功能表
Debug As(除錯為)	呈現所登錄之除錯啟動捷徑的子功能表。啟動捷徑可支援工作台或作用中編輯器選項的感應式啟動。
Debug...(除錯...)	這個指令會瞭解啟動配置對話框，以管理除錯模式下的

名稱	功能
	啟動配置。
Inspect(視察)	當執行緒暫停時，這個指令會使用「表示式」視圖，顯示在該執行緒之堆疊框或變數的環境定義下，視察所選表示式或變數的結果。
Display(顯示)	當執行緒暫停時，這個指令會使用「Display」視圖，顯示在該執行緒之堆疊框或變數的環境定義下，評估所選表示式的結果。如果目前作用中的部分是「Java Snippet Editor(Java 片段編輯器)」，則會在其中顯示結果。
Execute(執行)	執行
Step into Selection	這些指令可以逐步執行所要除錯的程式碼。
Externakl Tools(外部工具)	外部工具

「Windows」功能表

這個功能表可以顯示、隱藏，以及另行在工作台中操作各種視圖、視景和動作。

名稱	功能
New Window(開新視窗)	這個指令會開啟一個新的工作台視窗，其中含有與現行視景相同的視景。
Open Perspective(開啟視景)	這個指令會在此工作台視窗中開啟新視景。可以在「Window」→「Preferences」→「Workbench」→「Perspectives」頁面中變更這個喜好設定。在工作台視窗內開啟的所有視景都會顯示在捷徑列上。
Show View(顯示視圖)	這個指令會在現行視景中顯示選取的視圖。可以在「Window」→「Preferences」→「Workbench」→「Perspectives」頁面中配置開啟視圖的方式。可能會想開啟的視圖會最先列示；這份清單視現行視景而定。從其他... 子功能表中，可以開啟任何視圖。視圖會依照「Show View」對話框中的各個種類來排序。

名稱	功能
Customize Perspective(自訂視景)	每個視景包含一組預先定義的動作,可以從功能表列和工作台工具列存取這些動作。
Save Perspective As(另存新視景)	這個指令可以儲存現行視景，以及建立自己的自訂視景。儲存視景之後，可以使用「Window」→「Show View」→「Other...」功能表項目來開啟更多這類型的視景。
Reset Perspective(重設視景)	這個指令會將現行視景的佈置變更為其原始的配置。
Close Perspective(關閉視景)	這個指令會關閉作用中的視景。
Close All Perspectives(關閉所有視景)	這個指令會關閉工作台視窗中的所有已開啟視景。
Navigation(導覽)	這個子功能表包含用於在工作台視窗中的視圖、視景及編輯器之間導覽的按鍵。 ■ 顯示系統功能表：顯示用來重新調整大小、關閉或固定現行視圖或編輯器的功能表。 ■ 顯示視圖功能表：顯示可在作用中視圖的工具列中存取的下拉功能表。 ■ 將作用中的視圖或編輯器最大化：使作用中的部分占用整個畫面，如果已占用整個畫面，就使它返回原始狀態。 ■ 啟動編輯器：使現行編輯器作用中。 ■ 下一個編輯器：啟動最近使用的編輯器清單中的下一個開啟的編輯器。 ■ 上一個編輯器：啟動最近使用的編輯器清單中的上一個開啟的編輯器。 ■ 切換至編輯器：顯示一個對話框，用來切換到已開啟的編輯器。顯示一個對話框，用來切換到已開啟的編輯器。 ■ 下一個視圖：啟動最近使用的視圖清單中的下一個開啟的視圖。

名稱	功能
	■ 上一個視圖：啟動最近使用的編輯器清單中的上一個開啟的編輯器。 ■ 下一個視景：啟動最近使用的視景清單中的下一個開啟的視景。 ■ 上一個視景：啟動最近使用的視景清單中的上一個開啟的視景。
Preferences(喜好設定)	這個指令可以指出在使用工作台時的喜好設定。其中有各式各樣的喜好設定可用來配置工作台及其視圖的外觀，以及用來自訂在工作台中安裝的所有工具的行為。

「Help」功能表

這個指令提供有關使用工作台的說明。

名稱	功能
Welcome(歡迎使用)	這個指令會開啟歡迎使用內容。
Help Contents(說明內容)	這個指令顯示說明視圖。說明視圖含有說明書籍、主題，以及與工作台和已安裝特性的相關資訊。
Tips and Tricks(要訣和技巧)	這個指令會開啟可能尚未探索之有興趣的生產力特性的清單。
Cheat Sheets(提要)	這個指令會開啟選取提要的對話框。
Software Updates(軟體更新)	這個指令群組可以更新產品以及下載及安裝新特性。
About Eclipse Platform(關於Eclipse平台)	這個指令顯示產品、已安裝特性及可用外掛程式的相關資訊。

2.8.2 圖示和按鈕

「Navigator」視圖圖示

「Navigator」視圖中可能會出現下列圖示：

圖示	說明
	專案（開啟）
	資料夾（開啟）
	專案（已關閉）
	一般檔

編輯區標記列

標記列（編輯區左側）中可能會出現下列標記：

圖示	說明
	書籤
	岔斷點
	作業標記
	搜尋結果
	錯誤標記
	警告標記
	資訊標記

「Tasks」視圖

「Tasks」視圖可能會出現下列標記：

圖示	說明
	資訊作業
	高優先順序作業
	低優先順序作業
	已完成作業
	警告作業

圖示	說明
	錯誤作業

工具列按鈕

下列按鈕可能會出現在工作台工具列、視圖的工具列以及捷徑列中：

圖示	說明	圖示	說明
	開啟新視景		儲存作用中的編輯器內容
	儲存所有編輯器的內容		以新的名稱或位置儲存編輯器內容
	開啟搜尋對話框		列印編輯器內容
	開啟資源建立精靈		開啟檔案建立精靈
	開啟資料夾建立精靈		開啟專案建立精靈
	開啟「匯入」精靈		開啟「匯出」精靈
	執行增量建置		執行程式
	除錯程式		執行外部工具或 Ant
	剪下選擇至剪貼簿		複製選擇至剪貼簿
	從剪貼簿貼上選擇		復原最近的編輯
	重做最近的復原編輯		導覽至清單中的下一個項目
	導覽至清單中的上一個項目		向前導覽
	向後導覽		導覽上一層
	新增書籤或作業		開啟視圖的下拉功能表
	關閉視圖或編輯器		固定編輯器以防止自動重複使用
	過濾作業或內容		移至編輯器中的作業、問題或書籤
	還原預設內容		以樹狀結構顯示項目
	重新整理視圖內容		按字母順序排序清單
	取消執行過久的作業		刪除選取的項目或內容

外部工具和 Ant 圖示

物件

圖示	說明

	Ant 建置檔
	包含錯誤的 Ant 目標
	無效的專案建置器
	預設目標
	公用 Ant 目標（含說明的目標）
	Ant 內部目標（不含說明的目標）
	Jar 檔
	Ant 內容
	Ant 作業
	Ant 類型
	Ant 匯入作業
	Ant macrodef 作業

啟動配置

圖示	說明
	啟動外部工具
	Ant 啟動配置
	程式啟動配置
	「主要」標籤
	「重新整理」標籤
	「建置」標籤
	「目標」標籤
	「內容」欄標
	「類別路徑」標籤

Ant 視圖

圖示	說明
	Ant 視圖
	新增建置檔

	透過搜尋來新增建置檔
	執行選取的建置檔或選取的目標檔
	移除選取的建置檔
	移除所有的建置檔
	內容

除錯視圖

指令	名稱	說明
	回復	這個指令會讓已暫停的執行緒恢復執行。
	暫停	這個指令會暫停執行目標中所選取的執行緒，可以瀏覽或修改程式碼、視察資料、逐步執行等。
	終止	這個指令會終止所選取的除錯目標。
 (僅快速功能表)	終止並移除	這個指令會終止所選取的除錯目標，並將之從視圖中移除。
 (僅快速功能表)	全部終止	這個指令會終止視圖中所有作用中的啟動作業。
	切斷連線	這個指令會切斷除錯器和所選取的除錯目標間的連線(如果是遠端除錯的話)。
	移除全部終止的啟動	這個指令會將所有已終止的除錯目標從視圖顯示中清除。
	使用逐行過濾器	這個指令會切換逐行過濾器（開/關）。當它開啟時，所有的逐行功能都會套用逐行過濾器。
	進入副程序	這個指令會進入強調顯示的陳述式。
	跳過副程序	這個指令跳過強調顯示的陳述式。在下一行會以相同方法繼續執行或（如果位於方法結尾）使用呼叫現行方法的方法繼續執行。 游標會跳到方法的宣告處，並選取這一行。
	執行到 Return	這個指令會跳出現行方法的副程序。這個選項會在結束現行方法後停止執行。

	顯示完整名稱	這個選項可以切換成顯示或隱藏完整名稱。
 (僅快速功能表)	複製堆疊	這個指令會將已暫停執行緒中所選取的堆疊以及執行中之執行緒的狀態，複製到剪貼簿中。
	放到頁框	這個指令可以放回與重新輸入指定的堆疊框。這項特性類似「回頭執行」再整個重新啟動程式。 如果要放回堆疊框，再重新輸入指定的堆疊框，請選取要「放置」的指定堆疊框，再選取放入堆疊框。 請注意下列有關這項特性的警告： ■ 不能在堆疊中放入原生方法。 ■ 全體資料不受影響，仍維持其現行值。舉例來說，不會清除內含元素的 Static 向量。 ***附註：只有在基礎 VM 支援這項特性時，才會啟用這個指令。***
 (僅快速功能表)	重新啟動	這個指令會重新啟動所選除錯目標。
(僅快速功能表)	內容	這個指令會顯示所選取的啟動作業的內容。此外，也可以檢視所選程序的完整指令行。

2.9 視景

2.9.1 新視景

有幾種方法可在這個工作台視窗內開啟新視景：

- 利用捷徑列中的「Open Perspective」按鈕  。
- 從「Window」→「Open Perspective」功能表中選取一個視景。

如果要利用捷徑列按鈕來開啟一個視景，請執行下列動作：

I. 按一下「Open Perspective」按鈕  。

II. 這時會出現一個功能表，顯示和「Window」→「Open Perspective」功能表相同的選項。請從功能表中選擇「Other...」。

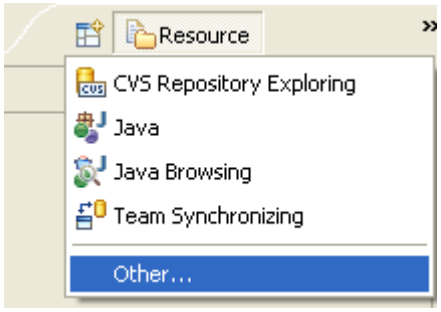


圖 2.14

III. 在「Select Perspective」對話框中，選擇 Debug，然後按一下 OK。

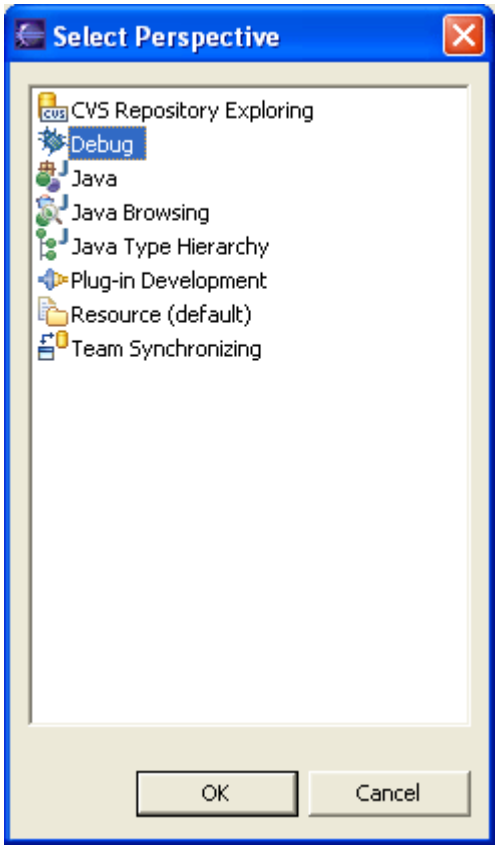


圖 2.15

這時會顯示「Debug」視景。

IV. 另外還有些有趣的事情值得注意。

- 現在，視窗標題會指出「Debug」視景正在使用中。
- 捷徑列包含幾個視景：原始「Resource」視景、新的「Debug」視景，以及少數幾個其他視景。「Debug」視景按鈕是已經下按的，表示它是現行視景。
- 如果要顯示視景的完整名稱，請用滑鼠右鍵按一下視景列，再勾選 Show Text。

圖 2.16

V. 在捷徑列中，按一下「Resource」視景按鈕。這時「Resource」視景又會成為現行視景。請注意，每個視景所擁有的一組視圖各不相同。

2.9.2 新視窗

除了在現行工作台視窗中開啟視景之外，也可以在另一個視窗中開啟新的視景。

依預設，新視景會開啟在現行視窗中。可以利用「Window」→「Preferences」→「Workbench」→「Perspectives」來配置這個預設行為。

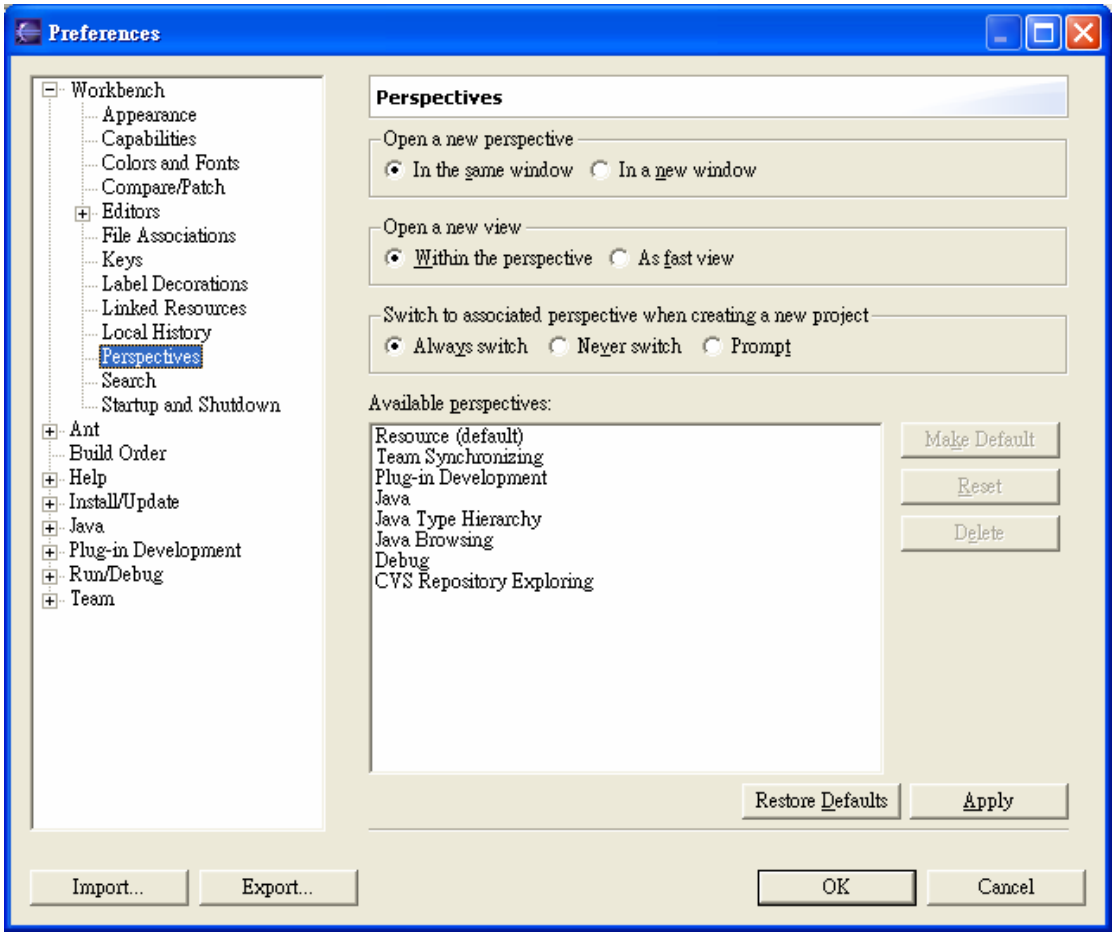


圖 2.17

2.9.3 儲存視景

可以利用工作台來儲存自己喜好的佈置，供未來使用。

I. 在捷徑列中，按一下「Resource」視景。現在「Resource」視景是在作用中。

II. 拖曳「Outline」視圖，將它和「Navigator」視圖堆放在一起。

III. 選擇「Window」→「Save Perspective As...」

IV. 「Save Perspective As...」對話框可用來重新定義現有的視景，或建立新視景。

按一下 OK 來更新「Resource」視景，並在後續的確認對話框中按一下 OK。如果重設視景或開啟新視景，就會使用新的視景佈置。

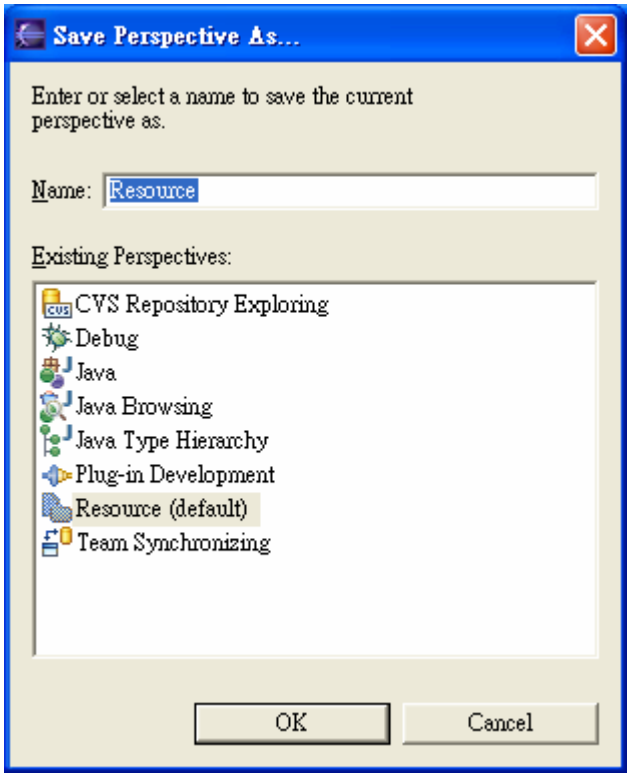


圖 2.18

V. 在「Resource」視景中移動「Outline」視圖，現在，它和「作業」視圖堆放在一起。

VI. 選擇「Window」→「Reset Perspective」。請注意，「Outline」視圖會和「Navigator」堆放在一起。原先，當第一次啟動工作台時，它是在導覽器下面，但由於儲存視景時將「儲存庫」和「Outline」堆疊起來，因此，它現在就以此為起始佈置。

VII. 選擇「Window」→「New Window」來開啟第二個視窗，以顯示資源視景。請觀察它，它在使用新儲存的佈置。

VIII. 關閉第二個視窗。

在變更過「Resource」視景之後，還有一個方法可用來回復原始佈置。如果要將「Resource」視景重設回程式佈置：

I. 選擇「Window」→「Preferences」。

II. 展開 Workbench，然後選取 Perspective。

III. 選取 Resource 視景， 然後按一下 Reset 按鈕，再按一下 OK。

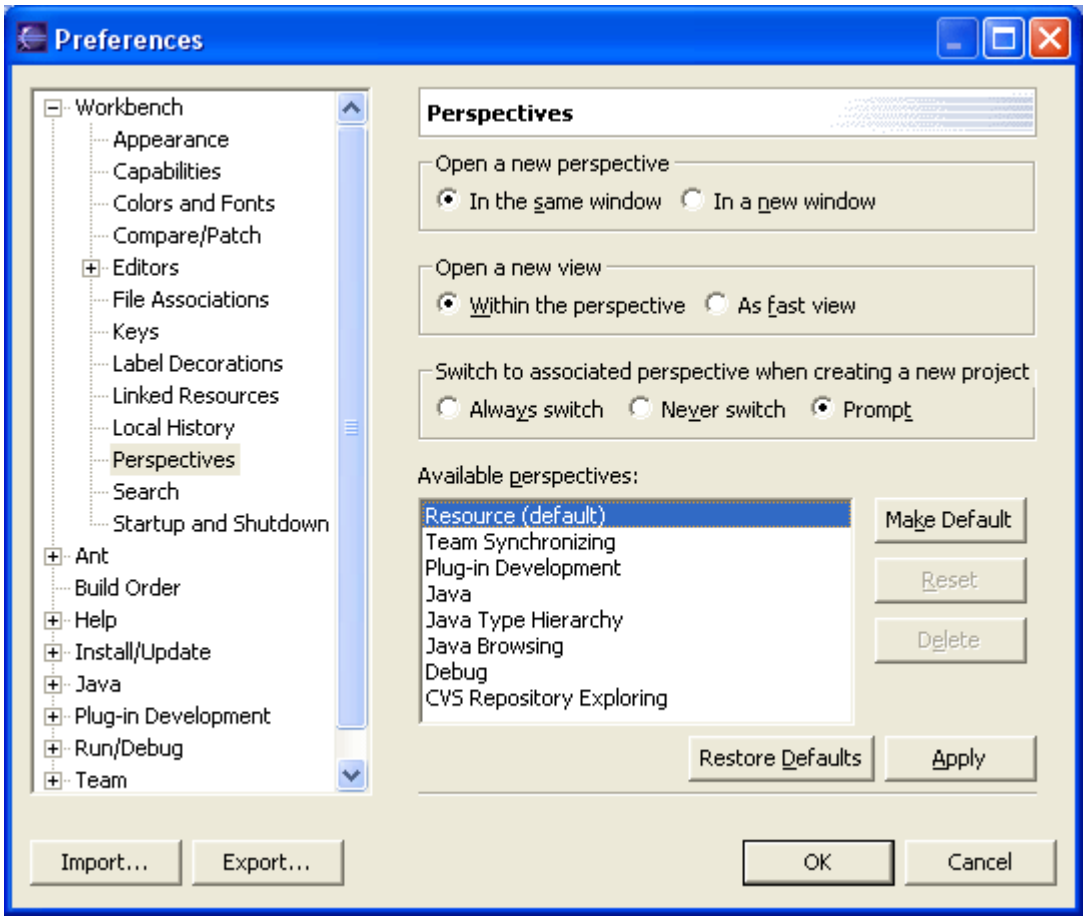


圖 2.19

IV. 現在，視景儲存狀態的任何變更尚未完成。如果要更新正在處理的「Resource」視景現行複本，請從工作台的功能表列中選取「Window」→「Reset Perspective」。

2.9.4 配置視景

除了配置視景的佈置之外，也可以控制視景的若干其他主要方面。其中包括：

- 「New Window」。
- 「Window」→「Open Perspective」。
- 「Window」→「Show View」。
- 工具列所顯示的各組動作。

請嘗試自訂這些項目之一。

I. 在捷徑列中，按一下「Resource」視景。

II. 選取「Window」→「Customize Perspective...」

III. 選取 Commands 標籤。

IV. 勾選 Launch，然後按一下 OK。

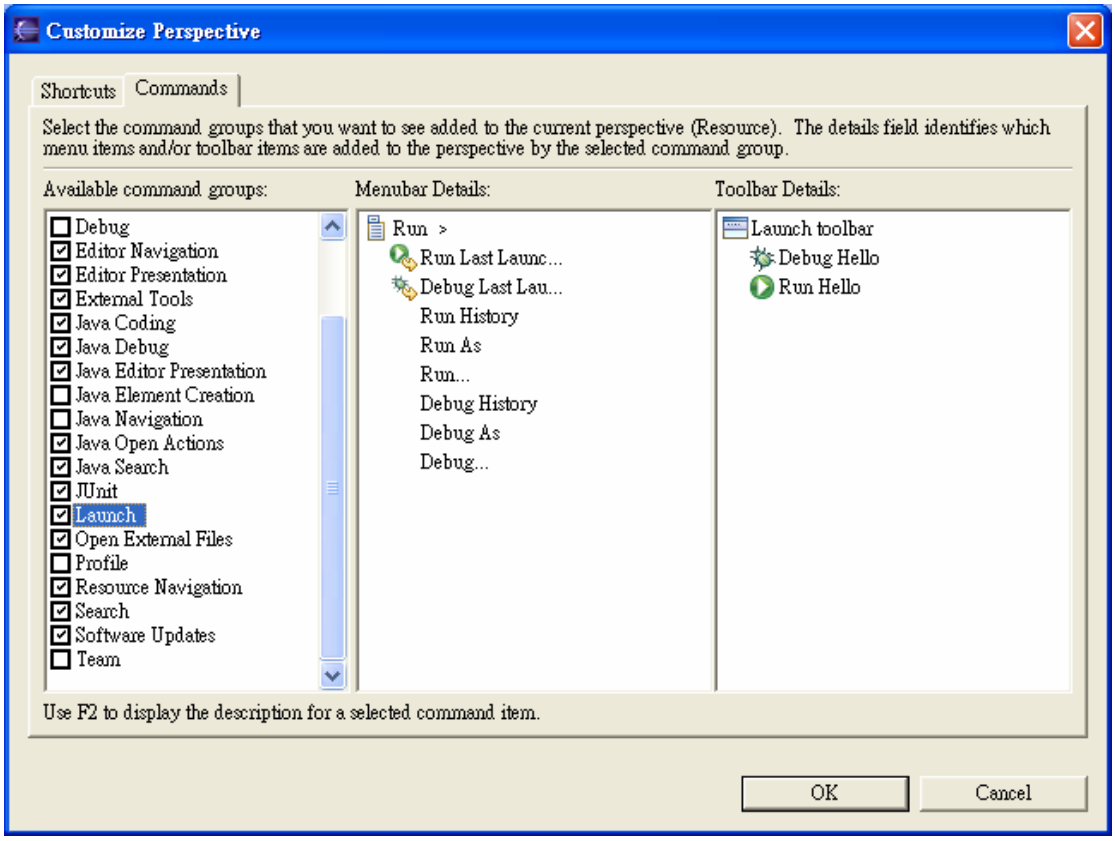
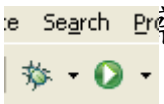


圖 2.20



序工具列，現在它含有除錯 / 執行啟動的按鈕。

VI. 嘗試過「Customize Perspective...」對話框中的其他選項之後，請選擇「Window」→「Reset Perspective」，讓視景返回原始狀態。

2.10 作業和標記

標記有許多類型，包括書籤、作業標記、除錯岔斷點和問題。這

一節的重點是作業和「Tasks」視圖。

「Tasks」視圖會顯示工作台中的所有作業。這個視圖會顯示與特定檔案、特定檔案中的特定行的相關作業，以及沒有與任何特定檔案相關的一般作業。

2.10.1 不相關的作業

未關聯的作業不會關聯於任何特定資源。如果要建立未關聯的作業：

I. 在「Tasks」視圖中，按一下「Add Task」按鈕  。這時會出現新作業對話框。

II. 輸入作業的簡要說明，再按 Enter 鍵。如果在輸入說明時要取消對話框，請按 Esc。這時「Tasks」視圖中會出現新的作業。

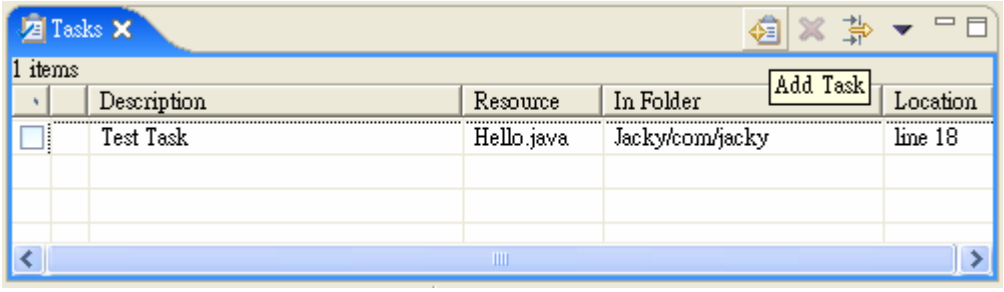


圖 2.21

2.10.2 相關的作業

相關作業會關聯於資源中的某特定位置。如果要將作業關聯於 Hello.java，請執行下列動作：

I. 在「Navigator」視圖中，按兩下開啟 Java 程式 (Hello.java)。

II. 在文字編輯器任何一行左側的編輯器區域中，從標記列來存取快速功能表。標記列主要文字區域左側的垂直列。

III. 從標記列的快速功能表中，選取 Add Task。

標記列會顯示包括書籤、(相關作業的)作業標記和/或除錯岔斷點在內的任何標記。可以直接從檔案中特定行左側的標記列中，存取快速功能表來將各種標記關聯於特定行。

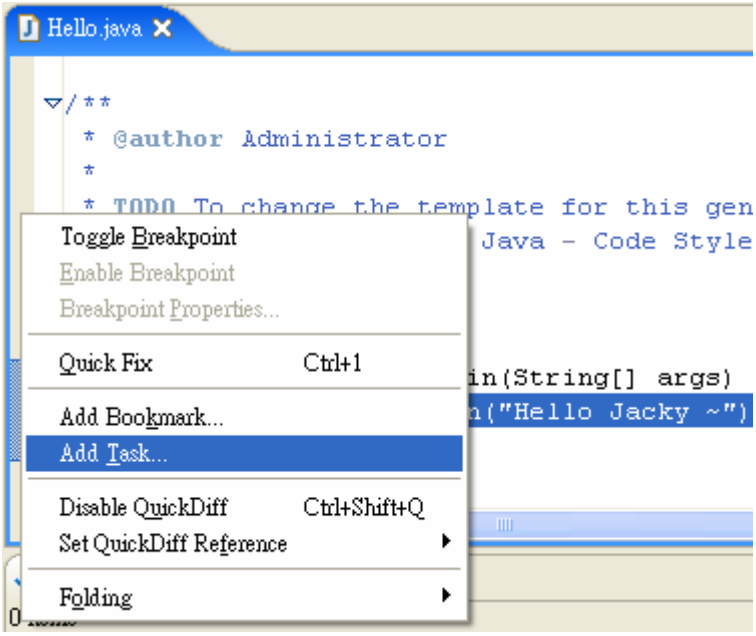


圖 2.22

IV. 在 Description 欄位中，輸入要關聯於文字檔中的這一行之作業的簡要說明。

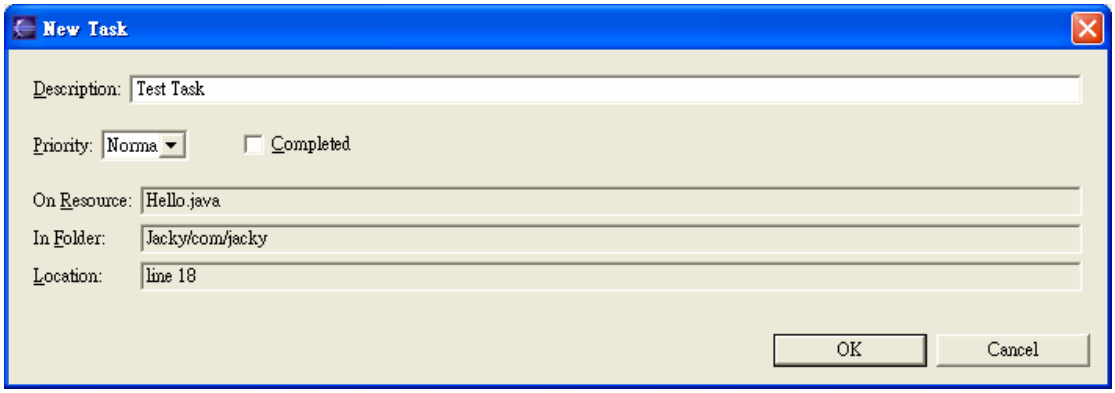


圖 2.23

V. 完成之後，按一下 OK。

VI. 請注意，新的作業標記會出現在標記列中，就在新增作業的那一行左側。另外，也請注意，新作業會出現在「Tasks」視圖。

VII. 新增作業之後，請按一下編輯器中的第一行，或新作業所關聯的行上面的任何其他行。

VIII. 在這點上，新增若干文字行到檔案中。

IX. 請注意，當在上面新增了若干文字行時，作業標記會跟著檔案中相關的行而在標記列中下移。當儲存檔案時，「Tasks」視圖中的行號會被更新。

X. 在「Tasks」視圖中，存取剛建立之作業的快速功能表。

XI. 選取 Mark Completed。

XII. 現在，從標記的快速功能表選取 Delete Completed Tasks。

XIII. 請注意，這時標記列的作業標記會消失，且會從「Tasks」視圖中移除作業。

2.10.3 開啓檔案

「Tasks」視圖提供兩個開啟作業的相關檔案的方法：

- 選取作業，然後從快速功能表中，選擇 Go To
- 按兩下作業

這兩種方法都會開啟檔案編輯器，且會標示出選取的作業所關聯的那一行。

2.11 書籤

書籤是導覽至常用資源最簡單的方式。這一節要看看書籤的設定和移除，以及在「Bookmarks」視圖中檢視它們。

「Bookmarks」視圖會顯示工作台中的所有書籤。如果要顯示「Bookmarks」視圖，請在「Resource」視景中，選取「Window」→「Show View」→「Bookmarks」。

2.11.1 新增和檢視書籤

工作台可以用書籤來標示個別檔案或檔案內的位置。這一節要示範如何利用「Bookmarks」視圖來設定若干書籤及檢視它們。

I. 從功能表列中，選取「Window」→「Show View」→「Bookmarks」。這時「Bookmarks」視圖會出現在工作台中。

II. 編輯 Hello.java 檔。

III. 將游標放在檔案中任何一行旁的編輯器標記列上。然後，從標記列的快速功能表中，選取 Add Bookmark。

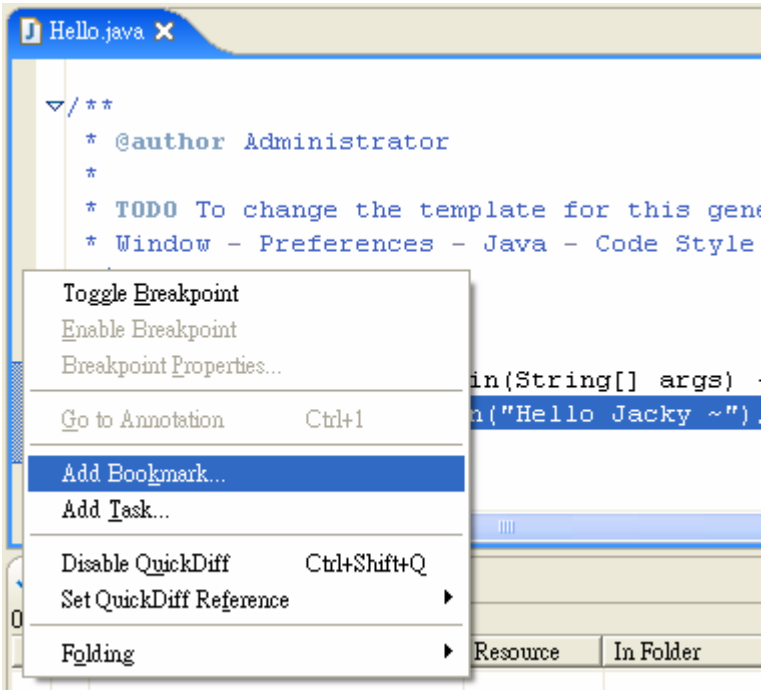


圖 2.24

當「Add Bookmark」對話框開啟時，輸入這個書籤的說明。請輸入「我的書籤」。

IV. 請注意，標記列中會出現一個新書籤。

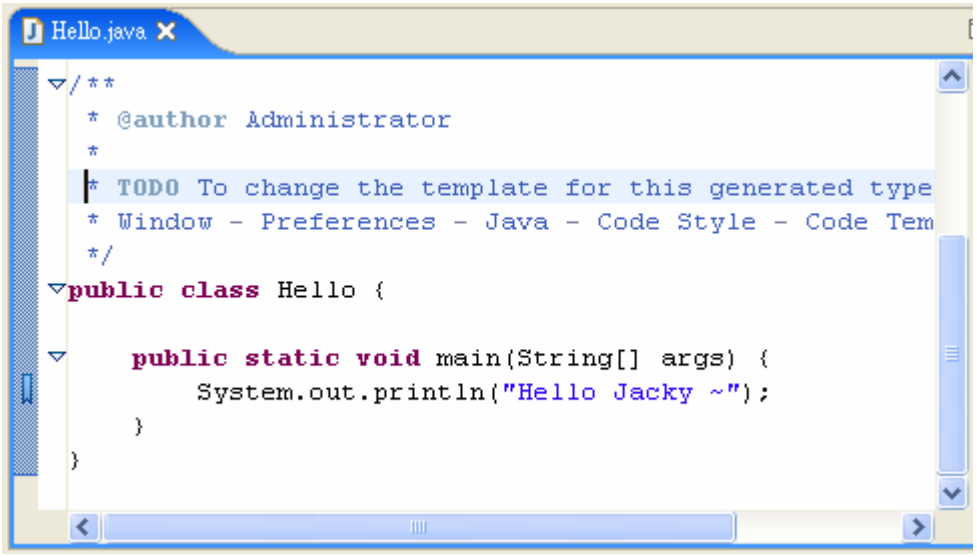


圖 2.25

新書籤也會出現在「Bookmarks」視圖中。

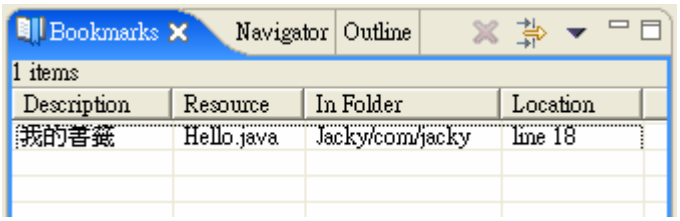


圖 2.26

V. 在「Navigator」視圖中，選取 Hello.java 檔。從主工作台功能表中，選取「Edit」→「Add Bookmark」。

這將會使用檔案名稱說明書籤，來建立檔案的書籤。現在，請看看「Bookmarks」視圖，其中有兩個書籤。

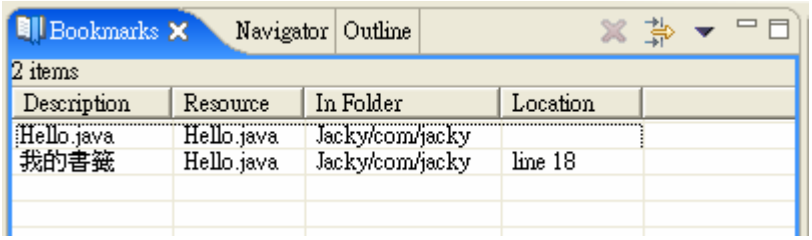


圖 2.27

2.11.2 使用書籤

建立好若干書籤之後，現在，將提供一些指示來說明如何取得書籤相關檔案的指示。

I. 在編輯器區域中，關閉所有檔案。

II. 在「Bookmarks」視圖中，按兩下所建立的第一個書籤（我的書籤）。

III. 請注意，這時會有開啟的編輯器顯示書籤所關聯的檔案，且會標示出書籤所關聯的那一行。

附註：「Bookmarks」視圖支援用另一種方式來開啟所選書籤的相關檔案，只要從書籤的快速功能表中選取 Go To 就行了。

在「Bookmarks」視圖中，選取導覽器中的相關檔案。

I. 在「Bookmarks」視圖中，選取「我的書籤」。

II. 從書籤的快速功能表中，選擇 Shoe in Navigator。

III. 請注意，現在可以見到「Navigator」視圖，且會自動選取 Hello.java 檔。Hello.java 是「我的書籤」所關聯的檔案。

2.11.3 移除書籤

I. 在「Bookmarks」視圖中，選取 Hello.java（我們建立的第二個書籤），再執行下列其中一項動作：

- 按一下視圖工具列中的「刪除」按鈕  。
- 從書籤的快速功能表中，選取「刪除」。
- 按一下鍵盤上的 Delete 鍵。

請注意，書籤已從「Bookmarks」視圖中移除。

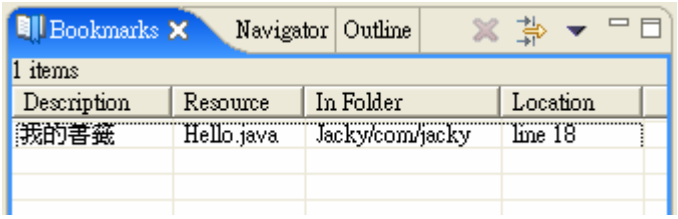


圖 2.28

II. 這時應該還有一個書籤。這個書籤與 Hello.java 檔其中一行相關。另外還有兩種方法可以移除這個書籤。

- 使用 Hello.java 編輯器標記列中的 Remove Bookmark。請記住，最初建立書籤時，在標記列中使用 Add Bookmark。

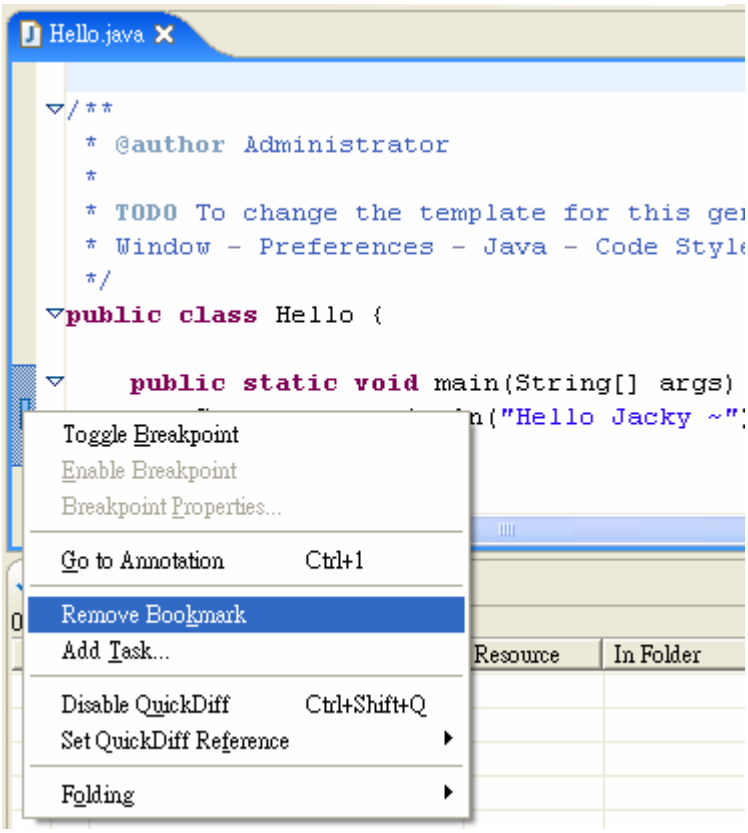


圖 2.29

- 在「Bookmarks」視圖中，利用書籤蹦現功能表中的刪除來刪除書籤（如上面所執行的動作）。

以下是第二個方式。

III. 確定有編輯器開啟了 Hello.java。

雖然編輯器實際上不需要開啟，但刪除書籤時，可以檢視編輯器

更新。

IV. 在「Bookmarks」視圖中，選取 Hello.java（剩下的書籤）。 按一下視圖工具列中的「刪除」按鈕  。請注意，書籤已從「Bookmarks」視圖及 Hello.java 編輯器中移除。

2.12 快速視圖(Fast View)

快速視圖是隱藏而可以快速顯示的視圖。它們的運作方式和一般視圖相同，唯一不同之處是它們在隱藏時不會佔據工作台視窗的畫面空間。

這一節要說明如何將「Navigator」視圖轉換成快速視圖。

2.12.1 建立快速視圖

快速視圖是隱藏而可以快速顯示的視圖。這些指示開始於從「Navigator」視圖建立快速視圖，之後，再說明完成快速視圖之後要如何使用它。

以下是兩個建立快速視圖的方法

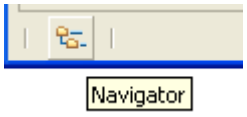
- 使用拖放技術。
- 使用視圖「系統」功能表所提供的功能表作業。

請依照下列方式，利用拖放技術來建立快速視圖。

I. 在「Navigator」視圖中，按一下標題列，將它拖曳到視窗左下方的捷徑列中。

II. 當游標到了捷徑列，它會變成一個"快速視圖"游標。請放開滑鼠按鈕，將導覽器放在捷徑列中。

現在，捷徑列中會有導覽器快速視圖的按鈕



如果要利用第二種方法來建立快速視圖，首先在「Navigator」視圖的標籤上蹦現快速功能表。請從這個功能表中，選取 Fast View。

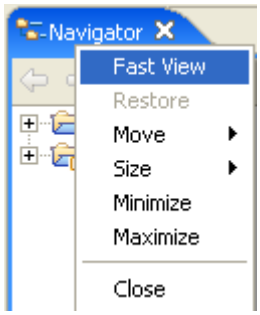
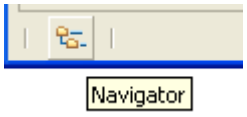


圖 2.30

2.12.2 使用快速視圖

現在，導覽器已轉換成快速視圖。這一節要示範它現在能做什麼。

請確認視窗左下方的捷徑列仍有「Navigator」視圖，且外觀如下：



I. 在捷徑列中，按一下「Navigator」快速視圖按鈕。

II. 觀察「Navigator」視圖從視窗左側出現。

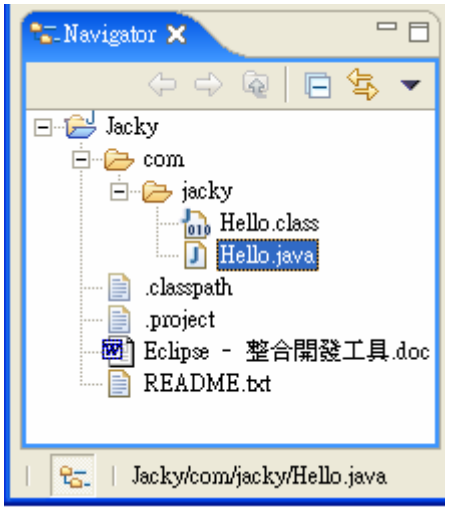


圖 2.31

III. 可以依照正常方式來使用「Navigator」快速視圖。如果要調整快速視圖的大小，請將滑鼠移到快速視圖右緣，游標在該處會變成雙箭頭。之後，請按住滑鼠左鍵來移動滑鼠。

IV. 如果要隱藏快速視圖，請按一下另一個視圖或編輯器，或在快速視圖的工具列中按一下最小化按鈕。

附註： 如果從「Navigator」快速視圖開啟檔案，快速視圖會自動隱藏起來，讓能夠使用檔案。

如果要將快速視圖轉換回正規視圖，請執行下列動作之一：

- 從視圖左上角圖示的快速功能表中，選擇快速檢視。
- 從工具列拖曳快速檢視圖示，然後將它放置在工作台視窗某處。

2.13 比較

工作台可用來比較多項資源以及在特殊的比較編輯器中呈現結果。

開始比較之前，必須建立一些檔案。

I. 利用專案的蹦現功能表來建立一個叫做 file1.txt 的檔案。在 file1.txt 的編輯器中，輸入下面這幾行文字，再將檔案儲存起來：

This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.

II. 在導覽器中，選取 file1.txt，再利用 Ctrl+C 來複製檔案。

III. 使用 Ctrl+V（貼上）來建立副本。在出現的名稱衝突對話框中， 將檔案重新命名為 file2.txt。

（在 Mac 中，請使用 Command+C 和 Command+V。）

現在，有兩個相同的檔案 file1.txt 和 file2.txt。

2.13.1 簡單比較

在導覽器中，選取 file1.txt 和 file2.txt，然後從快速功能表中，選取「Compare With」→「Each Other」。

這時會出現一個對話框，指出兩個檔案相同。

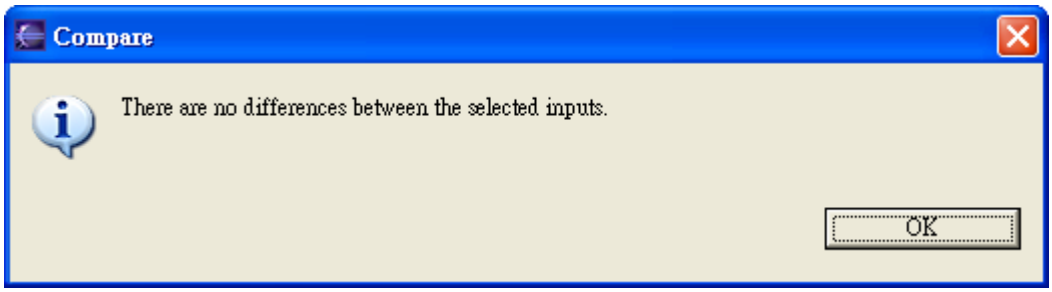


圖 2.32

依照下列方式來編輯 file1.txt：

I. 刪除第 1 行："*This is line 1.*"

II. 將第 3 行改成 "This is a much better line 3."

III. 插入第 4a 行（在第 5 行之前），內容為："This is line 4a and it is new"

現在，檔案（file1.txt）的內容應該如下：

This is line 2.

This is a much better line 3.

This is line 4.

This is line 4a and it is new

This is line 5.

儲存檔案的內容，方法是選取「File」→「Save」(或按 Ctrl+S)。

如果要比較檔案，請再次選取 file1.txt 和 file2.txt，從「導覽器」的快速功能表中，選取「Compare With」→「Each Other」。

這時會開啟一個特殊比較編輯器。下一節將說明如何使用這個比較編輯器。

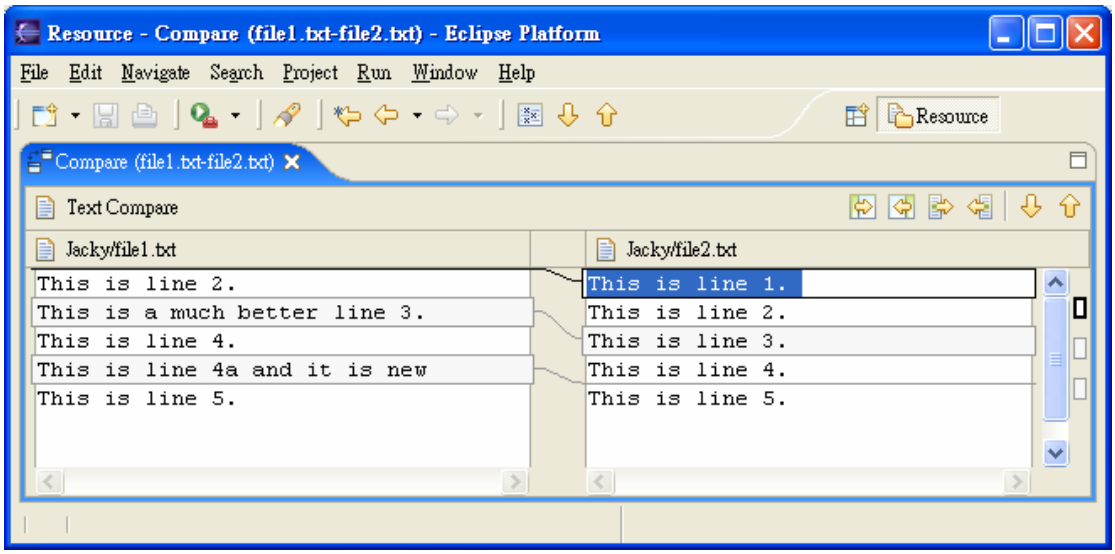


圖 2.33

2.13.2 瞭解比較

請比較在下列比較編輯器中產生的 file1.txt 和 file2.txt。左側顯示 file1.txt 的內容，右側顯示 file2.txt 的內容。連接左側和右側的線表示檔案之間的差異。

如果需要更多空間來查看比較，可以按兩下編輯器標籤，將編輯器最大化。

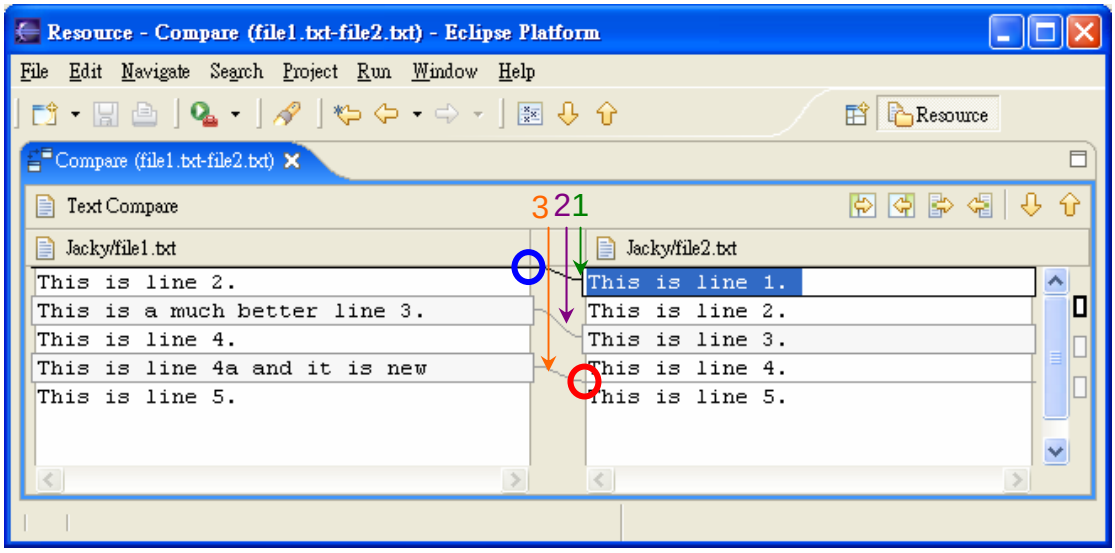


圖 2.34

不同編輯器左側的編號變更如下：

I. 從最上面一行開始（左窗格），差異列（在藍圈區）指出左側檔案的最頂端遺漏了什麼。請遵循右側檔案的差異群（請參閱 #1）。它含有 "This is line 1"。

II. 下一行 "This is line 2." 是白色，指出它符合右側檔案。

III. 移至下一行（背景顏色是彩色），可以看到左側檔案和右側檔案這一行的內容不同（請參閱 #2）。

IV. 下一行（第 4 行）又是白色，因此，可以跳過它。

V. 下一行是在左側檔案中，但由於它使用背景顏色，可以沿著它的右側差異列（請參閱#3），注意到右側檔案並沒有包含這一行（請參閱紅色圓圈）。

開始時，比較編輯器會有點令人氣餒，但當沿著左側向下作業，將焦點放在有灰色標示的項目以及左側中所沒有的項目時，就不會像原先那麼不好處理。

2.13.3 使用比較

請比較在下列比較編輯器中產生的 file1.txt 和 file2.txt。這

一節要示範如何使用比較編輯器來解析兩個檔案之間的差異。

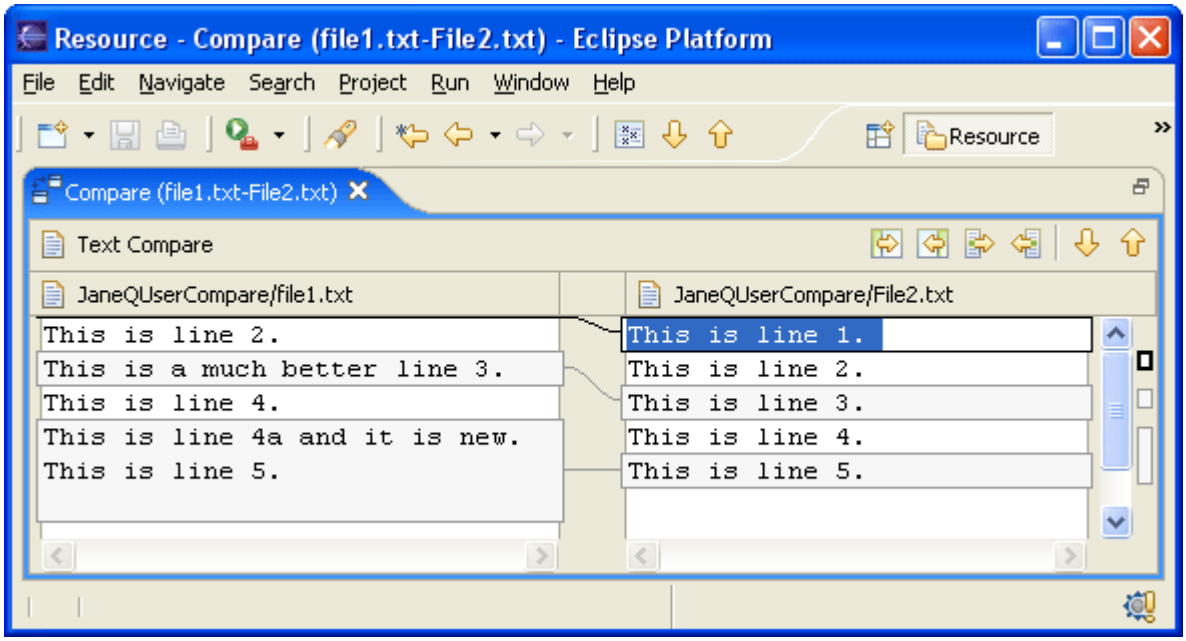
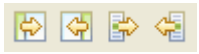


圖 2.35

比較編輯器的本端工具列有兩個部分。請利用右側的本端工具列按鈕群組來移至下一個或上一個變更。



I. 按一下「選取下一個變更」按鈕  。請觀察它如何選取下一個差異。

II. 再按一次「選取下一個變更」按鈕，移至下個變更。

III. 按一下「選取上一個變更」按鈕。

如果要將左側檔案的變更合併到右側檔案中，請使用左側的本端工具列按鈕群組，反之亦然。可以執行四類型的合併：

- 由左向右複製整份文件。
- 由右向左複製整份文件。
- 由左向右複製現行變更。
- 由右向左複製現行變更。

通常，當左或右側的整個檔案可由其他檔案的內容來取代時，都會使用複製整份文件的動作。

「複製現行變更」按鈕可以合併單一變更。

I. 確定已選取第二個差異（如下所示）：

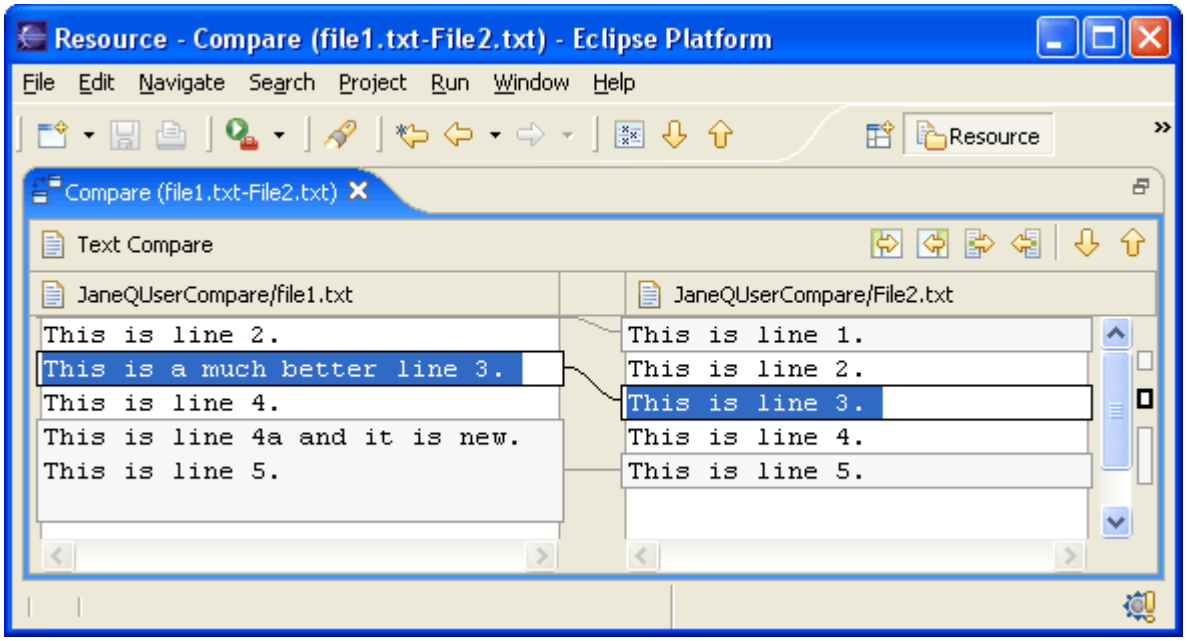


圖 2.36

II. 按一下從右向左複製現行變更 。觀察右側檔案中的所選文字，現在已複製到左側檔案中。

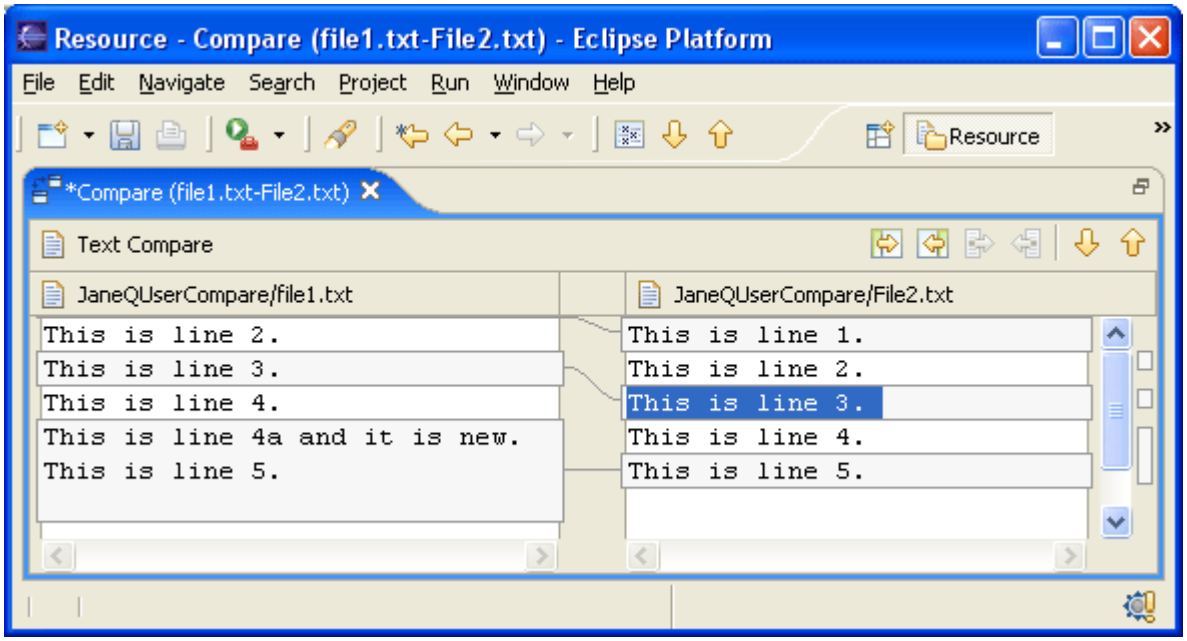


圖 2.37

III. 關閉比較編輯器，然後選擇 OK 來儲存變更。另外，也可以選擇「File」→「Save」（Ctrl+S）來儲存變更。

2.14 歷史紀錄

每次在工作台中儲存可編輯的檔案時，工作台都會更新這個檔案的歷史紀錄，且會將變更記載下來。之後，只要所需狀態不是太久以前，仍在儲存歷程中，就可以存取檔案的歷史紀錄且可以回復到先前所儲存的檔案複本。

I. 建立名稱為 sampleFile.txt 的新檔案。

II. 在 sampleFile.txt 的編輯器中修改資源，先新增 "change1" 這一行，再將檔案儲存起來。

III. 輸入新的一行 "change2"，再重新儲存它，以重複這個動作。

IV. 新增第三行 "change3"，再重新儲存它。

V. 從「Navigator」視圖中的資源的快速功能表中，選取「Replace With」→「Local History...」。

VI. 這時會開啟「從歷史紀錄取代」對話框，且會顯示檔案先前的歷史紀錄。

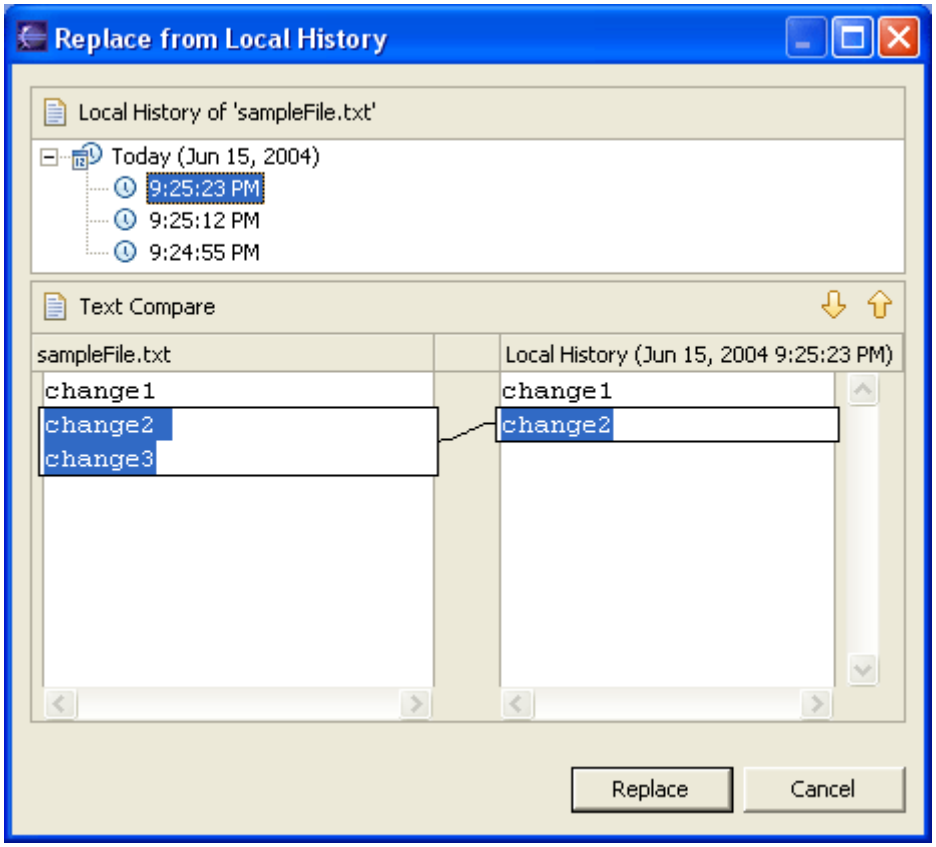


圖 2.38

對話框左窗格含有檔案的工作台副本。上圖顯示工作台包含的複本有完整的三行，也就是工作台編輯區目前所顯示的相同複本。歷史紀錄中的第一個項目（請參閱上面）含有最後儲存的檔案副本。這是只有兩行文字的副本。樹狀結構中最終項目是檔案的第一個副本。

對話框底端區域會顯示工作台檔案和歷史紀錄中所選取的特定檔案複本的差異。

VII. 選取歷史紀錄中的第一個項目（如上所示）。右窗格應該會顯示一行的文字。

VIII. 按一下取代。這會以所選歷史紀錄項目來取代 sampleFile.txt 的工作台副本。

IX. 觀察 sampleFile.txt 編輯器，現在它有兩行。

2.15 回應 UI

依預設，所有 Eclipse 作業都是在使用者介面執行緒中執行的。當使用接受序列化程式碼執行緒作業的回應 UI 時，仍可以在 Eclipse 的其他位置中作業。如果沒有回應 UI 支援，當遇到速度慢的作業時，會被鎖定，因而無法執行任何其他動作。

雖然部分作業會自動在背景中執行（如自動建置），但在許多情況下，都會顯示一個對話框來提供在背景中執行作業的選項。比方說，手動建置專案有時要多花一些時間，在這期間，仍可以在 Eclipse 中繼續使用其他功能。

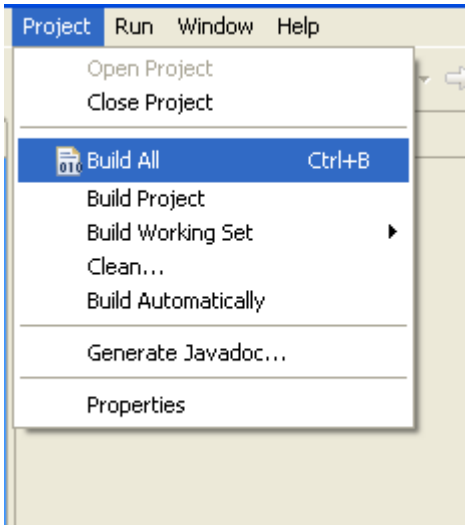


圖 2.39

當建置專案時，請從「Project」對話框中選取 Run in Background，回應 UI 可以在 Eclipse 中執行其他作業。

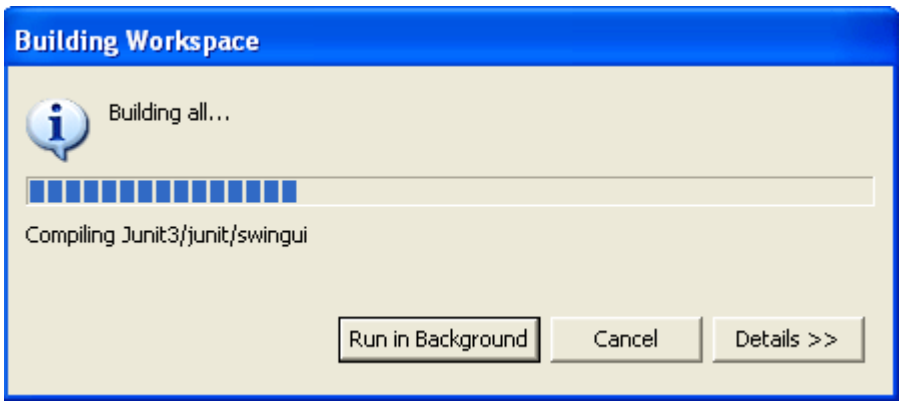
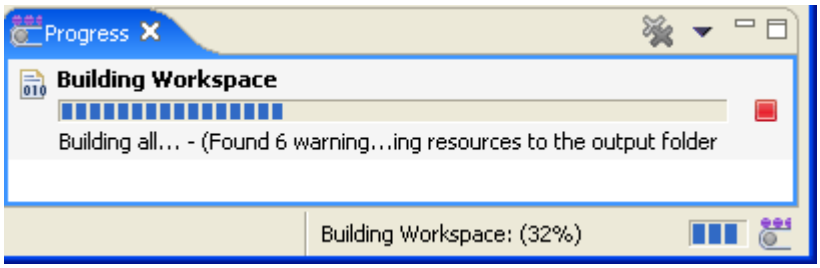


圖 2.40

如果需要動作狀態及目前在執行的其他作業的相關資訊，請按一下 Details。



Detail 畫面會顯示即將執行的作業以及可能同時在執行中的任何其他作業的狀態資訊。

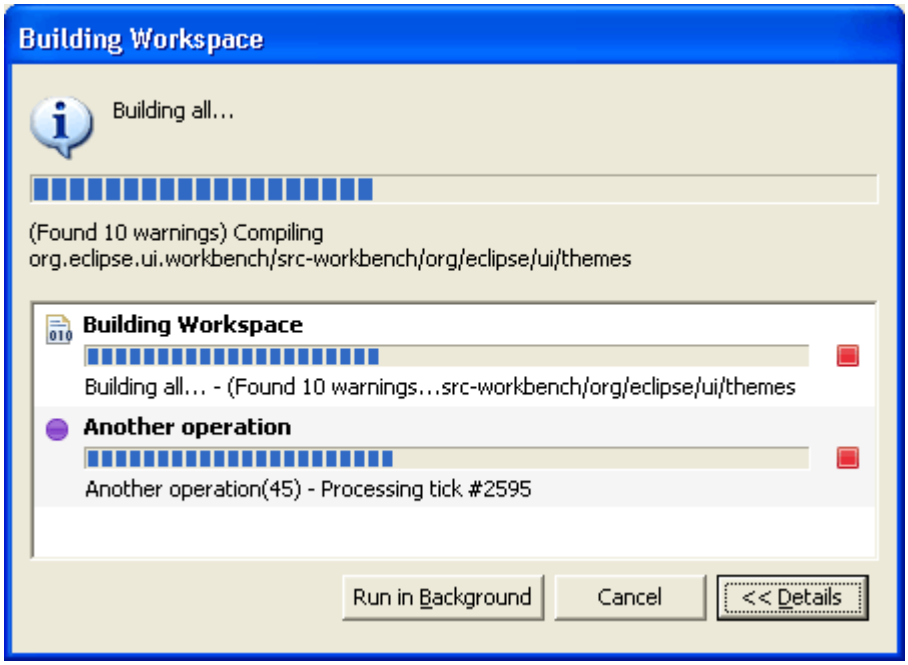


圖 2.41

當某項作業被其他作業鎖定時，「Progress Information」對話框也會加以指示。

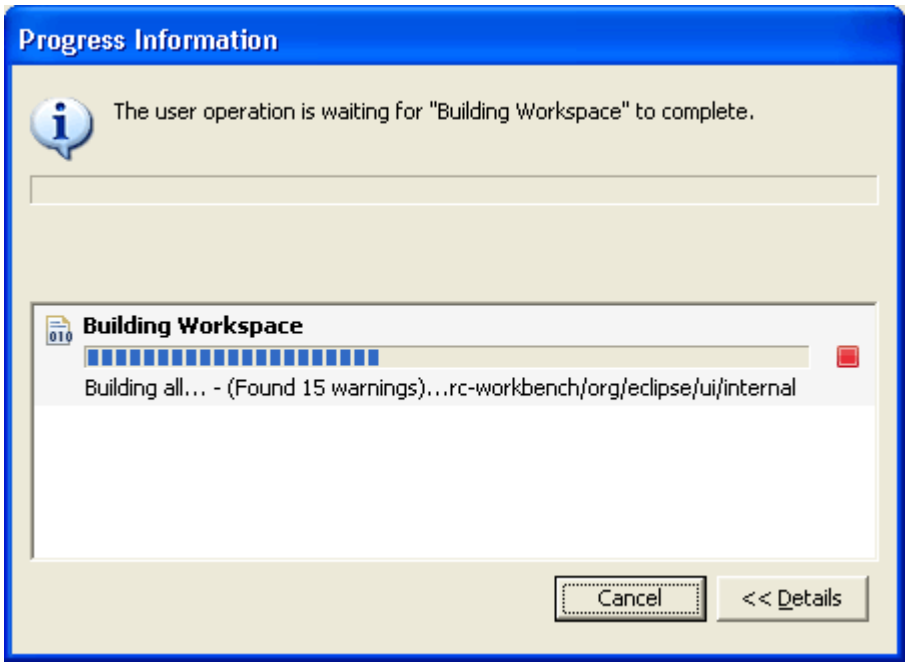


圖 2.42

如果要將在背景中執行的作業設為預設值，請選取「Window」→「Preferences」→「Workbench」，再勾選 Always run in background。

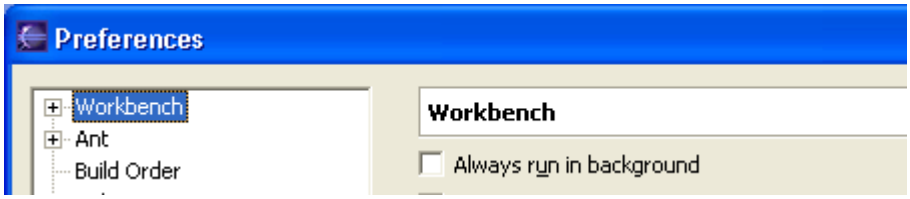


圖 2.43

3.喜好設定(Preferences)

「Preferences」對話框是用來設定使用者喜好設定的對話框。可以從「Window」→「Preferences」找到這個對話框。由外掛程式組成的喜好設定頁面也可以在這個對話框中找到。

喜好設定大部分的功能都是由其個別頁面所定義，但對話框提供了兩個一般功能：

- Import：匯出會將對預設喜好設定的任何變更寫入到使用者指定的檔案中。
- Export：匯入會套用使用者指定的檔案中的喜好設定。

喜好設定對話框的外觀如下：

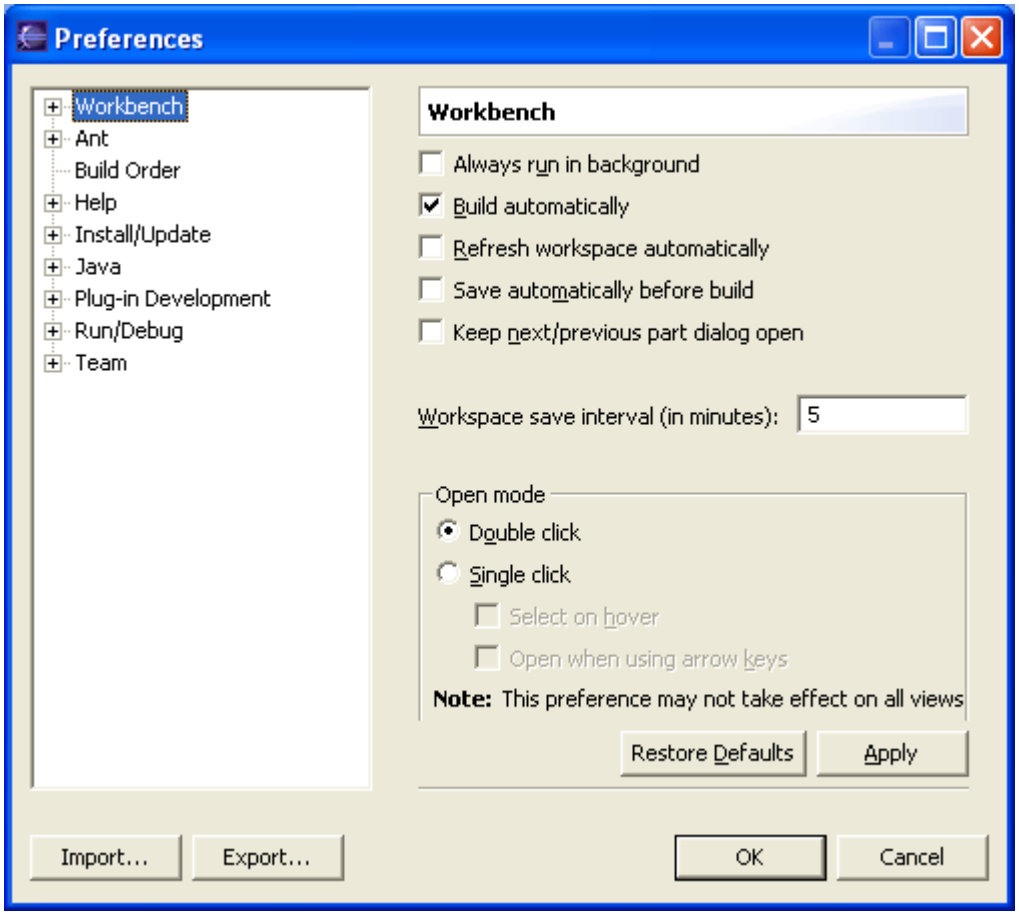


圖 3.1

3.1 工作台(Workbench)

*工作台*一詞指的是桌面開發環境。

每一個工作台視窗都含有一個或多個視景。視景則包含視圖和編輯器，以及控制在某些功能表和工具列上出現的項目。在任何給定的時間裡，可有多個工作台視窗存在於桌面上。

可以在工作台頁面中變更下列喜好設定。

選項	說明	預設值
Always run in background(一律在背景中執行)	如果開啟這個選項，工作台會在背景中執行特定動作，不會打擾到使用者。	關閉
Build automatically(自動建置)	如果開啟這個選項，每次儲存修改過的資源時，工作台都會自動執行建置動作。	開啟
Refresh workspace automatically(自動重新整理工作區)	如果開啟了這個選項，工作區資源自動與檔案系統中對應的資源同步化。 **附註：**這有可能會是一項冗長的作業，這會隨著工作區中的資源數目而不同。	關閉
Save automatically before build(建置之前自動儲存)	如果開啟這個選項，每次執行手動建置（從功能表列中，選取可用選項的「專案」）時，工作台都會自動儲存前次執行建置之後又修改過的所有資源。	關閉
Keep next/previous part dialog open(保持開啟下一個/上一個組件對話框)	如果開啟了這個選項，當編輯器和視圖循環對話框的啟動鍵放開時，對話框仍會維持開啟狀態。通常在按鍵組合放開時，會立即關閉對話框。	關閉
Workspace save interval (in minutes)(工作區儲存間隔（以分鐘為單	這個數字指出工作區的狀態自動儲存至磁碟的頻率。	5

選項	說明	預設值
位））		
Open mode...（開啟模式...）	可以選取下列方法之一，來開啟資源： 按兩下 - 按一下資源將會選取它，按兩下資源則會在編輯器中開啟它。 ■ 按一下（浮動說明時選取）- 將滑鼠游標橫越在資源上將會選取它，在資源上按一下則會在編輯器中開啟它。 ■ 按一下（使用方向鍵時開啟）- 以方向鍵選取資源時，會在編輯器中開啟它。 請注意，視哪一個視圖具有焦點而定，選取及開啟資源可能有不同的行為。	按兩下

工作台喜好設定頁面看起來如下：

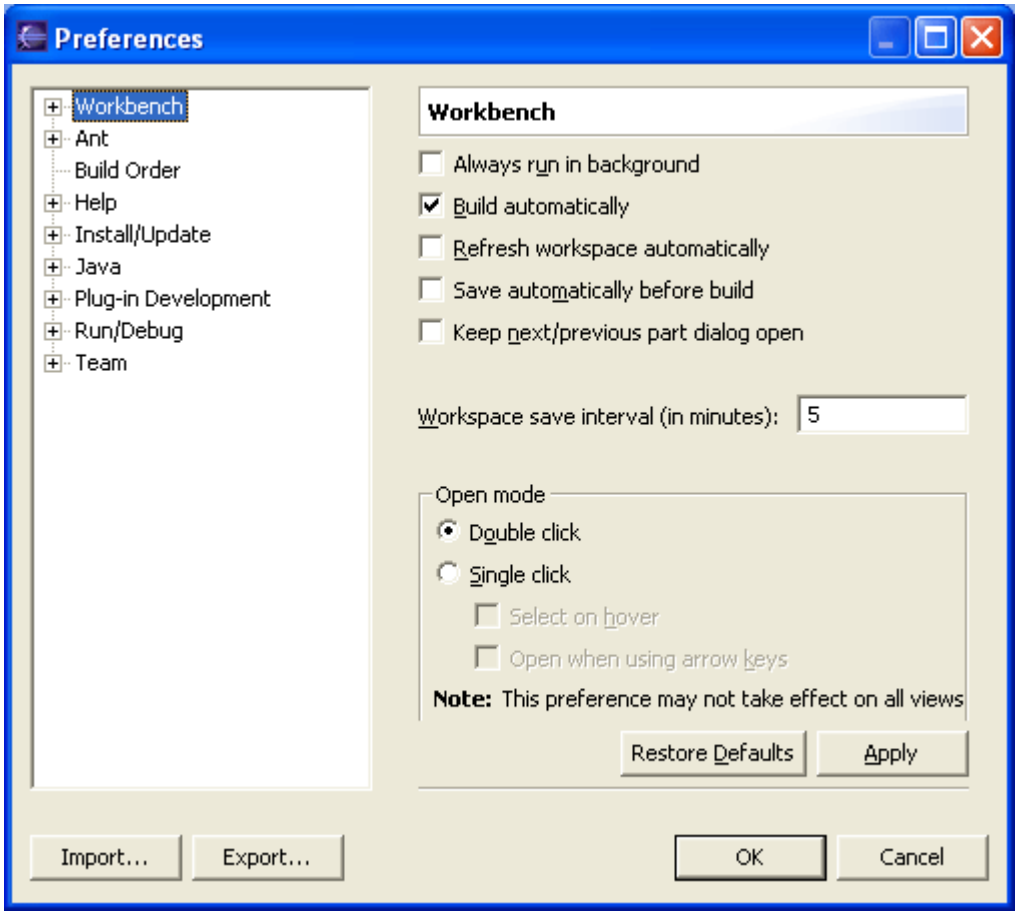


圖 3.2

3.1.1 外觀(Appearance)

可以在「外觀」頁面中變更下列喜好設定。

選項	說明	預設值
Tab positions - Editors(標籤位置 - 編輯器)	指定頂端或底端，以指出希望堆疊的編輯器的標籤出現的位置。	頂端
Tab positions - Views(標籤位置 - 視圖)	指定頂端或底端，以指出希望堆疊的視圖的標籤出現的位置。	頂端
Perspective switcher position(視景切換器位置)	請指定視景切換器列的位置	右上
Current presentation(現行呈現方式)	請指定目前作用中的呈現方式（外觀和操作方式）。	3.0 呈現方式
Current theme(現行主題)	請指定目前在作用中的主題（顏色和字型集）。	3.0 主題
Show text on perspective bar(在視景列顯示文字)	請指定應不應該在視景列及圖示中顯示標籤。	已啟用
Show traditional style tabs(顯示傳統樣式標籤)	請指定應不應該用傳統（方塊）標籤來取代曲線標籤	已停用

「外觀」喜好設定頁面看起來如下：

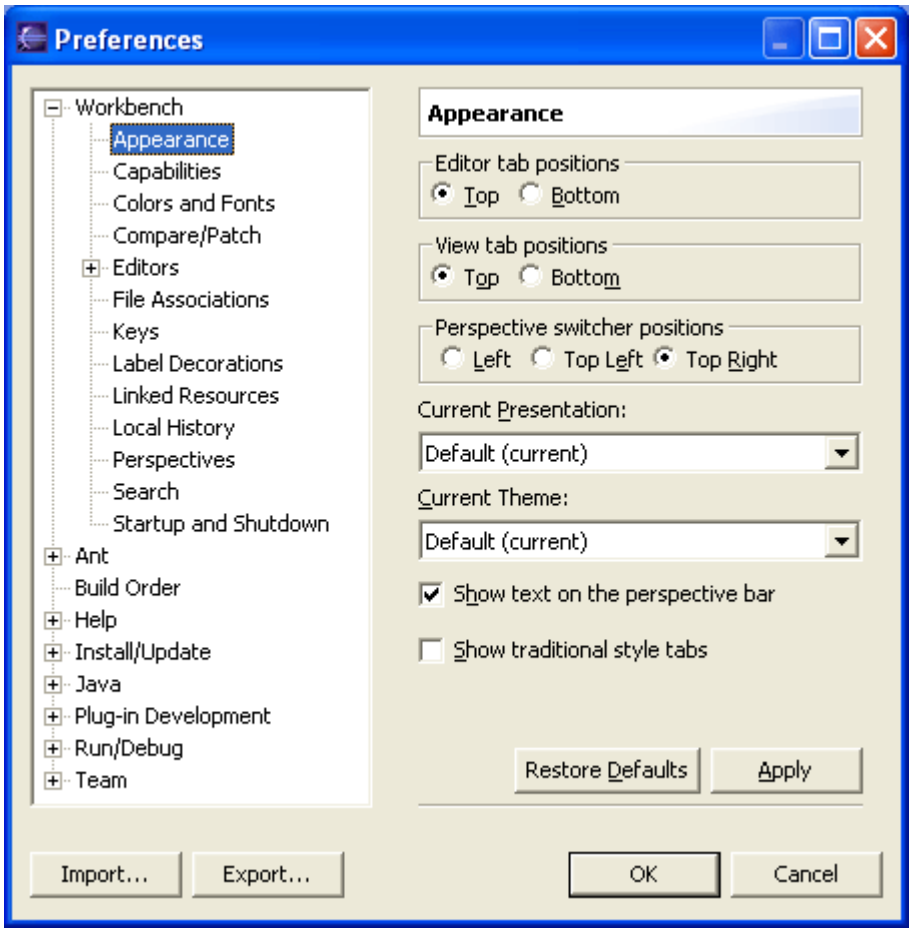


圖 3.3

3.1.2 功能(Capabilities)

「功能」喜好設定頁面可以啟用或停用各種產品元件，如 Java 開發和外掛程式開發。

附註：部分功能選項會相依於其他功能，停用某必要功能，卻仍啟用相依的功能，結果只會重新啟用它們。當取消選取 Java 開發和核心團隊支援時，就是如此。

「功能」喜好設定頁面看起來如下：

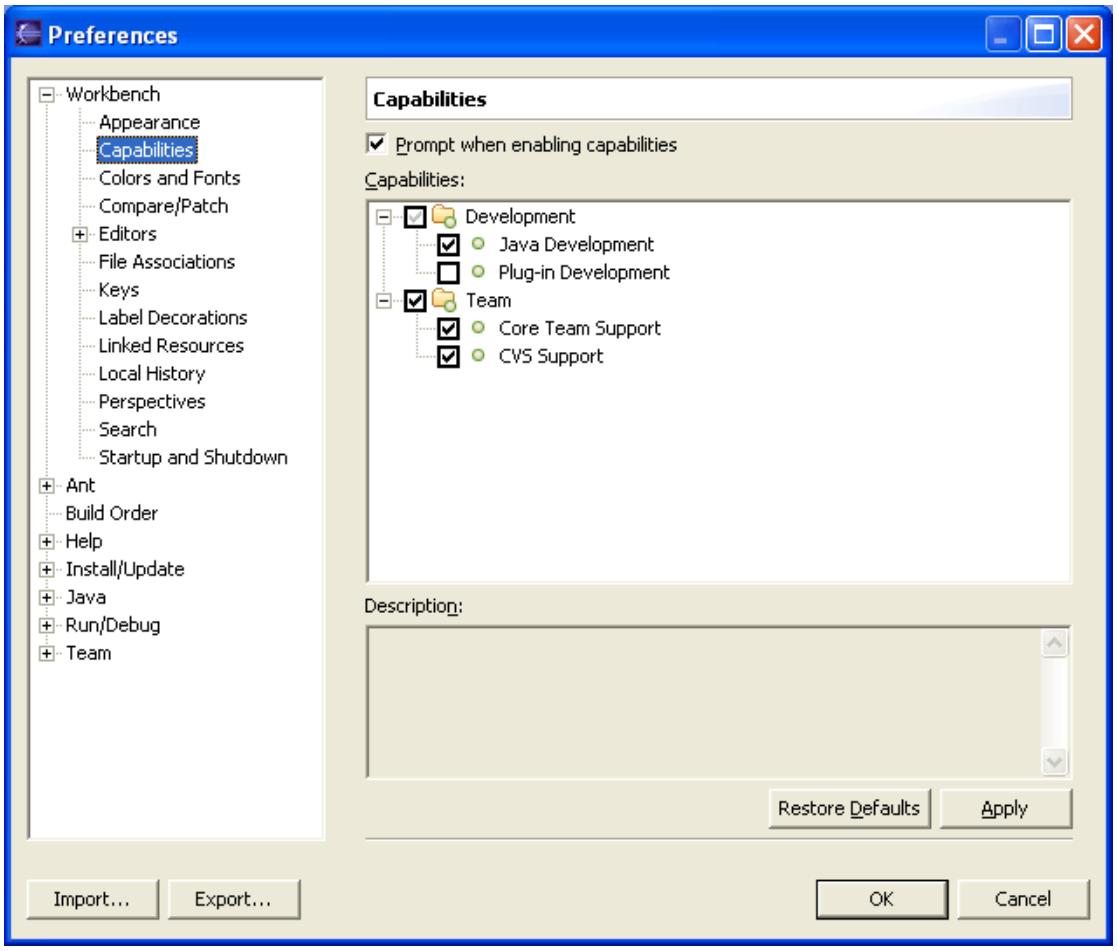


圖 3.4

當試圖啟用某個動作，但它的功能先前已停用或尚待喜好設定頁面予以啟用時，會出現下列 Confirm Enablement 提示，供確認確實要啟用必要的功能。請按一下 Details 來顯示功能的說明。

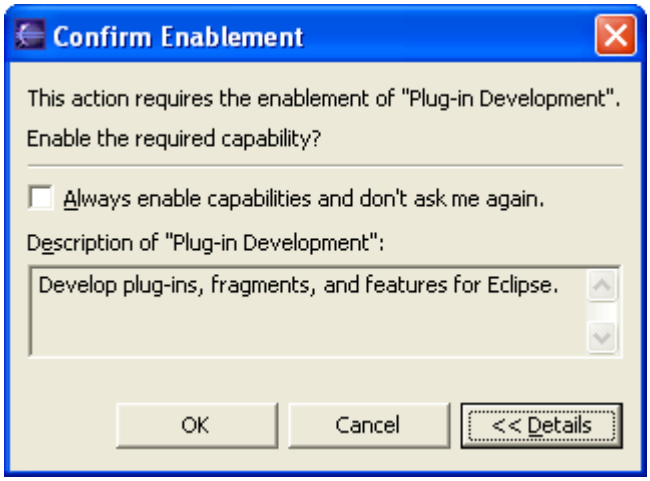


圖 3.5

3.1.3 顏色和字型(Colors and Fonts)

可以利用「顏色和字型」喜好設定頁面來設定 Eclipse 元件所用的許多字型和顏色。

樹狀結構用來導覽各種顏色和字型，以及顯示各種顏色和字型的預覽。任何字型的現行樣式（不是大小）預覽都會出現在它的標籤中。顏色的預覽則會出現在標籤的相關圖示中。另外，部分種類（尤其是工作台）會提供更詳細的構成要素預覽。這個預覽如果可用的話，會顯示在說明之下。

可以從清單中選取字型區，再按一下 Use System Font 來選取作業系統字型設定，或按一下 Change 來開啟選取字型的對話框，以變更字型設定。Reset 可用來返回預設值。

當選取了某個字型時，可以按一下樹狀結構區右側的顏色按鈕來變更顏色設定。Reset 可用來返回預設值。

「顏色和字型」喜好設定頁面看起來如下：

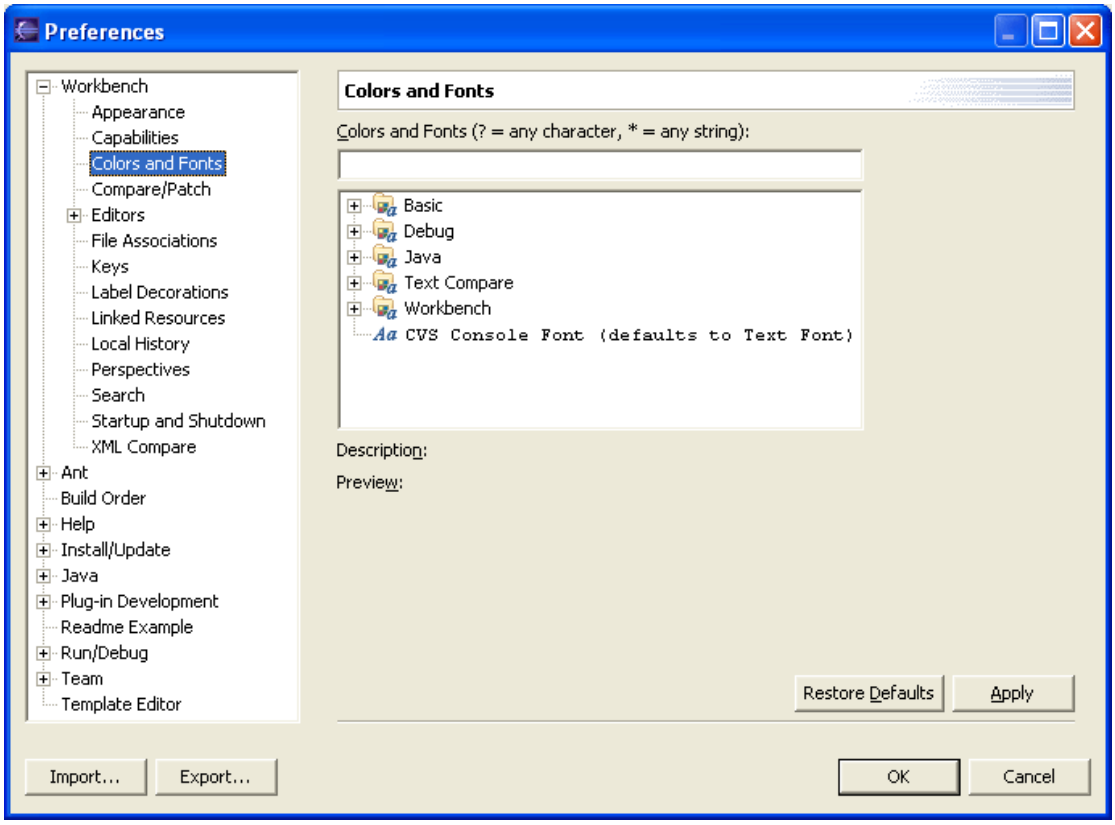


圖 3.6

「顏色和字型」文字欄位可用來過濾內容。只需要輸入一個項目，任何相符的結果都會保留在樹狀視圖中。

當選取工作台顏色和字型設定時，會提供說明和預覽。

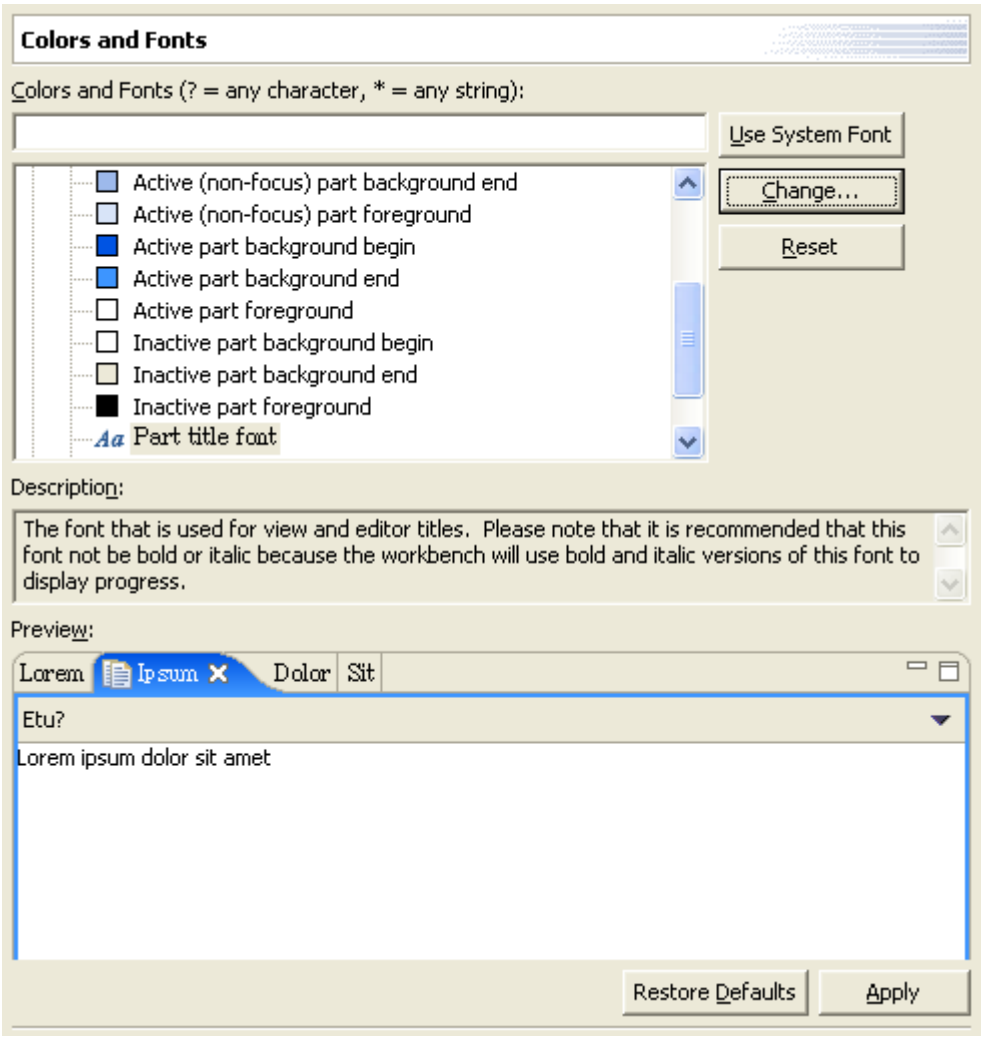


圖 3.7

3.1.4 比較/修正(Compare/Patch)

可以在「比較/修正」頁面中變更下列喜好設定。

一般選項(General)

選項	說明	預設值
Open structure	這個選項控制是否要在每當內容完成時自動	開啟

選項	說明	預設值
compare automatically(自動開啟結構比較)	執行結構比較。 如果不想看到結構上的差異，請關閉這個選項。	
Show additional compare information in the status line(在狀態行顯示其他的比較資訊)	如果開啟這個選項，則狀態行中會顯示關於變更的其餘資訊。 如果有意瞭解變更的其餘資訊，請開啟這個選項。	
Off(關閉)		
Ignore white space(忽略空格)	這個選項控制比較檢視器中是否要顯示空白變更。 如果想看到空白變更，請開啟這個選項。	關閉
Automatically save dirty options before patching(在修正前自動儲存變動過的選項)	這個選項可控制在套用修正前是否要自動儲存任何尚未儲存的變更。 如果要自動儲存變更，請開啟這個選項。	關閉

文字比較選項

選項	說明	預設值
Synchronize scrolling between panes in compare viewers(同步化捲動比較檢視器中的窗格)	兩個比較檢視器將會與對方一起「鎖定捲動」(lock scroll)，以便使每一個窗格內的程式碼的相同及對應部分並列顯示。 如果不希望比較檢視器鎖定捲動，請關閉這個選項。	開啟
Initially show ancestor pane(起始顯示上	有時會想比較資源的兩個版本與衍生它們的先前的版本。這稱為它們的*共同上代*，在三向比較期間，它會出現在其自己的比較窗格內。	關閉

選項	說明	預設值
代窗格)	如果希望上代窗格固定在比較開始時出現，請開啟這個選項。	
Show pseudo conflicts(顯示虛擬衝突)	顯示虛擬衝突，此衝突是在兩位開發人員進行相同的變更（例如，兩者都新增或移除完全相同的程式碼行或註解）時發生。 如果要虛擬衝突出現在比較瀏覽器中，請開啟這個選項。	關閉
Connect ranges with single line(包含單行的連線範圍)	控制不同的範圍是否要由單行或兩行所區隔的範圍來進行視覺化連線。	開啟

「比較」喜好設定頁面看起來如下：

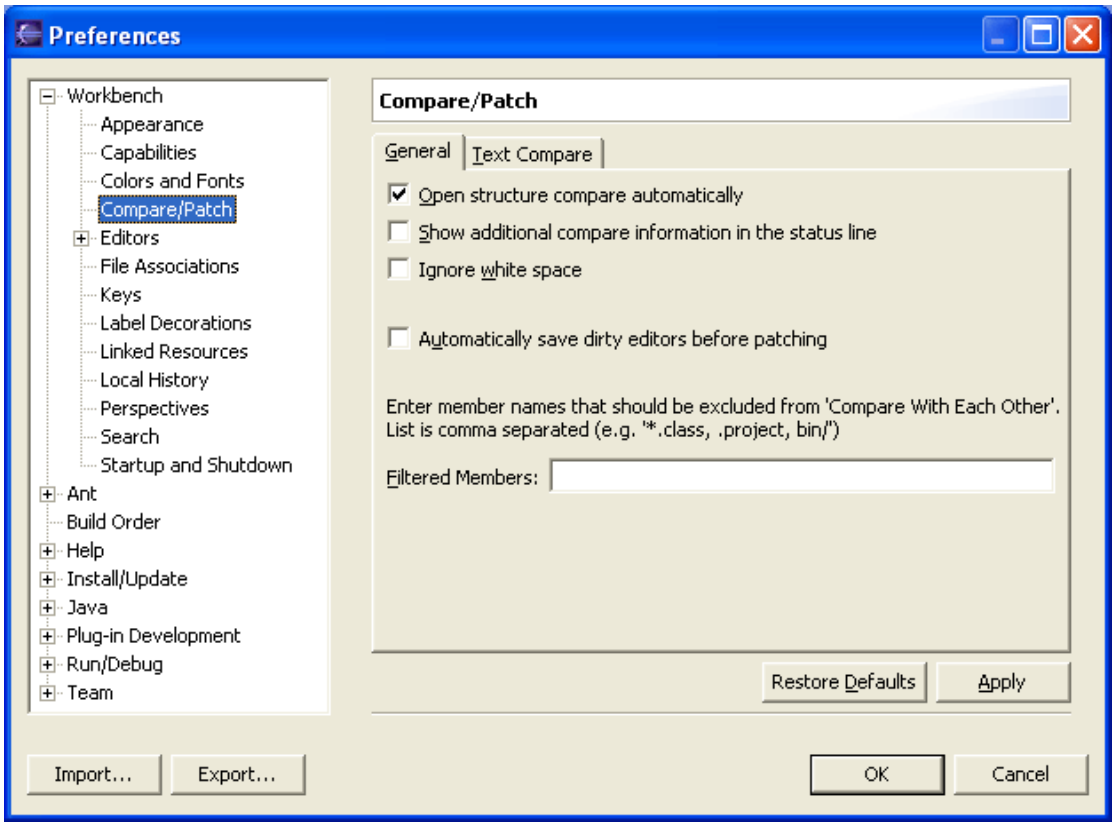


圖 3.8

3.1.5 編輯器(Editors)

可以在「編輯器」頁面中變更下列喜好設定。

選項	說明	預設值
Size of recently opened files list(最近開啟的檔案清單的大小)	以在編輯器中開啟的每一個檔案而言,它會被儲存在最近使用檔案的清單中。這個選項控制顯示在「檔案」功能表中的這個清單中的檔案數目。	4
Show multiple editor tabs(顯示多重編輯器標籤)	指定是否要顯示多重編輯器標籤。如果關閉的話,編輯器活頁簿會有一個大標籤,所有不可見的編輯器都只能從 chevron 存取。	開啟
Close all editors on exit(結束時關閉所有的編輯器)	這個選項是用來指定是否要在結束工作台時關閉所有的編輯器。啟用這個選項時,可以加速 Eclipse 的啟動, 因為它會減少啟動 Eclipse 所需要的工作量。	關閉
Close editors automatically(自動關閉編輯器)	這個選項用來指定是否要在工作台中重複使用編輯器。如果開啟,則可以指定在循環使用編輯器之前要使用的編輯器數目 (預設值是 8)。也可以指定當所有的編輯器都已用過時,是否要開啟提示對話框或新的編輯器。一旦開啟,就會將「固定編輯器」動作新增至工具列和編輯器標籤功能表。固定編輯器並不會循環使用。	關閉
Text File Encoding(文字檔編碼)	請使用這個選項來指定在編輯器中儲存文字檔時要使用的編碼。	預設值 (CP1252)

「編輯器」喜好設定頁面看起來如下：

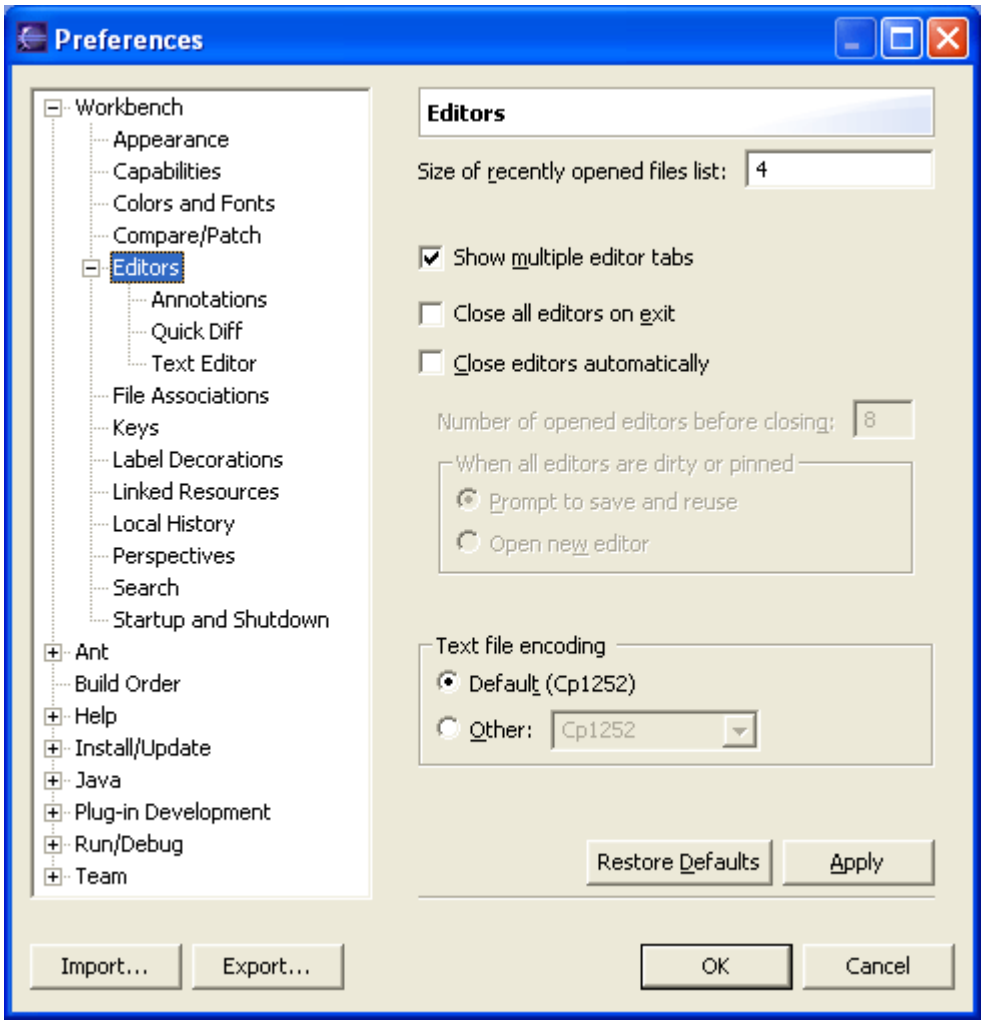


圖 3.9

3.1.6 檔案關聯(File Associations)

在「檔案關聯」喜好設定頁面上，可以新增或移除工作台可辨識的檔案類型。也可以建立編輯器與檔案類型清單中的檔案類型的關聯性。

檔案類型清單(File Associations)

- 新增：將新的檔案或檔案類型（副檔名）新增至預先定義的清單。在產生的「新檔案類型」對話框中，輸入檔案的名稱或副檔名。如果要新增副檔名，則必須在檔案類型前面輸入一點或

"*."（例如，".xml" 或 "*.xml"，而非只是 "xml"）。

- 移除：從清單中移除所選取的檔案類型

建立新檔案類型的對話框：

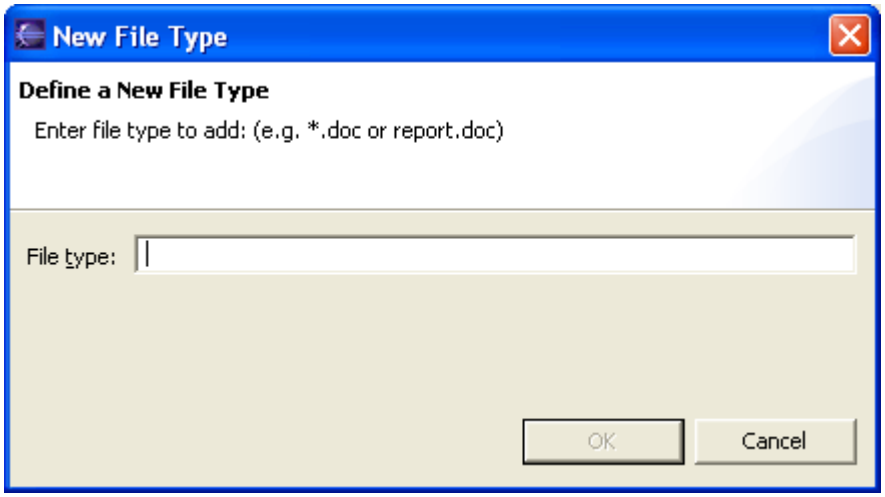


圖 3.10

相關編輯器清單(Associated Editors)

- 新增：將新編輯器新增至與上述選取之檔案類型相關聯的編輯器清單。在產生的「編輯器選擇」對話框中，可以選擇編輯器，以在工作台內（內部）或工作台外（外部）啟動；如果所要的編輯器未顯在清單中，請按一下 Browse 來自行尋找編輯器。
- 移除：移除編輯器與上述選取之檔案類型之間的關聯。
- 預設值：將所選取的編輯器設為上述選取之檔案類型的預設編輯器。該編輯器會移至「相關聯的編輯器」清單的頂端，以表示它是該檔案類型的預設編輯器。

建立新檔案關聯的對話框：

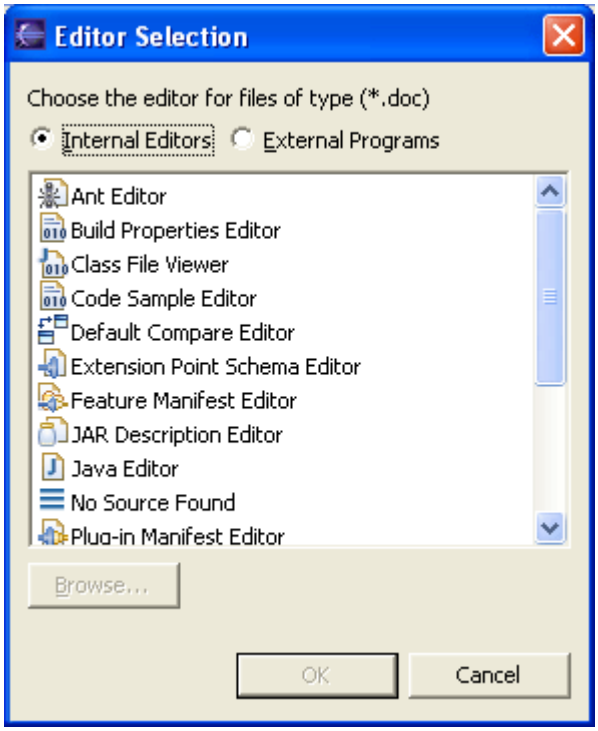


圖 3.11

「檔案關聯」喜好設定頁面看起來如下：

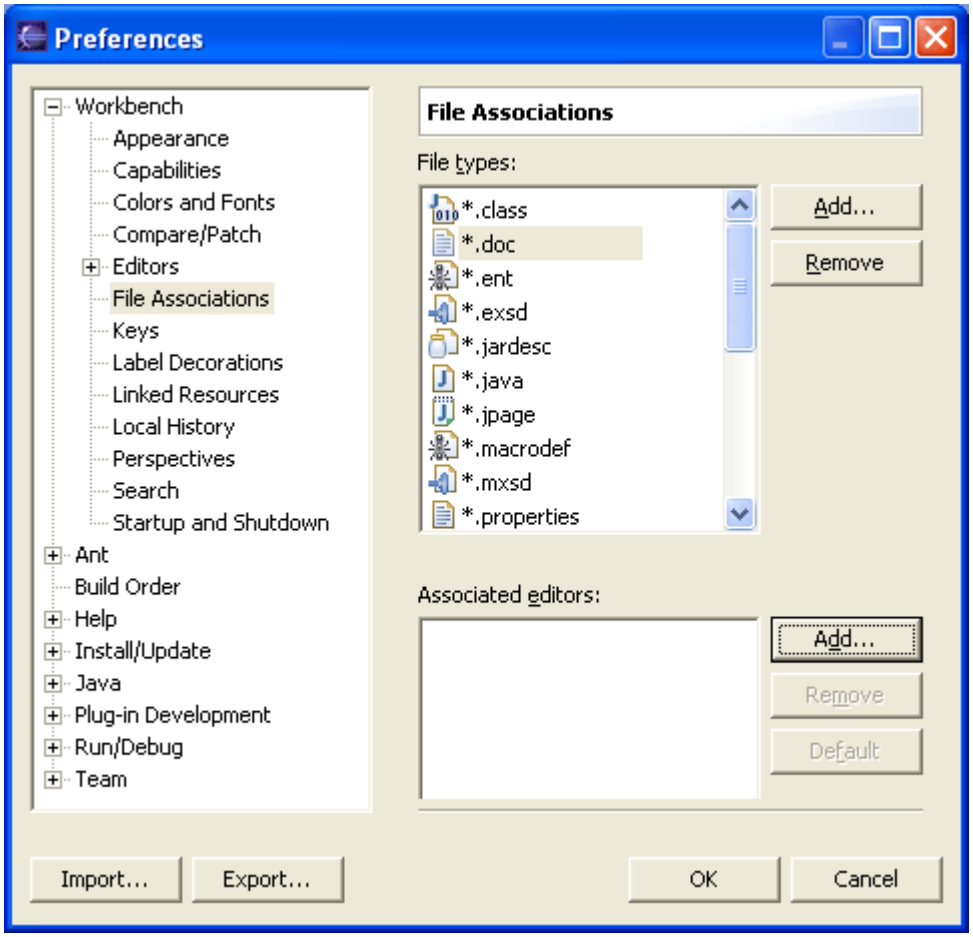


圖 3.12

3.1.7 按鍵(Keys)

在 Eclipse 中，可以廣泛自訂鍵盤的功能。Eclipse 中指定了許多按鍵作用和按鍵順序來呼叫特定的指令。

按鍵作用(Key Strokes)、按鍵順序(Key Sequences)和按鍵連結(Key Bindings)

「按鍵作用」是指按下鍵盤上的某個按鍵，同時選擇性按住一或多個下列修正鍵：Ctrl、Alt(在 Macintosh 上為 Option)、Shift 或 Command(只有 Macintosh 才有。)比方說，先按住 Ctrl，然後按 A 的時候，就會產生按鍵作用 Ctrl+A。單獨按下修正鍵時，並不會組成按鍵作用。

「按鍵順序」是指一或多個按鍵作用。傳統上，emacs 會指定兩個或三個按鍵作用的按鍵順序給特定的指令。例如，在 emacs 中，指派給全部關閉的正常按鍵順序是 Ctrl+X Ctrl+C。如果要輸入這個按鍵順序，必須按下按鍵作用 Ctrl+X，然後按下按鍵作用 Ctrl+C。雖然 Eclipse 支援任意長度的按鍵順序，但鍵盤捷徑的長度最好不超出四鍵。

「按鍵連結」是將按鍵順序指派給指令。

配置(Configurations)

「配置」是指一組按鍵連結。Eclipse 包含兩種配置：

- 預設值
- Emacs（延伸預設）

*預設*配置包含一組通用的按鍵連結，在許多情況下，使用者可以將它們視為傳統的按鍵順序。例如，Ctrl+A 是指派給全選，Ctrl+S 是指派給儲存等。

Emacs 配置包含一組 Emacs 使用者非常熟悉的按鍵連結。例如，Ctrl+X H 是指派給全選，Ctrl+X S 是指派給儲存等。

必須瞭解為什麼 *Emacs* 配置說它「延伸預設」，這一點很重要。與*預設*配置不同，*Emacs* 配置並不是一組完整的按鍵連結。相反的，它會盡可能借用*預設*配置，而且只會針對與*預設*配置不同的地方來定義明確的 Emacs 樣式按鍵連結。通常，諸如全選、儲存等常用的指令才會關聯於特定的 Emacs 按鍵順序。

使用者可以變更按鍵喜好設定頁面中的「作用中的配置」設定，來決定最喜歡使用的配置。如果使用者選擇*預設*配置，便會忽略所有 *Emacs* 按鍵連結。如果使用者選擇 *Emacs* 配置，則明確指派的 Emacs 樣式按鍵順序會優先於*預設*配置中任何衝突的指派。

環境定義(Contexts)

按鍵連結可能會因為 Eclipse 的現行環境定義而有所不同。

有時候，例如，作用中的組件可能會是 Java 檔案編輯器，這時指派不同組的按鍵順序可能會比作用中的組件是 HTML 檔案編輯器更合適。Ctrl+B 是一項特定的範例，在 Java 檔案編輯之類的環境定義中，Ctrl+B 通常是指派給建置，而在 HTML 檔案編輯之類的環境定義中，則是指派給將文字變為粗體字。這個環境定義通常是由作用中的組件來決定，但它也可能受到作用中的視窗或對話框的影響。如果作用中的組件沒有選擇特定的環境定義，工作台會將作用中的環境定義設定為*在視窗中*。

Eclipse 包含七種不同的環境定義。它們是：

- 在對話框和視窗中
- 在視窗中（延伸「在對話框和視窗中」）
- 在對話框中（延伸「在對話框和視窗中」）
- 編輯文字（延伸「在視窗中」）
- 編輯 Java 程式碼（延伸「編輯文字」）
- 除錯（延伸「在視窗中」）
- Java 除錯（延伸「除錯」）

環境定義與配置類似，它們可以延伸其他的環境定義。比方說，*編輯 Java 程式碼*環境定義會借用*編輯文字*環境定義的按鍵連結，後者又會從*在視窗中*環境定義借用按鍵連結。

附註：不建議將按鍵連結提升到它所延伸的環境定義。比方說，最好不要將*編輯文字*按鍵連結移到*在對話框和視窗中*環境定義。這可能會有非預期的結果。

可以讓某些按鍵連結在對話框中運作。這些按鍵連結會指派給*在對話框和視窗中*環境定義。「剪下」的按鍵連結就是這類按鍵連結的一個範例。可以變更這些按鍵連結。比方說，可以用 Ctrl+X 當成對話框的「剪下」功能，而以 Ctrl+W 作為視窗中的「剪下」功能。

平台和語言環境(Platform and Locale)

在不同的平台和語言環境下，按鍵連結也會不同。在 Macintosh 平台上，Command+S 是指派給儲存，而不是常用的 Ctrl+S。在中文語言環境中 (zh)，Alt+/ 是指派給內容輔助，而不是常用的 Ctrl+空白鍵。

當 Eclipse 啟動時，會決定現行的平台和語言環境，而且在 Eclipse 實例過程中並不會改變。

自訂按鍵連結(Customizing Key Bindings)

在自訂按鍵連結時，如果有多鍵的按鍵順序、配置和環境定義，則會有許多事項須記住。為了簡化，所有的按鍵自訂都是在「按鍵」喜好設定頁面中完成。

選取「Window」→「Preferences」→「Workbench」→「Keys」來進入「按鍵」喜好設定頁面。

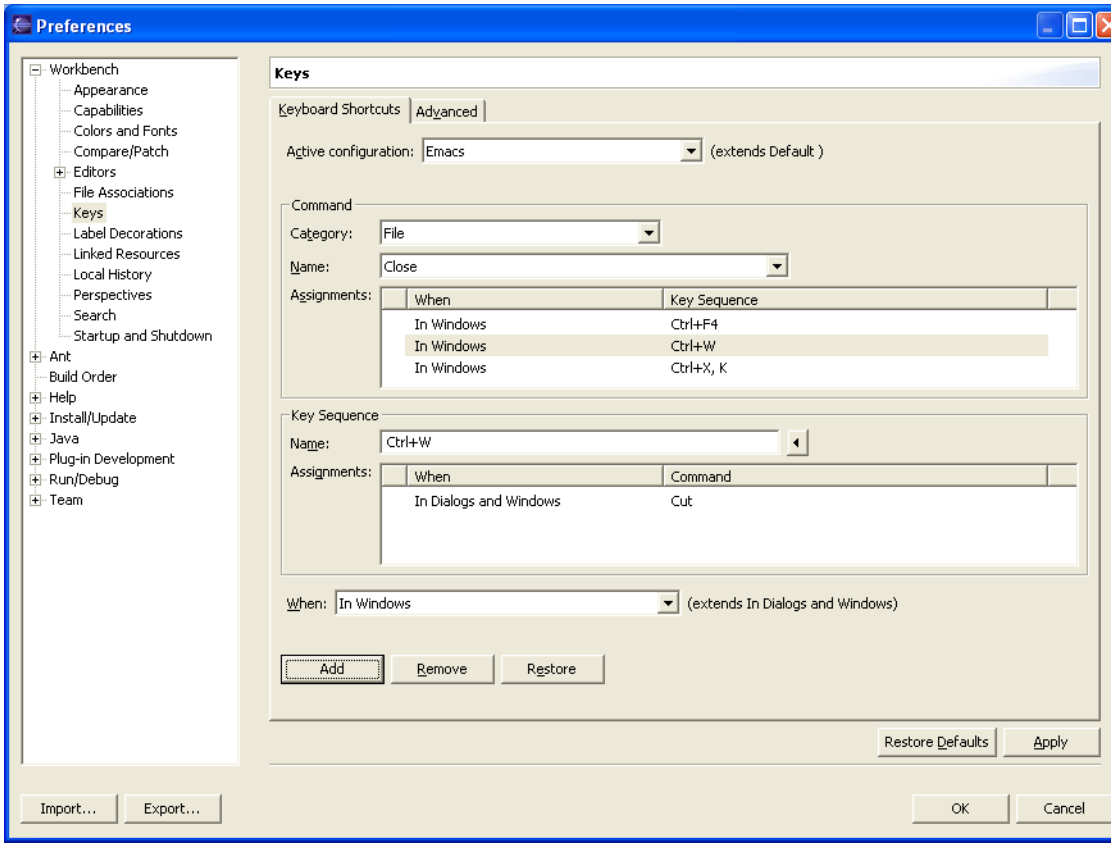


圖 3.13

在這個範例中，我們已經選擇 *Emacs* 作為作用中的配置，而且

已經從指令清單中選擇關閉指令。這個指令的資訊以及指令的現行按鍵連結都會顯示出來。

請注意，關閉已經指派了三個按鍵順序：*預設*配置中的 Ctrl+F4 和 Ctrl+W 以及 *Emacs* 配置中的 Ctrl+X K。兩者的指派環境定義都是*在視窗中*。因此，如果使用者將作用配置設為*預設*，就會將 Ctrl+F4 和 Ctrl+W 指派給關閉，而 Ctrl+X K 則不會。不過，如果使用者將作用中的配置設定為 *Emacs*， Ctrl+X K 就會指派給關閉。同時，由於 *Emacs* 配置也會從*預設*配置借用按鍵連結，因此，Ctrl+F4 和 Ctrl+W 也會指派給關閉，不過，這些按鍵連結必須尚未指派給 *Emacs* 配置中的另一個指令。在這個範例中，"Ctrl+W" 連結於 Emacs 按鍵配置中的剪下。

以下是指派給關閉的按鍵順序清單，還有一個地方可以新增或移除按鍵連結。依預設，它選取的環境定義是*在視窗中*。

輸入按鍵順序 Ctrl+W，就會啟用「新增」按鈕。同時，指定按鍵順序 Ctrl+W 的所有指令的清單會顯示在「新增」按鈕下面。這時可以看到 Ctrl+W 目前是在*在視窗和對話框中*的環境定義中指派給剪下指令。按一下「新增」指令來將 Ctrl+W 指派給關閉。

現在可以看到 Ctrl+W 已經新增至指派給關閉的按鍵順序清單中。請注意，小型的「變更」圖形 表示這個按鍵連結會變更現有的按鍵連結。請確定關閉的 Ctrl+W 按鍵連結在 Emacs 按鍵配置中能夠運作。剪下的連結仍存在，但只能在對話框中運作（也就是說，我們是「在對話框和視窗中」，而不是「在視窗中」）。我們可以隨時移除這個變更，方法是選取新的按鍵連結，然後按一下「移除」按鈕。這時就會自動還原將 Ctrl+W 指派給剪下的指派。

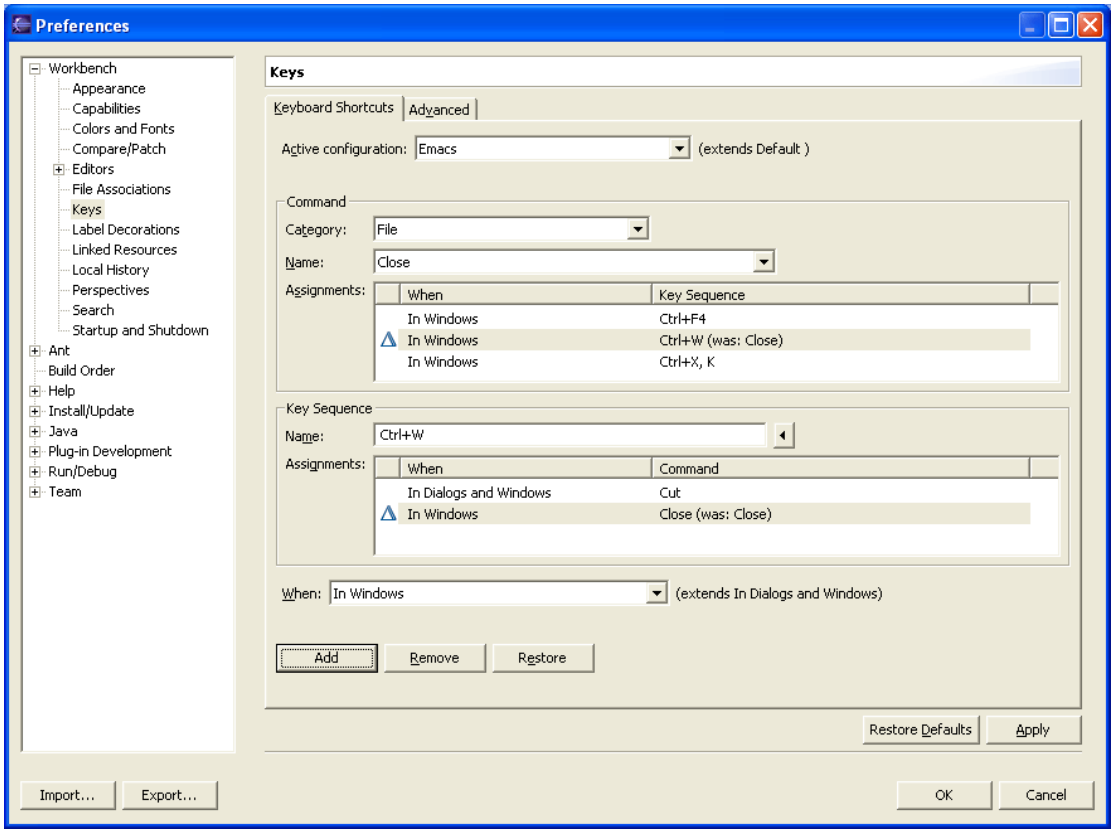


圖 3.14

選取剪下指令，就可以看到變更的結果。請注意，這個圖 表示按鍵連結已經移除。我們可以隨時還原這個按鍵連結，方法是在此處選取它，然後按一下「還原」按鈕，這樣就能有效移除在前一個步驟中新增的按鍵連結。

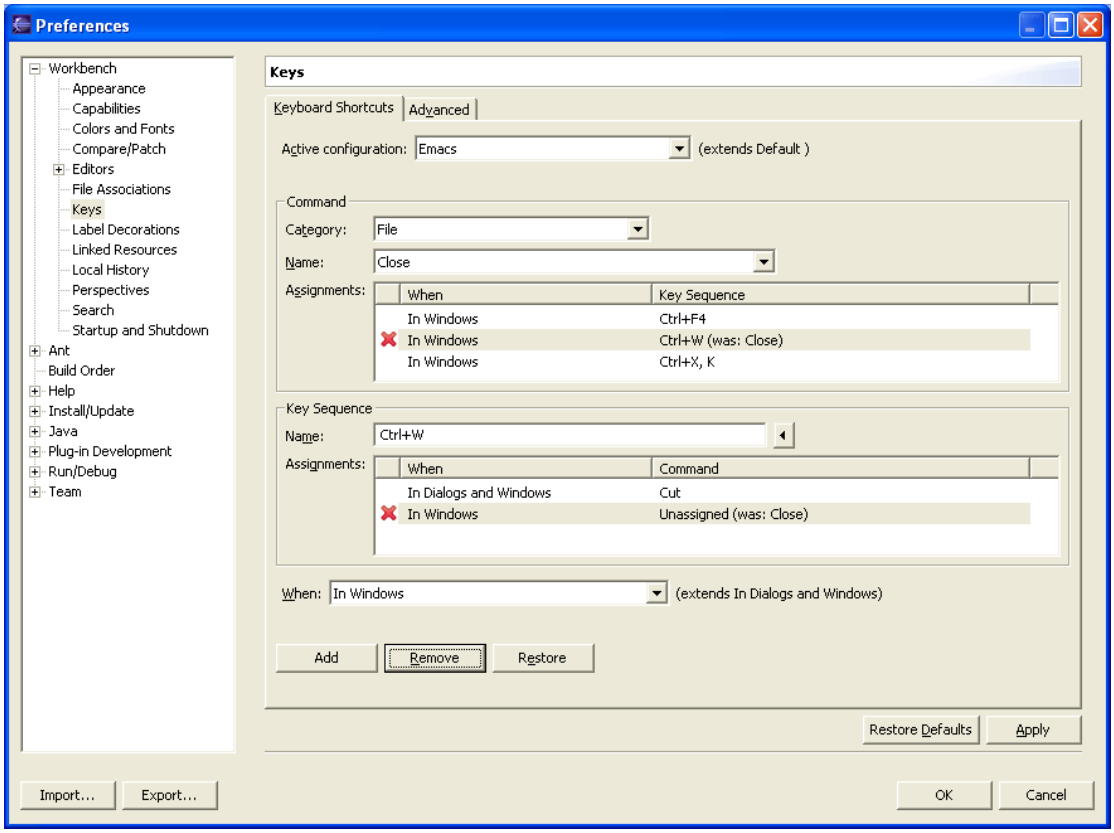


圖 3.15

假設在 *Emacs* 配置中已經選擇指定另一個按鍵給剪下（例如，Ctrl+Alt+W），以前一個方法來新增這個按鍵時，會產生下列結果。請注意，小型的「加號」圖形 表示使用者已新增按鍵連結，而這個按鍵連結之前並未指定：

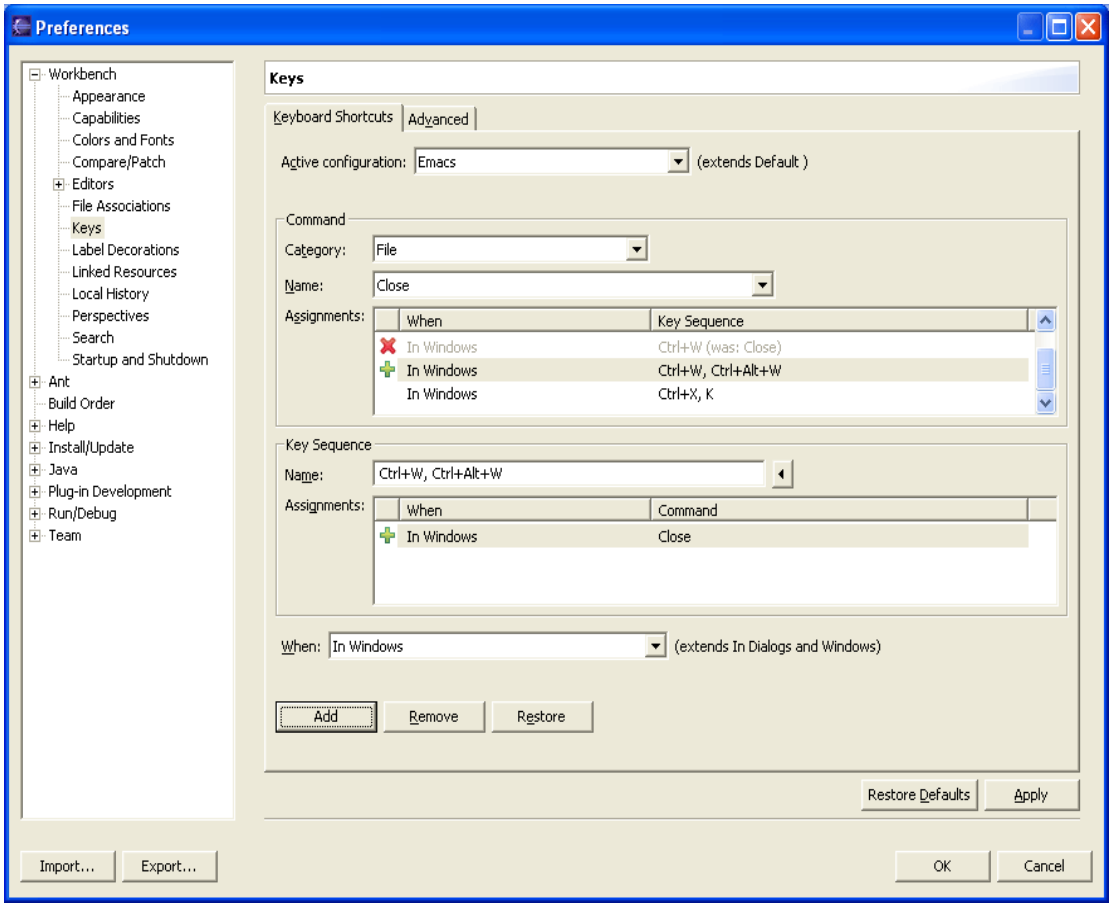


圖 3.16

按鍵連結的動態本質(The Dynamic Nature of Key Bindings)

按鍵連結是由外掛程式提供，而在 Eclipse 中，可以新增或移除外掛程式。這樣就能新增或移除由這些外掛程式所宣告的按鍵連結。Eclipse 在儲存自訂按鍵連結時，可以自動補償這個問題。比方說，在上面的範例中，在 *Emacs* 配置中，Ctrl+Alt+W 是指派給剪下。假設使用者安裝一個新的外掛程式，將 Ctrl+Alt+W 指定至特定指令。Eclipse 會將使用者的指派保留給剪下，但是會顯示有小型「變更」圖型的按鍵連結，而不會顯示含有「加號」圖型的按鍵連結。

衝突解決(Conflict Resolution)

只有少數簡單、常用的按鍵作用可以指派給多個指令。許多配置、環境定義、平台和語言環境的所有分割鍵順序在指派到網域中時，並沒有彼此衝突。如果環境定義不存在，請考量上述 Ctrl+B 的情況。有一個外掛程式將 Ctrl+B 指派給建置，則其他的外掛程式會將

Ctrl+B 指派給將文字變為粗體字。那麼 Eclipse 將如何正確地解決這個衝突呢？

雖然可藉由上述的機制來大量減少衝突，但衝突仍然可能發生。兩個相對獨立的外掛程式可以將相同按鍵順序指派給含相同環境定義、配置、平台和語言環境的不同指令。請考量如果外掛程式於*在視窗中*環境定義中指派了 Ctrl+F4，且將*預設*配置指派給它的其中一個指令。這會與將 Ctrl+F4 指派給相同環境定義和配置中之關閉指令的 Eclipse 造成直接衝突。

這就是衝突。同時呼叫兩個指令是不正確的，也不能只選擇其中一個指令來接收按鍵作用。唯一能做的，就是忽略這兩個按鍵連結，使 Ctrl+F4 在這個環境定義和配置中實際上沒有用。

「按鍵」喜好設定頁面顯示這個本質的衝突。請注意紅色的文字和 "[衝突]" 一字：

如果要解決這類衝突，使用者可以將按鍵順序明確指派給其中一個指令。

另一類的衝突可能是因為按鍵順序有多重按鍵作用。例如，在 *Emacs* 配置中，有許多多重按鍵作用的按鍵順序是以 Ctrl+X 的按鍵作用作為開頭。Ctrl+H K 是指派給關閉。Ctrl+X H 是指派給全選。

如同之前的說明，*Emacs* 配置會從*標準*配置借用按鍵連結。在標準配置中，Ctrl+X 是指派給剪下。雖然 *Emacs* 配置沒有明確重新定義 Ctrl+X，但是它的許多按鍵連結都需要按下 Ctrl+X。在 *Emacs* 配置中，按下 Ctrl+X 時，就等於要進入其中一個可能已經指定的按鍵順序。但我們並不希望在這時候呼叫剪下動作。

對於這類衝突，其規則是忽略已指派給剪下的 Ctrl+X。否則，就無法完成 *Emacs* 配置中的許多按鍵連結。

3.1.8 標籤裝飾(Label Decorations)

「標籤裝飾」可讓其餘資訊顯示在項目的標籤和圖示中。

「標籤裝飾」喜好設定頁面提供每一個裝飾的說明，並可以選擇要讓哪些裝飾看得到。

「標籤裝飾」喜好設定頁面看起來如下：

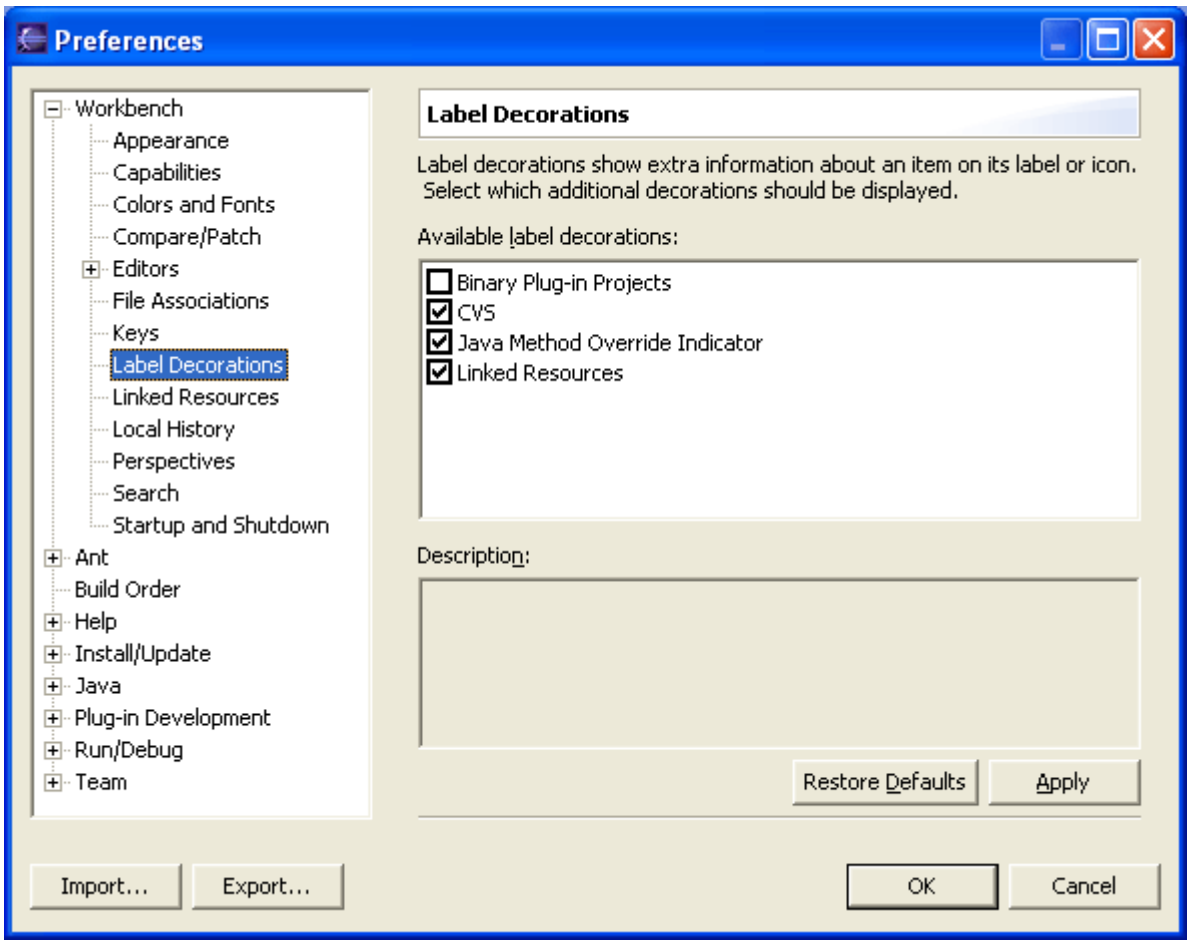


圖 3.17

3.1.9 鏈結資源(Linked Resources)

在使用鏈結資源時，會使用這個喜好設定頁面。Enable linked Resources 喜好設定是用來整體啟用或停用整個工作區的鏈結資源特性。依預設，鏈結資源是啟用的。如果停用鏈結資源，就無法建立任

何新的鏈結資源或匯入含有鏈結資源的現有專案。

並非所有的工作台版本都支援鏈結資源並且可將它們識別為鏈結資訊。如果打算與其他使用者共用工作區資料，可能不要使用鏈結資源。如果其他使用者無法使用鏈結資源，請停用這個喜好設定。

這個頁面的其他部分是用來定義在建立鏈結資源時所使用的路徑變數。請使用 New 按鈕來定義新的變數，也可以使用 Edit 按鈕來變更現有變數的值，或者使用 Remove 按鈕來移除現有的變數。請注意，如果變更的路徑變數正在使用中，就需要對這些專案執行本端重新整理，以 "探索" 檔案系統中是否有任何不同之處。可以開啟該項資源的「導覽器」快速功能表，然後選取 Refresh，來重新整理資源。建議不要移除目前正在使用的路徑變數。

「鏈結資源」喜好設定頁面的外觀如下：

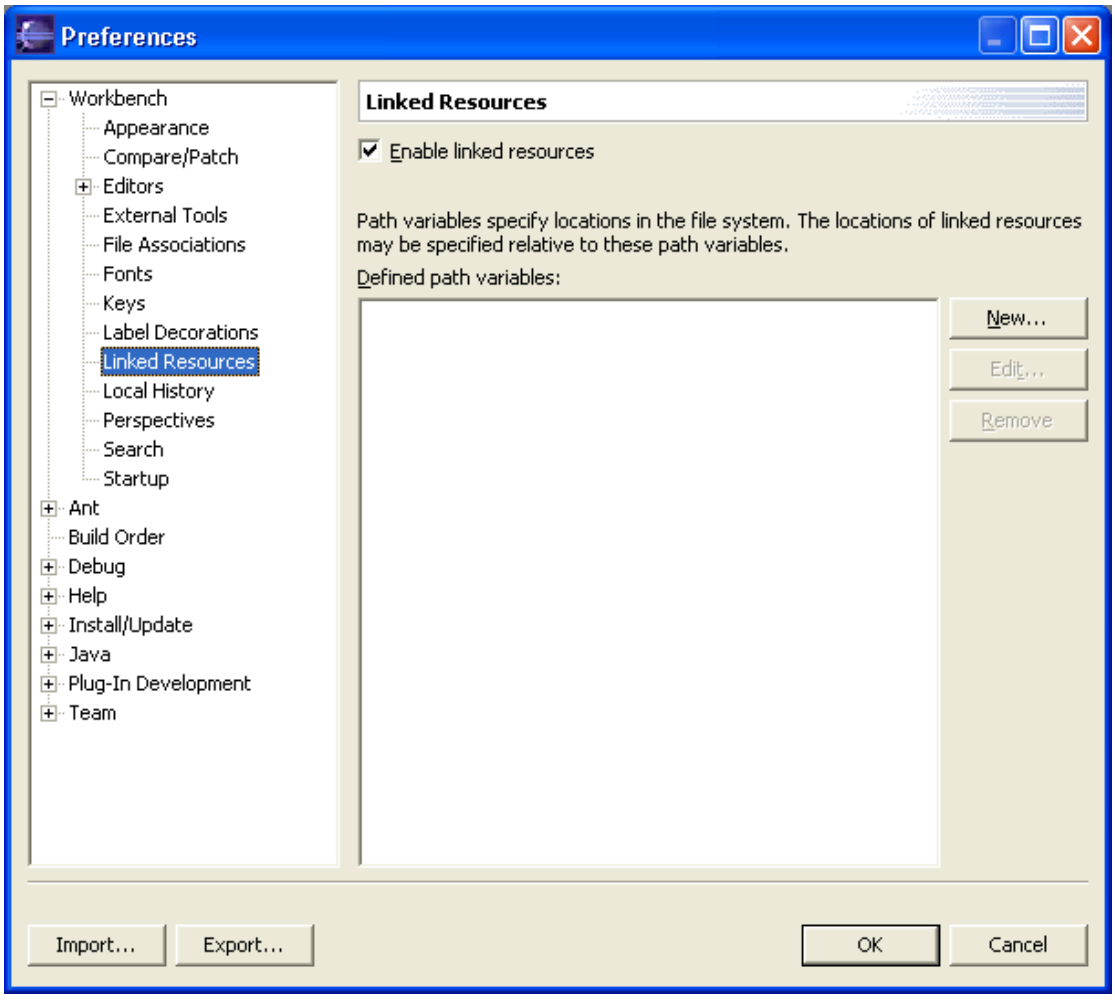


圖 3.18

3.1.10 歷史紀錄(Local History)

可以在「歷史紀錄」頁面中變更下列喜好設定。

選項	說明	預設值
Days To Keep Files(檔案的天數)	指出要在歷史紀錄中維護變更多少天。這個值以外的歷程狀態將會流失。	7 天
Entries Per File(檔案的項目數)	指出在歷史紀錄中每個檔案要維護多少歷程狀態。如果超過這個值，將會失去較舊的歷程，以挪出空間供新歷程使用。	50 個項目
Maximum File Size(MB)(最大檔案大小(MB))	指出歷程儲存庫中的個別狀態的大小上限。如果檔案超過這個大小，它將不會被儲存。	1 MB

「歷史紀錄」喜好設定頁面看起來如下：

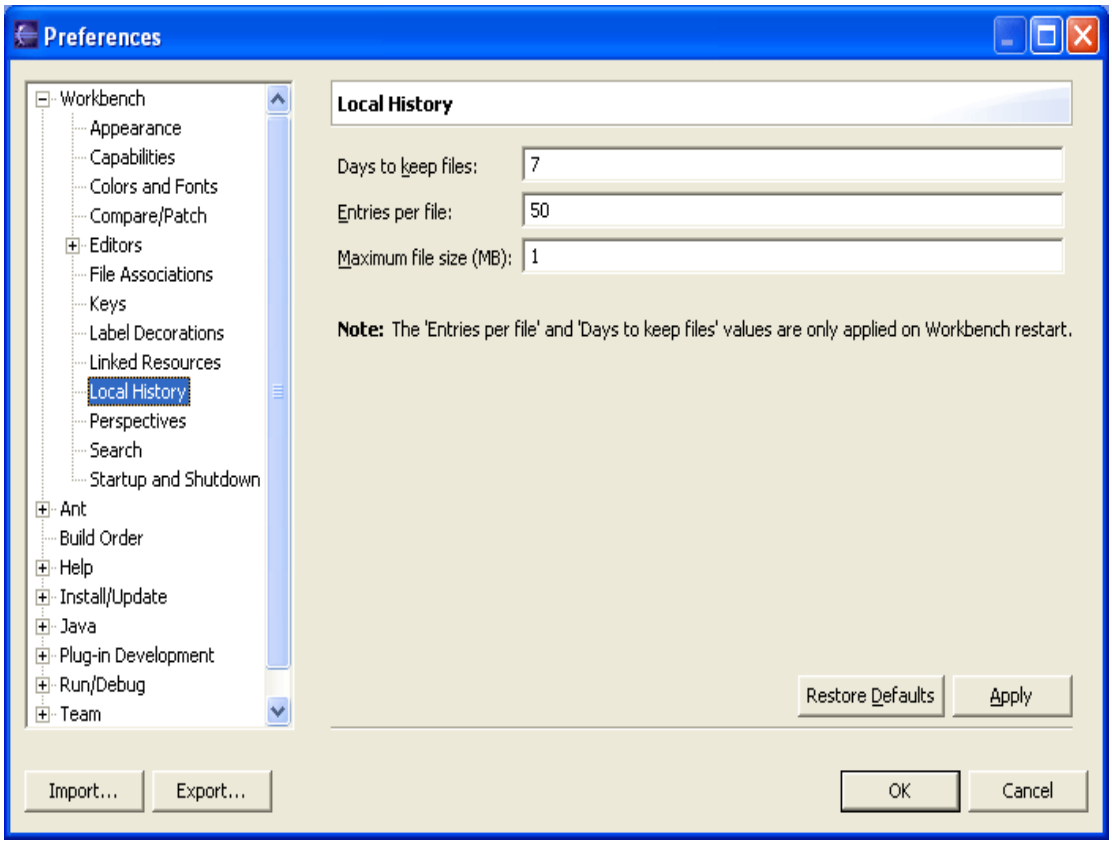


圖 3.19

3.1.11 視景

在「視景」喜好設定頁面上，可以管理各種定義於工作台的視景。

選項	說明	預設值
Open a new perspective(開啟新視景)	請使用這個選項來設定開啟新視景時會發生的情況。要在現行工作台視窗內或是新視窗內開啟視景？	在相同視窗中
Open a new view(開啟新視圖)	請使用這個選項來指定新的視圖開啟時會發生什麼情況。它會在其位於現行視景內的預設位置上開啟，或是開啟為快速視圖並定置到現行視景的側邊。	在視景內
New project options(新專案選項)	請使用這個選項來指定建立新專案時的視景行為。可以將它設定為：將現行視景切換至與專案類型相關聯的視景，並在同一個工作台視窗中開	在同一視窗開啟視景

選項	說明	預設值
	啟視景以作為現行視景、切換視景並在新的工作台視窗中加以開啟，或是完全不切換視景。	

可用的視景選項：

選項	說明	預設值
Make Default(設為預設值)	將所選取的視景設為預設視景。	資源
Reset(重設)	將選取視景的定義重設成預設配置。這個選項僅適用於已經使用「視窗」→「另存新視景...」來改寫的內建視景。	n/a
Delete(刪除)	刪除所選取的視景。這個選項僅適用於使用者定義的視景（無法刪除內建視景）。	n/a

「視景」喜好設定頁面看起來如下：

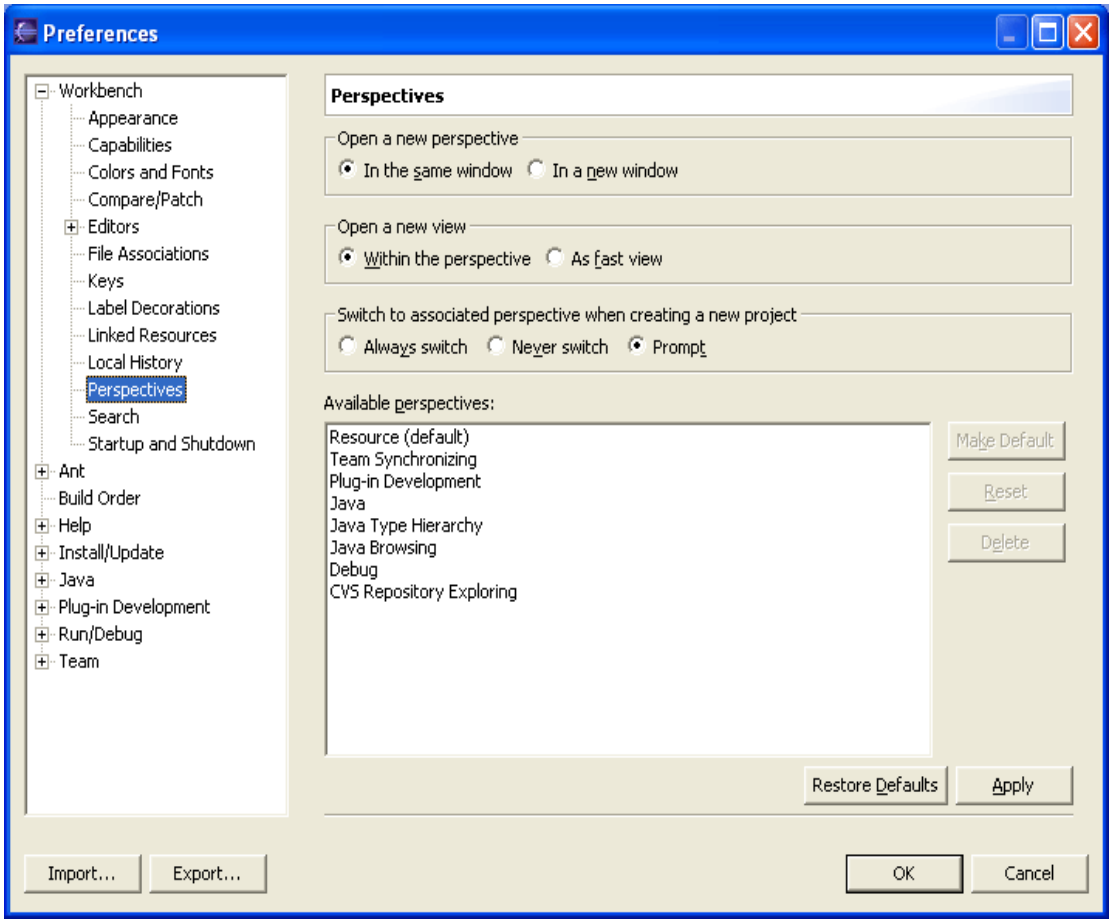


圖 3.20

3.1.12 搜尋(Search)

「搜尋」喜好設定頁面可讓使用者設定搜尋的喜好設定。Reuse editors to show matches(重複使用編輯器來顯示相符項目)選項可讓使用者繼續使用同一個編輯器來搜尋結果，以減少開啟的編輯器數目。Emphasize inexact matches(強調不完全相符)的項目是一個選項，可以在「搜尋」視圖中強調顯示這些項目。如果搜尋引擎不完全確定相符項目，則該相符項目會被視為不精確。也可以設定 Foreground color for inexact matches (不完全相符項目的前景顏色)。如果只要察看完全相符的項目，請勾選 Ignore inexact matches (忽略不完全相符的項目)。Default perspective for the Search view(視圖的預設視景)可以定義有新搜尋結果時要將哪一個視景移至最前面。

「搜尋」喜好設定頁面看起來如下：

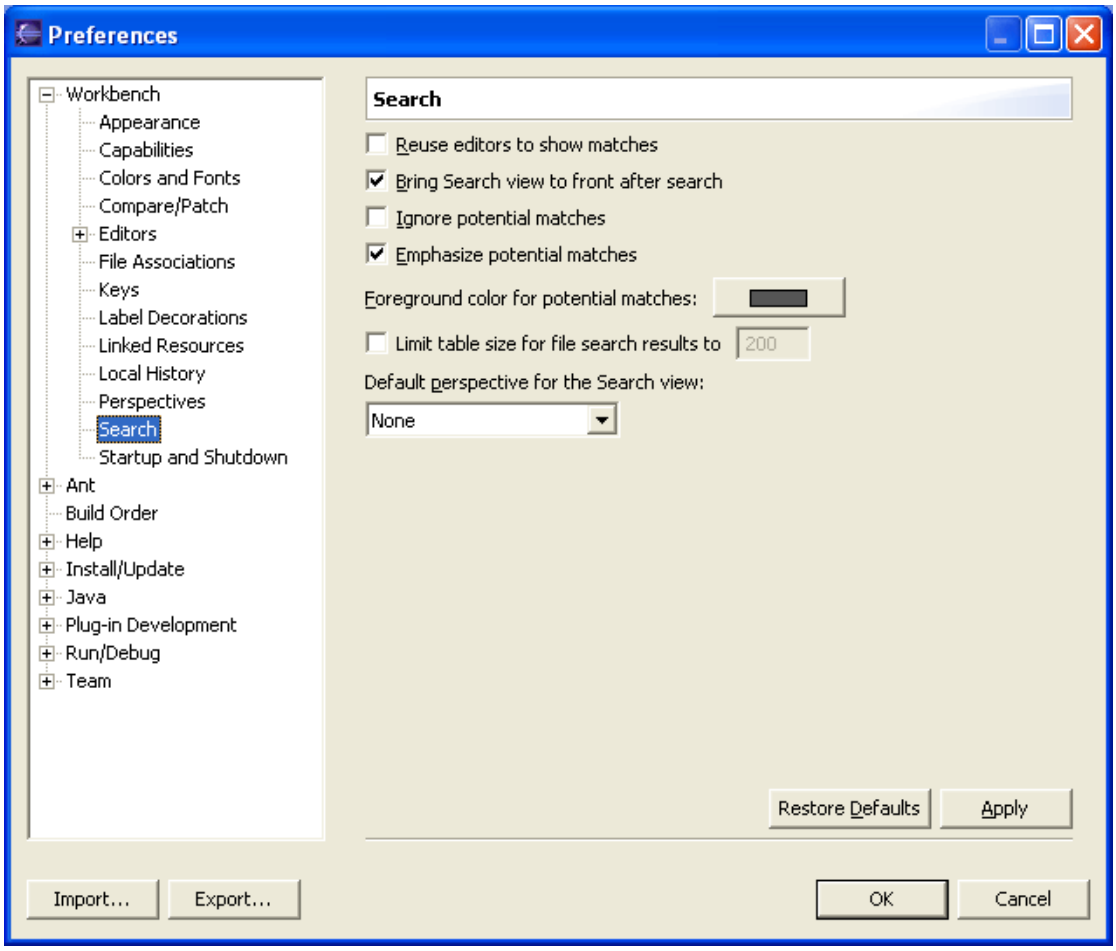


圖 3.21

3.1.13 啓動和關閉(Startup and Shutdown)

「啟動和關閉」喜好設定頁面可以選擇要在工作台啟動期間自動啟動的外掛程式。

一般而言，要到需要外掛程式時才會加以啟動。不過，某些外掛程式可能會指定它們希望在啟動期間被啟動。這個喜好設定頁面可以選擇在啟動期間將會實際啟動這其中的哪些外掛程式。

選項	說明	預設值
Prompt for workspace on startup(啟動時發出工作區	如果開啟這個選項，每次啟動工作台時，工作台都會提示使用者要用哪個工作區。	開啟

選項	說明	預設值
提示)		
Refresh workspace on startup(啟動時重新整理工作區)	如果開啟這個選項，在啟動時，工作台會與檔案系統同步化它的內容。	關閉
Confirm exit when closing last window(關閉最後一個視窗時確認結束)	如果開啟這個選項，當關閉最後一個視窗時，工作台都會問使用者是否要結束。	開啟

「啟動和關閉」喜好設定頁面看起來如下：

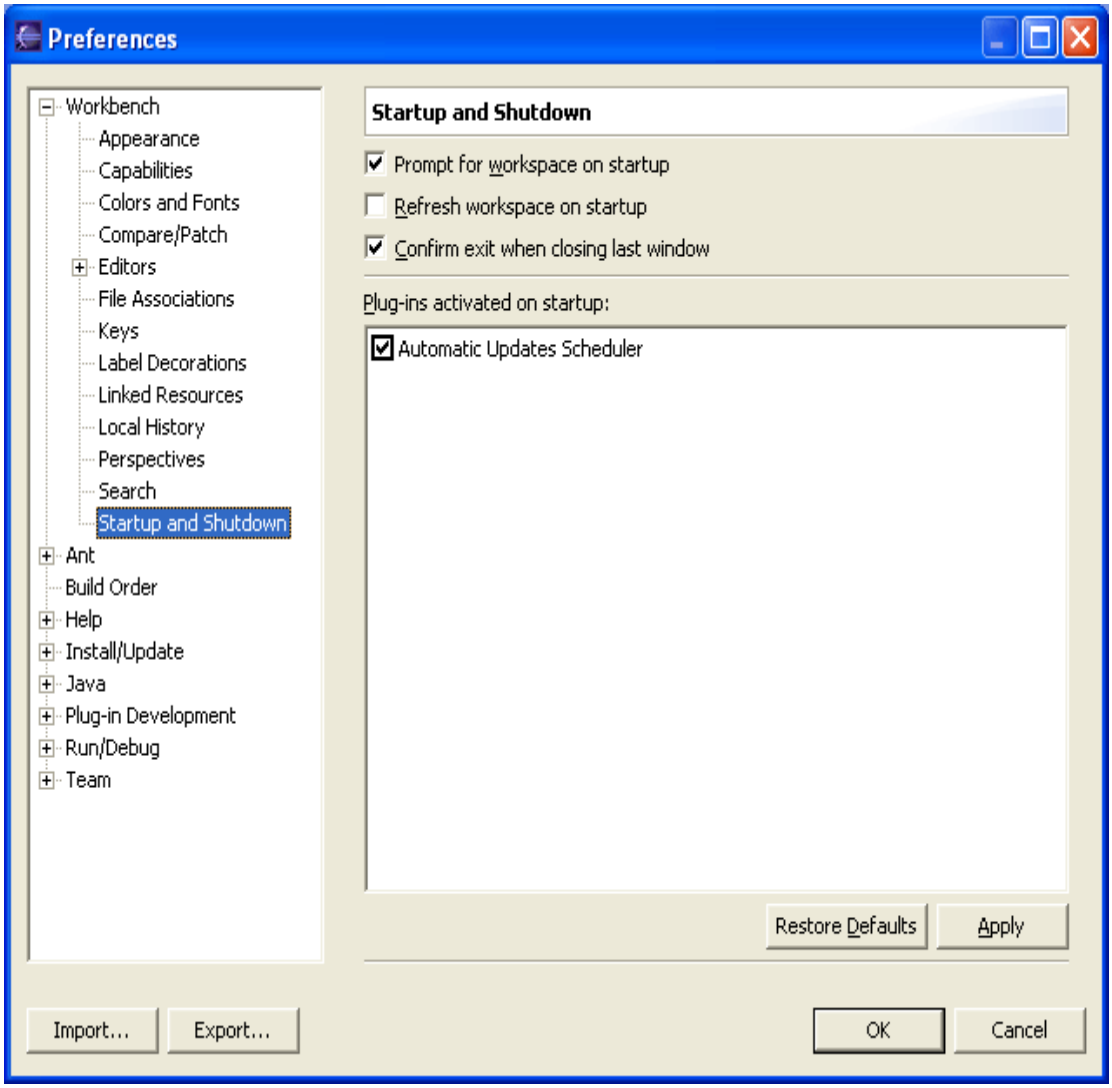


圖 3.22

3.2 Ant

可以在 Ant 頁面中變更下列喜好設定。

可以配置 Ant 的 "-find" 模擬要搜尋的建置檔。

也可以配置 Ant 建置輸出的顏色。

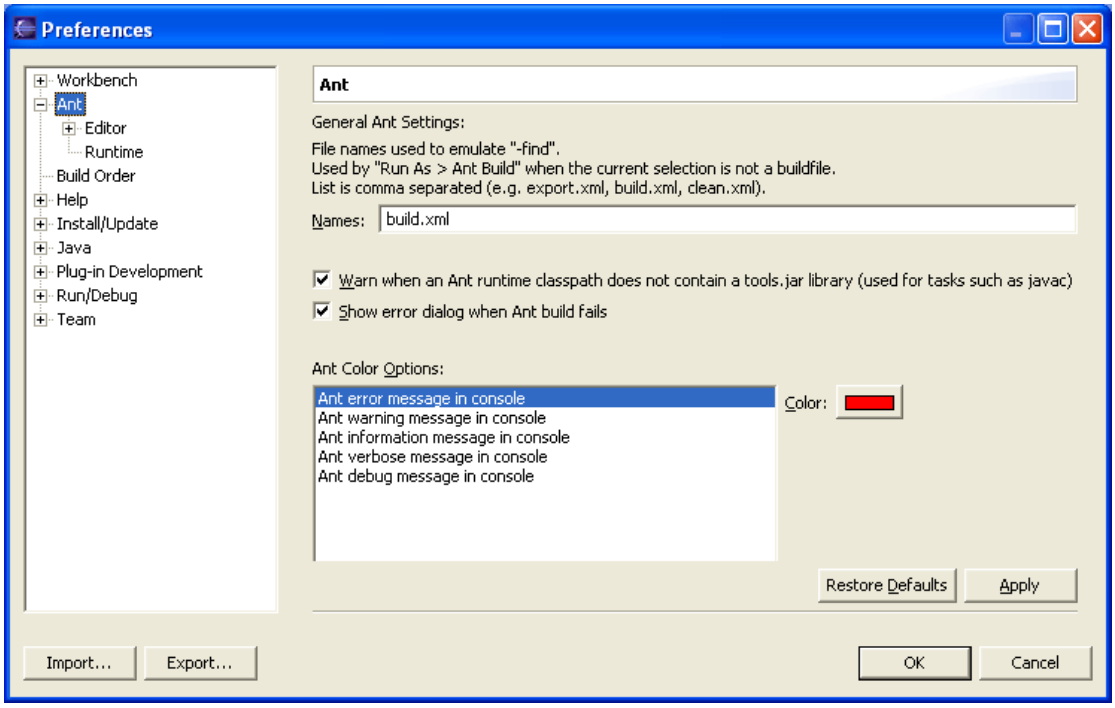


圖 3.23

3.2.1 Ant 編輯器(Ant Editor)

可以在「Ant 編輯器」頁面中變更下列喜好設定。

外觀選項(Appearance Options)

選項	說明	預設值
Print margin column(列印邊距直欄)	這個選項可以設定 Ant 編輯器的列印邊距。	80

選項	說明	預設值
Displayed tab width(顯示的欄標寬度)	這個選項用來控制在 Ant 編輯器中，要用多少空格來顯示欄標。	4
Insert spaces for tabs when typing(在輸入時插入欄標空格)	這個選項用來控制在 Ant 編輯器中輸入時，是否要用空格來取代欄標。	關閉
Show overview ruler(顯示概觀尺規)	這個選項用來控制 Ant 編輯器右側是否要顯示概觀尺規。	開啟
Show line numbers(顯示行號)	這個選項用來控制 Ant 編輯器左側是否要顯示行號。	關閉
Highlight current line(高亮度顯示現行行)	這個選項用來控制是否要強調顯示現行行。	開啟
Show print margin(顯示列印邊距)	這個選項控制是否可以看到列印邊距。	關閉
Appearance color options(外觀顏色選項)	這個選項控制各種外觀顏色。	

在外觀頁面中，提供 Ant 編輯器的外觀選項。

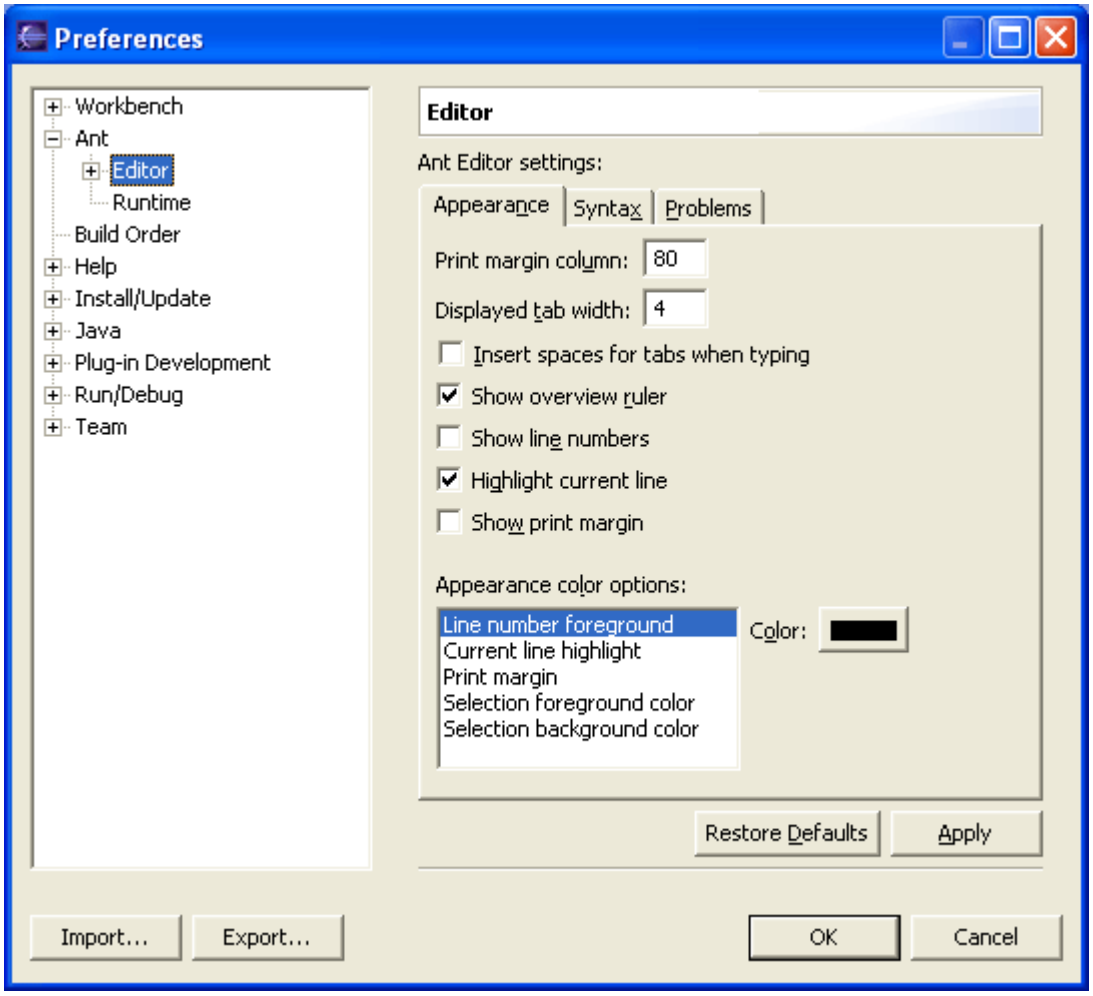


圖 3.24

3.2.2 Ant 執行時期(Ant Runtime)

在類別路徑頁面上，可以將定義作業和類型的其他類別新增至 Ant 類別路徑。

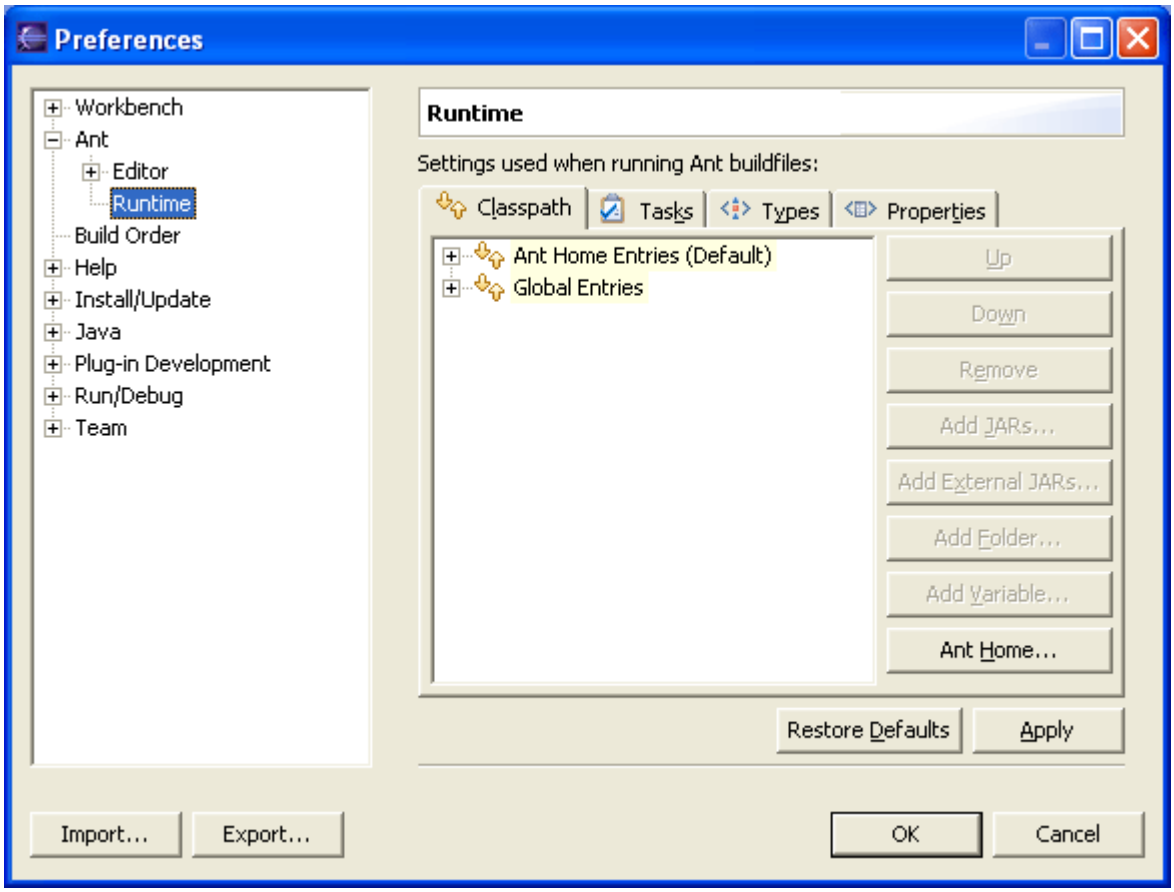


圖 3.25

在作業頁面上，可以新增定義於類別路徑上的其中一個類別的作業。

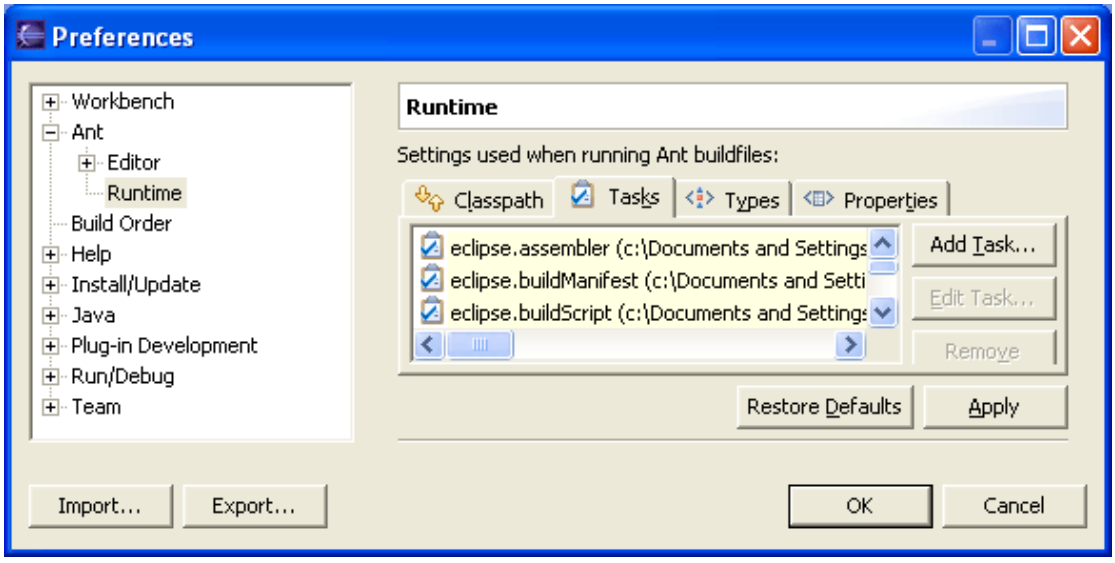


圖 3.26

在類型頁面上，可以新增定義於類別路徑上的其中一個類別的類

型。

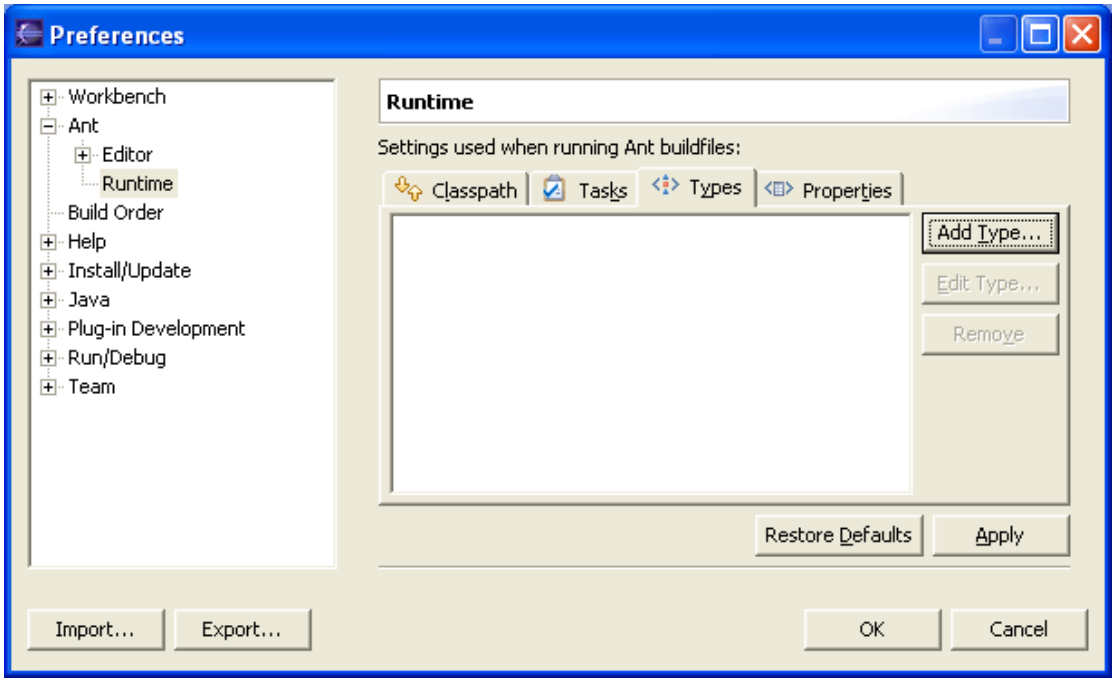


圖 3.27

在內容頁面中，可以新增要傳送到 Ant 的內容和內容檔。

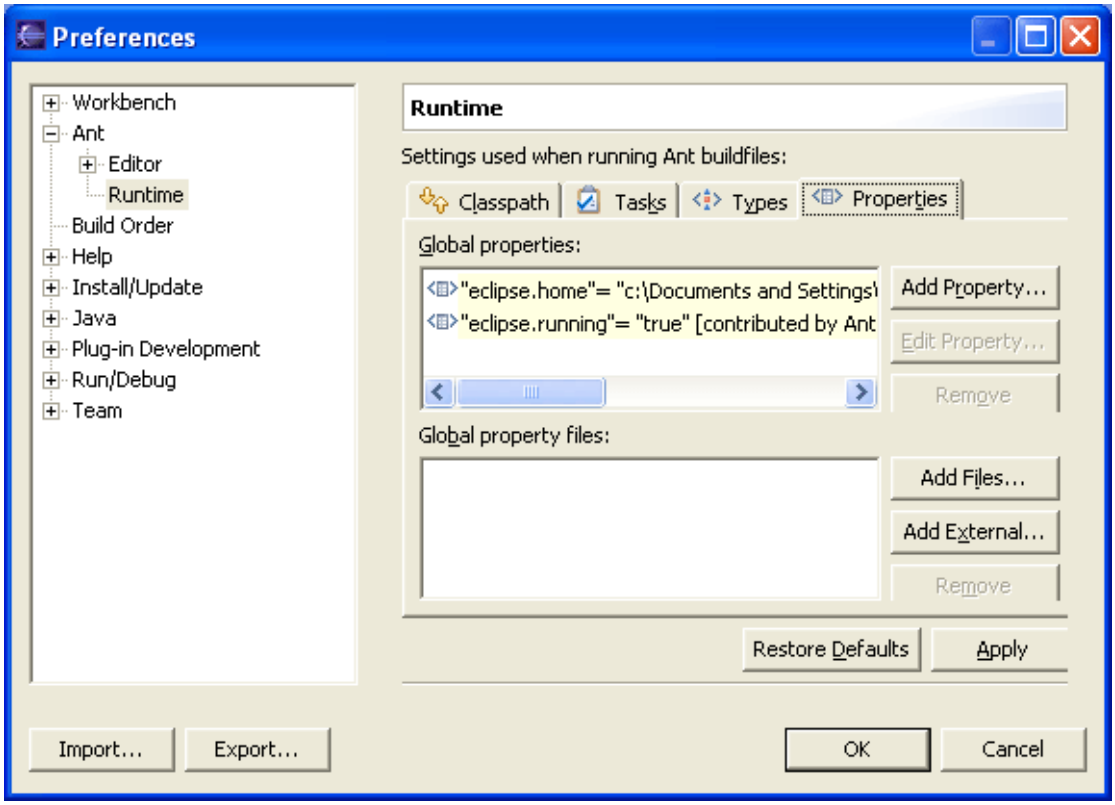


圖 3.28

3.3 建置次序(Build Order)

通常，專案建置的次序是很重要的。比方說，如果某個專案需要在另一個專案中定義的 Java 類別，則必須先建置第一個專案的必備類別，然後再建置第一個專案。工作台可以讓使用者明確地定義建置專案的次序。此外，使用者可以讓平台藉由將專案參照解譯為必備關係，來計算建置次序。在建置整個工作區或專案群組上，都會套用建置次序。

可以在「建置次序」喜好設定頁面中變更這個次序。一開始，*使用預設建置器次序*選項是開啟的，在此情況下，平台會計算建置次序。關閉這個選項時，就可以存取專案清單，而且可以操作專案的次序。請選取專案並使用上和下按鈕來變更建置次序。可以使用 Add Projec 及 Remove Project 按鈕，在建置次序中新增和移除專案。從建置次序中移除的專案*將*會被建置，但是會在建置次序中的所有專案都已建置完成之後才會建置。

在這個頁面底端，有一個喜好設定可用來處理包含循環的建置次序。在理想的狀況下，應該避免在專案之間使用循環參照。包含循環的專案在邏輯上仍屬於單一專案，所以它們會盡可能收合成為一個專案。然而，如果一定要有循環，建置次序可能需要好幾個疊代，才能正確地建置每一個項目。變更這個喜好設定時，會改變工作台嘗試在建置次序上進行疊代多少次之後，才會放棄。

「建置次序」喜好設定頁面看起來如下：

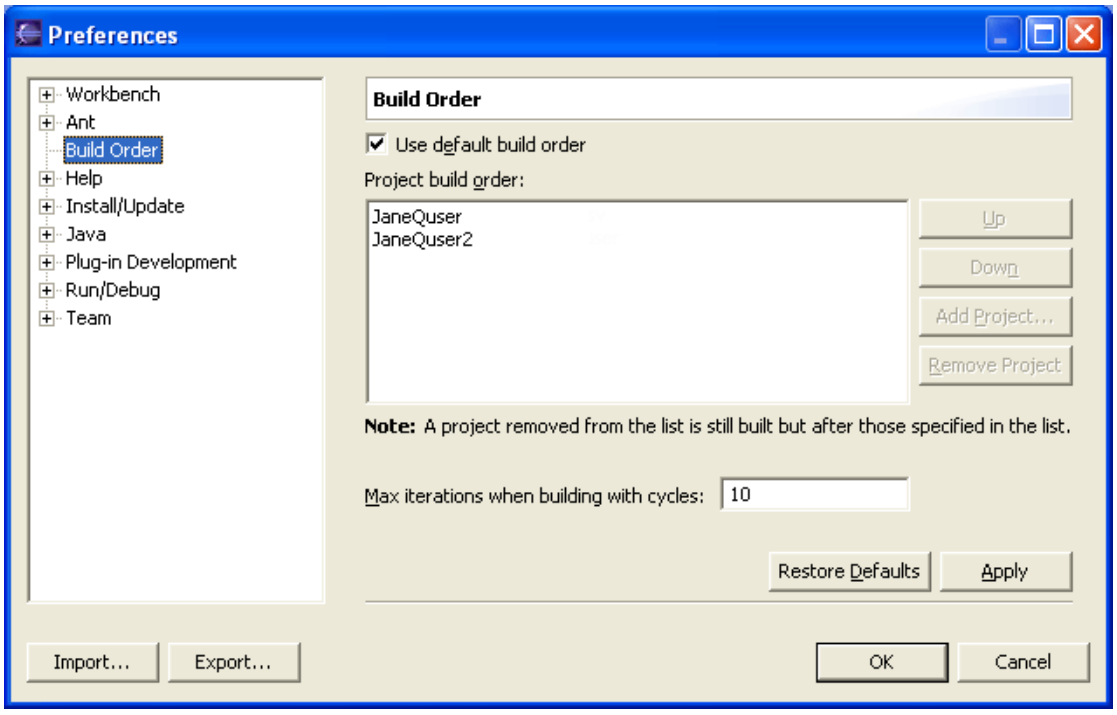


圖 3.29

3.4 說明(Help)

在「說明」喜好設定頁面上，可以指出如何顯示說明書籍。

選項	說明	預設值
Always use external browsers(固定使用外部瀏覽器)	如果的系統支援內嵌 Web 瀏覽器，可能的話，說明會利用內嵌說明瀏覽器來顯示說明，但仍可以使用這個選項。請選取它來強迫說明使用清單中的外部瀏覽器。	關閉
Current web browser adapter(現行 Web 瀏覽器配接器)	說明系統利用 Web 瀏覽器配接器，在外部瀏覽器中顯示線上說明。如果有多個配接器可以在的系統中開啟瀏覽器，就會使用選取的配接器來顯示說明。	視作業系統而定

Custom browser command(自訂瀏覽器指令)	如果從瀏覽器配接器清單中選取「自訂瀏覽器」，就會使用這個欄位來指定要啟動瀏覽器程式的指令。如果 URL 不是瀏覽器程式可以接受的最後一個參數，請使用 "%1" 字串來指出 URL 在指令中的位置。	"C:\Program Files\Internet Explorer\IEXPLORE.EXE" %1 - Windows，mozilla %1 - 其他平台（會顯示這個欄位）

「說明」喜好設定頁面看起來如下：

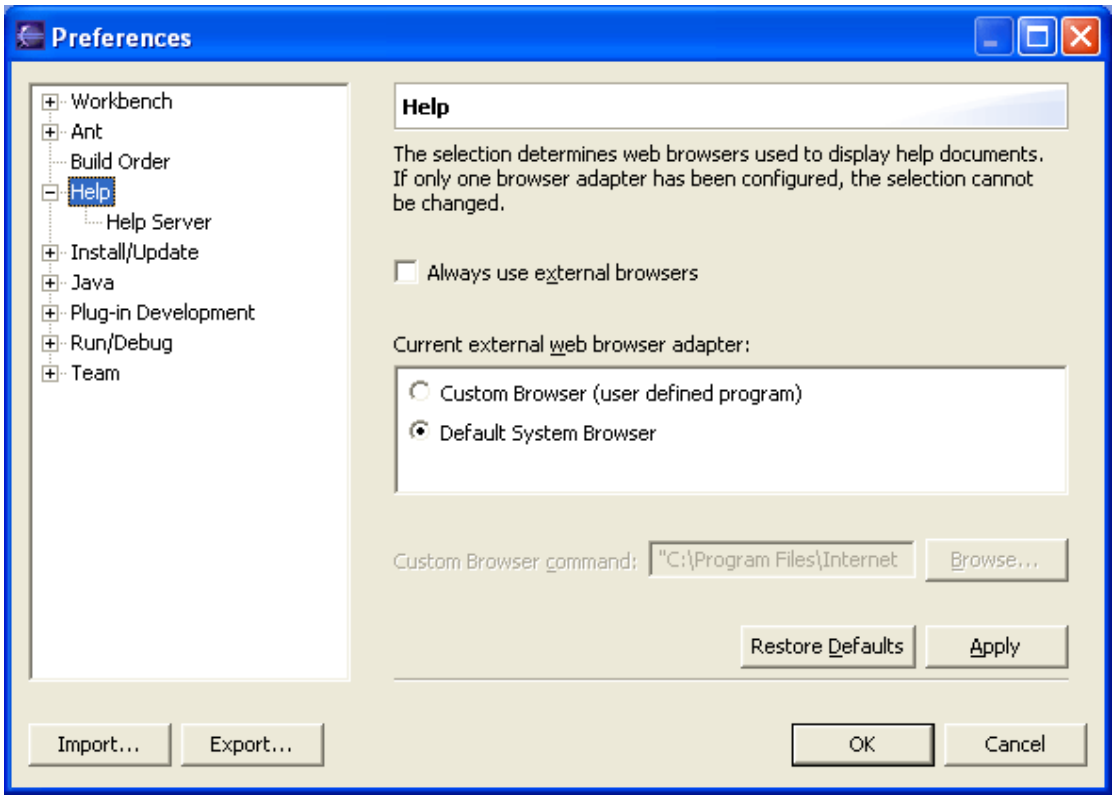


圖 3.30

附註：在這個頁面中所執行的選項會影響說明視圖的呈現方式。如果選取的瀏覽器沒有和 Internet Explorer 或 Mozilla 完全相容，或者已經停用 JavaScript，則瀏覽器中所顯示的說明視圖可能是簡化的版本。

3.4.1 說明伺服器(Help Server)

說明系統包含一個內部伺服器，可提供說明內容給瀏覽器。可以使用這個喜好設定頁面來變更伺服器所使用的介面和埠。只有當遇到問題且無法使用預設的喜好設定來檢視說明時，才應該變更這些設定。

選項	說明	預設值
Host(主機)	伺服器所使用的本端 IP 介面的名稱或位址。	空白
Port（埠）	伺服器要接聽的 IP 埠。如果將這個值指定成 0，就會由作業系統來指定這個埠。	0

「說明伺服器」喜好設定頁面的外觀如下：

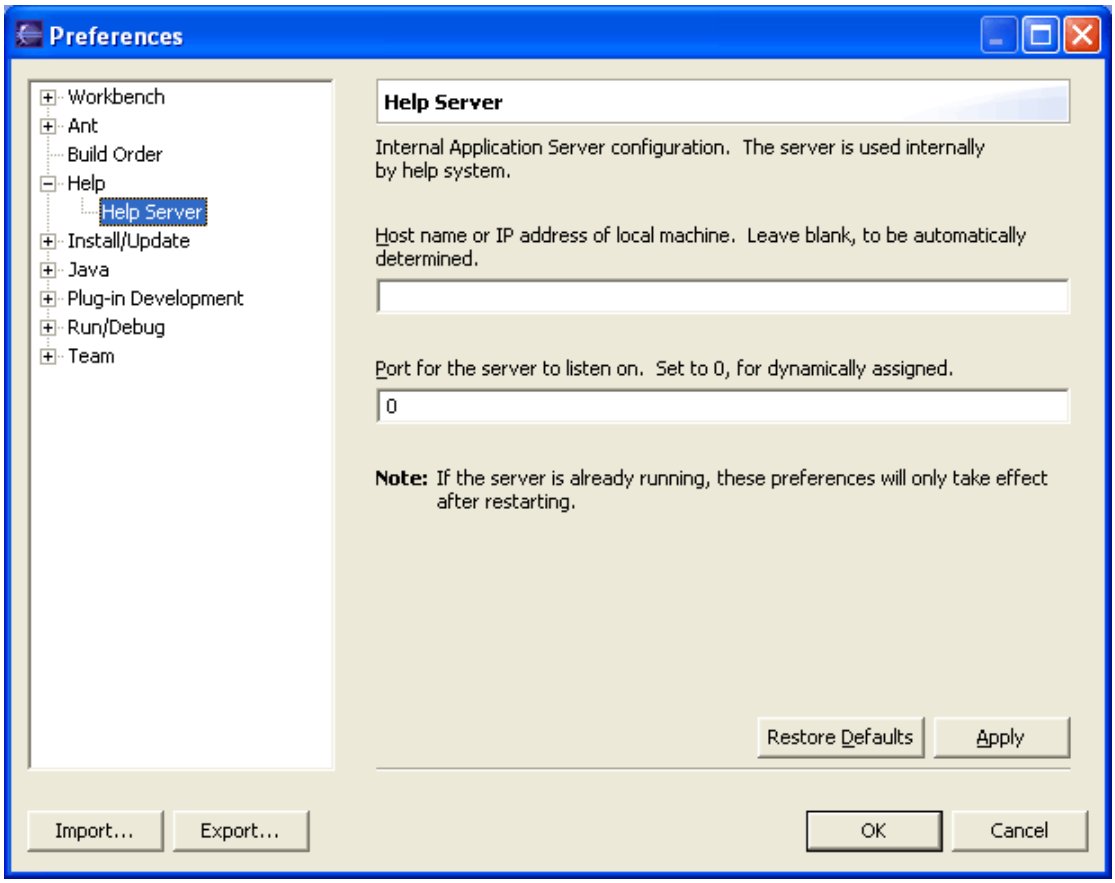


圖 3.31

3.5 自動更新(Install/Update)

可以在「自動更新」頁面中變更下列喜好設定：

選項	說明	預設值
Automatically search for updates and notify me（自動搜尋更新項目並通知我）	當選取時，更新管理程式會依照更新時程表所定義來自動搜尋更新項目	未選取
Update Schedule	每次啟動時尋找更新項目，或在每天或每週某天的某個預定時間尋找更新項目。	在啟動時

選項	說明	預設值
(更新時程表)		

3.6 Java

這個頁面可以指出在一般 Java 設定方面的喜好設定。

選項	說明	預設值
Update Java views(更新 Java 視圖)	**只在儲存時(On save only)：** 在儲存編譯單元前，不會更新「概要」視圖以外之所有視圖中顯示的 Java 元素。視圖會反映工作區的現行狀態，但不會顧及到工作副本。 **編輯時(While editing)：** 所有視圖中顯示的 Java 元素恆會反映工作區的現行狀態，包括工作副本。	編輯時
Action on double click in the Package Explorer(在「套件瀏覽器」中按兩下後的動作)	**進入所選元素(Go into the selected element)：** 當按兩下儲存器時，就會執行 Go Into 指令。請從導覽功能表查看 Go Into。 **展開所選元素(Expand the selected element)：** 當按兩下儲存器時，會將之展開，並顯示其子項。	展開所選取的元素
When opening a Type Hierarchy(當開啟類型階層時)	**開啟新的「類型階層」視景(Open a new Type Hierarchy Perspective)** 只要一開啟「類型階層」視圖，即開啟新的「類型階層」視景。 **在現行視景中顯示「類型階層」視圖(Show the Type Hierarchy View in the current perspective)** 在現行視景中顯示「類型階層」視圖。	在現行視景中顯示「類型階層」視圖

選項	說明	預設值
	附註：在「工作台」喜好設定頁面中，可以選擇要在新視窗或現行視窗中開啟新視景，或者以新視景取代現有的視景。	

3.6.1 外觀(Appearance)

在這個喜好設定頁面上，可以配置視圖中之 Java 元素的外觀。選項如下：

選項	說明	預設值
Show method return types(顯示方法傳回類型)	當啟用時，視圖中的方法會顯示傳回類型。	關閉
Show override indicators in outline and hierarchy(以概要及階層顯示置換指示器)	當啟用時，則在「概要」與「類型階層」視圖中，會針對已置換與實作的方法顯示一個指示器。	開啟
Show members in Package Explorer(在套件瀏覽器中顯示成員)	當啟用時，亦會顯示 Java 檔與類別檔層次下的 Java 元素。	開啟
Compress package name segments(壓縮套件名稱區段)	當啟用時，則會根據壓縮型樣壓縮套件名稱。	關閉

選項	說明	預設值
Stack views vertically in the Java Browsing perspective(垂直堆疊「Java 瀏覽」視景中的視圖)	當啟用時，則「Java 瀏覽」視景中的視圖會採垂直（而非水平）方式堆疊。	關閉

3.6.2 類別路徑變數(Classpath variables)

可配置的變數(Configurable variables)

可在「Java 建置路徑」中使用類別路徑變數，以免參照本端檔案系統。當使用變數項目時，類別路徑中只含有一個變數，且建置路徑可供整個團隊共用。必須在這個頁面中配置變數的值。

指令	說明
New...(新建...)	新增變數項目。在產生的對話框中，指定新變數的名稱和路徑。可以按一下 File 或 Folder 按鈕來瀏覽以找出路徑。
Edit...(編輯...)	可讓編輯所選取的變數項目。在產生的對話框中，編輯該變數的名稱與（或）路徑。可以按一下 File 或 Folder 按鈕來瀏覽以找出路徑。
Remove(移除)	移除選取的變數項目。

保留類別路徑變數(Reserved class path variables)

某些類別路徑變數會設定在內部，且無法在「類別路徑變數」喜好設定中變更：

- JRE_LIB：這個保存檔中含有目前所用之 JRE 的執行時期 JAR 檔。
- JRE_SRC：為目前所用 JRE 的程式檔保存檔。
- JRE_SRCROOT：為目前所用 JRE 之程式檔保存檔中的根路徑。

3.6.3 程式碼格式製作器(Code Formatter)

這個頁面中的預覽窗格會示範這些選項對編輯器中之 Java 程式碼所產生的結果。

選項	說明	預設值
Insert a new line before an opening brace(在左大括弧前插入新行)	編輯器在新的左大括弧前插入換行符號。換句話說，左大括弧恆起自新行的開頭。	關閉
Insert new lines in control statements(在控制陳述式中插入新行)	編輯器在新的控制陳述式前插入一行。換句話說，控制陳述式（如：if、else、catch、finally 等）恆起自新行的開頭。	關閉
Clear all blank lines(清除所有空白行)	編輯器刪除檔案中的所有空白行。	關閉
Insert new line between 'else if'(在'else if'之間插入新行)	編輯器在 else-if 陳述式中之 "else" 與 "if" 字之間插入一行新行。	關閉

選項	說明	預設值
Insert a new line inside an empty block(在空區塊內插入新行)	編輯器在空大括弧間插入一個換行。換句話說，在一個空大括弧組中的左右大括弧會出現在不同行。一個例外情況是，如果右大括弧後跟著一個關鍵字，則兩個大括弧會出現在同一行。	開啟
Maximum line length(行長度上限)	這是任何單行的最大長度。當字行超過這個長度時，則會折行。如果輸入 0，則完全停用折行特性。	80
Compact assignment(精簡指派)	編輯器會移除變數與指派陳述式間的任何空格，使其不對稱（如 a= b;）。	關閉
Insert a space after a cast(在強制轉型後插入空格)	編輯器會在強制轉型與下列表示式之間插入一個空格。	開啟
Insert tabs for indentation, not spaces(插入 tab（而非空格）以內縮)	編輯器使用 tab（而非空格）來呈現內縮。	開啟
Number of spaces representing an indentation level(代表內縮層次的空格數目)	如果編輯器是使用空格而非 tab 來呈現內縮，則這是指出一個內縮是由多少空格組成。	4
Preview pane(「預覽」	使用這個頁面目前所示的設定值，以範例顯示 Java 程式碼的模樣。	n/a

選項	說明	預設值
窗格)		

3.6.4 程式碼產生(Code generation)

程式碼產生喜好設定分成兩個區段：

- 名稱
- 程式碼和註解

名稱(Names)

這個頁面定義欄位（Static 和非 Static)、參數和區域變數的命名慣例。對於每一個變數類型，有可能配置字首或字尾清單，或兩者的清單。

產生 Getter 和 Setter 動作，以及所有建立欄位、參數和區域變數的動作和「快速修正」提議，都會使用命名慣例。

動作	說明
Edit...(編輯...)	開啟一個對話框，編輯目前選取之變數類型的字首和字尾清單

程式碼和註解(Code and Comments)

程式碼和註解頁面含有產生程式碼之動作所使用的程式碼範本。範本含有當套用範本時將替代的變數。某些變數可用在所有範本中，某些變數則是範本特有的。

動作	說明
Edit...(編輯...)	開啟一個對話框，編輯目前選取的程式碼範本。
Import...(匯入...)	從檔案系統匯入程式碼範本。

動作	說明
Export...(匯出...)	匯出所有選取的程式碼範本至檔案系統。
Export All...(匯出全部...)	匯出所有程式碼範本至檔案系統。
Automatically add comments for new methods and types(自動新增方法和類型的註解)	這個設定指定註解程式碼範本是否會自動新增至所有新的方法。如果停用，僅在明確地新增註解(如使用新增 Javadoc 註解動作）時，才會使用註解程式碼範本。請注意，這個設定並不套用至程式碼範本（如新建 Java 檔案）中所含的註解

註解範本(Comment templates)

註解範本可包含 ${tags} 變數，這個變數將被已加註元素的標準 Javadoc 標示 (@param, @return..) 所替代。此外，「置換方法」註解可包含 ${see_to_overridden} 範本

- 建構子註解：指定新建構子註解的範本
- 類型註解：指定新類型註解的範本。請注意，可在「新建 Java 檔」範本中參照這個範本
- 方法註解：指定新方法（不置換基礎類別中的方法）註解的範本
- 置換方法註解：指定新方法（置換基礎類別中的方法）註解的範本。依預設，註解會定義成非 Javadoc 註解(Javadoc 將把這個註解換成已置換方法的註解)。如果想要的話，可以將這個註解變更為真實的 Javadoc 註解

新建 Java 檔範本(New Java files template)

當建立新檔案時，「類別」和「介面」精靈就會使用「新建 Java 檔」範本。範本可以指定要新增註解之處。請注意，範本可以含有

${typecomment} 變數，這個變數將被類型註解範本的評估值所替代。

Catch 區塊主體範本(Catch block body template)

當建立 catch 區塊主體時，就會使用「Catch 區塊主體」範本。它可以使用 ${exception_type} 和 ${exception_var} 變數。

方法主體範本(Method body template)

當建立含有主體的新方法時，就會使用「方法主體」範本。它含有解析為 return 陳述式或/和 super 呼叫的 ${body_statement} 變數。

建構子主體範本(Constructor body templates)

當建立含有主體的新方法或建構子時，就會使用「建構子主體」範本。它含有解析 super 呼叫的 ${body_statement} 變數。

「程式碼範本」對話框(Code Template dialog)

對話框中的欄位與按鈕如下：

選項	說明
Description(說明)	範本的說明
Pattern(型樣)	範本的型樣。
Insert Variables...(插入變數...)	顯示預先定義之範本特有變數的清單。

3.6.5 編譯器(Compiler)

下列各段將分別說明編譯器的喜好設定：

- 問題
- 樣式
- 相容和類別檔

■ 建置路徑

問題(Problems)

選項	說明	預設值
Unreachable code(無法呼叫到的程式碼)	無法呼叫到程式碼，可選擇性地報告成錯誤、警告，或者加以忽略。位元組碼一律產生最佳化程式碼。請注意，根據 Java 語言規格,無法呼叫到的程式碼應該是一個錯誤。	錯誤
Unresolvable import statements(無法解析的 import 陳述式)	無法解析的 import 陳述式可選擇性地報告成錯誤、警告，或加以忽略。請注意，根據 Java 語言規格,無法解析的 import 陳述式應該是一個錯誤。	錯誤
Unused local variables (i.e. never read)(未使用的區域變數（如從未讀取）)	當啟用時,編譯器會針對未用的區域變數(亦即：從未讀取的變數)，發出錯誤或警告。	忽略
Unused parameters (i.e. never read)(未使用的參數（如從未讀取）)	當啟用時,編譯器會針對未用的方法參數(亦即：從未讀取的參數)，發出錯誤或警告。	忽略
Unused imports(未用的匯入)	當啟用時，編譯器會針對未用的匯入參照，發出錯誤或警告。	警告
Unused private types, methods or fields (未用的 private 類型、方法或欄位)	當啟用時，每當宣告 Private 方法或欄位時，但從未在同一單元內使用時，編譯器將發出錯誤或警告。	忽略
Usage of non-externalized strings(未提出字串	當啟用時，編譯器將為未提出的字串文字發出錯誤或警告（如，未標示的 //$NON-NLS-<n>$)。	忽略

的用法)		
Usage of deprecated API(已停用的 API 的用法)	當啟用這個選項時，編譯器會將使用已停用的 API 標為錯誤或警告。	警告
Signal use of deprecated API inside deprecated code(已停用的程式碼內之已停用的 API 的信號使用)	一旦啟用，編譯器將發出信號，指出在已停用的程式碼內使用已停用的 API。問題的嚴重性是由「已停用的 API 的用法」選項來控制。	關閉
Maximum number of problems reported per compilation unit(各編譯單元所能報告的問題數上限)	指定各編譯單元所能報告的問題數上限。	100

樣式(Style)

選項	說明	預設值
Methods overridden but not package visible(已置換但套件看不到的方法)	套件的預設方法在另一套件中看不到，因此無法置換。當啟用這個選項時，編譯器會將這類情況標為錯誤或警告。	警告
Methods with a constructor name(建構子名稱中的方法)	如果以建構子名稱來命名方法，通常會被視為較差的程式設計風格。當啟用這個選項時，編譯器會將這類情況標為錯誤或警告。	警告
Conflict of interface	當啟用時，每當介面定義一個與非繼承「物件」方法不相容的方法時，編譯器將發出錯誤或警	警告

method with protected 'Object' method(介面方法與受保護的「物件」方法發生衝突)	告。直到解決這個衝突之前，將無法實作如此的介面，如
	interface I {
	int clone();
	}	

Hidden catch blocks(隱藏的 catch 區塊)	在本端環境下對於 try 陳述式而言，某些 catch 區塊可能會隱藏其他者，例如：
	try { throw new java.io.CharConversionException();
	} catch (java.io.CharConversionException e) {
	} catch (java.io.IOException e) {}.
	當啟用這個選項時，編譯器會針對對應至所檢查之異常狀況的快取區塊隱藏，發出錯誤或警告。
		警告

Non-static access to a static member(Static 成員的非 Static 存取權)	當啟用時，每當以表示式接收器存取 Static 欄位或方法時，編譯器將發出錯誤或警告。應該以類型名稱限定 Static 成員的參照。	警告
Access to a non-accessible member of an enclosing type(存取含括類型中無法存取的成員)	當啟用時，只要其模擬存取含括類型中無法存取的成員，編譯器即會發出錯誤或警告。這類存取可能涉及效能。	忽略
Assignment has no effect (e.g. 'x = x')(指定沒	當啟用時，每當指派沒有效果(如 'x = x')時，編譯器將發出錯誤或警告。	警告

有生效（例如 'x=x'））		
Using a char array in string concatenation(在字串連結中使用 char 陣列)	當啟用時，每當在下列「字串」連結中使用 char[] 表示式時，編譯器就會發出錯誤或警告："hello" + new char[]{'w','o','r','l','d'}	警告

相容和類別檔(Compliance and Class files)

選項	說明	預設值
Compiler compliance level(編譯器相容層次)	指定 JDK 編譯器相容層次。	1.3
Use default compliance settings(使用預設相容設定)	當啟用時，在編譯器的相容層次方面，會套用預設的相容設定。	開啟
Generated class files compatibility(所產生的類別檔相容性)	指定所產生的類別檔相容性。	1.1
Source compatibility(程式檔相容性)	指定程式檔和 1.3 或 1.4 相容。從 1.4 開始，"assert" 為保留給主張支援的關鍵字。	1.3
Report 'assert' as identifier(將 'assert'報告成識別碼)	當啟用時，只要 'assert'（為 JDK 1.4 中的保留關鍵字）被當成識別碼使用，編譯器即會發出錯誤或警告。	忽略

Add variable attributes to generated class files(新增變數屬性到產生的類別檔中)	當啟用時，會在類別檔中新增變數屬性。這會讓區域變數名稱顯示在除錯器中(位於明確指定變數之處)。產生的 .class 檔會變大。	開啟
Add line number attributes to generated class files(新增行號屬性到產生的類別檔中)	當啟用時，會在類別檔中新增行號資訊。這會在除錯器中強調顯示出程式碼。	開啟
Add source file name to generated class file(新增程式檔名稱到產生的類別檔中)	當啟用時，會在類別檔中新增程式檔名稱。這會讓除錯器顯示對應的程式碼。	開啟
Preserve unused local variables(保留未用的區域變數)	當啟用時，則不會將未用的區域變數(亦即，從未讀取)從類別檔中除去。如果除去這項，有可能會改變除錯。	開啟

建置路徑(Build Path)

選項	說明	預設值
Incomplete build path(不完整的建置路徑)	指出當類別路徑上的項目不存在、不合規定或看不見(如關閉了參照專案)時，所報告的問題的嚴重性。	錯誤
Circular dependencies(循環相依項)	指出在循環中併入專案時所報告的問題的嚴重性。	錯誤

Duplicated resources(重複的資源)	指出當多次出現的資源將複製到輸出位置時所報告的問題的嚴重性。	警告
Abort building on build path errors(當建置路徑錯誤時中止建置)	容許如果類別路徑無效，將建置器切換至中止。	開啟
Scrub output folders on full build(進行完整建置時清除輸出資料夾)	指出是否容許「Java 建置器」在執行完整建置作業時清除輸出資料夾。	開啟
Enable using exclusion patterns in source folders(啟用在來源資料夾中使用排除型樣)	當停用時，專案類別路徑上沒有項目可與排除型樣相關聯。	開啟
Enable using multiple output locations for source folders(啟用對來源資料夾使用多個輸出位置)	當停用時，專案類別路徑上沒有項目可與特定輸出位置相關聯，因而防止使用多個輸出位置。	開啟
Filtered resources(過濾的資源)	以逗點分格方式列出不複製到輸出資料夾中的檔案型樣。	''

3.6.6 Java 編輯器(Java editor)

這個頁面可以設定如下的 Java 編輯器喜好設定：

- 外觀
- 語法
- 程式碼協助
- 問題指示

外觀(Appearance)

外觀指定 Java 編輯器的外觀。

選項	說明	預設值
Displayed tab width(Tab 的顯示寬度)	指定 Tab 的顯示寬度（以空格為單位）。	4
Print margin column(列印邊距直欄)	指定其後要顯示列印邊距的直欄。 如果要顯示列印邊距，請啟 Show print margin 選項；如果要指定列印邊距的顏色，請使用 Appearance color options 喜好設定。	80
Synchronize outline selection on cursor move(在游標移動時將概要選項同步化)	當啟用時，「概要」視圖恆會在 Java 編輯器中選取含括游標的 Java 元素。	關閉
Show overview ruler(顯示概觀尺規)	當啟用時，Java 編輯器右邊框中會出現概觀尺規，並顯示整個可視文件的問題。	開啟
Show line numbers(顯示行號)	當啟用時，Java 編輯器左邊框中的垂直尺規會顯示可視文件的行號。 可在 Appearance color options 中指定行號的	關閉

選項	說明	預設值
	顏色。	
Highlight matching brackets(強調顯示對稱的括弧)	當啟用時，只要游標是位於小括弧、方括弧或大括弧旁，即會強調顯示其相對的左或右括弧。 可在 Appearance color options 中指定括弧的強調顯示顏色。	開啟
Highlight current line(強調顯示現行行)	當啟用時，則會強調顯示游標所在現行行的背景。 可在 Appearance color options 中指定現行行的背景顏色。	開啟
Show print margin(顯示列印邊距)	當啟用時，會顯示列印邊距。 可使用 Print margin column 和 Appearance color options 喜好設定，來判定列印邊距的位置與顏色。	關閉
Appearance color options(外觀顏色選項)	可在這裡指定各種 Java 編輯器外觀特性的顏色。 ■ Line number foreground(行號前景)：行號的顏色。 ■ Matching brackets highlight(相符的方括弧強調顯示)：括弧的強調顯示顏色。 ■ Current line highlight(現行行強調顯示)：現行行的強調顯示顏色。 ■ Print margin(列印邊距)：列印邊距的顏色。 ■ Find scope(尋找範圍)：尋找範圍的顏色。 ■ Linked position(鏈結的位置)：程式碼輔助中所用鏈結位置的顏色。 ■ Link(鏈結)：鏈結顏色	預設顏色

語法(Syntax)

語法指定如何展現 Java 程式碼。

選項	說明	預設值

選項	說明	預設值
Background color(背景顏色)	System default(系統預設值)：作業系統所提供的預設背景顏色。 Custom(自訂)：使用者定義的背景顏色。	系統預設值
Foreground(前景)	下列的 Java 程式檔片段可以不同的顏色與樣式展現： ■ Multi-line comment(多行註解)：註解的格式為'/*...*/' ■ Single-line comment(單行註解)：註解的開頭為'//' ■ Keywords(關鍵字)：所有 Java 關鍵字。 ■ Strings(字串)：Java 字串和字元，以單引號與雙引號括住 ■ Others(其他)：預設 Java 程式碼 ■ Task tags(作業標示)：註解中的作業標示 ■ Javadoc keywords(Javadoc 關鍵字)：Javadoc 中所用的關鍵字，開頭為'@' ■ Javadoc HTML tags(Javadoc HTML 標示)：Javadoc 中所用的 HTML 標示 ■ Javadoc links(Javadoc 鏈結)：{@link reference}標示 ■ Javadoc others(Javadoc 其他)：預設 Javadoc 文字	預設顏色和樣式
Preview(預覽)	顯示和現行顏色和樣式有關之 Java 程式碼的預覽。	n/a

程式碼輔助(Code assist)

程式碼輔助指定程式碼輔助的行為與外觀。

選項	說明	預設值
Completion inserts/Completion	如果開啟了完成插入(Completion inserts)，完成文字將插入在脫字符號 (^) 位置，所以它絕不會改寫任何現有的文字。	完成插入

選項	說明	預設值
overwrites(完成插入/完成改寫)	如果開啟了完成改寫(Completion overwrites)，完成文字將取代脫字符號 (^) 位置之後直到字尾的字元。	
Insert single proposals automatically(自動插入單一提議)	當啟用時，程式碼輔助會自動選擇與插入單一提議。	開啟
Show only proposals visible in the invocation context(在呼叫環境定義中只顯示可見的提議)	當啟用時，則會使用顯示規則來限制 Java 元素的提議。例如，不顯示其他類別的 private 欄位提議。	開啟
Present proposals in alphabetical order(按字母順序來提供提議)	當啟用時，則提議會按字母順序排序。	關閉
Automatically add import instead of qualified name(自動新增匯入取代完整名稱)	當啟用時，則位於其他套件中的類型提議會呼叫以新增對應的匯入宣告。否則，會完整插入類型。	開啟
Fill argument names on method completion(方法完成時填入引數	當啟用時，則在選擇方法提議時，會插入該方法之宣告中指定的引數名稱。	關閉

選項	說明	預設值
名稱)		
Enable auto activation(啟用自動啟動)	當啟用時，可自動呼叫程式碼輔助。 可在自動啟動延遲(Auto activation delay)、自動啟動 Java 觸發(Auto activation triggers for Java)以及自動啟動 Javadoc 觸發(Auto activation triggers for Javadoc)中指定自動呼叫的條件。	開啟
Auto activation delay(自動啟動延遲)	一旦從遇到自動啟動觸發字元起到輸入新字元止的時間，超過自動啟動延遲，即會呼叫程式碼輔助。	500
Auto activation triggers for Java(自動啟動 Java 觸發)	如果 Java 程式碼內鍵有一個觸發字元(但不是鍵於 Javadoc 註解內)，則一旦在自動啟動延遲逾時前未輸入其他字元，即會呼叫程式碼輔助。	'.'
Auto activation triggers for Javadoc(自動啟動 Javadoc 觸發)	如果 Javadoc 內鍵有一個觸發字元，則一旦在自動啟動延遲逾時前未輸入其他字元，即會呼叫程式碼輔助。	'@'
Code assist color options(程式碼輔助顏色選項)	下列程式碼輔助 UI 元素所用的顏色： ■ Completion proposal background(完成提議背景)：完成提議視窗的背景顏色 ■ Completion proposal foreground(完成提議前景)：完成提議視窗的前景顏色 ■ Method parameter background(方法參數背景)：參數視窗的背景顏色 ■ Method parameter foreground(方法參數前景)：參數視窗的前景顏色 ■ Completion overwrite background(完成改寫背景)：完成改寫視窗的背景顏色 ■ Completion overwrite foreground(完成改	預設顏色

選項	說明	預設值
	寫前景)：完成改寫視窗的前景顏色	

附註(Annotations)

附註指定何時及如何顯示附註。

選項	說明	預設值
Analyze annotations while typing(輸入時分析附註)	如果啟用，則在使用者輸入時，就會更新附註。否則，直到編譯 Java 檔案之前，都不會更新附註。	開啟
Indicate annotations solvable with Quick Fix in vertical ruler(在垂直尺規中指出可利用快速修正解決的註釋)	針對每一個可透過快速修正解決的註釋，在 Java 編輯器左邊框的垂直尺規中，顯示一個燈泡。	開啟
Annotation presentation(附註呈現方式)	對於每一類型的附註，可以指定 以文字、以概觀尺規或兩者顯示附註 以何種顏色展現附註	

範本(Templates)

「範本」喜好設定頁面可以建立新範本與編輯現有範本。範本可方便程式設計師快速插入常重複出現的程式碼型樣。

下列按鈕可以操作與配置範本：

動作	說明
New...(新建...)	開啟對話框以建立新範本。

動作	說明
Edit...(編輯...)	開啟對話框以編輯目前所選取的範本。
Remove(移除)	移除所有選取的範本。
Import...(匯入...)	從檔案系統匯入範本。
Export...(匯出...)	將選取的所有範本匯出至檔案系統。
Export All...(匯出全部...)	將所有範本匯出至檔案系統。
Enable All(全部啟用)	啟用所有範本。
Disable All(全部停用)	停用所有範本。
Use Code Formatter(使用程式碼格式製作器)	當啟用時，在插入之前，會先根據程式碼格式製作器(Code Formatter)喜好設定中指定的程式碼格式設定規則，來設定範本的格式。否則，範本會按原樣插入，但內縮將不正確。 請參閱「程式碼格式製作器」喜好設定頁面

「範本」對話框(Template dialog)

新建範本與編輯現有範本所用的對話框相同，以下是其說明。

對話框中的欄位與按鈕如下：

選項	說明
Name(名稱)	範本的名稱。
Context(環境定義)	環境定義是決定哪些地方可使用範本，以及決定可用的一組預先定義的範本變數。 ■ Java：Java 環境定義

選項	說明
	■ Javadoc：Javadoc 環境定義
Description(說明)	範本的說明，會在使用者選擇範本時顯示。
Pattern(型樣)	範本的型樣。
Insert Variables...(插入變數...)	列出預先定義的環境定義特定變數。

範本變數(Template variables)

Java 和 Javadoc 環境定義皆會定義下列變數：

變數	說明
${cursor}	指出當離開範本編輯模式時的游標位置。當離開範本編輯模式時，如果希望游標應移至另一位置，而非移至範本尾端，則可善用這項。
${date}	評估成現行日期。
${dollar}	評估成錢幣符號 '$'。 另外，可以使用兩個貨幣符號：'$$'。
${enclosing_method}	評估成含括名稱的名稱。
${enclosing_method_arguments}	評估成含括方法的以逗點區隔之引數名稱清單。這個變數有助於為許多方法產生 log 陳述式。
${enclosing_package}	評估成含括套件的名稱。
${enclosing_project}	評估成含括專案的名稱。
${enclosing_type}	評估成含括類型的名稱。

變數	說明
${file}	評估成檔案的名稱。
${return_type}	評估成含括方法的傳回類型。
${time}	評估成現行時間。
${user}	評估成使用者名稱。

此外，Java 環境定義會定義下列變數：

變數	說明
${array}	評估成已宣告之陣列名稱的提議。
${array_element}	評估成已宣告之陣列的元素名稱的提議。
${array_type}	評估成已宣告之陣列的元素類型的提議。
${collection}	評估成實作 java.util.Collection 之已宣告集成的提議。
${index}	評估成未宣告之陣列索引疊代的提議。
${iterator}	評估成未宣告之集成疊代的提議。

3.6.7 JRE 安裝(JRE installations)

「類別路徑變數」喜好設定

選項	說明
Add...(新增...)	將新的 JRE 定義新增至工作台中。在產生的對話框中，指定下列： ■ JRE 類型：(從下拉清單中選取一種 VM 類型) ■ JRE 名稱：輸入這個 JRE 定義的名稱 ■ JRE 起始目錄：輸入或瀏覽以選取這項 JRE 安裝的根目錄 ■ Javadoc URL：輸入或瀏覽以選取 URL 位置。這個位置

選項	說明
	供 Javadoc 匯出精靈做為預設值，並供「開啟外部 Javadoc」動作使用。除錯器逾時值：輸入這個 JRE 之除錯器的預設逾時值（以毫秒為單位） ■ 如果要讓這個 JRE 採用預設程式庫位置，請勾選這個勾選框；如果要輸入或瀏覽以便為下列檔案指定程式庫位置，請清除這個勾選框： □ JAR 檔（例如：classes.zip） □ 程式檔（例如：source.zip） 可以按一下「Browse」按鈕，以瀏覽並找出所要的路徑。
Edit...(編輯...)	可讓編輯所選取的 JRE。
Remove(移除)	將所選取的 JRE 從工作台中移除。
Search...(搜尋...)	自動搜尋已安裝在本端檔案系統的 JRE,並在工作區中建立對應的 JRE 定義。

3.6.8 JUnit

選項	說明	預設值
Show the JUnit results view only when an error or failure occurs（只有在發生錯誤或失敗時,才顯示 JUnit 結果視圖）	當啟用時，則只有在發生錯誤或失敗時才會將 JUnit 視圖帶至前面。	開啟
Stack trace filter patterns(堆疊追蹤過濾器型樣)	測試失敗時，不應該顯示在堆疊追蹤的套件、類別或型樣。	預設過濾器型樣

3.6.9 新專案(New project)

選項	說明	預設值
Source and output folder(程式檔與輸出資料夾)	■ 專案：將程式檔與輸出位置皆設為專案的根目錄。 ■ 資料夾：可個別設定程式檔和輸出位置。如果要指定程式檔與輸出位置，請設定來源資料夾名稱(Source folder name)與輸出位置名稱(Output location name)。	專案
Source folder name(來源資料夾名稱)	程式檔的位置。	'src'
Output location name(輸出位置名稱)	輸出檔的位置。	'bin'
As JRE library use(使用 JRE 程式庫時)	指定所要使用的 JRE 程式庫。 JRE 儲存器 JRE 儲存器。 JRE_LIB 變數 *JRE_LIB* 變數指定的 JRE。	JRE 儲存器

3.6.10 組織匯入(Organize imports)

下列的喜好設定是定義「組織匯入」指令要如何在編譯單元中產生 import 陳述式。

「組織匯入」喜好設定

選項	說明	預設值
Import order list(匯入順序清	這份字首清單顯示套件匯入到 Java 編譯單元中的順序。每一個項目皆定義出一個區塊。區塊之	java javax

選項	說明	預設值
單)	間各空一行。	org com
New...(新建...)	新增套件名稱字首到匯入順序清單中。在產生的對話框中，輸入套件名稱或套件名稱字首。	n/a
Edit...(編輯...)	變更現有套件名稱字首的名稱。在產生的對話框中，輸入套件名稱或套件名稱字首。	n/a
Up(上)	在匯入順序清單中,將所選套件名稱字首往上移。	n/a
Down(下)	在匯入順序清單中,將所選套件名稱字首往下移。	n/a
Remove(移除)	將套件名稱字首從匯入順序清單中移除。	n/a
Load...(載入...)	從檔案載入套件名稱字首清單。	n/a
Save...(儲存...)	將套件名稱字首清單儲存至檔案。	n/a
Number of imports needed before .* is used(使用.*之前所需的匯入數目)	在使用 *<package>.** 之前,可來自相同套件的完整 import 陳述式的數目。	99
Do not create imports for types starting with a lower case letter(不為以小寫字母開頭的類型建立匯入)	當啟用時，則不會匯入以小寫字母開頭的類型。	開啟

3.6.11「重構」喜好設定(Refactoring preferences)

可在「重構」喜好設定頁面中設定下列的喜好設定。(「Window」→「Preferences」→「Java」→「Refactoring」。)

選項	說明	預設值
Confirm the	在這個區段中，請選取哪些種類的問題會使精靈	錯誤

選項	說明	預設值
execution of the refactoring if(如果發生下列情況，請確認執行重構)	在按下 Finish 之後，仍維持開啟狀態並顯示問題： ■ 會阻止實際重構執行的問題 ■ 搜尋工作台 ■ 工作台中的警告 ■ 前置條件檢查所產生的資訊 當問題的嚴重性比所選的層次還低時，則可讓進行重構，而不必先預覽結果。	
Save all modified resources automatically prior to refactoring(在重構前自動儲存所有修改過的資源)	如果啟用這個選項，則每當執行重構動作時，工作台會自動儲存自前次儲存以來所有已修改過的資源。	不勾選

3.6.12 作業標示(Task Tags)

在這個喜好設定頁面上，可以配置作業標示。當標示清單不是空的時候，每當編譯器在 Java 程式碼中的任何註解內遇到其中一個對應標示時，編譯器將發出作業標記。所產生的作業訊息將包括標示，以及直到下一個字行分隔字元或註解結尾的範圍。

指令	說明
New...(新建...)	新增作業標示。請在出現的對話框中，指定新作業標示的名稱和優先順序。
Remove(移除)	移除所選作業標示。
Edit...(編輯...)	可以編輯所選作業標示。請在出現的對話框中，編輯作業標示的名稱和/或優先順序。

有一個名為 TODO、優先順序為 Normal 的預設作業標示。

3.7 團隊(Team)

團隊喜好設定頁面包含會影響版本管理團隊支援的選項。

選項	說明	預設值
Use compressed folders as default Synchronize view layout(利用壓縮資料夾作為預設的「同步化」視圖佈置)	在初次將同步化加入「同步化」視圖時，請利用這個選項來配置所用的預設佈置。可以在視圖下拉功能表中後續變更這個佈置。	已啟用
Show all synchronization information in a resource's text label(將所有同步化資訊顯示在資源的文字標籤中)	請利用這個選項，將資源的同步化狀態顯示成資源標籤中的文字。依預設，只會用一個圖示裝飾元來識別資源的同步化狀態。	已停用
Switch to the associated perspective when a synchronize operation completes(當同步化作業完成時,切換到相關的視景)	請利用這個選項來配置執行同步化作業時會發生的狀況。選項如下： ■ Always(固定)：一律切換視景 ■ Never(絕不)：絕不切換視景 ■ Prompt(提示)：提示切換視景	提示

選項	說明	預設值
Perspective Switching(視景切換)	請利用這個選項來配置執行同步化作業時所顯示的視景。	「團隊同步化」視景

3.7.1 CVS

在 CVS 喜好設定頁面中，可以自訂 CVS 外掛程式的若干項目。

一般 CVS 喜好設定(General CVS Preferences)：

選項	說明	預設值
Prune empty directories(刪改空目錄)	可以使用這個選項來指定在更新以及同步處理視圖中刪改空白的目錄。雖然刪改的目錄不會顯示在工作台中，在儲存庫中仍會存有一個空目錄。這是有幫助的，因為 CVS 不會讓用戶端從伺服器移除目錄。	已啟用
Consider file contents in comparison(在比較時考量檔案內容)	當比較 CVS 資源時，請利用這個選項來比較所找到已變更的檔案之內容。通常會利用時間戳記來比較 CVS 檔案，這是目前最快的方法。不過，在某些情況下，經由比較檔案內容可得到較為精確的比較。停用這個選項會加快比較的速度，但可能會產生內容相同的比較項目。這個選項只適用於比較，不適用於合併和工作區同步化。	已啟用
Delete unmanaged resources on replace(取代時刪除未管理的資源)	可以使用這個選項，在取代為儲存庫的資源時，允許刪除不在 CVS 控制下的資源。	已啟用
Treat all new files as binary(將所有新	可以使用這個選項來置換檔案內容設定以及將所有新檔案視為二進位檔。	已停用

選項	說明	預設值
檔案視為二進位檔)		
Validate server version compatibility on first connection(第一次連線時驗證伺服器版本的相容性)	可以使用這個選項，在第一次連線時查詢 CVS 伺服器版本，以判斷伺服器的相容性。 伺服器版本會輸出到主控台，如果偵測到不相容，就會在連線時記載警告訊息。	已啟用
Confirm move tag on tag operation(確認在標示作業上移動標示)	請利用這個選項，以便在選取「移動標示」選項時得到提示。	已啟用
Display detailed protocol output to stdout(將詳細的通訊協定輸出顯示在標準輸出中)	請利用這個選項來顯示工作台和 CVS 伺服器之間的通訊追蹤。	已停用
Convert text files to use platform line neding(轉換文字檔來使用平台行尾)	請利用這個選項，將文字檔的行尾轉換成平台所用的行尾。如果將資源移出到 Windows 機器所裝載的 *nix 磁碟機，可以停用這個選項。	已啟用
Show revision comparisons in dialog(在對話框中顯示修訂比較)	請利用這個選項，將修訂比較顯示在對話框中，而不是顯示在比較編輯器中。	已停用

選項	說明	預設值
Communication timeout(通訊逾時)	可以使用這個選項來配置在逾時前要等待連線到 CVS 伺服器的時間總數（以秒為單位）。	60 秒
Quietness level(安靜層次)	設定針對指令的 CVS 列印狀態資訊總數。在有點安靜(Somewhat quiet)模式中，會抑止列印不重要的參考資訊。重要性的考量是根據每個指令。在非常安靜(Very quiet)模式中，除了完成指令的必要輸出外，所有的輸出都會被抑止。在非常安靜(Very quiet)模式中，有些 CVS 伺服器可能無法告知一些已經發生的錯誤的重要資訊。可能要考慮改用有點安靜(Somewhat quiet)模式。	細節
Default keyword substitution(預設的關鍵字替代)	可以使用這個選項來設定文字檔的預設關鍵字替代。	包含關鍵字展開項 -kkv 的 ASCII
Save dirty editors before CVS operations(在 CVS 作業之前儲存變動過的編輯器)	可以使用這個選項來配置在執行 CVS 作業時，如果已開啟的編輯器包含未儲存的變更時，會發生什麼情況。 選項包括： ■ Never(絕不)：繼續執行 CVS 作業，即使已開啟的編輯器中有尚未儲存的變更。 ■ Prompt(提示)：詢問使用者要如何處理已開啟的編輯器中的未儲存變更。 ■ Auto-save(自動儲存)：在每一個 CVS 作業前自動儲存已開啟編輯器中尚未儲存的變更。	提示

主控台喜好設定(Console preferences)：

選項	說明	預設值
Console text color settings(主控台文字的顏色設定)	可以使用這些選項來變更「CVS 主控台」中所顯示的文字的顏色。 ■ 指令行文字（黑色） ■ 訊息文字（藍色） ■ 錯誤文字（紅色）	

選項	說明	預設值
Show CVS output in Console view(在「主控台」視圖中顯示 CVS 輸出)	請利用這個選項，將 CVS 指令輸出顯示在「主控台」視圖中。啟用這個選項可以顯示非常有用的資訊，但指令作業速度會變慢。	已停用

「Ext 連線方法」喜好設定(Ext Connection Method preferences)：

Use external program vs. Use internal connection method(使用外部程式和使用內部連線方法)	這個頁面可以配置 ext 連線方法來使用外部程式，或配置另一個連線方法來連接到伺服器。後面一個選項是要讓 extssh 之類的自訂連線方法保持與外部 CVS 用戶端工具相容。	使用外部程式
CVS_RSH(CVS_RSH)	可以使用這個選項來配置在在連線到遠端 CVS 伺服器時要呼叫的程式。呼叫 RSH 指令時會使用下列呼叫型樣： *CVS_RSH 參數 CVS_SERVER*	ssh
Parameters(參數)	可以使用這個選項來配置傳送到 CVS_RSH 程式的參數。預設參數型樣為 {host} -l {user}。可使用 {host}、{user}、{password} 與 {port} 等變數來加以修改。	{host} -l {user}
CVS_SERVER(CVS_SERVER)	可以使用這個選項來配置要執行的遠端 CVS 伺服器程式的名稱。只有當遠端 CVS 伺服器二進位名稱與預設值不同時，才變更這個設定	cvs
Connection type(連線類型)	如果啟用使用另一個連線方法的選項，請利用這個選項來設定使用 ext 連線方法的儲存庫位置所用的連線方法。	

「標籤裝飾」喜好設定(Label Decorations preferences)：

Text(文字)	可以使用這個頁面中的選項來配置如何將 CVS 資訊新增至文字標籤。
Icons(圖	可以使用這個頁面中的選項來配置可用哪些圖示作為覆蓋，以便在

示)	視圖中顯示 CVS 特定的資訊。
General(一般)	可以使用這個頁面中的選項來配置有關裝飾元的幾個喜好設定： 計算資料夾的深度送出狀態(Compute deep outgoing state for folders) 可以使用這個選項來配置是否應該計算資料夾的送出指示器。停用這個選項時，可增進 裝飾元的效能，因為計算資料夾的已用過狀態時，需要計算所有子項資源的已用過狀態。(*預設是啟用的*)

密碼管理(Password Management)：

這個喜好設定頁面可以查看哪些儲存庫位置的密碼快取在金鑰環檔案中，且可以除去這些密碼。

SSH2 連線方法(SSH2 Connection Method)：

General(一般)	請利用這個頁面中的選項來配置 ssh 金鑰目錄的位置，以及連線時要將哪些金鑰傳給伺服器。
Proxy	請利用這個頁面中的選項來配置 HTTP 或 SOCKS5 Proxy。
Key Management(金鑰管理)	請利用這個頁面中的選項來建立、管理和匯出金鑰

監視/編輯喜好設定(Watch/Edit preferences)：

Configure projects to use Watch/edit on checkout(配置專案以便在移出時使用監視/編輯)	可以使用這個選項來指出從儲存庫移出的檔案都要變成唯讀。	停用
When read-only files are modified in an editor(在編輯器中修改	可以使用這個選項來配置當另一個工具或已開啟的編輯器在修改唯讀檔時，會發生何種狀況。 ■ 傳送 cvs 編輯通知給伺服器(Send a cvs edit notification to the server)：在容許寫入檔案前發出 cvs 編輯通知給伺服器。 如果檔案中有其他編輯器，就會提示使用者是否要繼	傳送 cvs 編輯通知給伺服器

唯讀檔案的時機)	續或取消。 ■ 編輯檔案而不通知伺服器(Edit the file without informing the server)：使檔案變成唯讀而不通知伺服器。	
Before a CVS edit notification is sent to the server(在傳送 CVS 編輯通知給伺服器之前)	請利用這個選項來配置當開啟的編輯器或另一個工具在修改唯讀檔，且啟用了將 CVS 編輯通知傳給伺服器(Send a cvs edit notification to the server)時，會發生何種狀況。 ■ Always Prompt(固定提示)：固定提示使用者進行確認 ■ Only prompt if there are other editors(只在有其他編輯器時才提示)：向使用者顯示現行編輯器的清單，讓使用者確認或取消編輯。 ■ Never Prompt(絕不提示)：傳送編輯通知，但不提示	只在有編輯器時才提示

3.7.2 忽略的資源(Ignored Resources)

在「Team」→「Ignored Resources」喜好設定頁面中，可以指定要排除在版本控制管理系統之外的檔案名稱型樣。它有一份檔案型樣清單，資源必須符合這些檔案型樣，才能成為版本控制候選項。這些型樣可含有萬用字元 "*" 和 "?"。型樣 "*" 代表任何零或多個字元的序列。型樣 "?" 代表任何一個字元。例如，可以指定型樣 "*~"，其將會比對任何以 "~" 結尾的暫存檔。在更新或確認作業期間，將會忽略任何符合任一型樣的檔案或目錄。

如果要將檔案類型新增至忽略清單，只需要按一下 Add 按鈕。在對話框中，請輸入檔案類型（例如 *.class)。如果要從忽略清單中移除檔案類型，只需要選取忽略清單中的檔案類型，按一下 Remove 按鈕。

可以從清單中取消選取某個檔案型樣，暫時停用這個檔案型樣的忽略功能。不必從清單中移除指定的檔案型樣，就可以暫時停用它。

3.7.3 檔案內容(File Content)

在「Team」→「File」喜好設定頁面中，可以將副檔名關聯到檔案所包含的資料類型。檔案內容類型的兩個選項是 ASCII 和二進位。然後，像 CVS 這類的儲存庫提供者就可以使用這項資訊來為內容類型提供適當的行為。例如，在 ASCII 檔案中，CVS 可確保行終止器符合 OS 平台的行終止器。

將項目新增至「檔案內容」頁面的方法有兩種。第一個方法是透過工作台外掛程式的幫助。整合到工作台的工具可以為工作台提供工具專用的副檔名的檔案內容類型。工作台本身也會定義在工作台中經常使用和出現的副檔名的檔案內容類型（例如，html、gif 等等）。

第二個方法是讓使用者在「檔案內容」喜好設定頁面中明確地新增檔案內容類型。如果要這麼做，使用者只需要按一下 Add 按鈕，然後輸入副檔名。之後，他們可以切換類型與副檔名之間的關聯，方法是在表格中選取副檔名的項目，然後按一下 Change。使用者可以選取項目，然後按一下 Remove，將項目從清單中移除。

4. Java 程式開發

在 Eclipse 中做任何事之前，都必須新增一個專案。Eclipse 可透過外掛支援數種專案(如 EJB 或 C/C++)，預設支援下列三種專案：

- Java Project – Java 開發環境
- Plug-in Project – 自行開發 plug-in 的環境
- Sample Project – 提供操作文件的一般環境

如圖 4.1

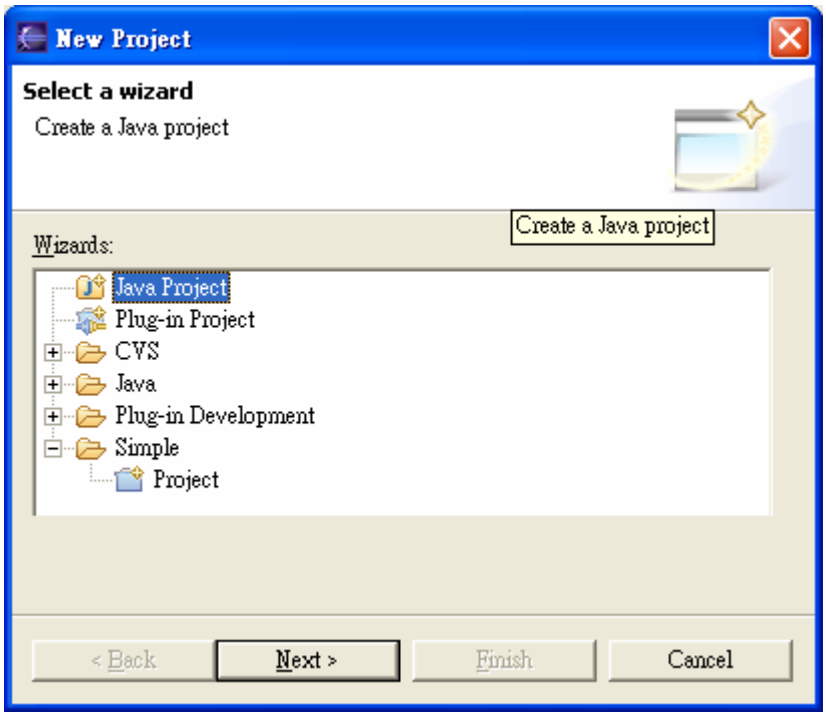


圖 4.1

4.1 建立 Java 專案

新增 Java 專案的步驟：

I. 選擇「File」→「New」→「Project」
(或是在『Package Explorer』視窗上按滑鼠右鍵，選擇「New」→「Project」選單選項)

（或是按工具列上 New Java Project 的按鈕  ）

II. 在 New Project 對話框(圖 4.1)，選 Java Project

（或是展開 Java 的資料夾，選 Java Project，如圖 4.2）

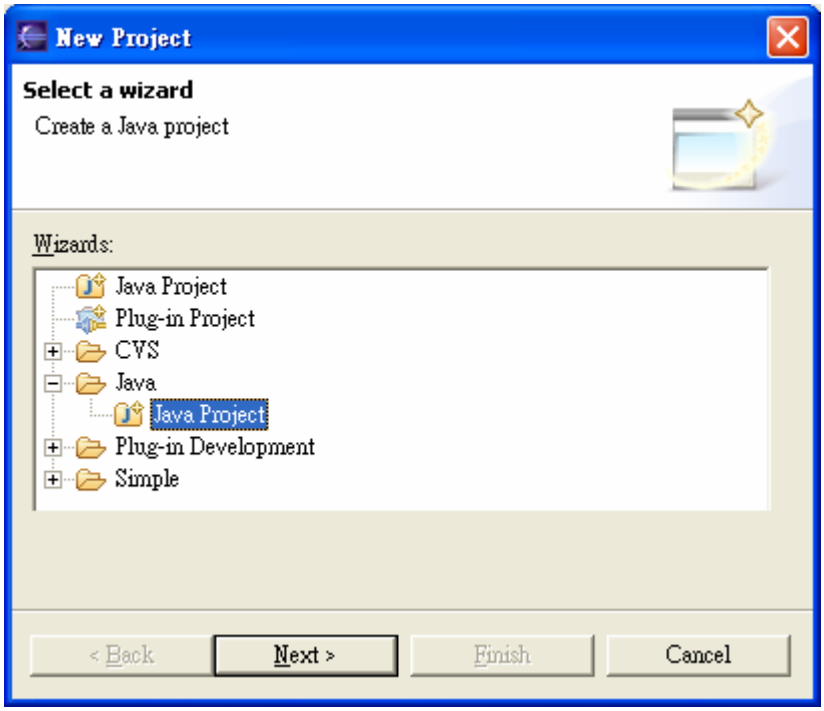


圖 4.2

III. 在 New Java Project 的視窗中輸入 Project 的名稱，如圖 4.3

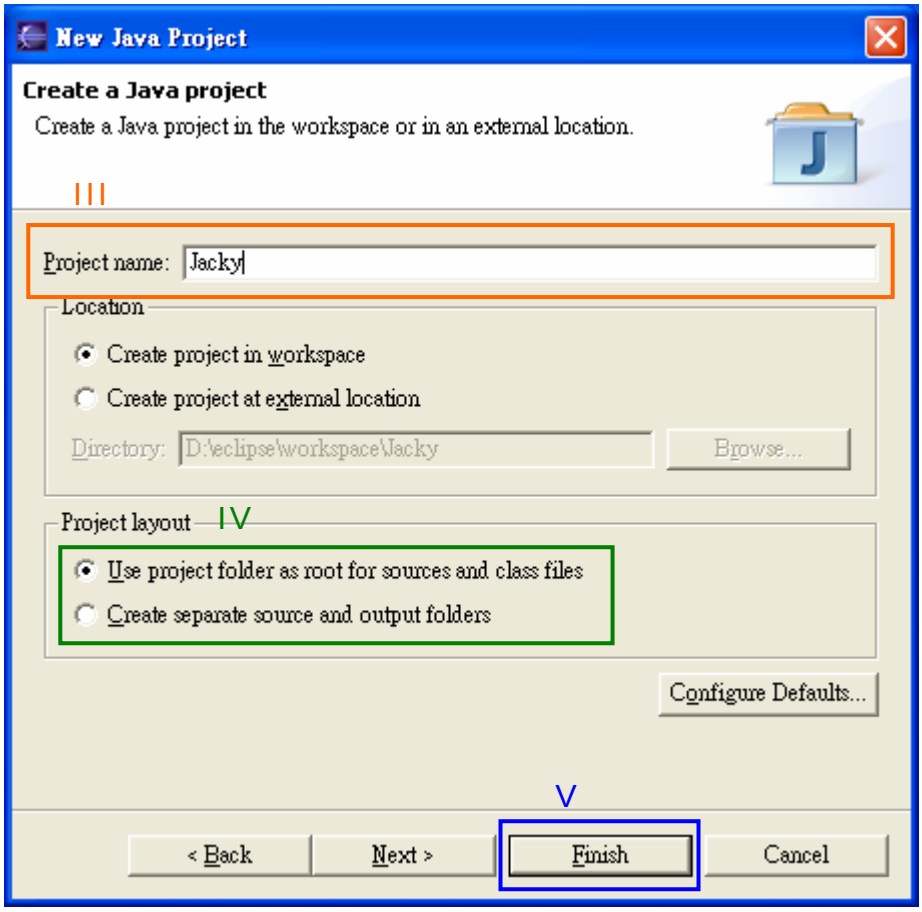


圖 4.3

IV. 在 Project Layout 中可以選擇編譯好的檔案是否要和原始檔放在同一個目錄下，如圖 4.3

V. 按下 Finish

4.2 建立 Java 類別

新增 Java 類別的步驟：

I. 選擇「File」→「New」→「Class」
(或是在『Package Explorer』視窗上按滑鼠右鍵，選擇「New」→「Class」選單選項)
(或是按工具列上 New Java Class 的按鈕)

II. 在 New Java Class 視窗中，Source Folder 欄位預設值是專案的資料夾，不需要更改。

III. Package 欄位輸入程式套件的名稱

IV. Name 欄位輸入 Class Name

V. 在 Which method would you like te creat 的部份，有勾選 public static void main(String[] args)的話，會 generate main method

VI. 按 Finish，會依套件新增適當的目錄結構及 Java 原始檔

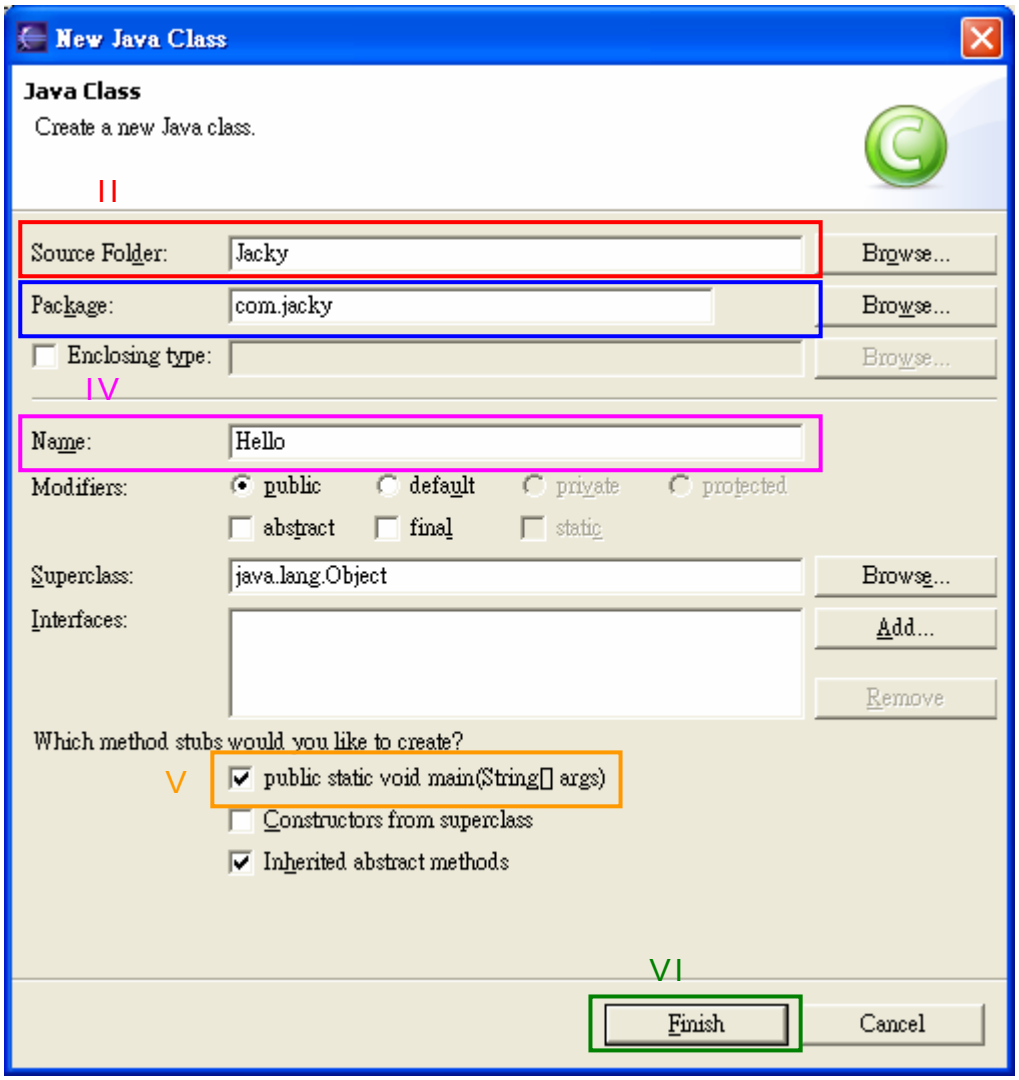


圖 4.4

- 在 Package Explorer 的視圖中可以看到程式的結構
- 在 Navigator 的視圖中可以看到套件的目錄架構

4.3 程式碼完成功能

4.3.1 Code Completion

在 Eclipse 中打左括號時會立刻加上又括號；打雙引號(單引號)時也會立刻加上雙引號(單引號)。

4.3.2 Code Assist

在輸入程式碼時，例如要打 System.out.println 時，打完類別名稱後暫停一會兒，Eclipse 會顯示一串建議清單，列出此類別可用的方法和屬性，並附上其 Javadoc 註解。可以直接捲動選出然後按 Enter。

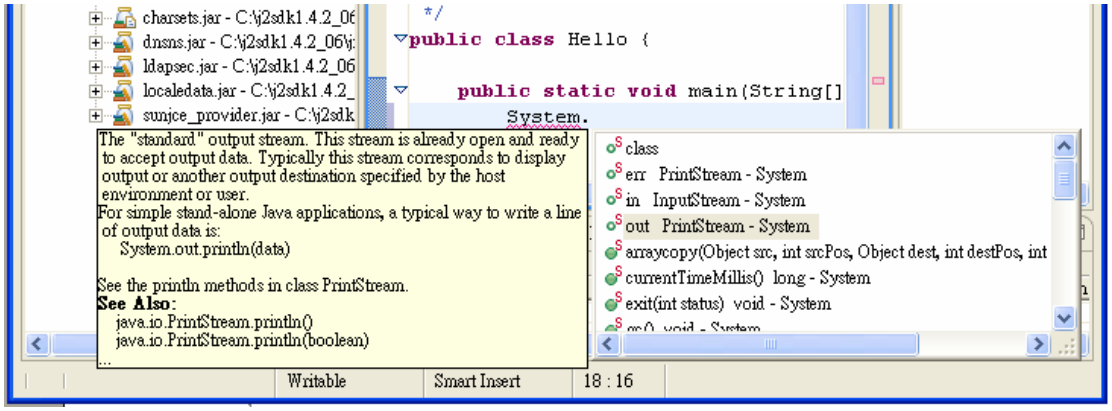


圖 4.5

也可以只打類別開頭的字母，然後按 Alt - /，一樣會顯示一串建議清單。

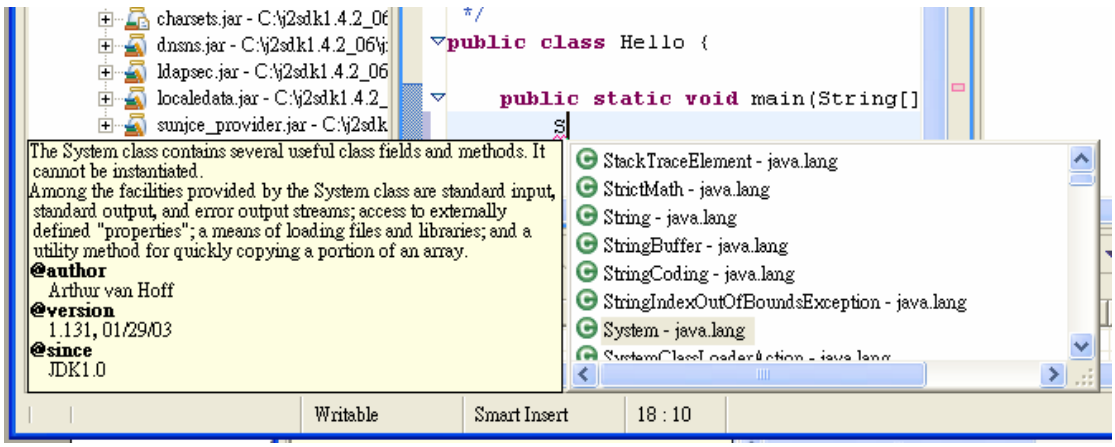


圖 4.6

Alt - /這個組合鍵不僅可以可以顯示類別的清單，還可以一併顯示已建立的模板程式碼，例如要顯示陣列的資訊，只要先打 for，在按 Alt - /這個組合鍵，就會顯示模板的請單。

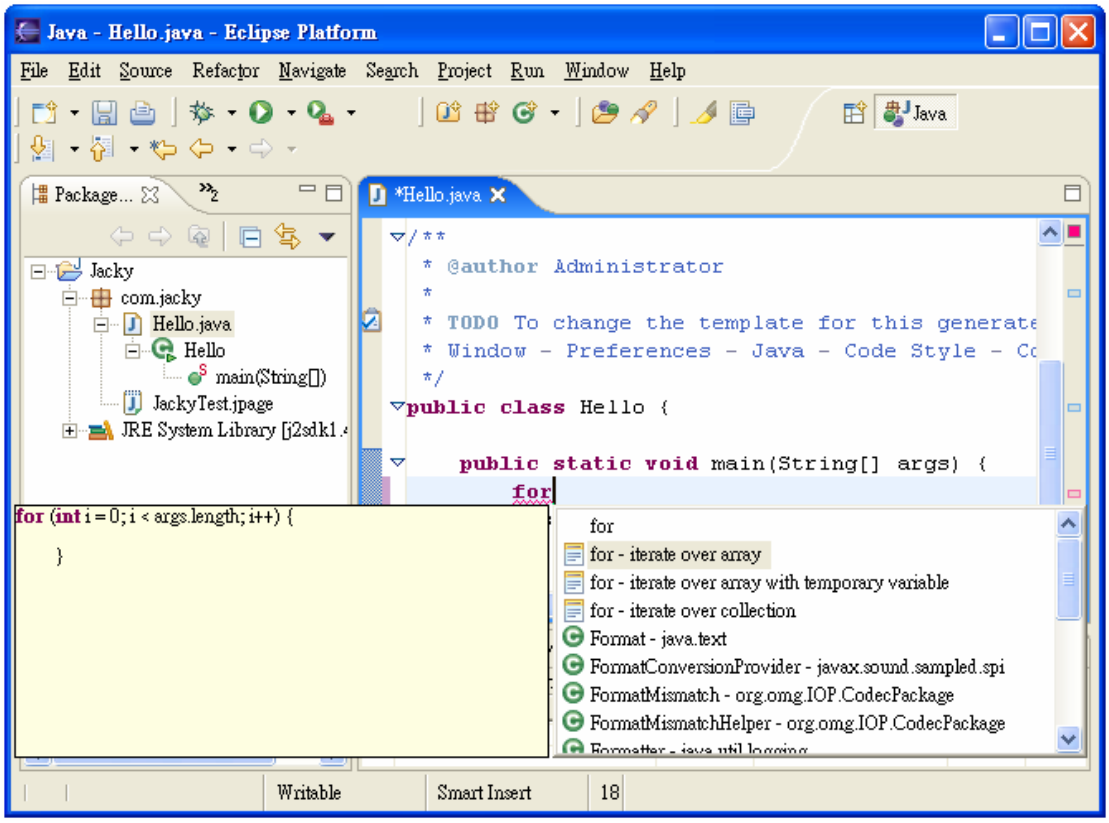


圖 4.7

4.4 執行 Java 程式

大多數的程式不需特定的啟動組態(Launch Configuration)，首先確定要執行的程式碼在編輯器中有選到(頁籤變藍色)，再執行下列步驟：

I. 選單選「Run」→「Run as」→「Java Application」

II. 若有修改過程式，Eclipse 會詢問在執行前是否要存檔

III. Tasks 試圖會多出 Consol 頁籤並顯示程式輸出

程式若要傳參數、或是要使用其他的 Java Runtim...等等，則需要設定程式啟動的相關選項，執行程式前，新增一個啟動組態或選用現有的啟動組態。

I. 選單選「Run」→「Run」，開啟 Run 的設定視窗

- **Main** 標籤用以定義所要啟動的類別。請在專案欄位中，輸入內含所要啟動之類別的專案名稱，並在主要類別欄位中輸入主要類別的完整名稱。如果想要程式每當在除錯模式中啟動時，在 main 方法中停止，請勾選 **Stop in main** 勾選框。
- 附註：不必指定一個專案，但這樣做可以選擇預設類別路徑、來源查閱路徑，以及 JRE。
- **引數(Arguments)**標籤用以定義要傳遞給應用程式與虛擬機器（如果有的話）的引數。也可以指定已啟動應用程式要使用的工作目錄。
- **JRE** 標籤用以定義執行或除錯應用程式時所用的 JRE。可以從已定義的 JRE 選取 JRE，或定義新的 JRE。
- **類別路徑(Classpath)**標籤用以定義在執行或除錯應用程式時所用類別檔的位置。依預設，使用者和 bootstrap 類別位置是從相關聯專案的建置路徑衍生而來。可以在這裡置換這些設定。
- **程式檔(Source)**標籤用以定義當除錯 Java 應用程式時，用來顯示程式檔之程式檔的位置。依預設，這些設定是從相關聯專案的建置路徑衍生而來。可以在這裡置換這些設定。
- **環境(Environment)**標籤會定義在執行 Java 應用程式或者對它進行除錯時，所要使用的環境變數值。依預設，這個環境是繼承自 Eclipse 執行時期。可以置換或附加至繼承的環境。
- **共用(Common)**標籤定義有關啟動配置的一般資訊。可以選擇將啟動配置儲存在特定檔案，以及指定當啟動配置啟動時，哪些

視景將變成作用中。

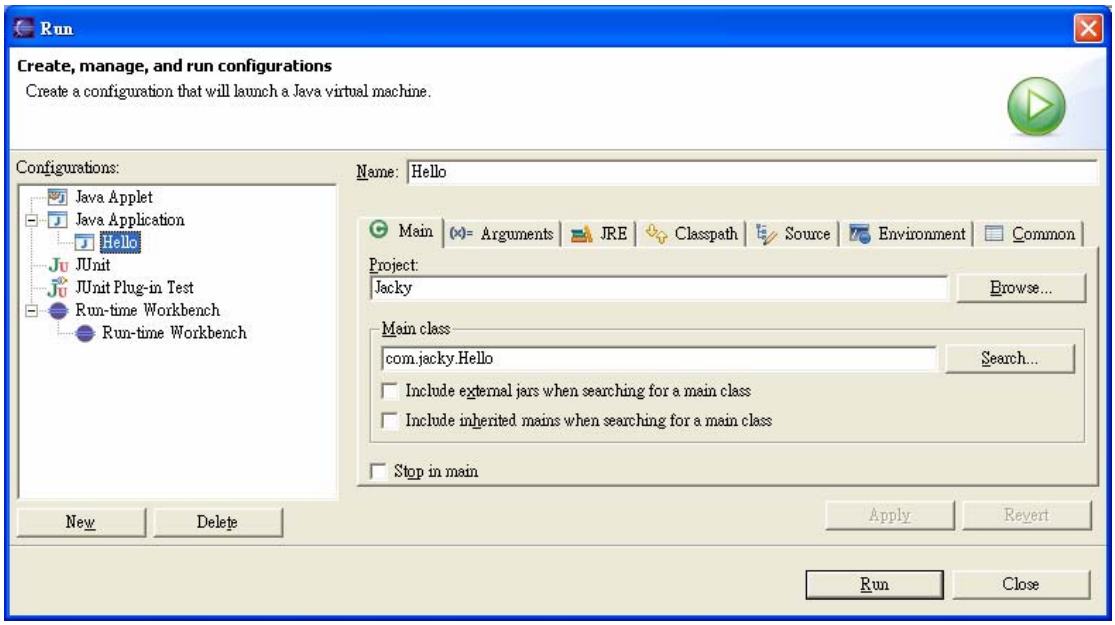


圖 4.8

II. 在 Arguments 的頁籤中輸入要傳入的值，若是多值的話，用空白鍵隔開

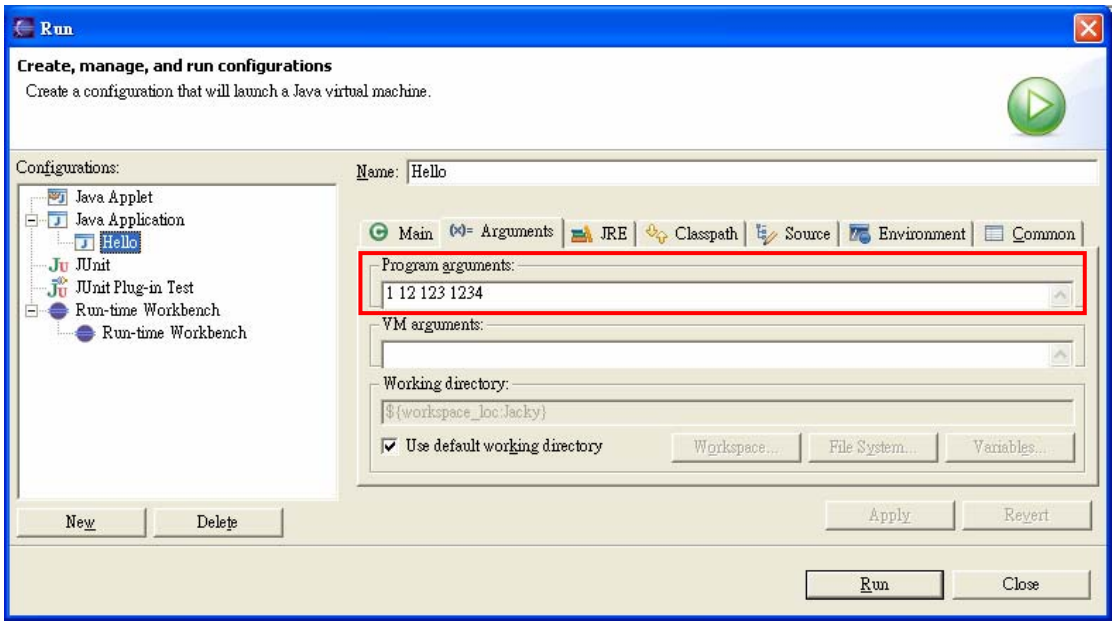


圖 4.9

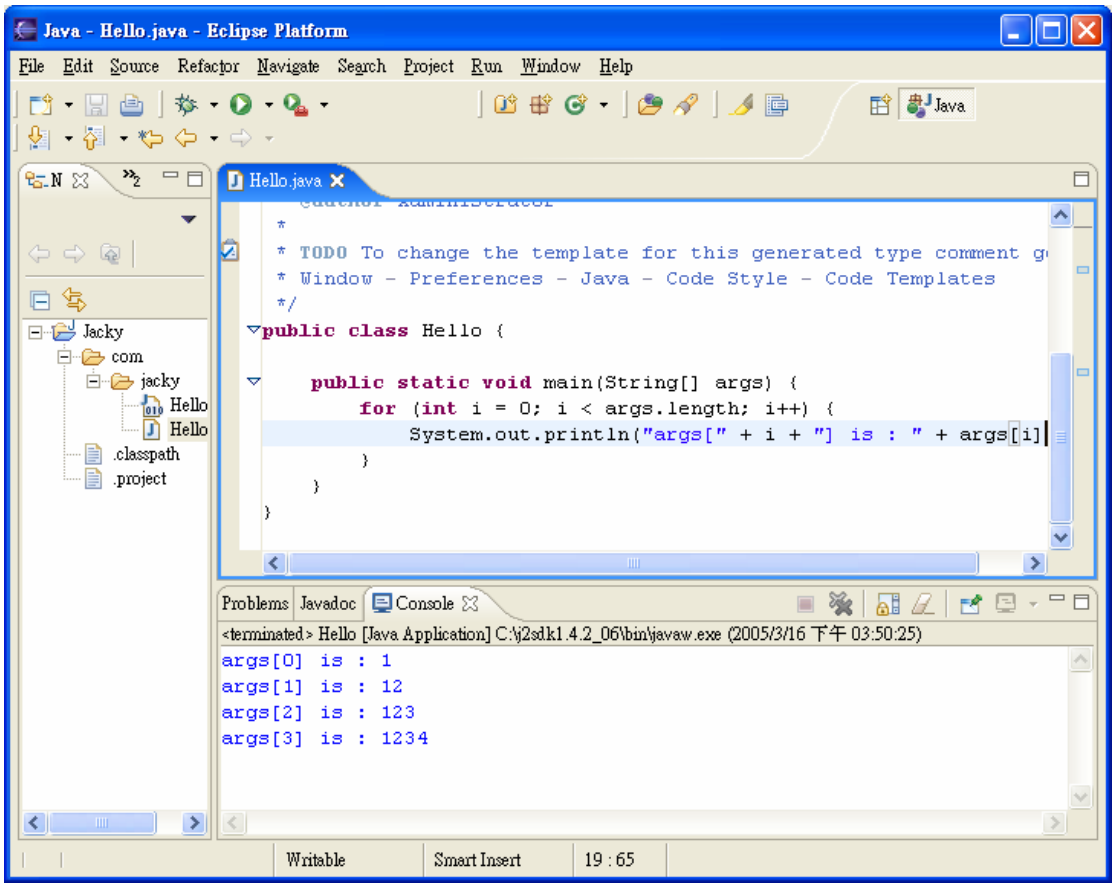


圖 4.10

4.5 Java 即時運算簿頁面(Java Scrapbook Page)

寫程式時可能會些其他的想法，但不知是否可行：多數情況是直接寫到程式再來 debug，或是另外寫各小程式。Eclipse 提供一種輕巧的替代方式，Java 即時運算簿頁面(Java Scrapbook Page)，藉由漸進式編譯器，可以在即時運算簿寫入任意的 Java 程式碼並執行，不需另寫在類別或方法中。

I. 切換至 Java 視景

II. 「File」→「New」→「Other...」→「Java」→「Scrapbook Page」

（在專案上按右鍵，「New」→「Other...」→「Java」→「Scrapbook Page」）

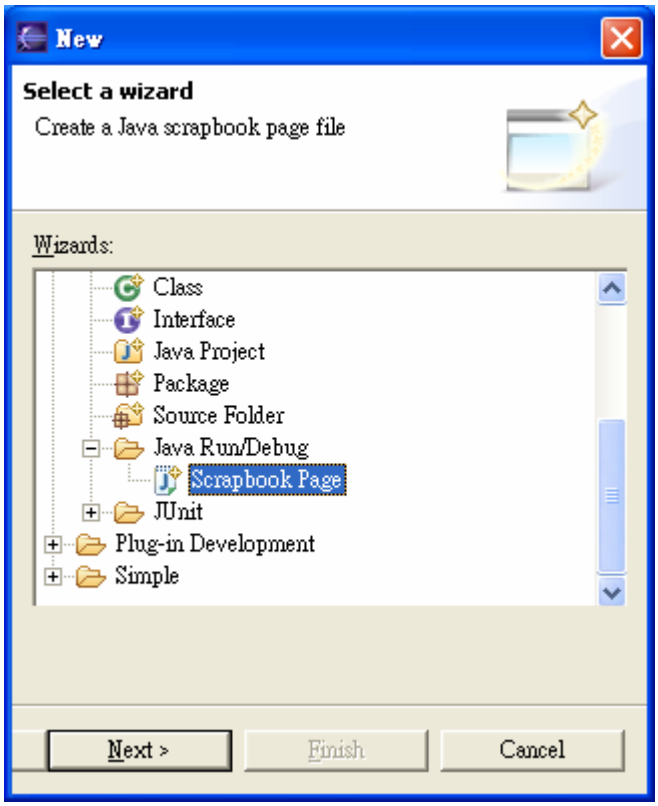


圖 4.11

III. 選擇要存放的地方

IV. 輸入檔名

IV. 按下 Finish

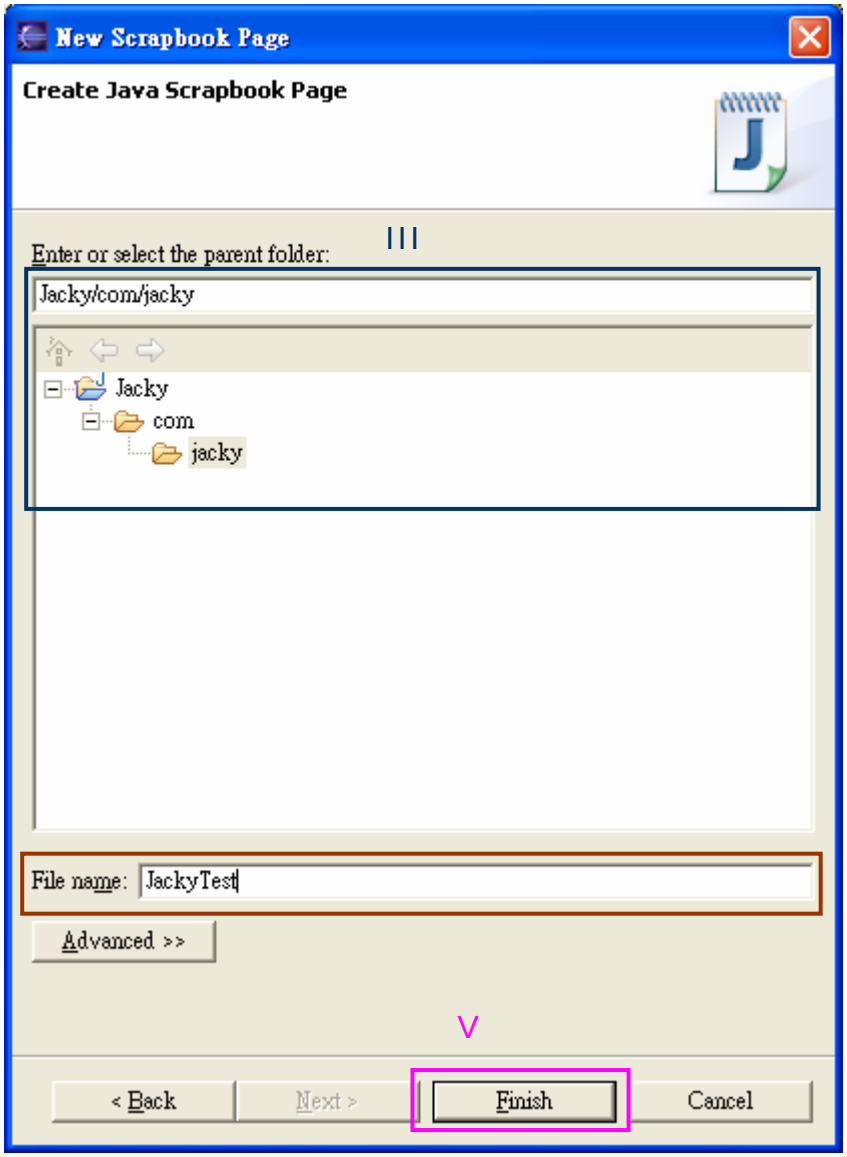


圖 4.12

在「Package Explorer」或是「Navigator」視圖會顯示剛剛建立的 JackyTest.jpage 檔案。

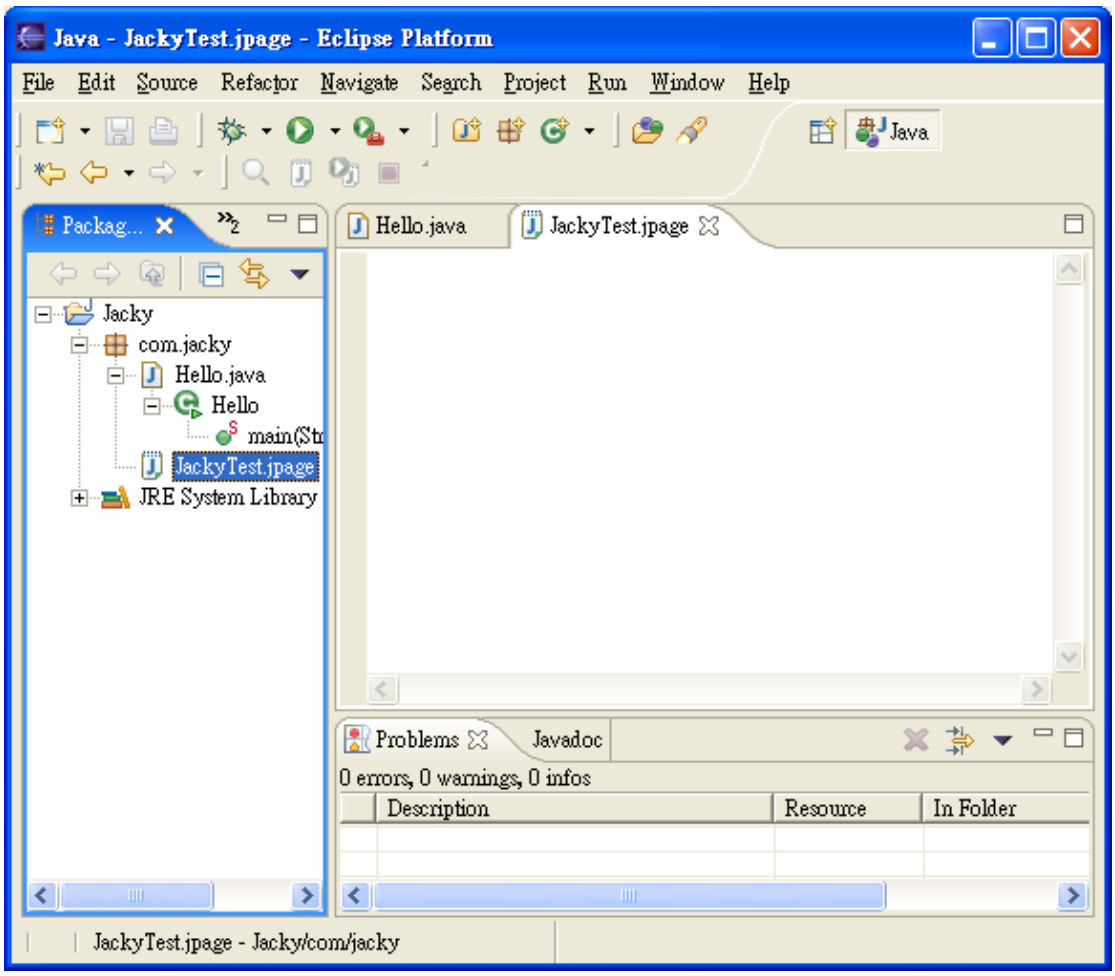


圖 4.13

可以輸入要測試的 Java 程式碼，例如

```
for (int i = 0; i < 5; i++) {
    System.out.println(Integer.toString(i));
}
```

I. 將這段程式碼反白

II. 在這段程式上按右鍵，選擇 Execute

III. Consol 視圖會顯示執行的結果

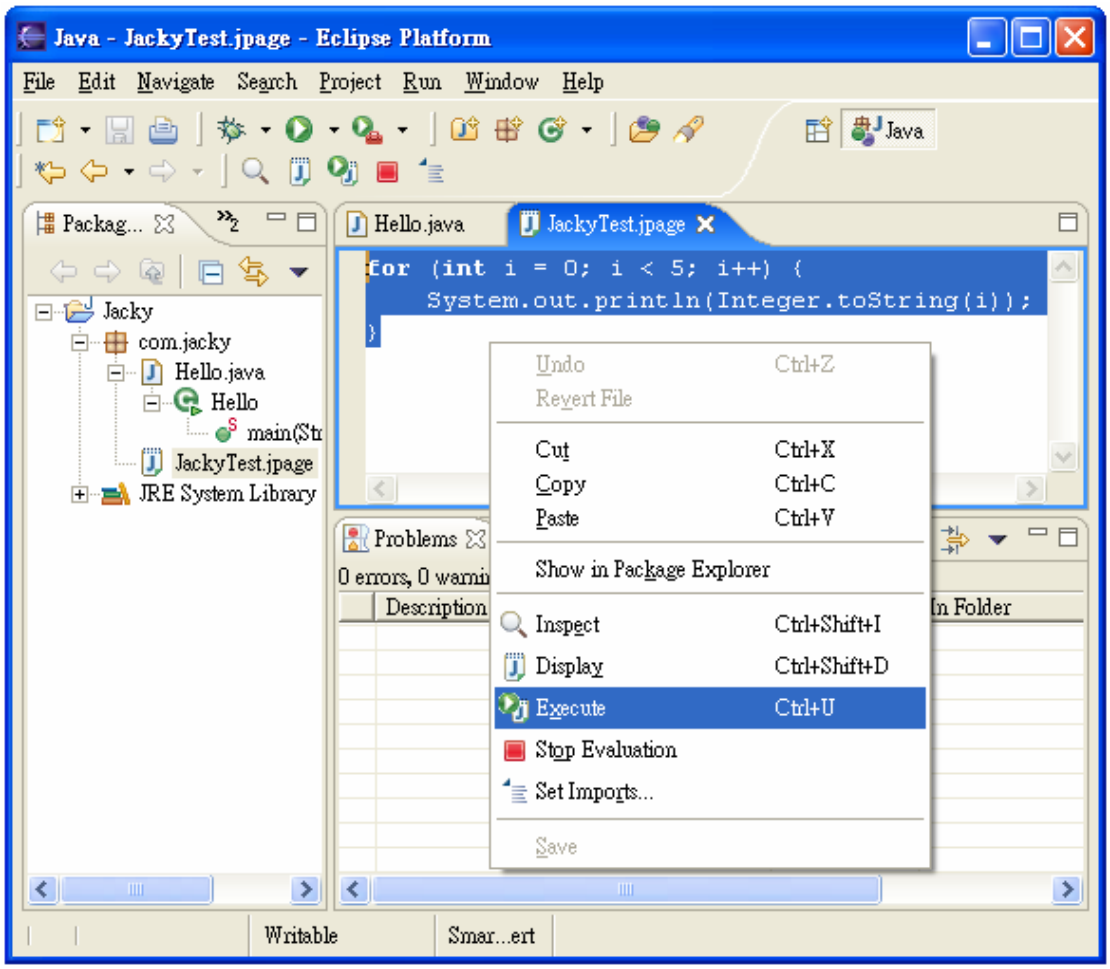


圖 4.14

若需要匯入套件；

I. 在編輯器視窗內按右鍵，選 Set Import...

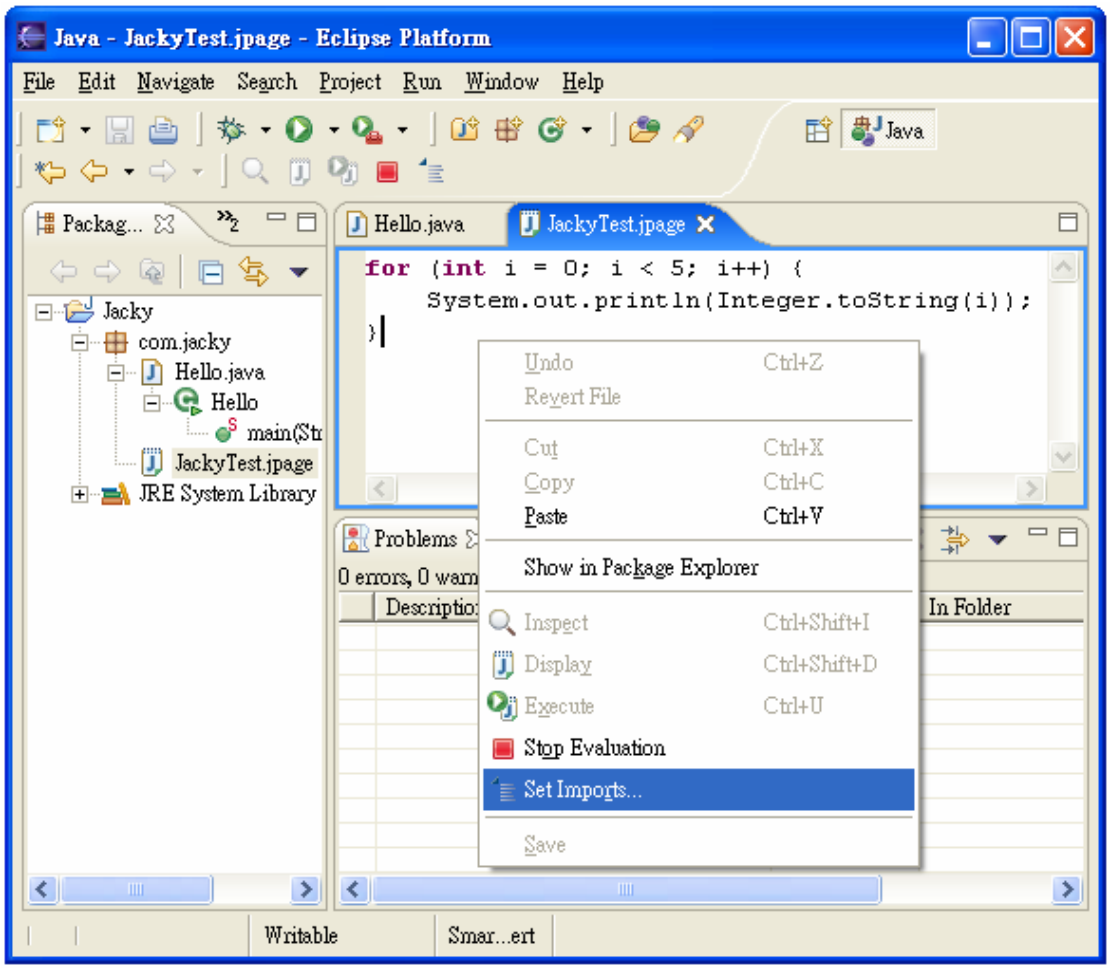


圖 4.15

II. 在 Java Snippet Imports 視窗中，按 Add Packages 的按鈕

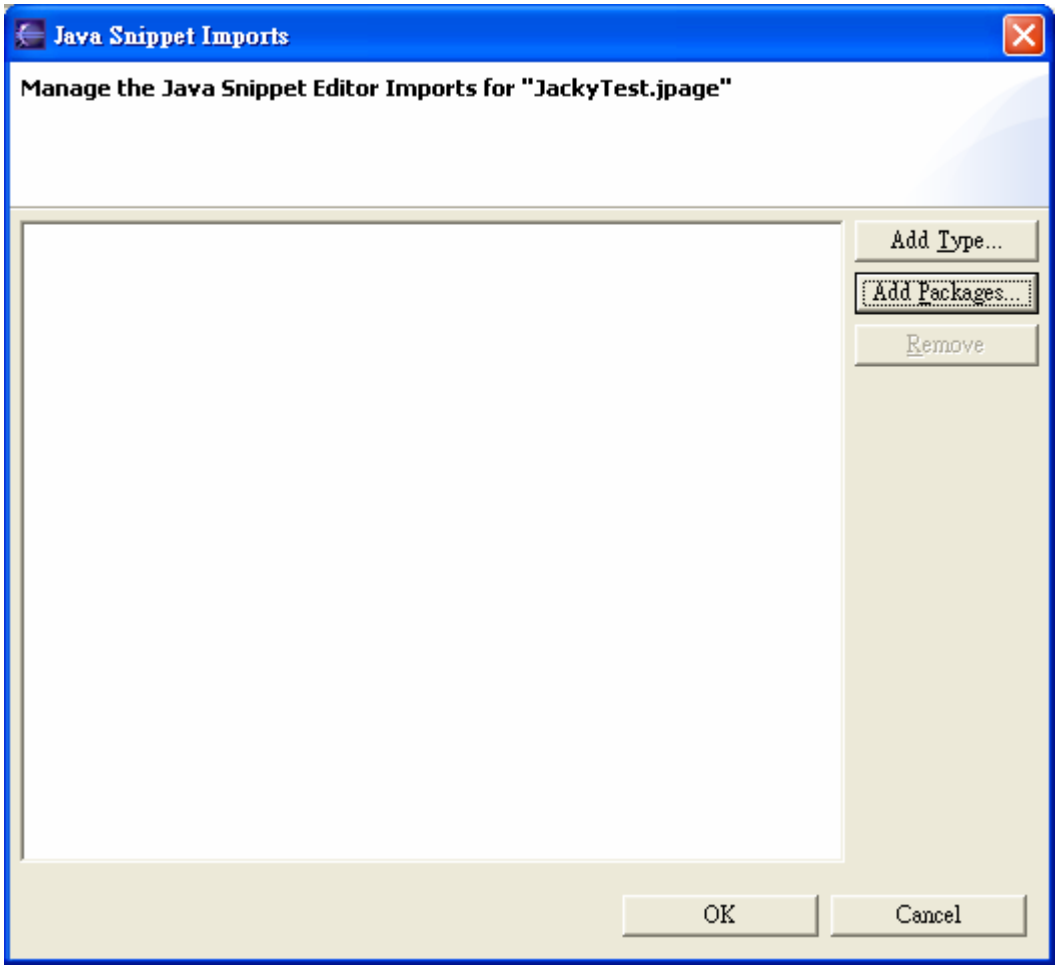


圖 4.16

III. 在 Add Packages as Imports 的視窗中，挑選要 import 的 package

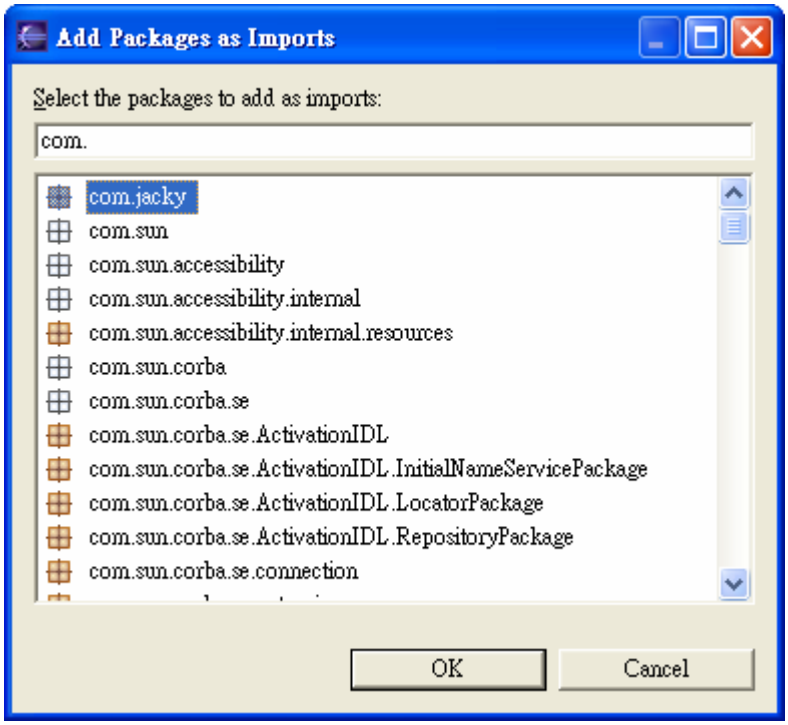


圖 4.17

IV. 按 OK

4.6 自訂開發環境

4.6.1 程式碼格式

在「Window」→「Preferences」→「Java」→「Code Style」→「Code Formatter」

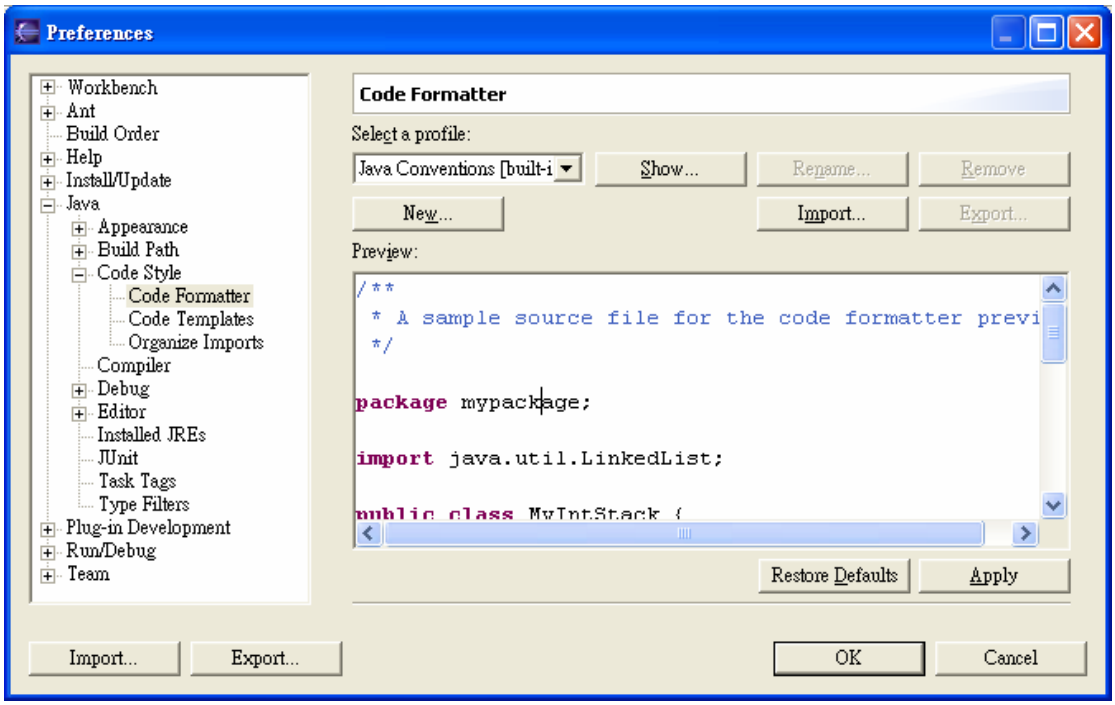


圖 4.18

按 Show 的按鈕，出現 Show Profile 的視窗，裡面的各個頁籤，可以設定 Java Code Style

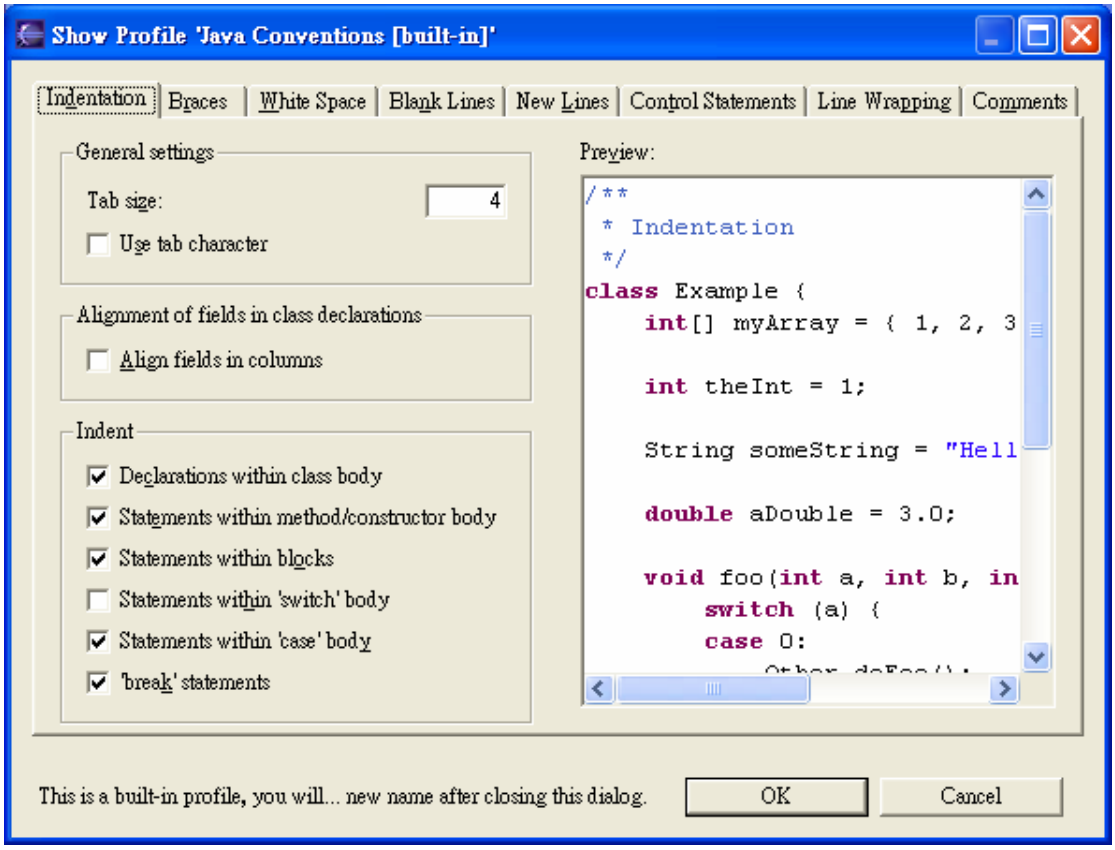


圖 4.19

設定完成後，可以 Export 成一個檔案；以利下次設定 Java Code Style 時，可以利用 Import 的方式，產生一致的程式風格。

4.6.2 程式碼產生模板

在開發 Java 時，可以把常用的流程控制建構式或是常用到的 API，建立成一個模板，可以加速程式開發。接下來以 System.out.println() 為例子，來說明如何建立模板：

I. 「Window」→「Preferences」→「Java」→「Editor」→「Templates」

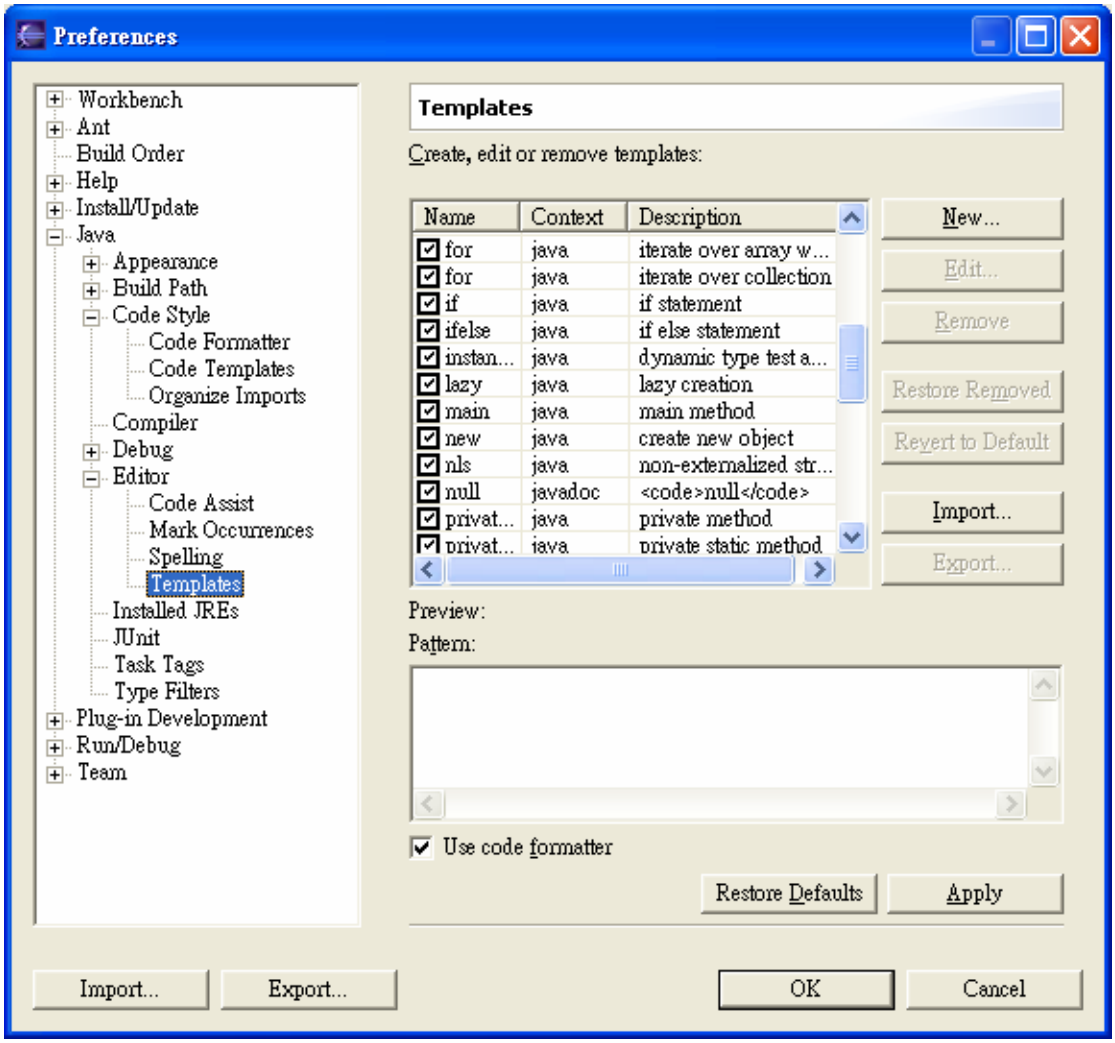


圖 4.20

II. 在 Preferences 視窗按 New 的按鈕

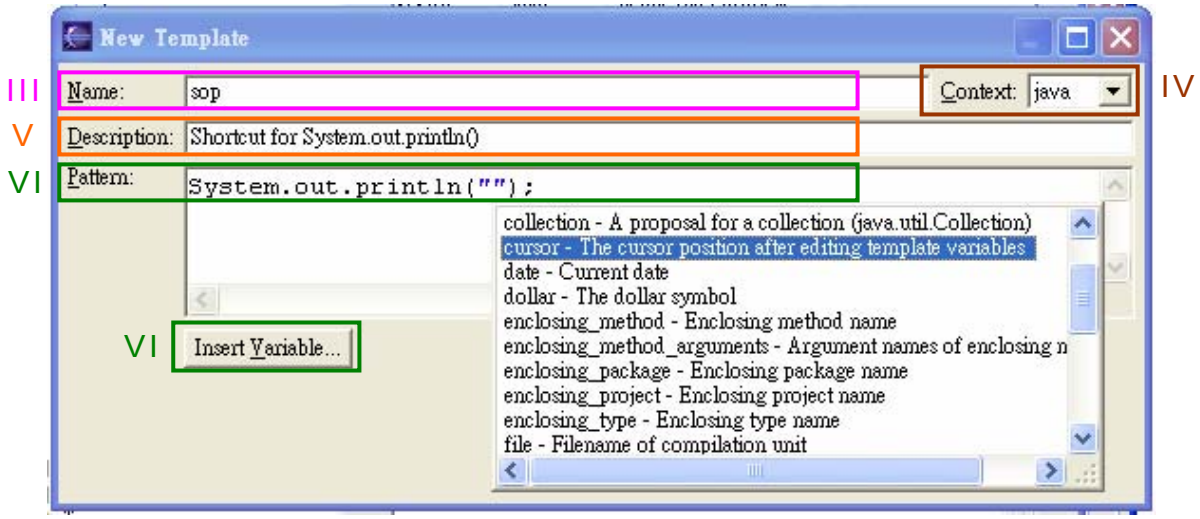


圖 4.21

III. 在 Name 的欄位輸入自己想要的名稱

IV. Context 選 java

V. 在 Description 的欄位輸入簡短的說明

VI. 在 Pattern 的欄位輸入 System.out.println("")後；把游標移到兩個雙引號的中間，再按下面 Insert Variable 的按鈕，選擇 cursor

VII. 再按兩次 OK

這裡的${cursor}變數代表插入模板的程式碼後，游標所在的位置。

使用此新模板，打 s(或是 sop)再按 Alt - /，從清單中選 sop，再按 Enter 即可。

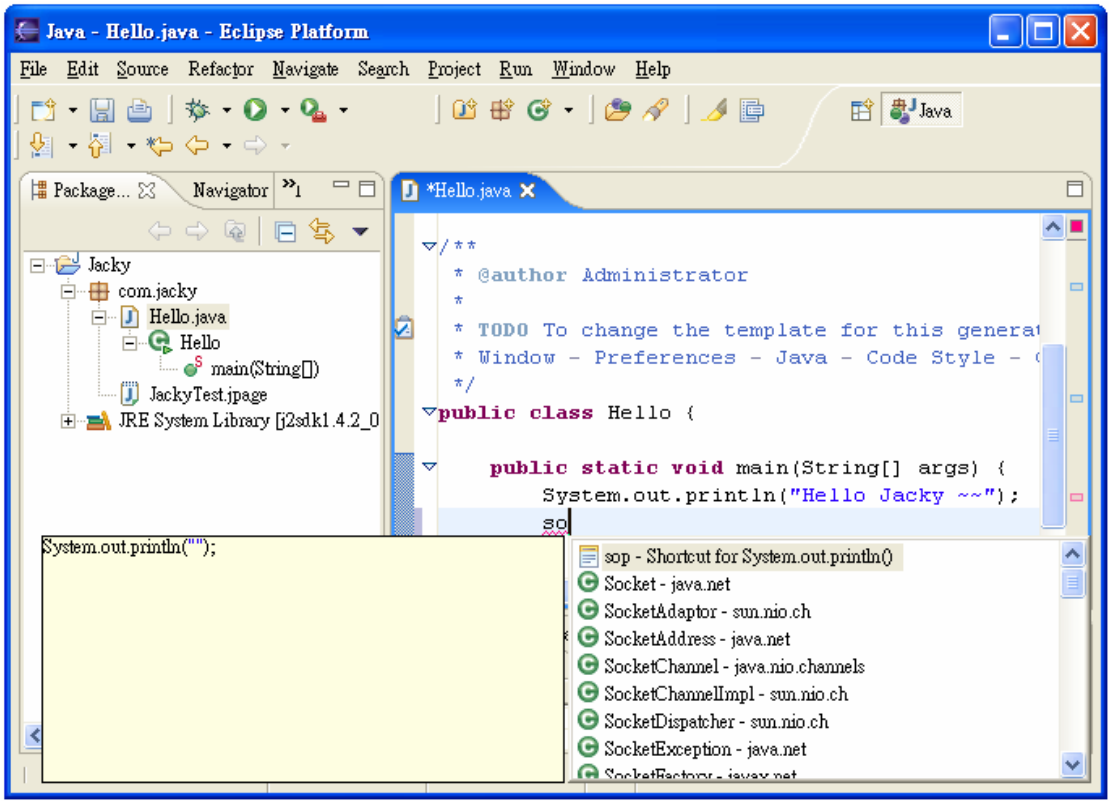


圖 4.22

4.6.3 Javadoc 註解

編輯新增類別後出現的文字。移除” To change the template for this generated...”這段前置文字，並自行擴充 Javadoc 註解。

I. 「Window」→「Preferences」→「Java」→「Code Style」→「Code Templates」

II. 選右邊畫面的「Code」→「New Java files」，按 Edit 按鈕

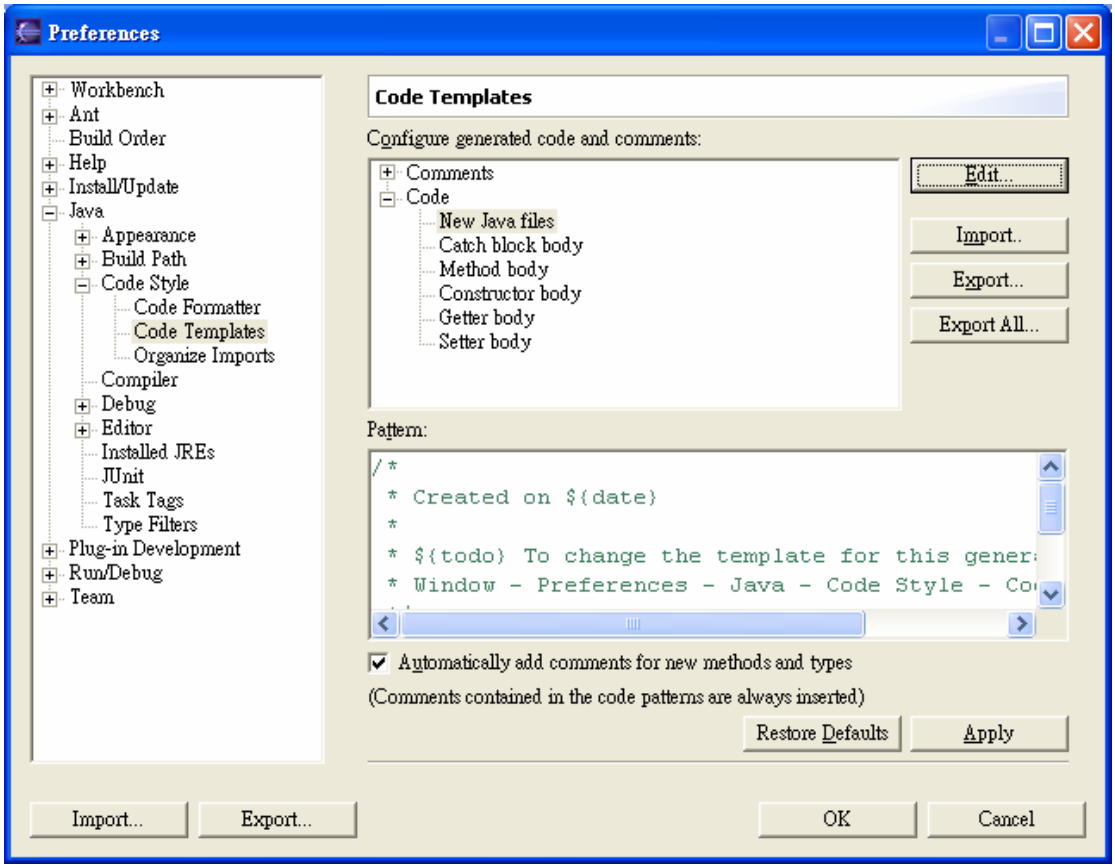


圖 4.23

III. 修改成需要的格式

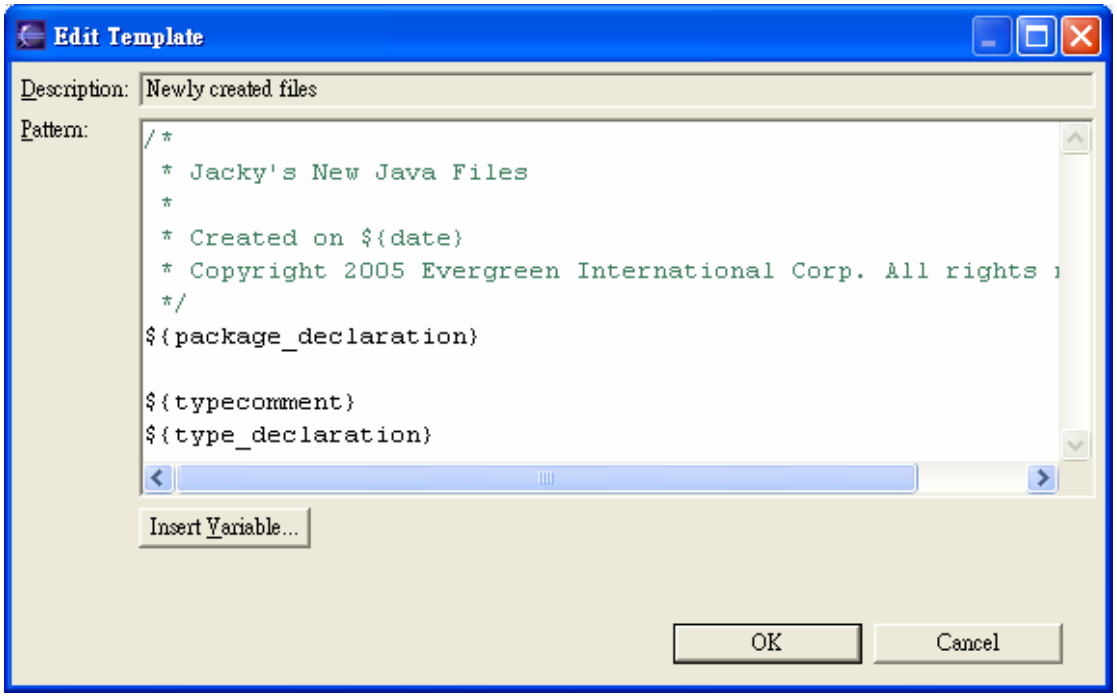


圖 4.24

IV. 按 OK

除了 New Java file 的模板外，還需要修改另一個模版-類型註解(Typecomment)。

I. 「Window」→「Preferences」→「Java」→「Code Style」→「Code Templates」

II. 選右邊畫面的「Comments」→「Types」，按 Edit 按鈕

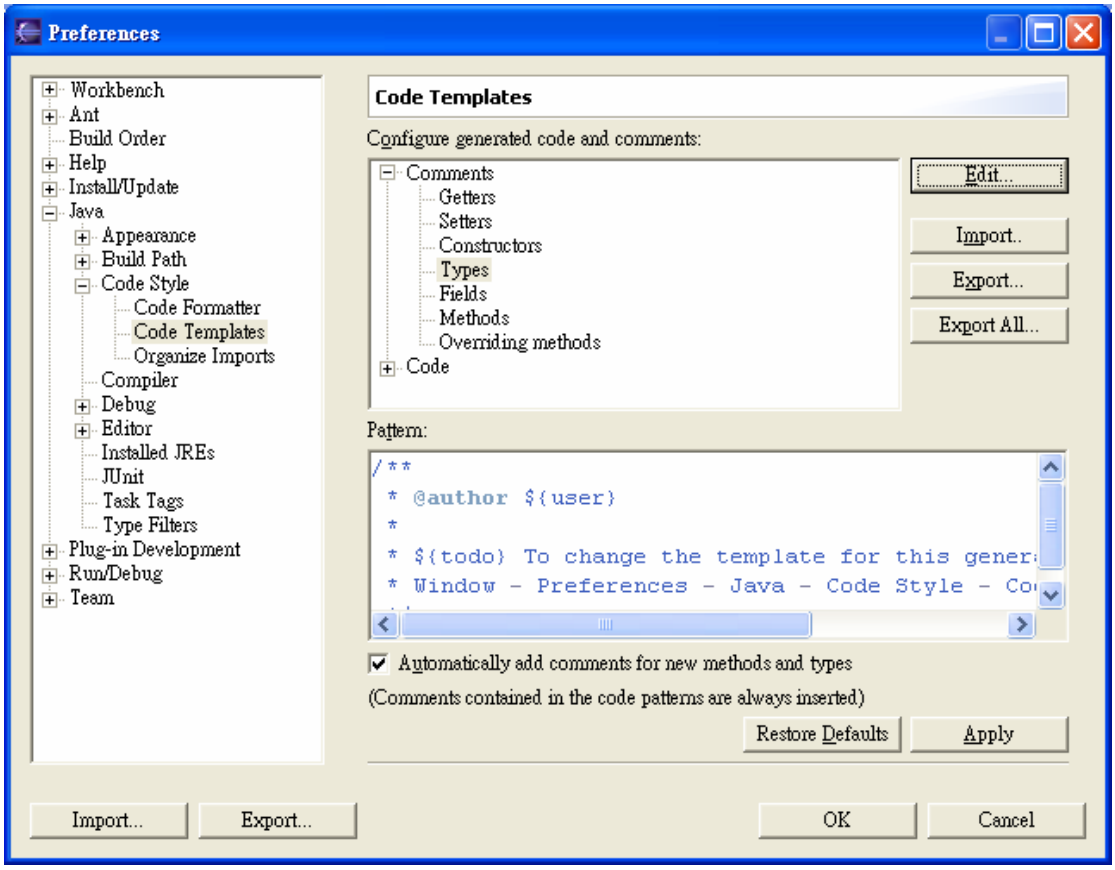


圖 4.25

III. 修改成需要的格式

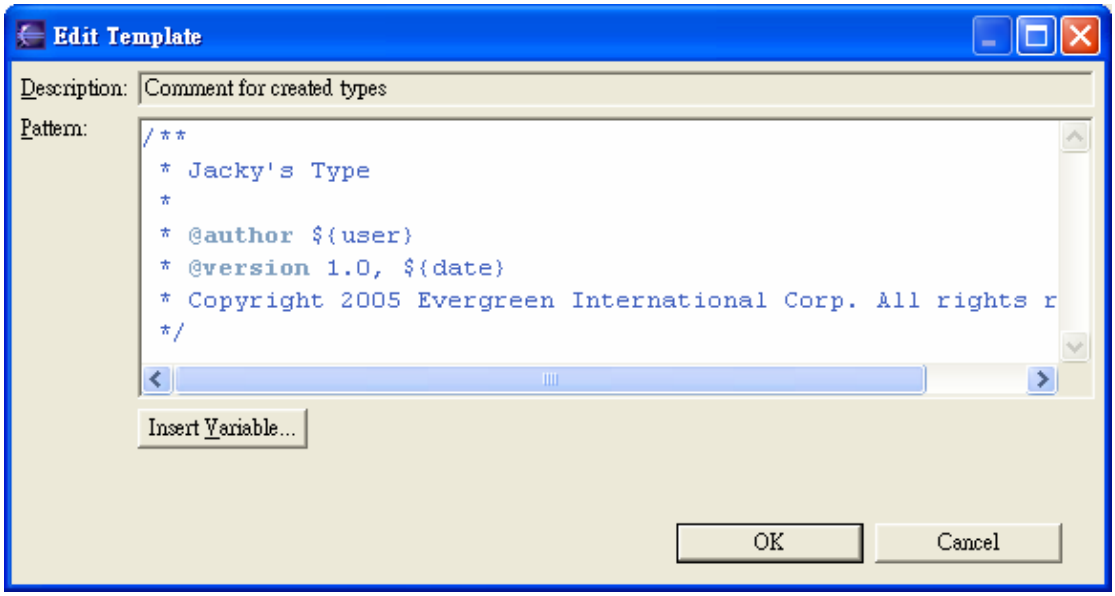


圖 4.26

IV. 按 OK

往後新增的類別檔案，就會套用現在註解。

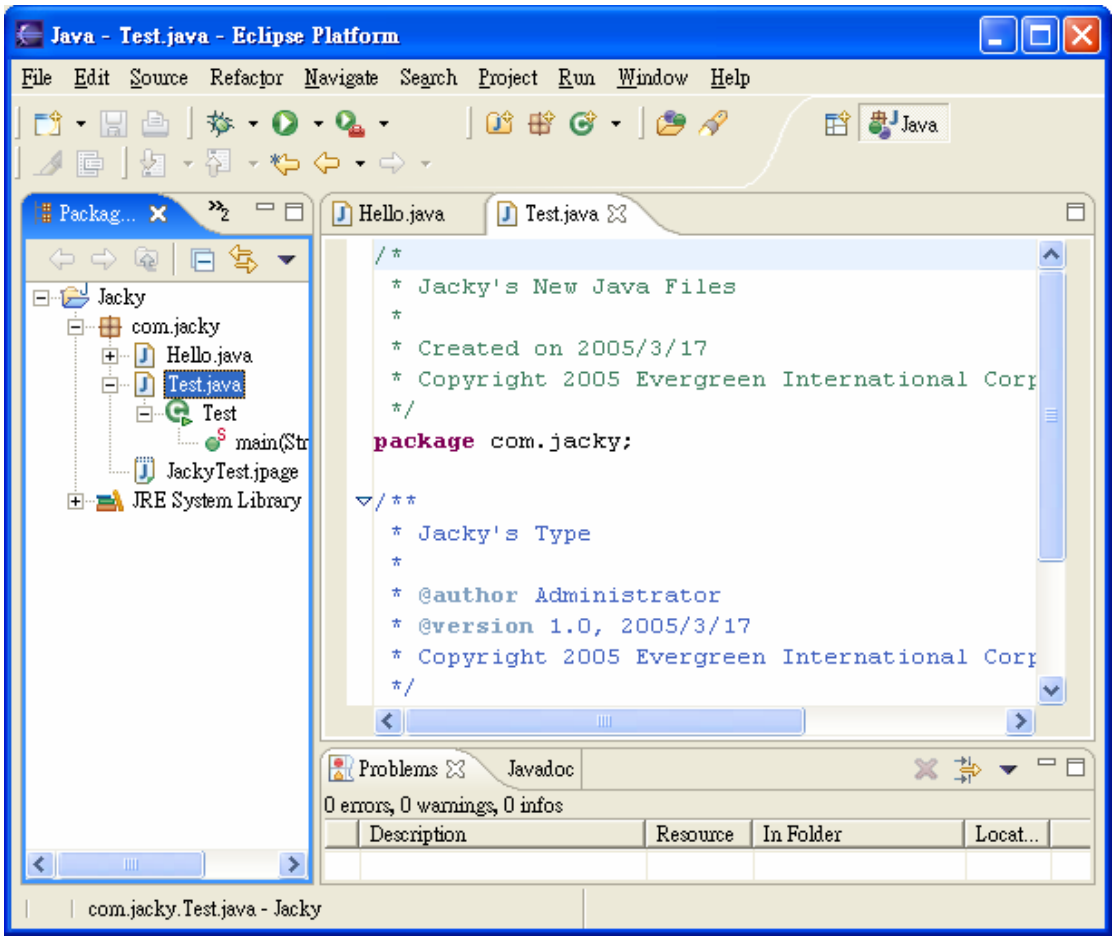


圖 4.27

Javadoc也可以產生模板，做法跟 4.6.2 程式碼產生模板類似，差別在於Context改選javadoc。

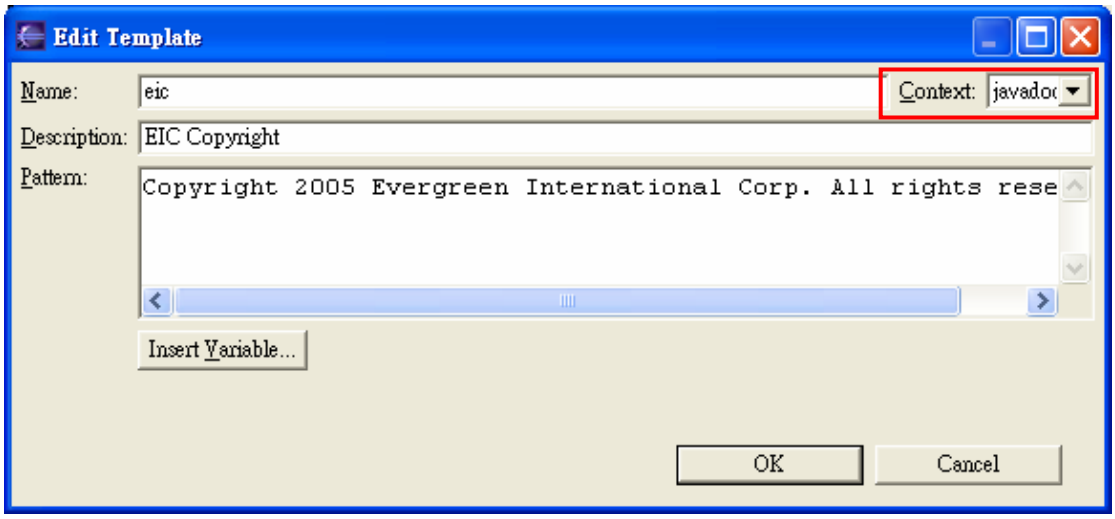


圖 4.28

在程式註解的地方，打 eic 再按 Alt - /，就可以出現清單可以選擇。

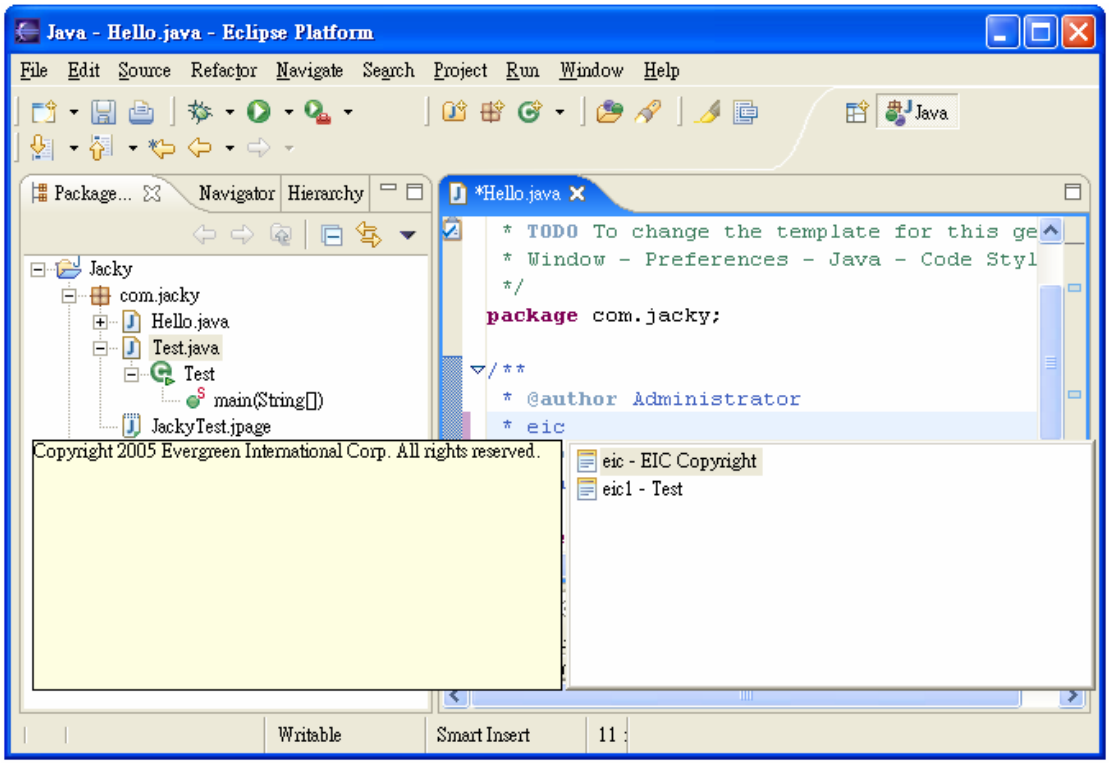


圖 4.29

4.7 產生 getter 與 setter

Java 編輯器可以為編譯單元內的類型欄位，產生存取元(accessors，也就是 getter 和 setter 的 method)。

I. 「Source」→「Generate Getter and Setter...」
(或是在 Java 編輯器按右鍵，「Source」→「Generate Getter and Setter...」)

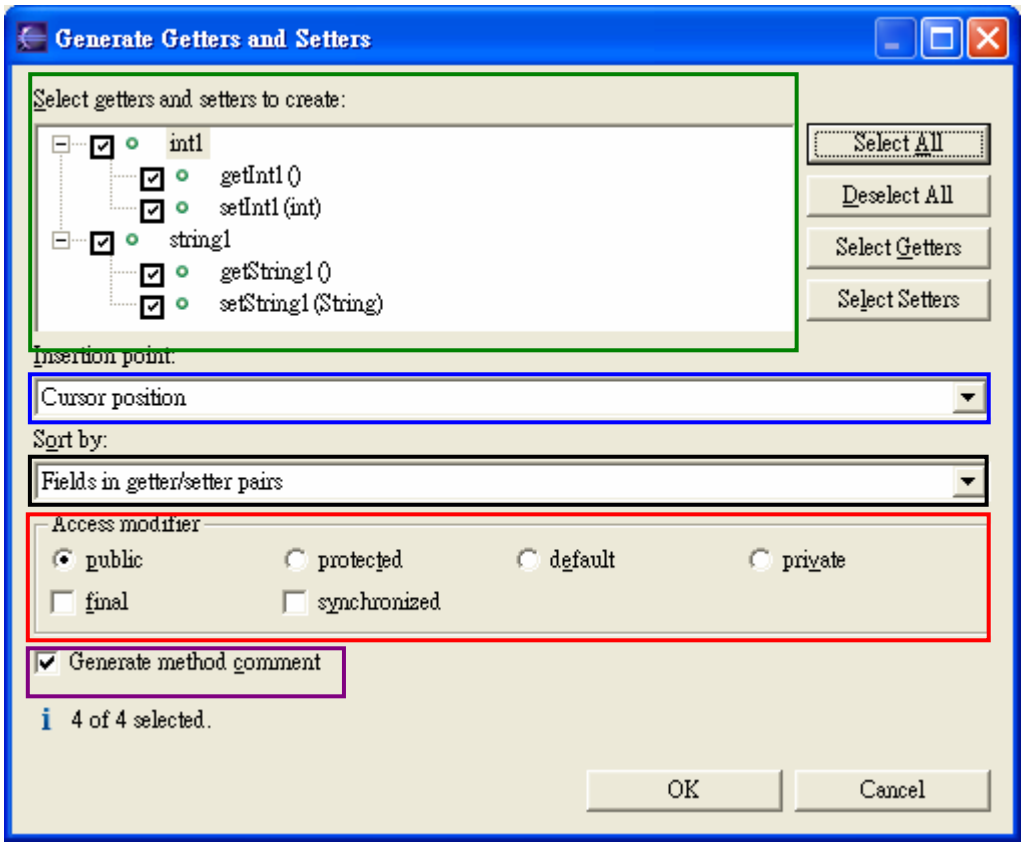


圖 4.30

II. 挑選哪些需要建立 getter 和 setter 的 method
III. 選擇 method 要建立的地方
IV. 排序的方式
V. 選擇 Access modifier
VI. 選擇是否需要建立註解
VII. 按 OK

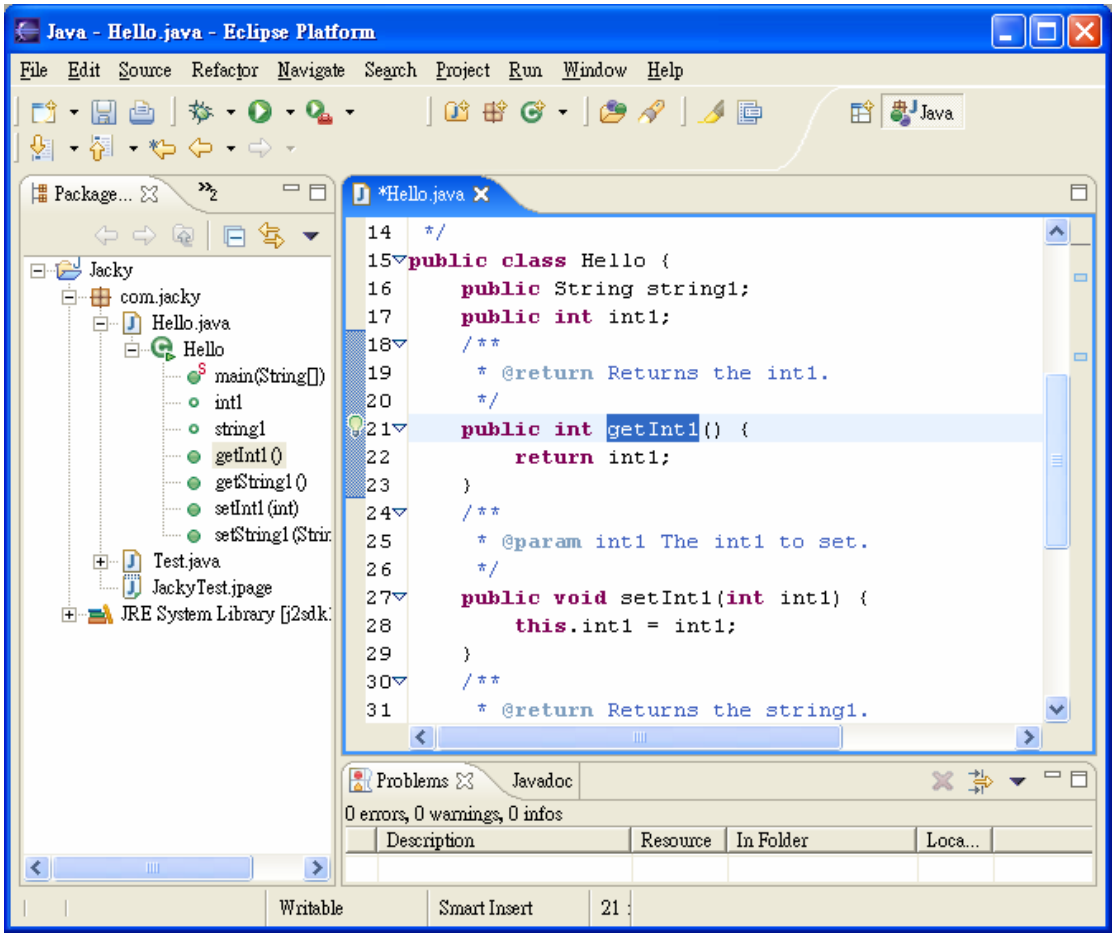


圖 4.31

4.8 建立 JAR 檔案

4.8.1 建立新的 JAR 檔案

如果要在工作台中建立新 JAR 檔，請執行下列動作：

I. 在「Package Explorer」中，可以選擇性地預選一或多個要匯出的 Java 元素。(在步驟 IV 中，這些會在 JAR Package Specification 精靈頁面中自動選出。)

II. 從快速功能表或從功能表列的 File 功能表，選取 Export。

III. 選取 JAR file，然後按一下 Next。

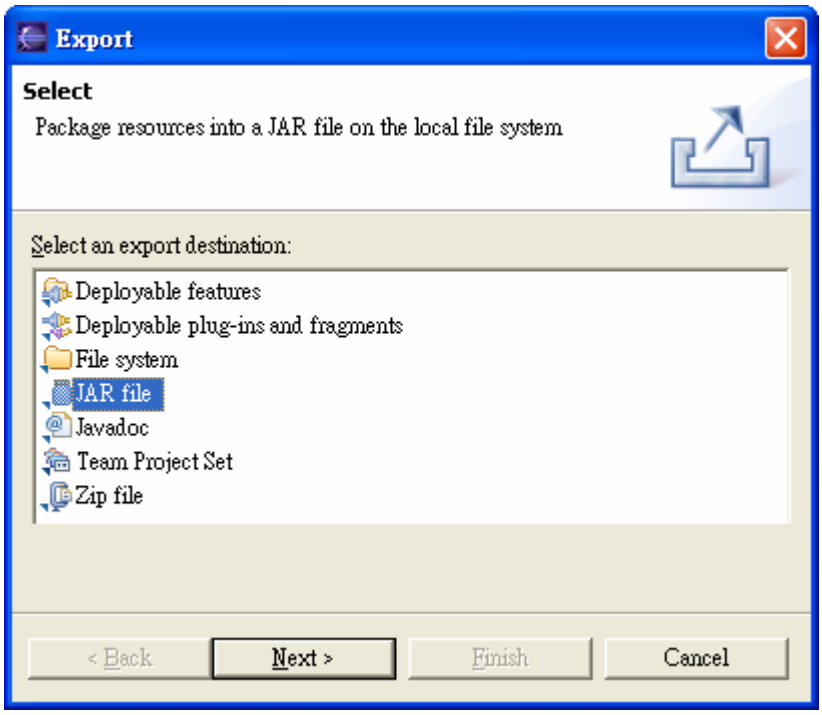


圖 4.32

IV. 在 JAR Package Specification 頁面的 Select the resources to export 欄位中，選取要匯出的資源。

V. 選取適當的勾選框，以指出想 Export generated class files and resourcess 或 Export java source files and resources。**附註：**這兩種情況皆會匯出所選的資源。

VI. 在 Select the export destination 欄位中，輸入或按一下 Browse 以選取 JAR 檔的位置。

VII. 選取或清除 Compress the contents of the JAR file 勾選框。

VIII. 選取或清除 Overwrite existing files without warning 勾選框。如果清除這個勾選框，則會提示確認是否要更換每一個將被改寫的檔案。

IX. **附註：**在撰寫 JAR 檔、JAR 說明與 Manifest 檔時，會套用改寫選項。

X. 有兩項選擇：

- 按一下 Finish 來立即建立 JAR 檔。
- 按一下 Next，使用「JAR 套裝選項」頁面，以設定進階選

項，建立 JAR 說明，或變更預設 manifest。

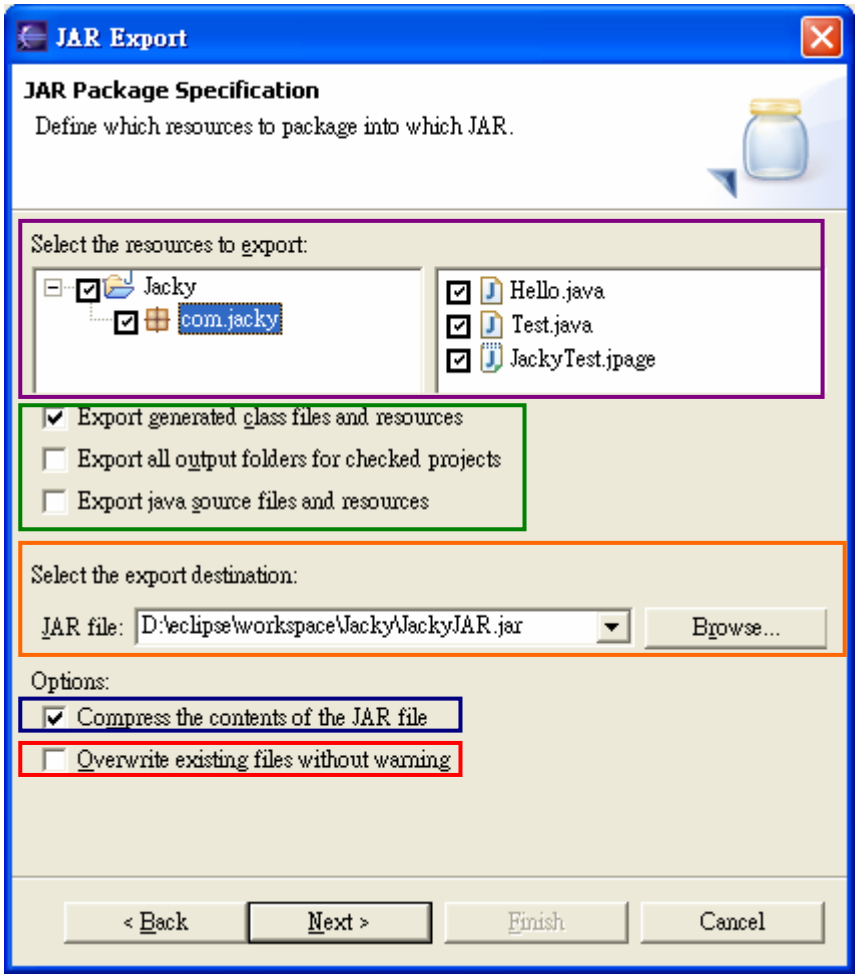


圖 4.33

4.8.2 設定進階選項

I. 遵循建立 JAR 檔的程序進行，但在最後一個步驟中按一下 Next，以移至「JAR 套裝選項」頁面。

II. 如果想儲存 JAR 檔說明，請選取 Save the description of this JAR in the workspace 勾選框。

III. 編譯器能產生 CLASS 檔，即使程式檔中有錯誤。可以選擇排除內含編譯錯誤的 CLASS 檔（但非程式檔）。如果有啟用報告特性的話，則結束時將會報告這些檔案。

IV. 可以選擇排除內含編譯警告的 CLASS 檔（但非程式檔）。在結束時將會報告這些檔案。

附註：這個選項不會自動排除內含編譯錯誤的類別檔。

V. 可以選擇包含來源資料夾路徑，方法是選取 Create source folder structure 勾選框。

VI. 如果希望讓匯出作業在建立 JAR 檔前先執行建置，請選取 Build projects if not built automatically 勾選框。

VII. 按一下 Finish，以立即建立 JAR 檔；如果想變更預設 manifest，請按一下 Next。

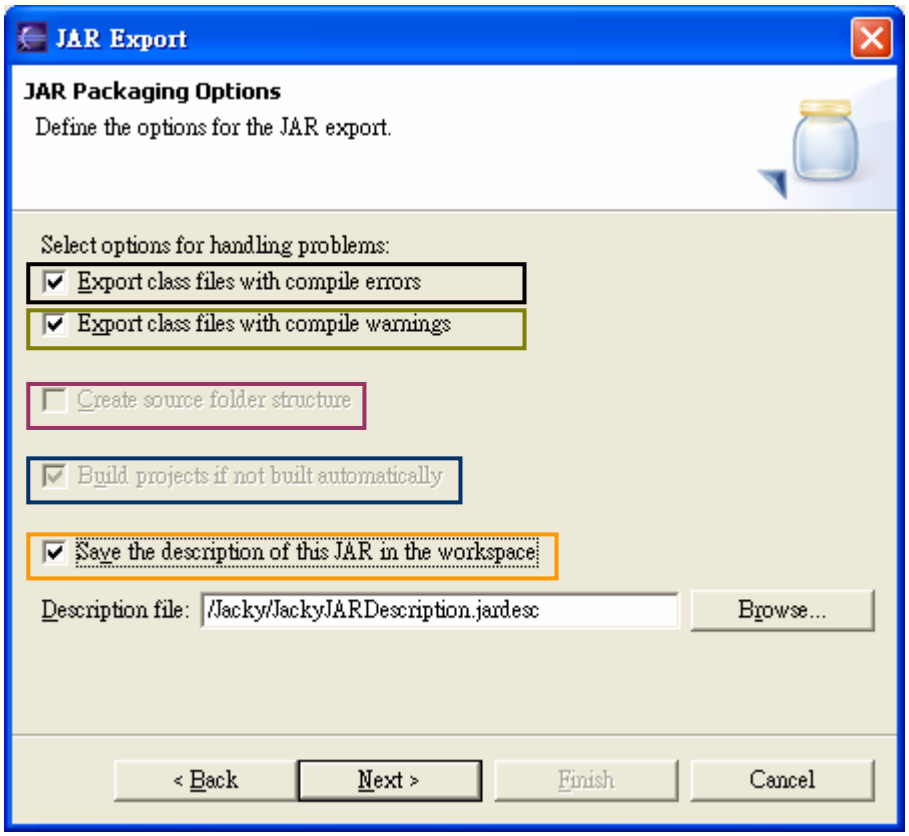


圖 4.34

4.8.3 定義 JAR 檔的 manifest

可以直接在精靈中定義 JAR 檔 Manifest 的重要部分，或使用

已存在於工作台中的 Manifest 檔。

建立新 Manifest

I. 遵循建立 JAR 檔的程序進行，但在最後一個步驟中按一下 Next，以移至「JAR 套裝選項」頁面。

II. 設定任何要設定的進階選項，再按一下 Next，移至「JAR manifest 規格」頁面中。

III. 如果尚未選取，請按一下 Generate the manifest file 按鈕。

IV. 這時可以選擇將 Manifest 儲存在工作台中。這會儲存 Manifest 以便日後使用。按一下 Save the manifest in the workspace，然後按一下 Manifest file 欄位旁的 Browse，以指定 Manifest 的路徑與檔名。

V. 如果在前一步驟中決定儲存 Manifest 檔，並在先前的精靈頁面中選擇儲存 JAR 說明，可以選擇在 JAR 說明中重複使用 Manifest（做法是選取 Reuse and save the manifest in the workspace 勾選框）。這表示當從 JAR 說明重建 JAR 檔時，將會使用所儲存的檔案。如果想先修改或更換 Manifest 檔，然後再從說明重建 JAR 檔，請善用這個選項。

VI. 可以選擇密封 JAR，以及選擇性地將某些套件排除在密封之外，或指定密封套件清單。依預設，不會進行任何密封。

VII. 按一下 Main Class 欄位旁的 Browse 按鈕來指定應用程式的進入點。

附註：如果的類別不在清單中，表示在一開始時忘了選取它。

VIII. 按一下 Finish。這會建立 JAR，並選擇性地建立 JAR 說明與 Manifest 檔。

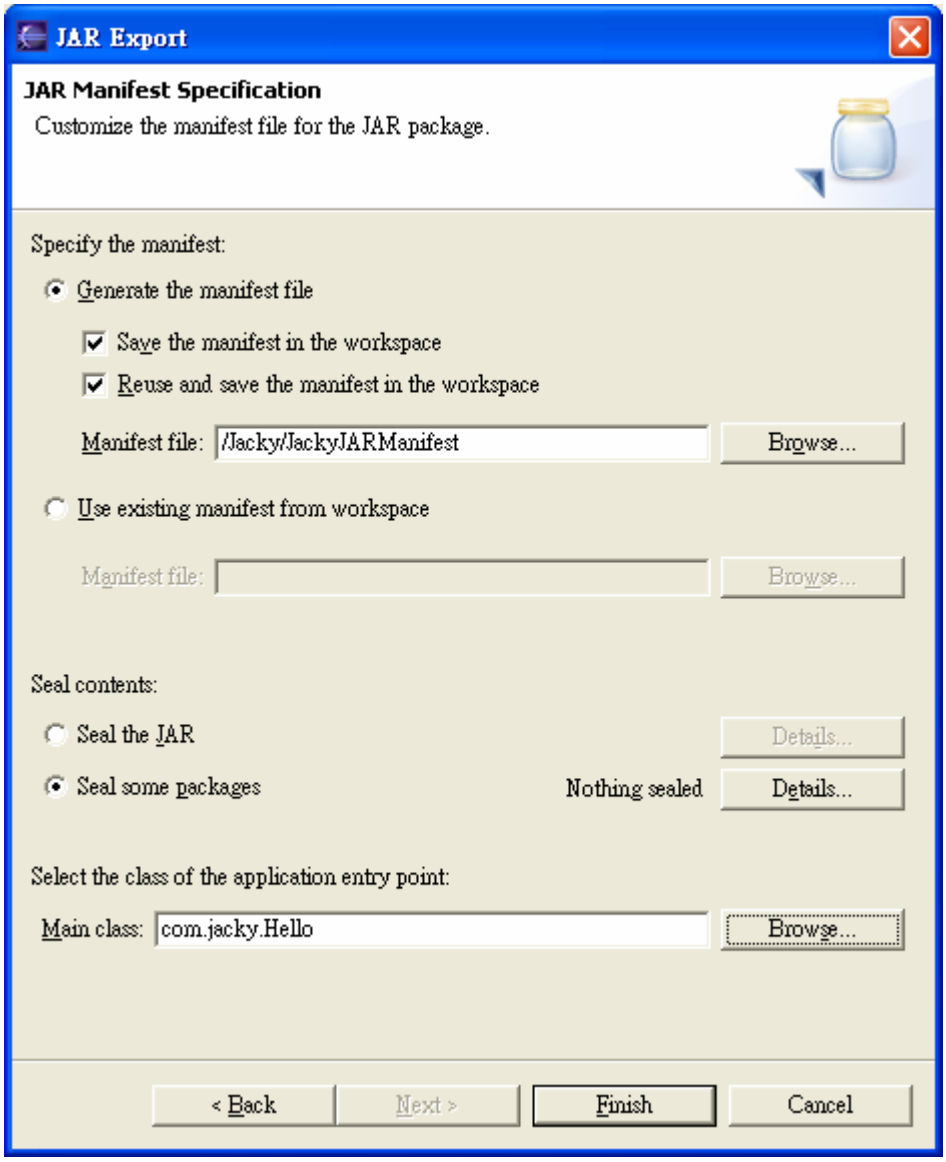


圖 4.35

使用現有的 manifest

可以使用已存在於工作台中的現有 Manifest 檔。

I. 遵循建立 JAR 檔的程序進行，但在最後一個步驟中按一下 Next，以移至「JAR 套裝選項」頁面。

II. 設定任何要設定的進階選項，再按一下 Next，移至「JAR manifest 規格」頁面中。

III. 按一下 Use existing manifest from workspace 圓鈕。

IV. 按一下 Browse 按鈕來從工作台中選取 Manifest 檔。

V. 按一下 Finish。這會建立 JAR，並選擇性地建立 JAR 說明。

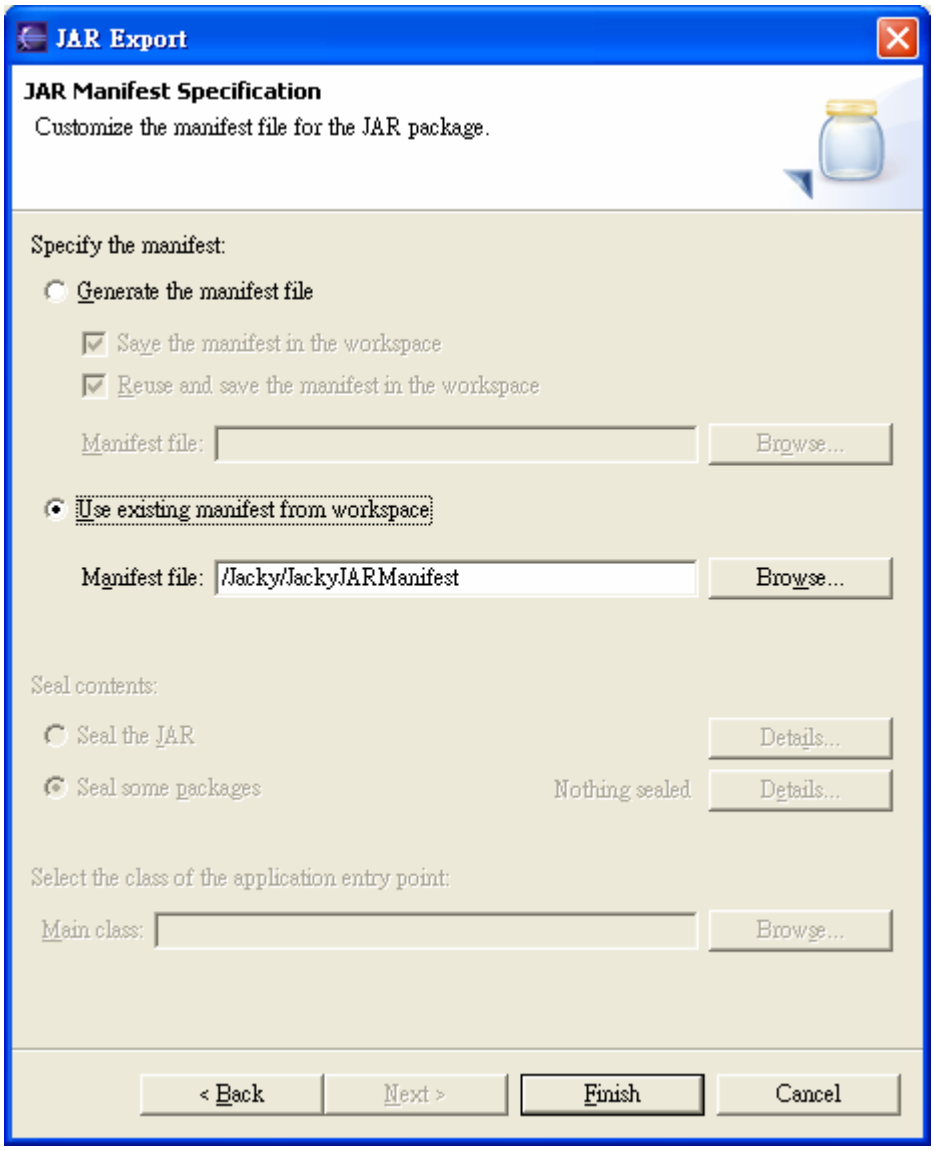


圖 4.36

4.8.4 重新產生 JAR 檔

可以使用 JAR 檔說明來重新產生先前所建立的 JAR 檔。

從選項的蹦現功能表中，選取 Create JAR。這時會重新產生 JAR 檔。

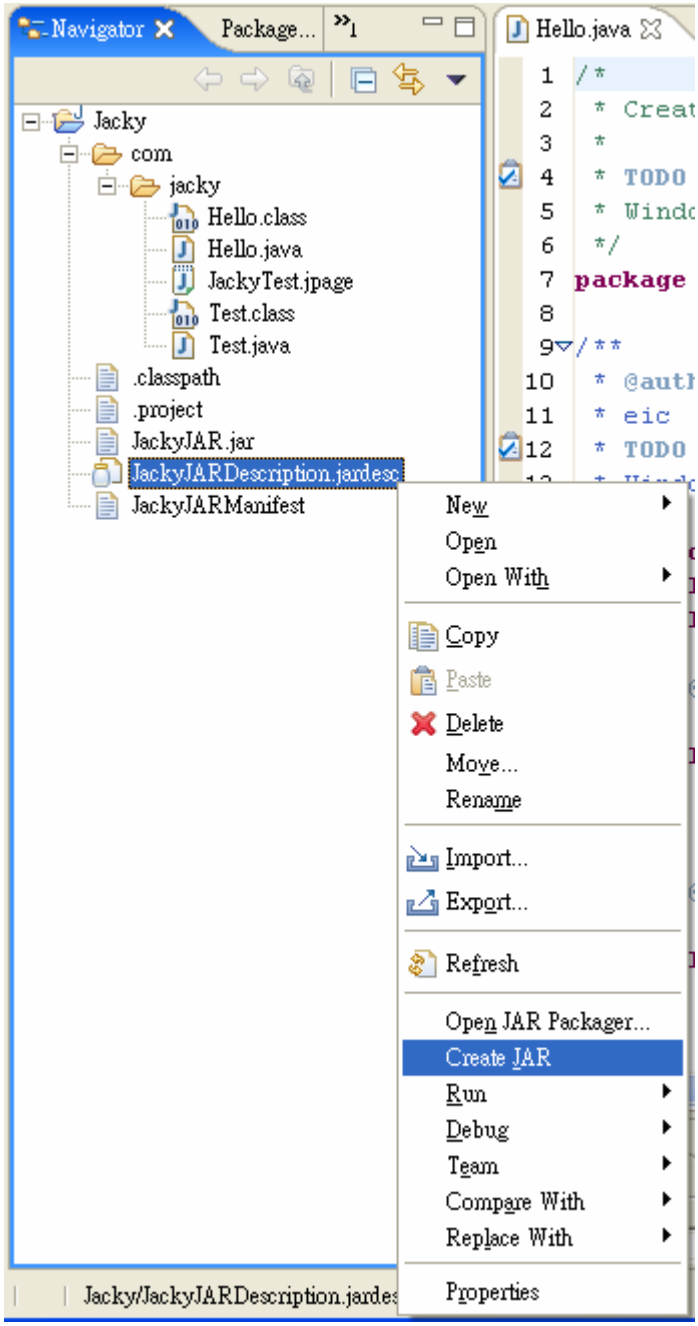

Navigator
Package...
Hello.java
Jacky
com
jacky
Hello.class
Hello.java
JackyTest.jpage
Test.class
Test.java
.classpath
.project
JackyJAR.jar
JackyJARDescription.jardesc
JackyJARManifest
New
Open
Open With
Copy
Paste
Delete
Move...
Rename
Import...
Export...
Refresh
Open JAR Packager...
Create JAR
Run
Debug
Team
Compare With
Replace With
Properties
Jacky/JackyJARDescription.jardes

圖 4.37

4.9.建立 Javadoc 文件

選取所要的套件、來源資料夾或專案組（內含一或多個元素），以便為其產生 Javadoc 文件。

執行下列之一，以開啟「匯出」精靈：

- 選取選擇項之蹦現功能表中的匯出；或
- 從功能表列中選取「File」→「Export...」。

在出現的對話框中，從清單中選取 Javadoc，並按下下一步。

4.9.1 選取產生 Javadoc 用的類型

I. 指定 Javadoc 指令的位置。

II. 在樹狀結構控制中，選取要的元素，以為其產生 Javadoc。

III. 使用為具有可見性的成員建立 Javadoc 下的圓鈕，選取可見性。

IV. 維持選取使用標準 doclet 圓鈕。
(或是自定 doclet)

V. 按下完成，為所選的元素產生 Javadoc，或按下一步，指定其他選項。

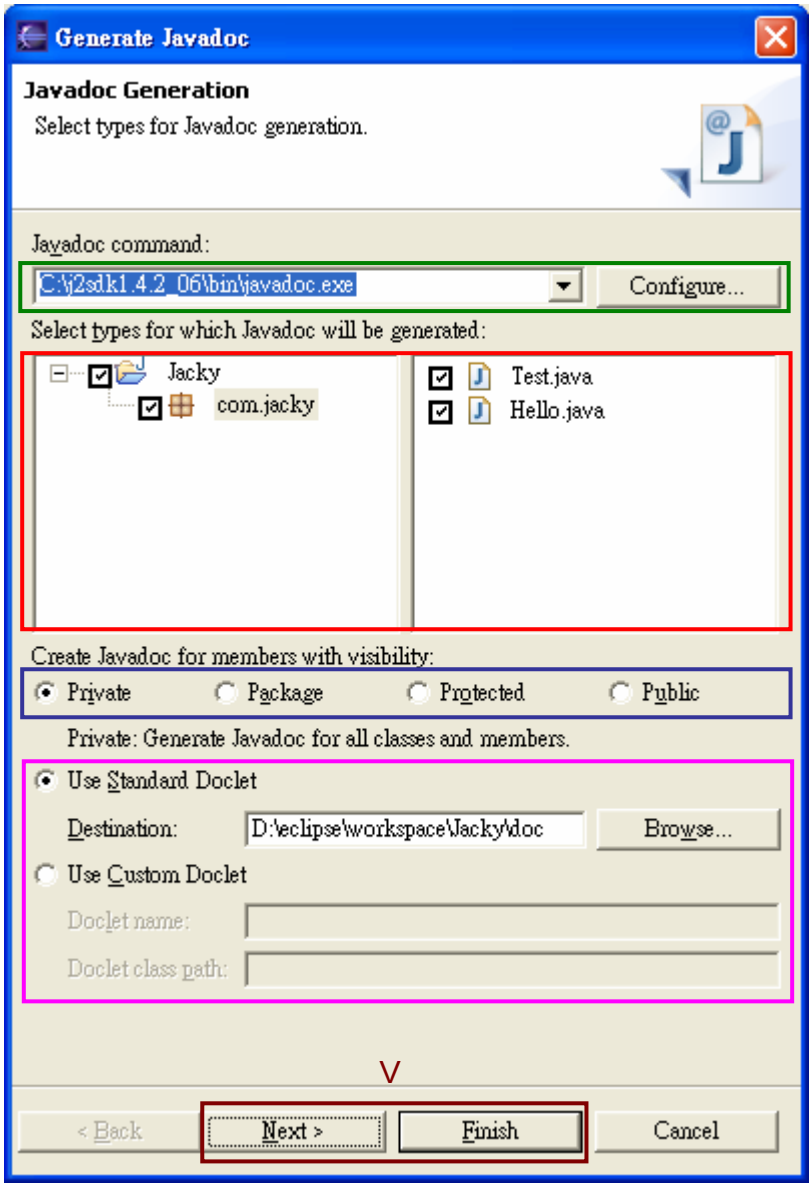


圖 4.38

4.9.2 爲標準 doclet 配置 Javadoc 引數

I. 標題名稱

II. 請使用基本選項下的勾選框，以指定 Javadoc 選項。

可以使用闡明這些標示群組中的勾選框，以變更所要闡明的標示。

III. 如果想讓程式庫中之類別的參照，鏈結至程式庫的 Javadoc，

請在清單中選取該程式庫，並按下配置，以指定程式庫之 Javadoc 的位置。

IV. 選擇 CSS 檔案

V. 按下完成，以產生 Javadoc；或按下一步，以指定其他的 Javadoc 產生選項。

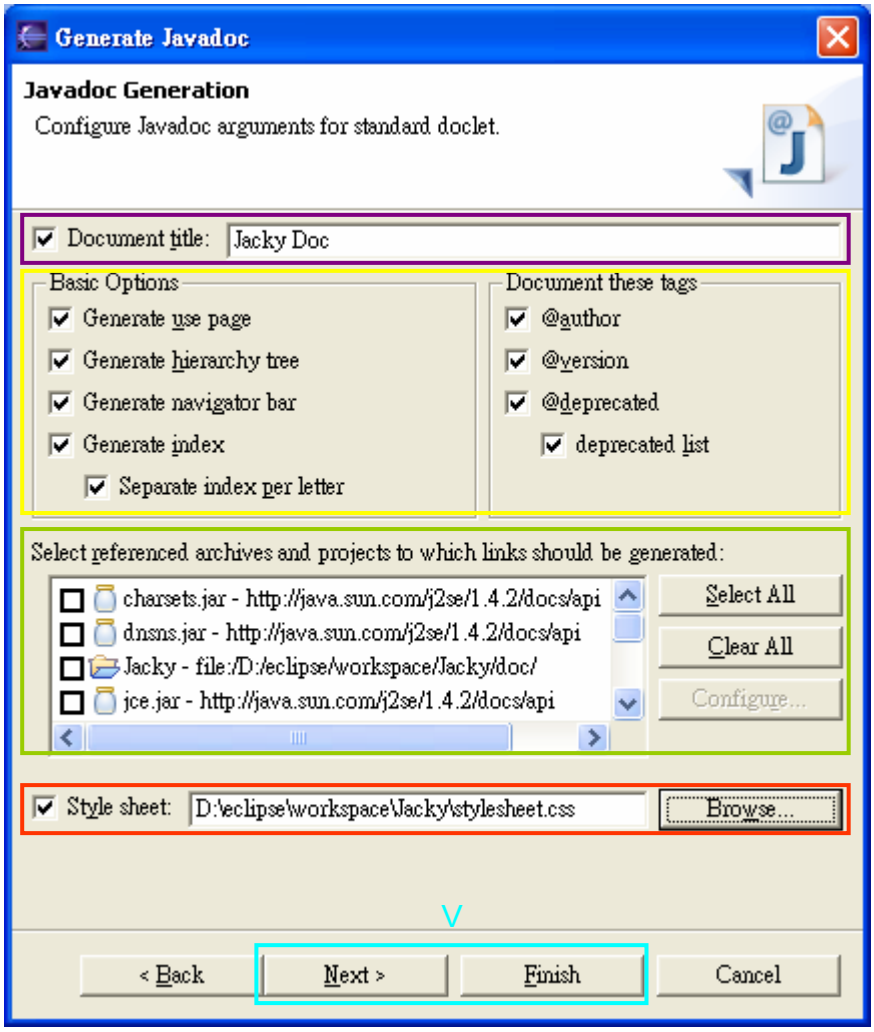


圖 4.39

4.9.3 配置 Javadoc 引數

I. 選取在瀏覽器中開啟產生的索引檔勾選框。

II. 可以為 Javadoc 指令指定多個特定選項，方法是在文字區中輸

入這些選項。

III. 選取將這個 Javadoc 匯出的設定儲存成 Ant Script 勾選框。

IV. 指定 Ant Script 的位置。

V. 按完成，以產生 Javadoc。

附註：Javadoc 指令產生的輸出（包括錯誤與警告）會顯示在「Console」視圖中。

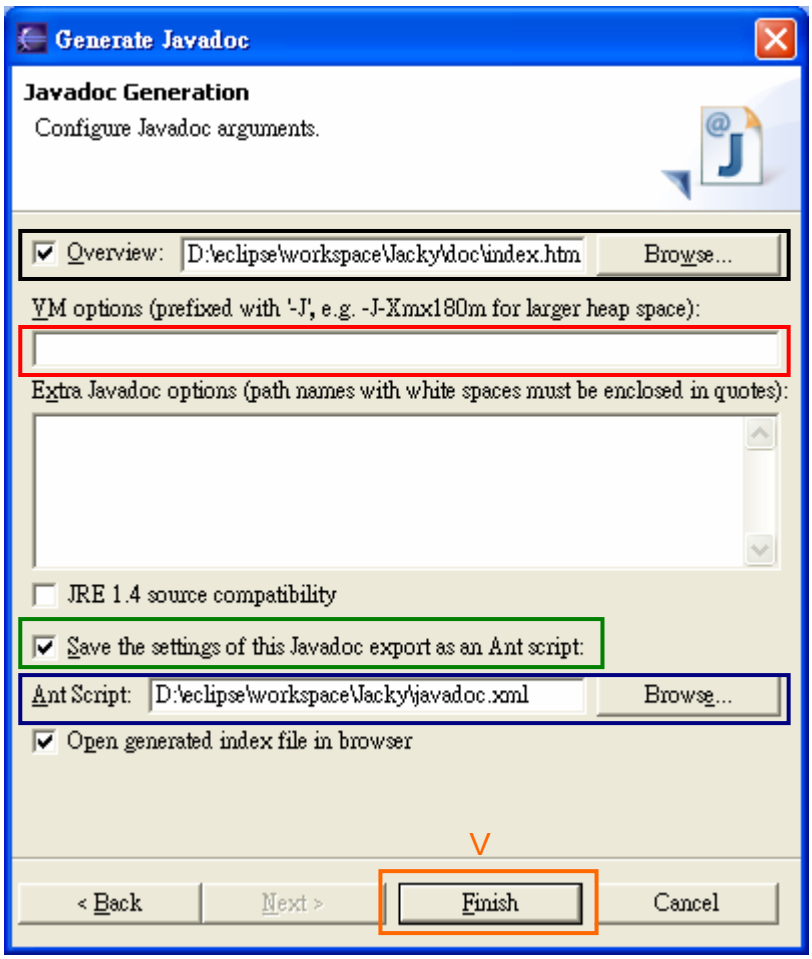


圖 4.40

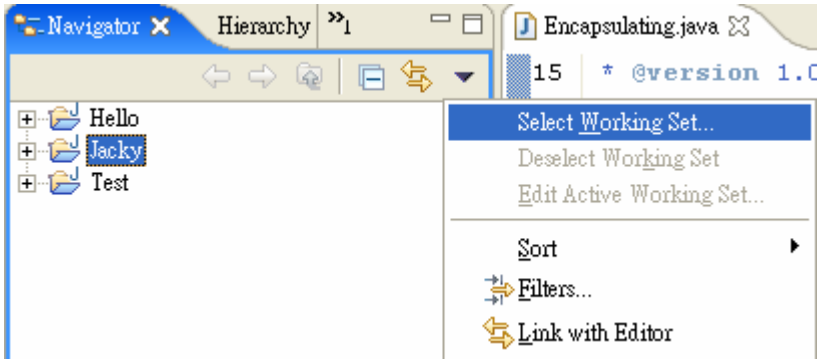
4.10 工作集(Working Sets)

工作集將元素分組，以顯示在視圖中，或對元素集進行作業。「Navigate」視圖使用工作集來限制顯示的資源集。如果在導覽器中選取了工作集，只會顯示資源、資源的子項，以及資源的母項。

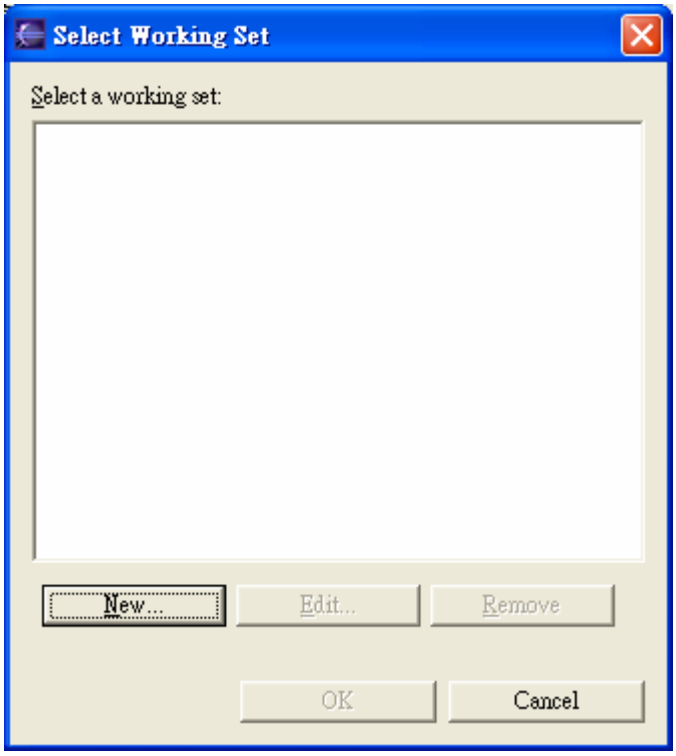
可以在「Tasks」視圖中使用工作集來限制作業的顯示，方法與「Navigate」視圖類似。在使用工作台搜尋機能時，可以使用工作集來限制可搜尋的元素集。不同的視圖提供不同的指定工作集方法。不過，它們通常使用下列工作集選項對話框來管理現有的工作集以及建立新的工作集。

4.10.1 新增工作集

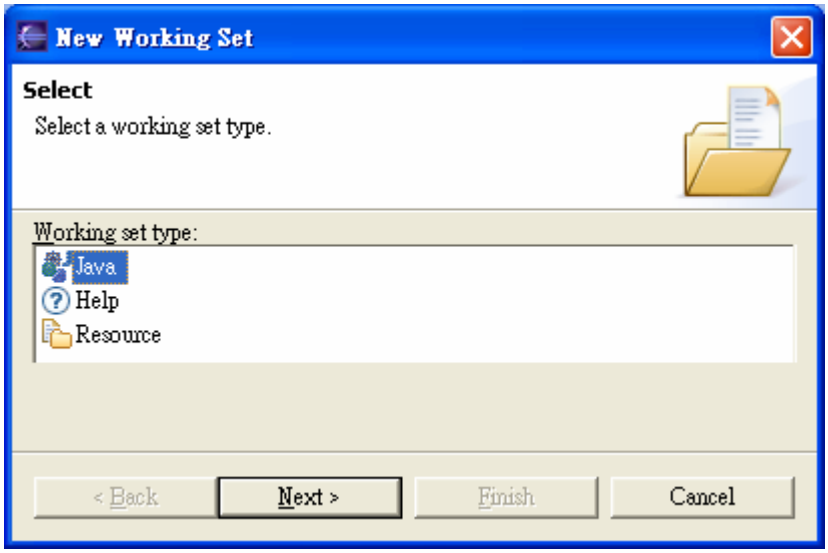
I. 在「導覽器」視圖的工具列上，按一下功能表按鈕，開啟顯示選項的下拉功能表。



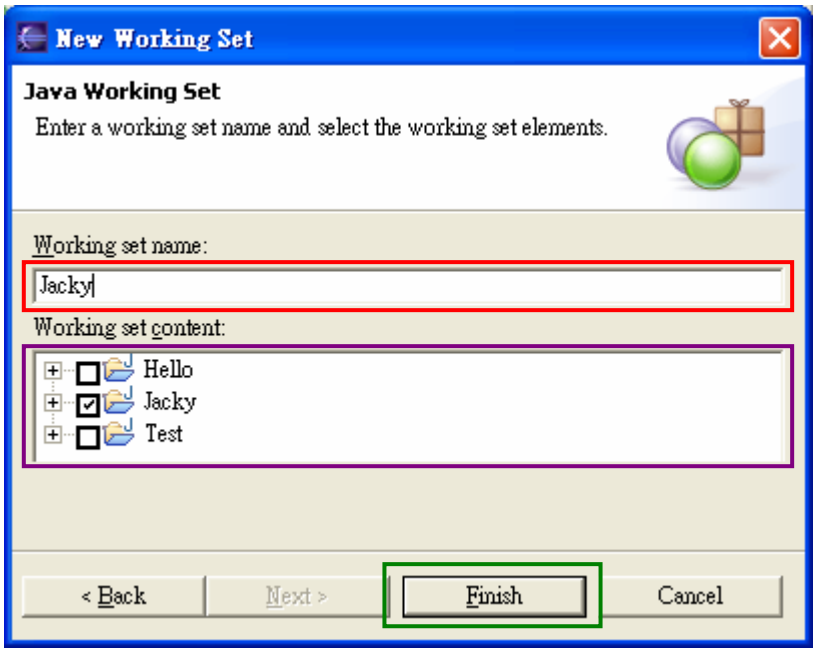
II. 選 Select Working Set 後，出現 Select Working Set 的視窗



III. 選擇 Java 後，按 Next



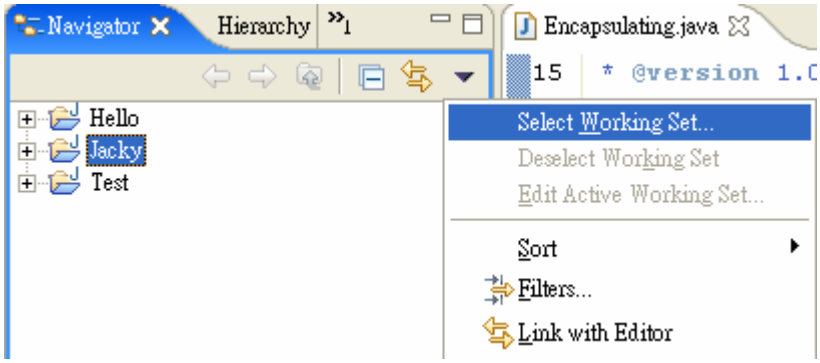
VI. 在 New Working Set 視窗輸入名稱和勾選所需的專案，最後按下 Finish 即可



附註：新建的資源不會自動併入作用中的工作集。如果它們是現有的工作集元素的子項，則會被隱含地併入工作集中。如果要在建立其他資源之後併入它們，必須明確地將它們新增至工作集。

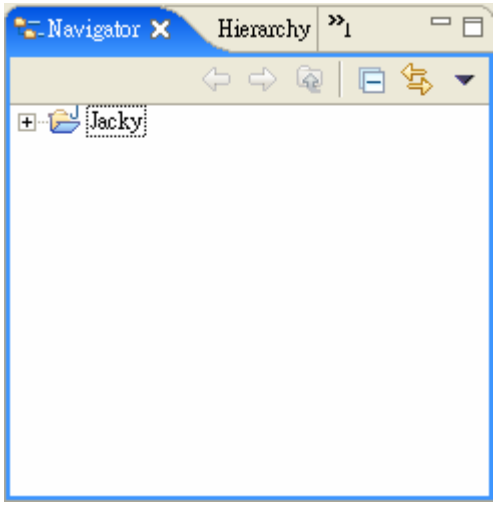
4.10.2 隱藏「導覽器」視圖中的檔案

I. 在「導覽器」視圖的工具列上，按一下功能表按鈕 ，開啟顯示選項的下拉功能表。



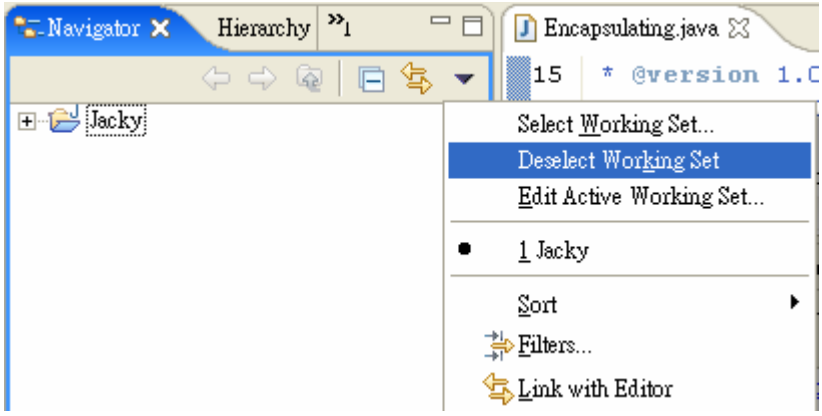
II. 選 Select Working Set 後，出現 Select Working Set 的視窗

III. 從清單中選取一個現有的工作集，來隱藏「導覽器」視圖中的檔案

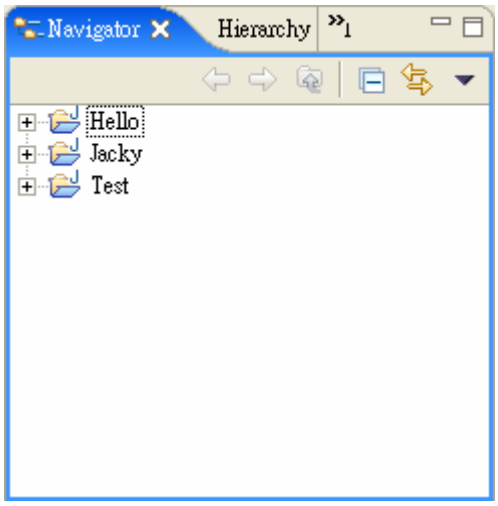


4.10.3 顯示「導覽器」視圖中的檔案

I. 在「導覽器」視圖的工具列上，按一下功能表按鈕 ，開啟顯示選項的下拉功能表。



II. 選 Deselect Working Set 後，就可以出現原有的檔案



5.除錯

我們的說明是採用邏輯錯誤，藉此追蹤下去；範例之後，要談一些更進階的除錯主題，例如設定除錯啟動組態，使用 Hot Code Replacement，暫停執行中且不會中斷的程式(例如無窮回圈)等等。對 IDE 而言，能夠和程式做互動式的除錯，是應該具備的功能。

5.1 錯誤的程式

錯誤的範例程式是要做階乘(n! = n * (n-1) * (n-2) *... * 1)。此範例會建立多層的堆疊框(stack frame)。

```
public class ErrorTest {
    public static void main(String[] args) {
        System.out.println(factorial(6));
    }

    public static int factorial(int value) {
        if (value == 0) {
            return value;
        } else {
            return value * factorial(value - 1);
        }
    }
}
```

此例中，是求 factorial(n)，這個方法會遞迴的呼叫自己，直到

此階乘被算盡為止。此例是要找出 6 的階乘，也就是 720，可惜第一次執行此例的結果是 0。

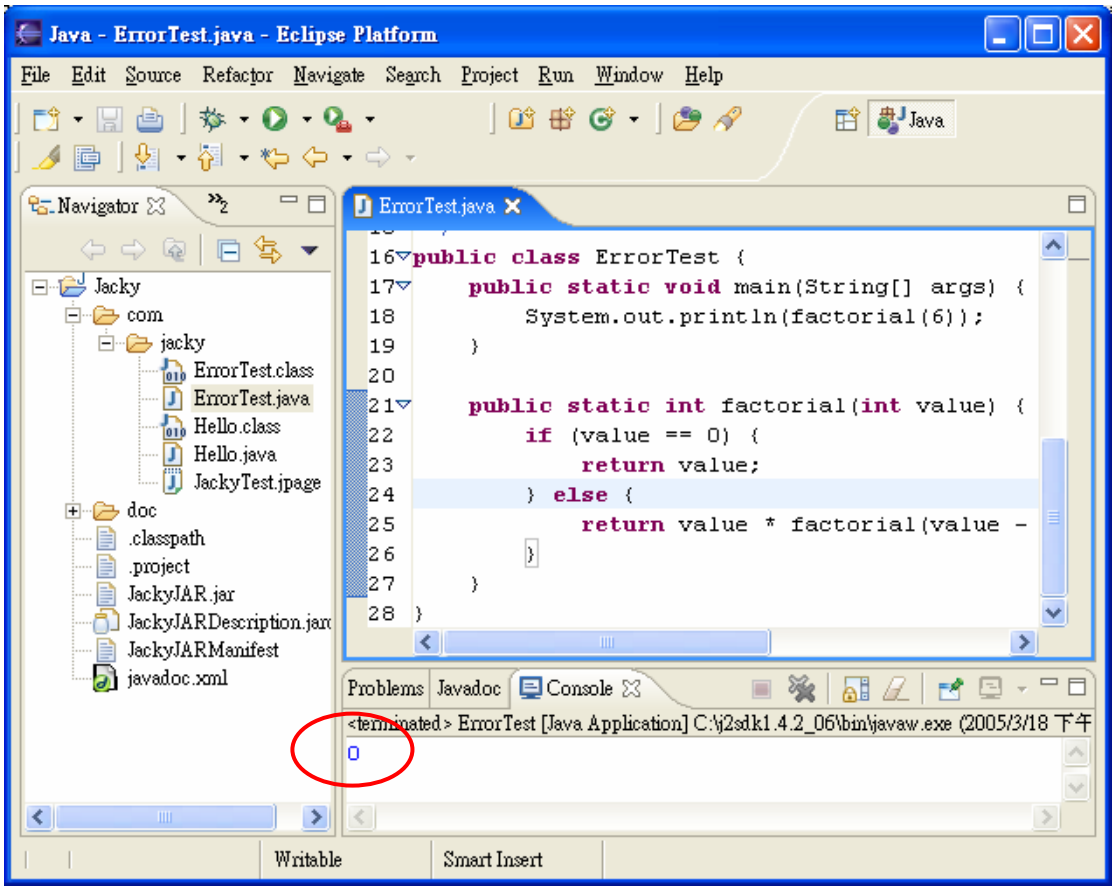


圖 5.1

5.2 設定岔斷點(Breakpoints)

由於沒有拋出任何的例外事件，所以問題是在程式的邏輯。要在程式執行期間檢視程式，所以要設定岔斷點(Breakpoints)來暫停程式。在要暫停的程式碼前面的「Marker Bar」點兩下(或是「Run」→「Toggle Line Breakpoint」)來設定岔斷點，稍後要移除岔斷點，只要再對該岔斷點按兩下即可。

要安插一個岔斷點到 return value * factorial(value - 1)的旁邊，這樣才能觀看連續呼叫 factorial()方法而建立的階乘值。在「Marker Bar」上有一個藍點。

開始除錯，「Run」→「Debug As」→「Java Application」(或是按 旁邊的箭頭選「Debug As」→「Java Application」)，還開啟「Debug」視景。

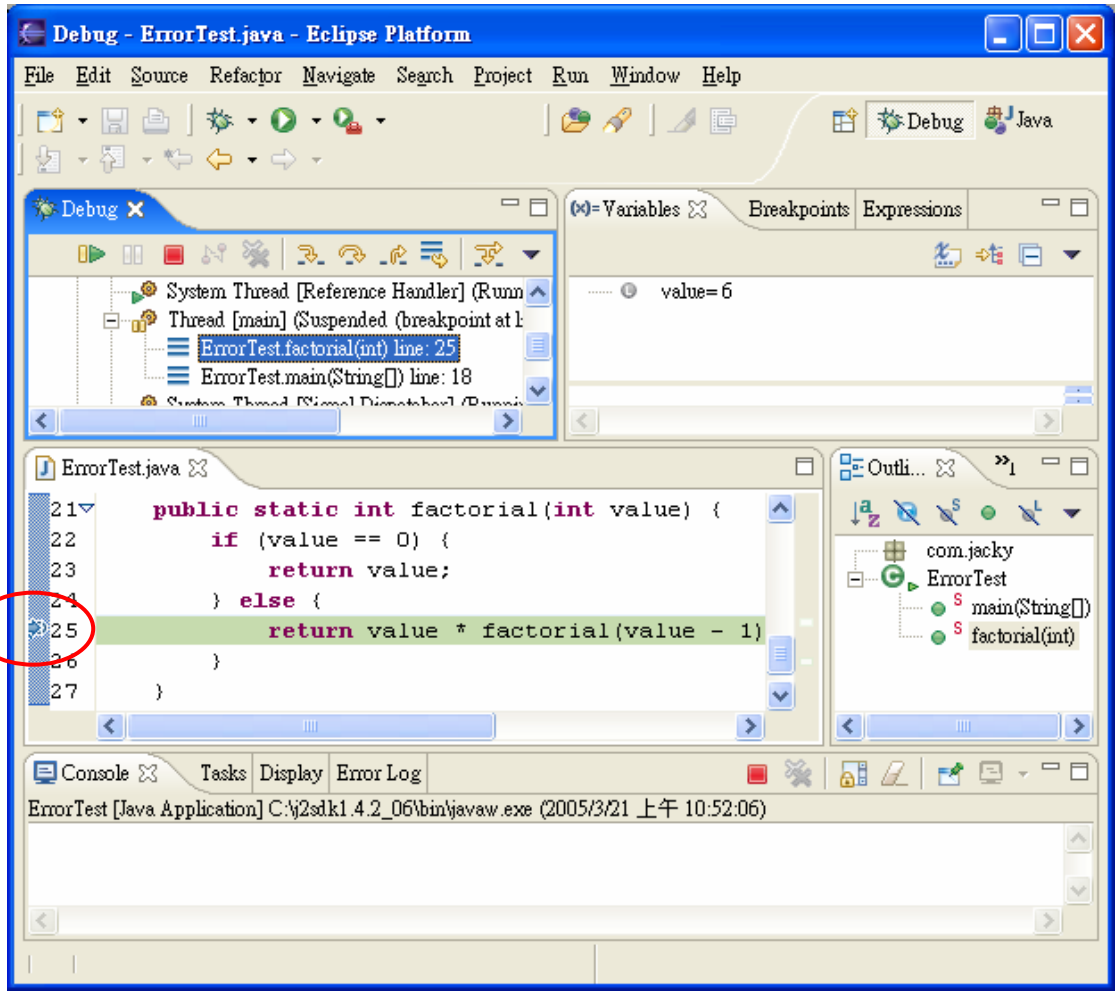


圖 5.2

程式執行到岔斷點會暫停，執行暫停處的該列程式會出現在「Debug」編輯器中，標上一個箭頭。

先了解「Debug」視景。在左上角的「Debug」視圖中可以看見正在除錯的程式構成項目。這裡的堆疊框都有標上三條橫棒 。此例中，我們正在 factorial()方法中，已經由 mail()方法所呼叫了。「Debug」視圖中由左到右的按鈕分別是， Resume 按鈕(在開始執行程式)、 Suspend 按鈕(暫停程式)、 Terminate 按鈕(中止除錯)、 Disconnet 按鈕、 Remove All Terminated Launches 按鈕(除去先

前 debug session)。

「Debug」視圖右邊是層疊的視圖。分別是「Variables」、「Breakpoints」和「Expressions」。

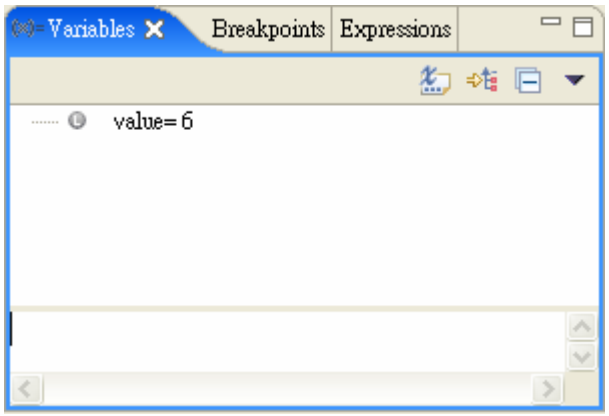


圖 5.3

「Variables」視圖可以檢視區域變數之值。在除錯程式時，可以編修區域變數之值(稍後會做)，這樣可以和程式互動以修正問題。Eclipse 會監視這些變數值，當這些變數值有變時，會改變顏色(改成紅色)。「Variables」視圖底端的部分稱為**詳細資料窗格**(Detail Pane)，會顯示更完整的資訊。

「Breakpoints」視圖管理程式中的岔斷點，對清單中的某各岔斷點按右鍵，在從選單中選擇「Enable」、「Disable」、「Remove」或「Remove All」。

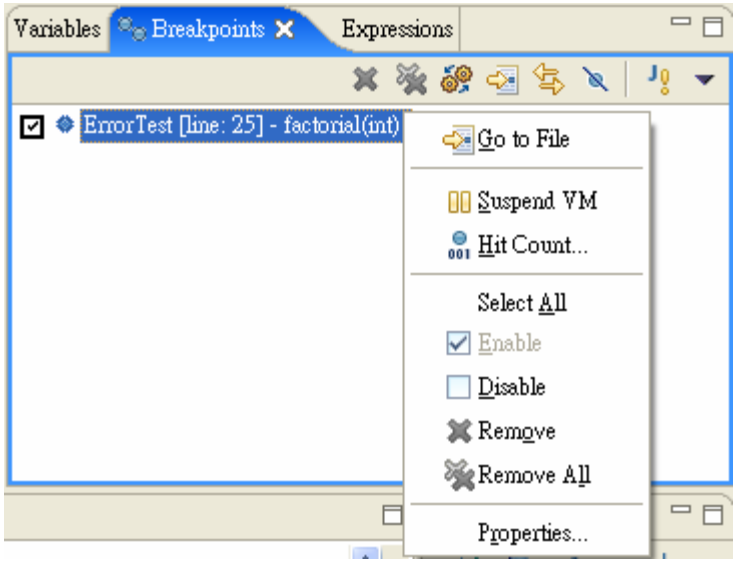


圖 5.4

「Expressions」視圖可以計算表示式(稍後會做)，在編輯器中選取一道表示式，按右鍵，選擇 Inspect 選項，就可以在「Expressions」視圖中予以計算。

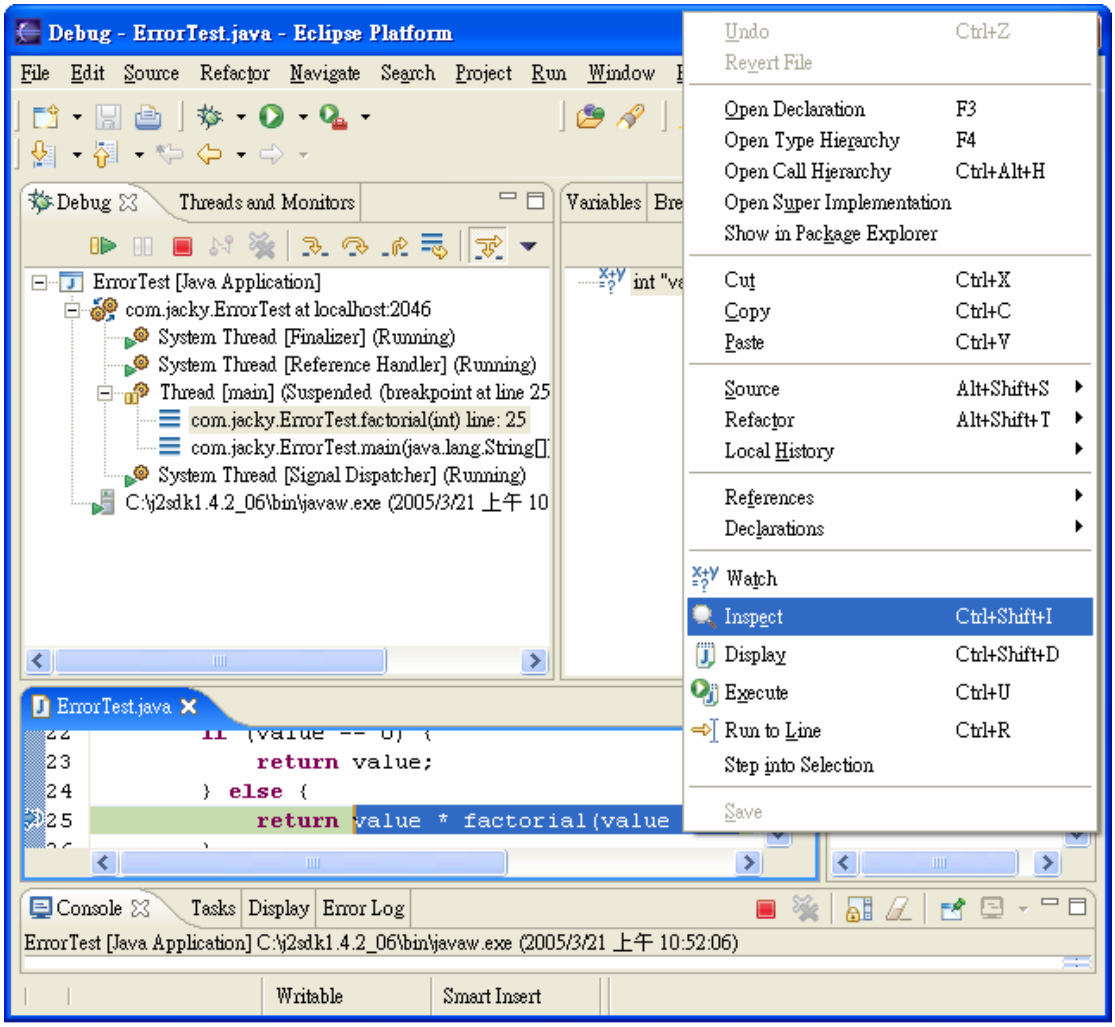


圖 5.5

若選 Display 選項時，計算的結果會顯示在「Display」視圖中。

「Debug」視景中的編輯器和「Java」視景中的編輯器本質上一樣的，但是「Debug」視景的編輯器可以檢視變數的值，只要滑鼠移到變數上即可。

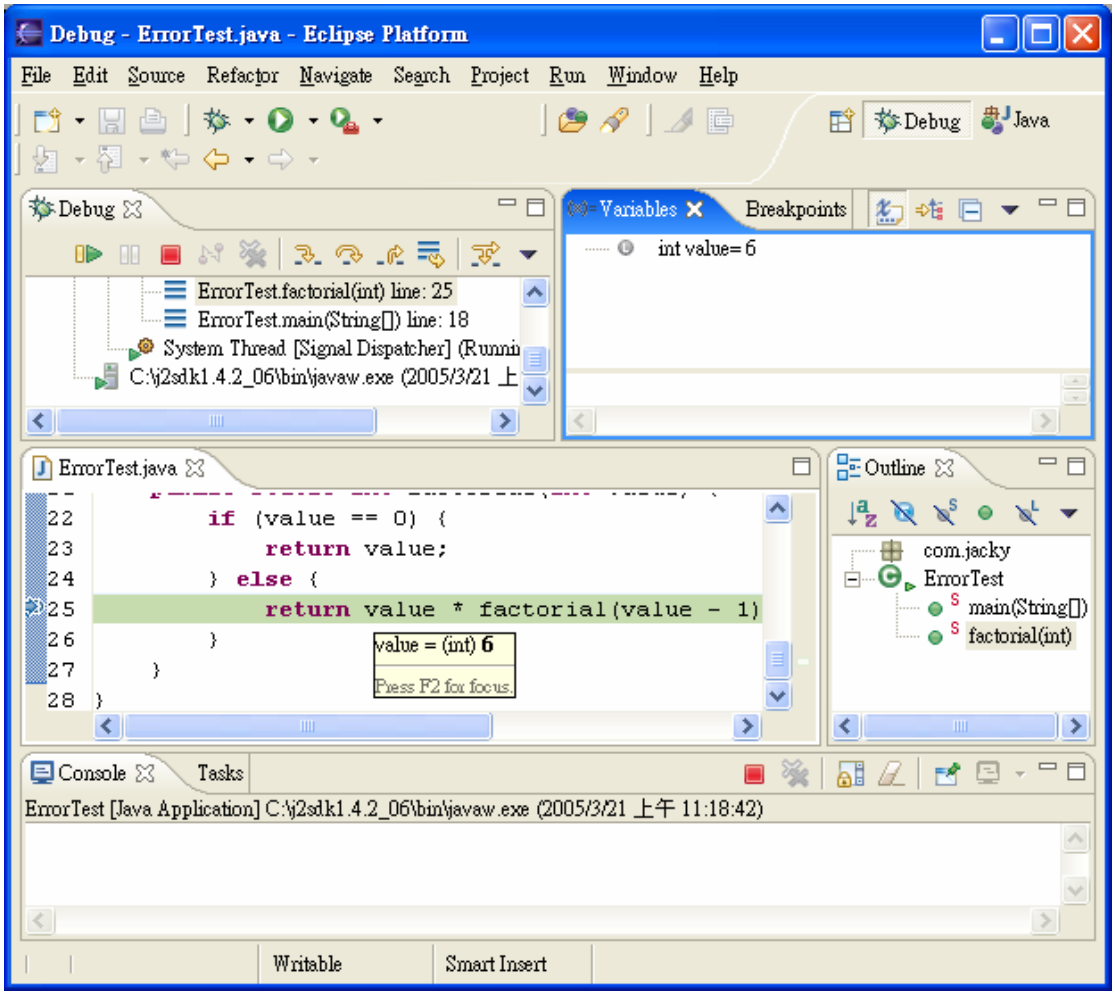


圖 5.6

5.3 逐步除錯

在暫停的程式中移動最基本的作法是採逐步法(single-steping)。Eclipse 提供下列的選項：

5.3.1 Step Into

按 按鈕(也可以按 F5)，進入選取的敘述內。如果該敘述是呼叫某方法，則進入執行該方法。

5.3.2 Step Over

按  按鈕(也可以按 F6)，掠過選取的敘述內。如果該敘述是呼叫某方法，則不會進入該方法。

5.3.3 Step Return

按  按鈕(也可以按 F7)，執行將回復，直到現行方法中下一個 return 陳述式要執行為止，且執行會暫停於下個可執行行上。

5.3.4 Drop to Frame

按  按鈕，這個指令可以放回與重新輸入指定的堆疊框。這項特性類似「回頭執行」再整個重新啟動程式。如果要放回堆疊框，再重新輸入指定的堆疊框，請選取要「放置」的指定堆疊框，再選取 Drop to Frame。

請注意下列有關這項特性的警告：

- 不能在堆疊中放入原生方法。
- 全體資料不受影響，仍維持其現行值。舉例來說，不會清除內含元素的 Static 向量。

附註： *只有在基礎 VM 支援這項特性時，才會啟用這個指令。*

5.3.5 Use Step Filters/Step Debug

按  按鈕(也可以按 Shift－F5)，當動作切換為開啟時，每一個逐行動作（over、into、return）都會套用使用者喜好設定所定義的逐行過濾器集(請參閱「Window」→「Preferences」→「Java」→「Debug」

→「Step Filtering」)。當呼叫逐行動作時，逐行作業會一直進行，直到到達未經過濾的位置，或是遇到岔斷點為止。

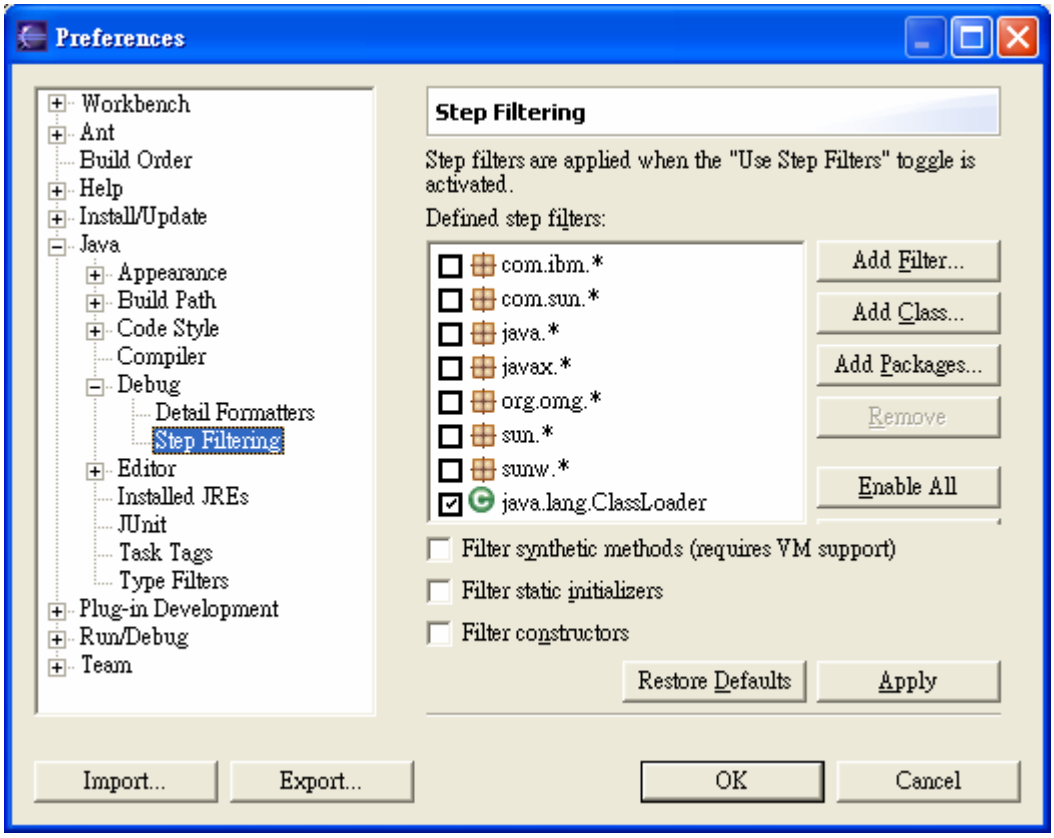


圖 5.7

例如，我們的 debug session 線在暫停在 return value * factorial(value - 1)這一列程式碼，按 F5，就會走進該列，也就是說會開始執行 factorial(value - 1)的呼叫，value 的變數之新值為 5。

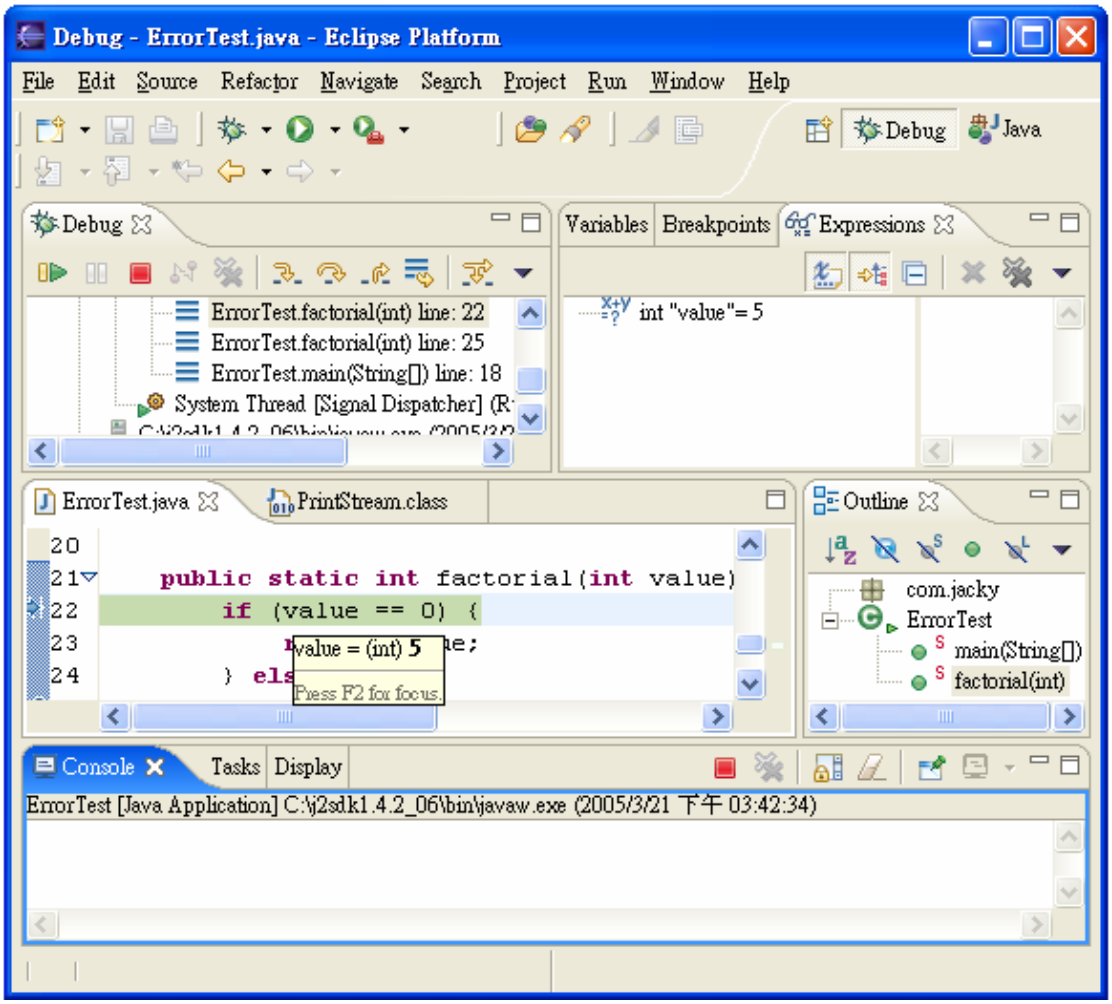


圖 5.8

5.4 繼續執行

我們已經做過程式逐步除錯，還可以繼續做下去，但是每次呼叫 factorial()，每一列程式碼都得跑一遍，實在有點煩。可以改成讓程式一直跑，直到碰到岔斷點。要這樣做只要按「Debug」視圖中 Resume 按鈕。

在這樣做之前，也可以設定去監看我們想要監視的變數。在編輯器中對該變數按右鍵，選 Watch 的選項，把該變數加到「Expressions」視圖中。

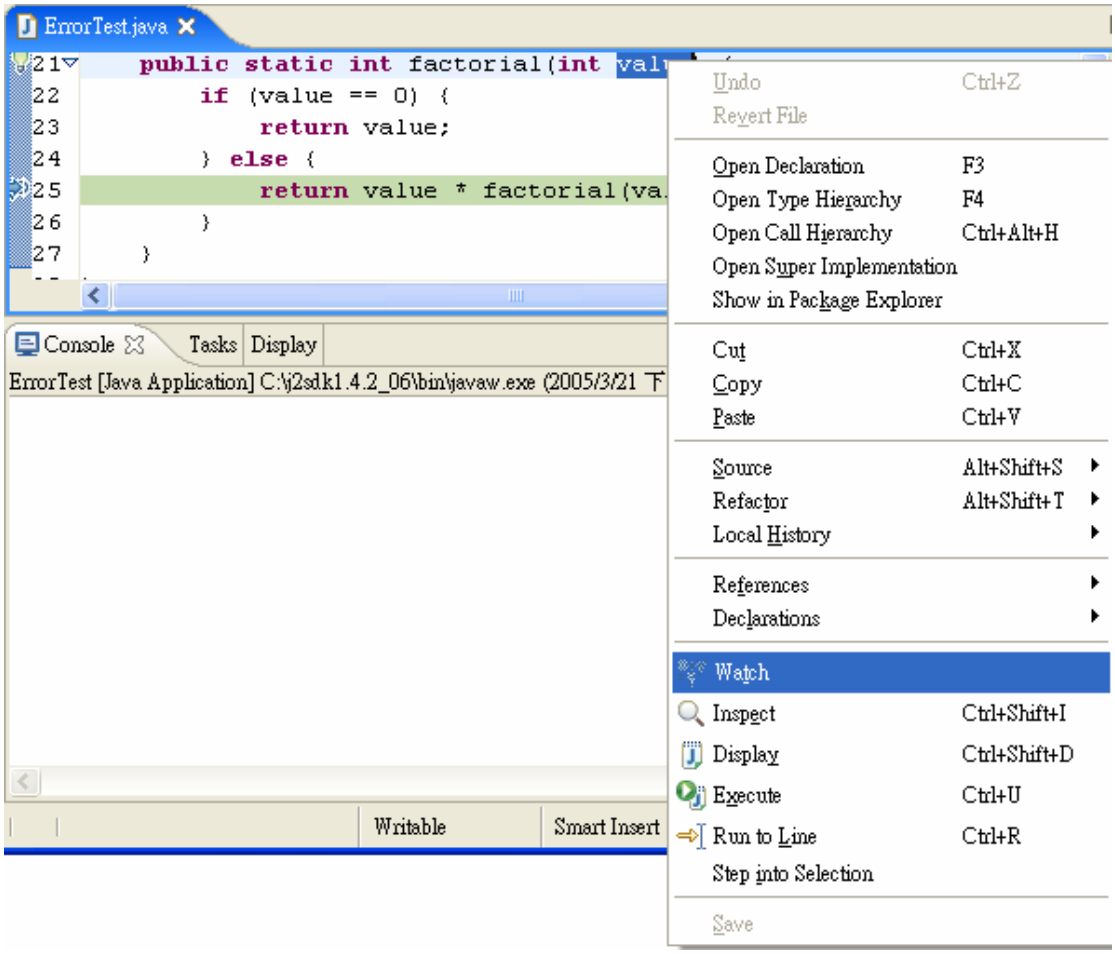


圖 5.9

現在點 Resume 按鈕，程式會繼續執行，直到碰到下一個岔斷點，看一下 value 的值為 5 之後，會發現扔然在同一個 factorial()之內，只要重複不斷按 Resume 按鈕，可以看出 value 值的變化。

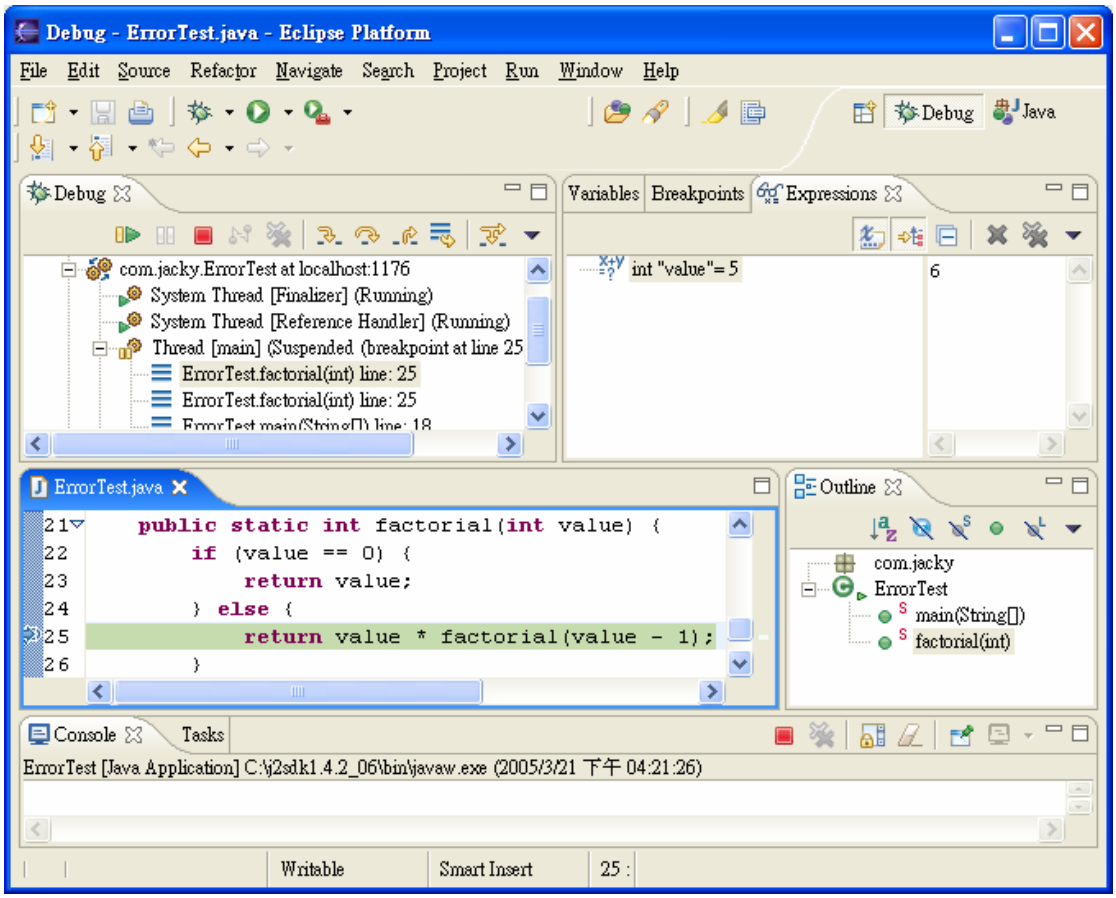


圖 5.10

5.5 設定岔斷點的 Hit Count

這個 factorial()方法要跑 6 次，所以需要按 6 次 Resume 按鈕；也可以設定 Hit Count 來節省時間。有 2 種方式設定：在「Breakpoints」視圖的岔斷點按右鍵

選擇 Properties

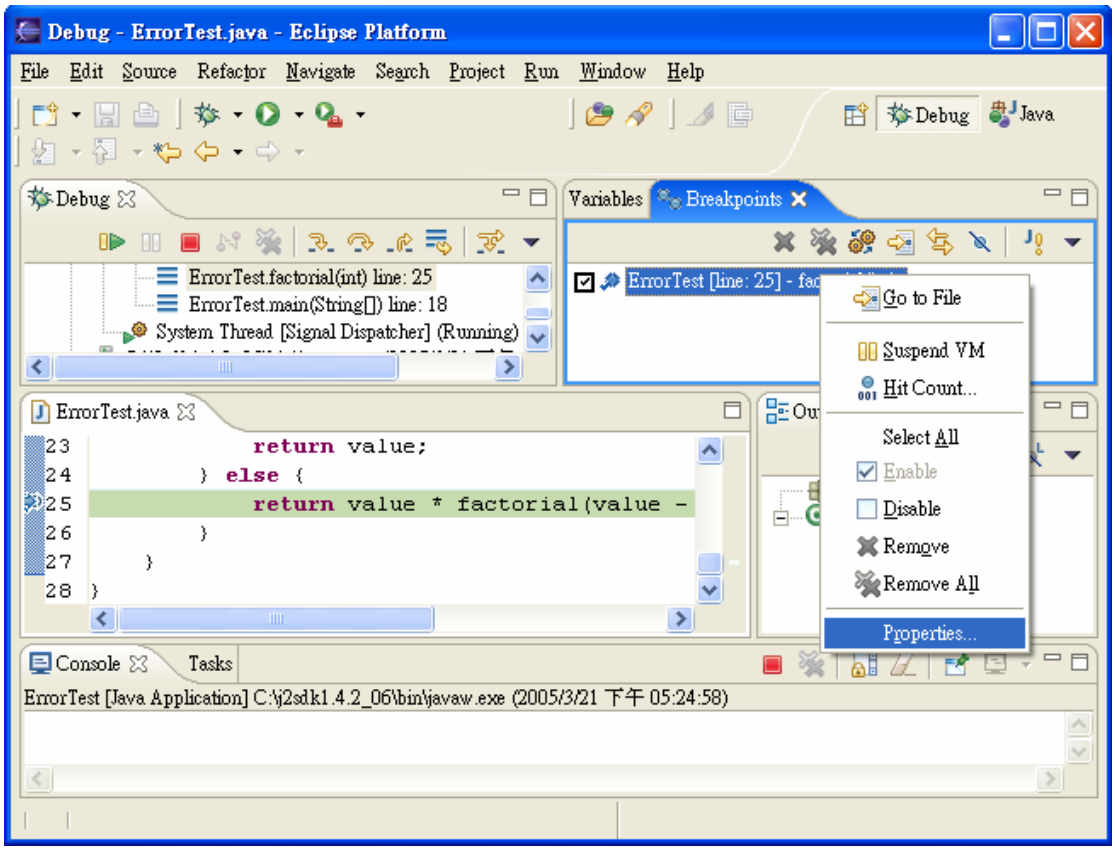


圖 5.11

開啟 Breakpoints Properties 視窗，勾選 Hit Count，並輸入 6

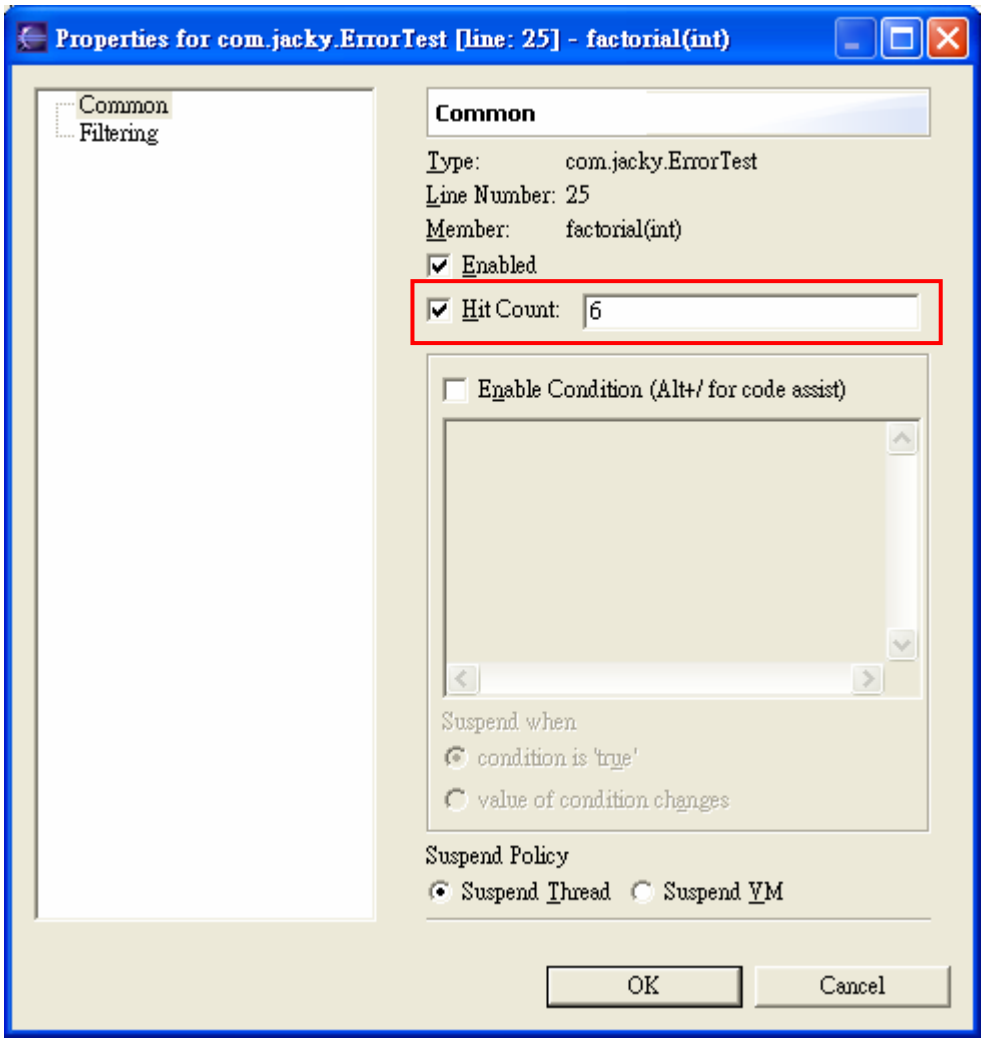


圖 5.12

（或是選擇 Hit Count。

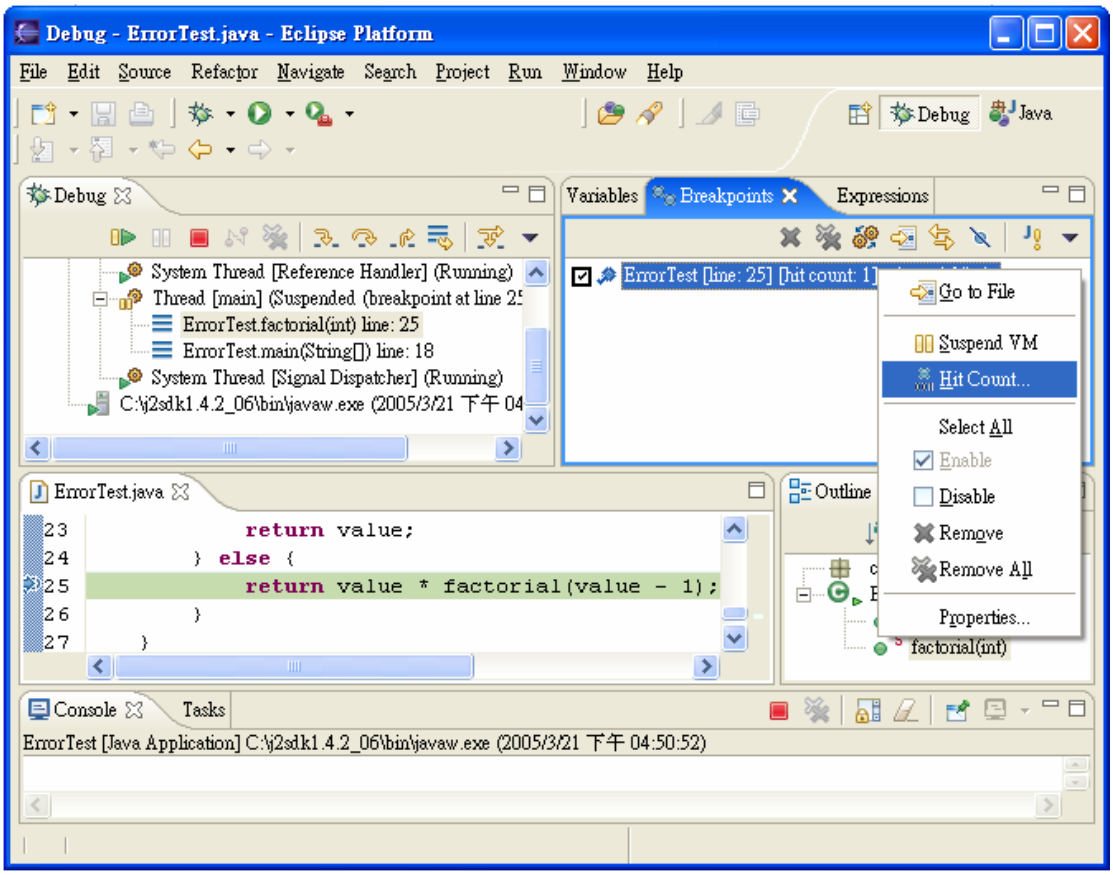


圖 5.13

開啟 Set Breakpoint Hit Count 視窗，並輸入 6

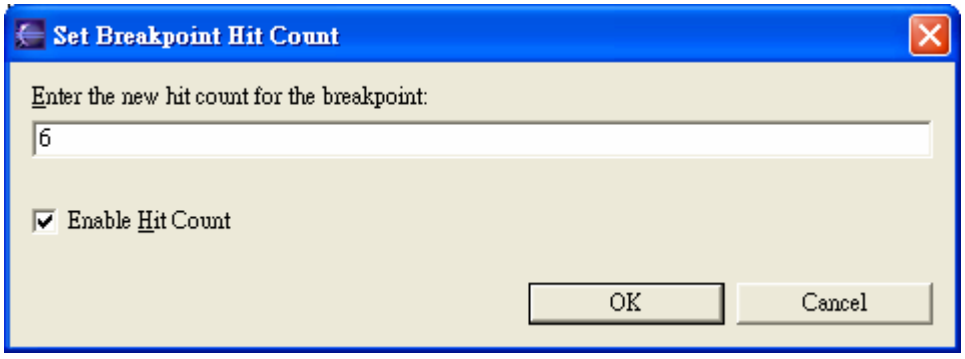


圖 5.14)

重新開始此 debug session，程式執行會在第 6 次碰上岔斷點時暫停，注意到 value 的值是 1，而在「Debug」視圖中可以看見 factorial() 連續呼叫的堆疊框。

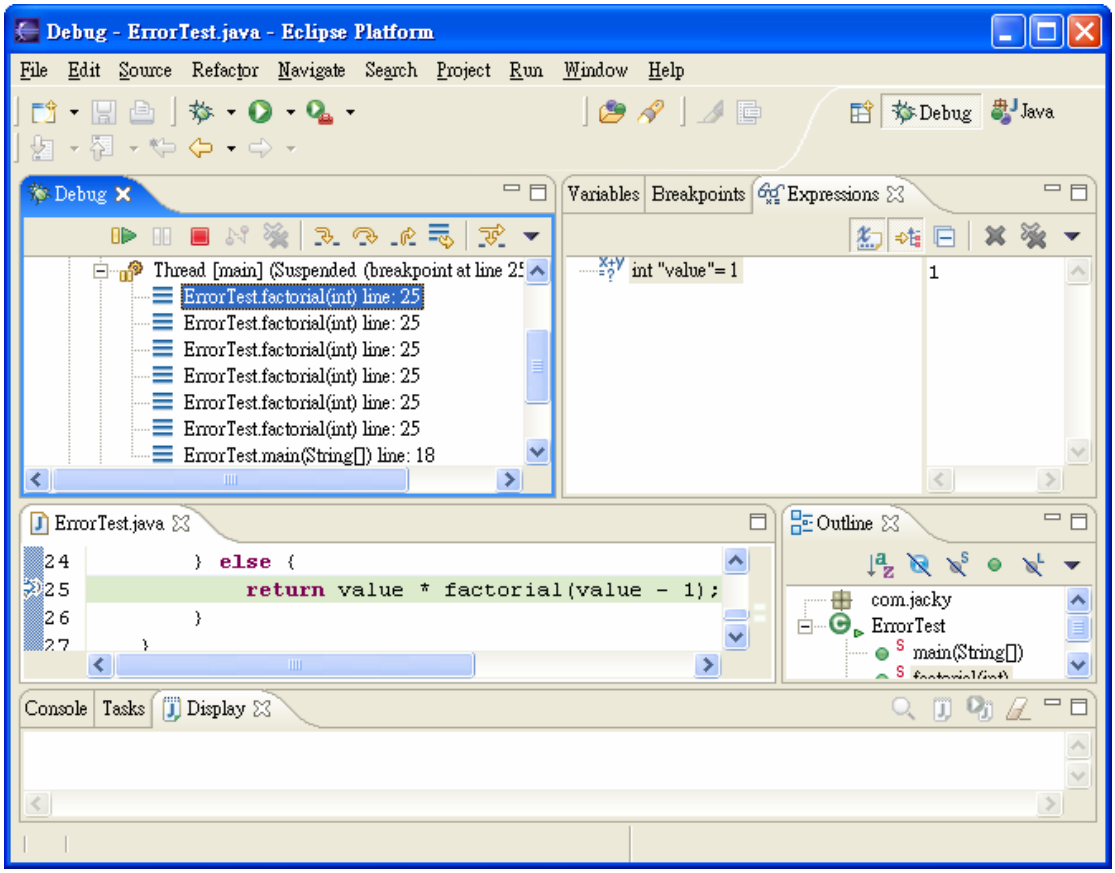


圖 5.15

要檢視「Debug」視圖中任何的堆疊框，以及其中區域變數的值，只要對某堆疊框點一下，使其成為現行堆疊框(active frame)。該區域變數都隨著堆疊框保留下來，例如當前堆疊框的 value = 1，下一個堆疊框 value = 2。

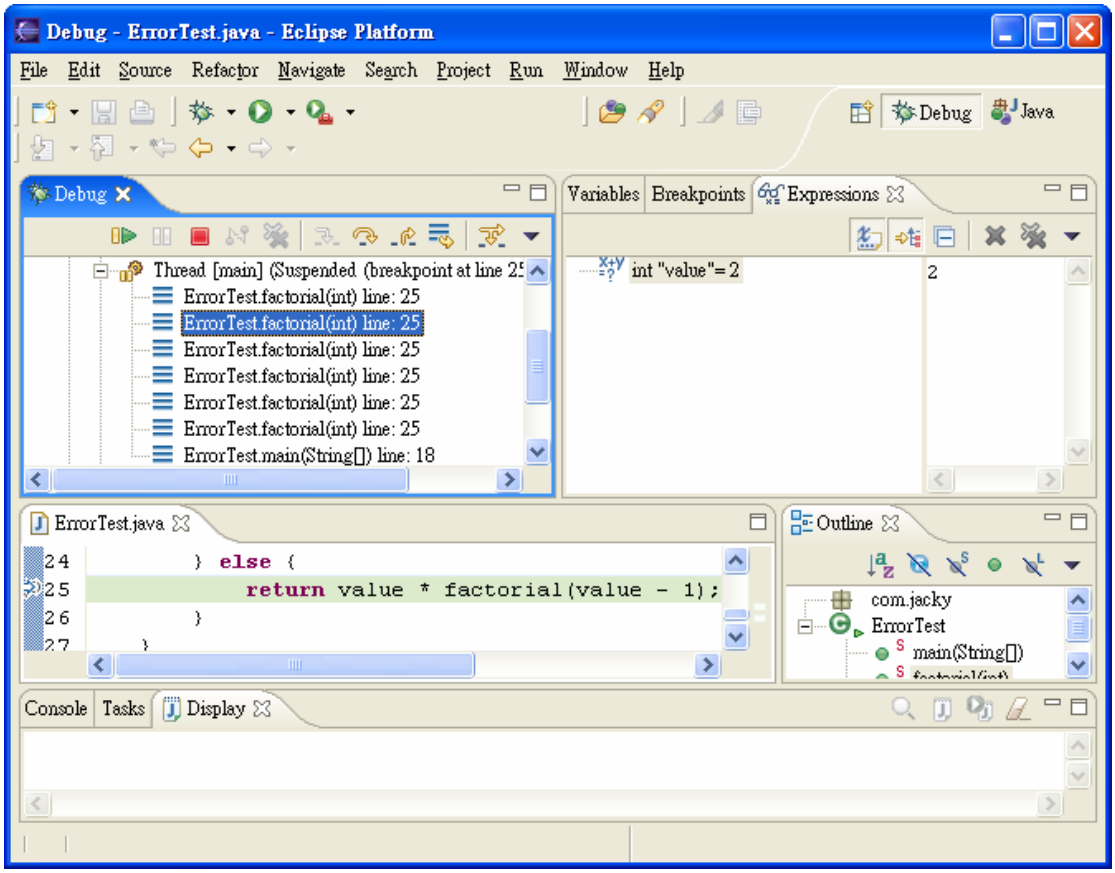


圖 5.16

以上的執行一切都沒問題，然而接下來再按下 Step Into 按鈕時，程式卻又跑進 factorial()方法，而「Expressions」視圖中的 value 之值變成 0，這是不對的。Value 值永遠不能為 0，因為階乘只能用到正整數。

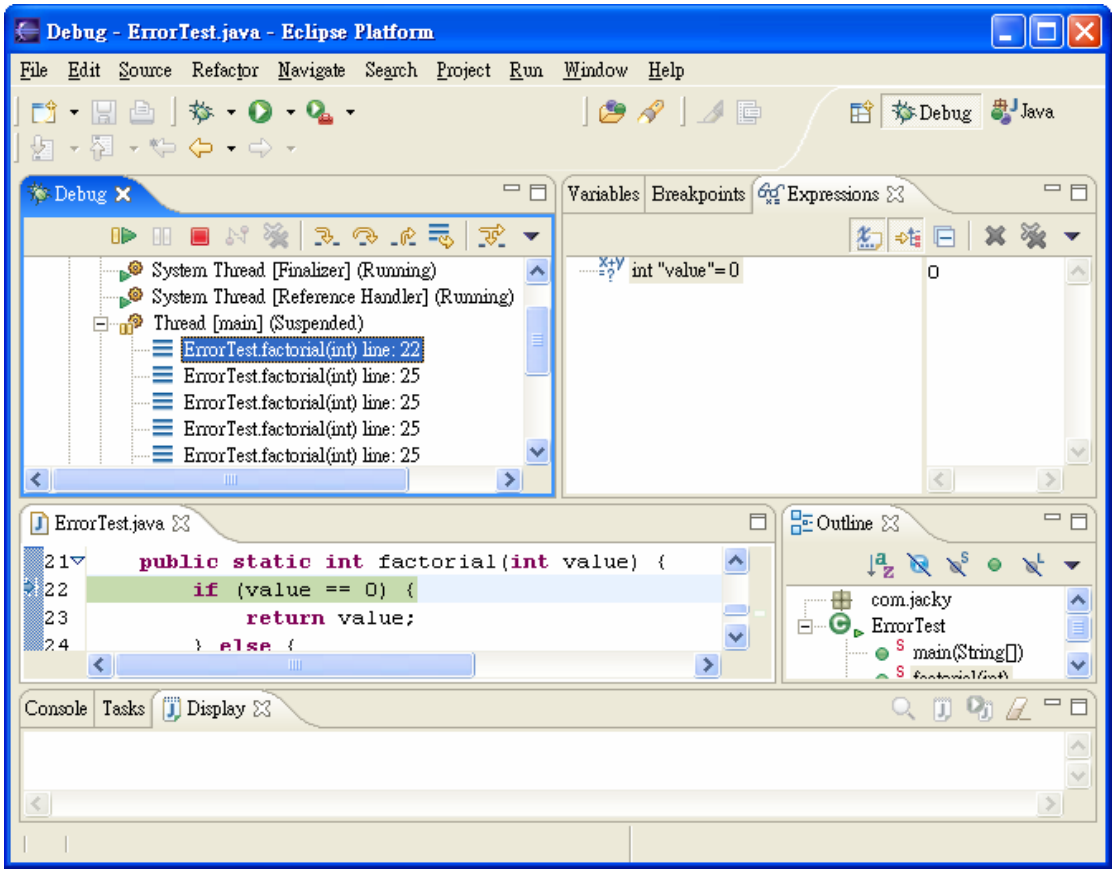


圖 5.17

終於找到問題的所在，讓 factorial()呼叫重複到 value = 1 的時候，而不是設為 0。

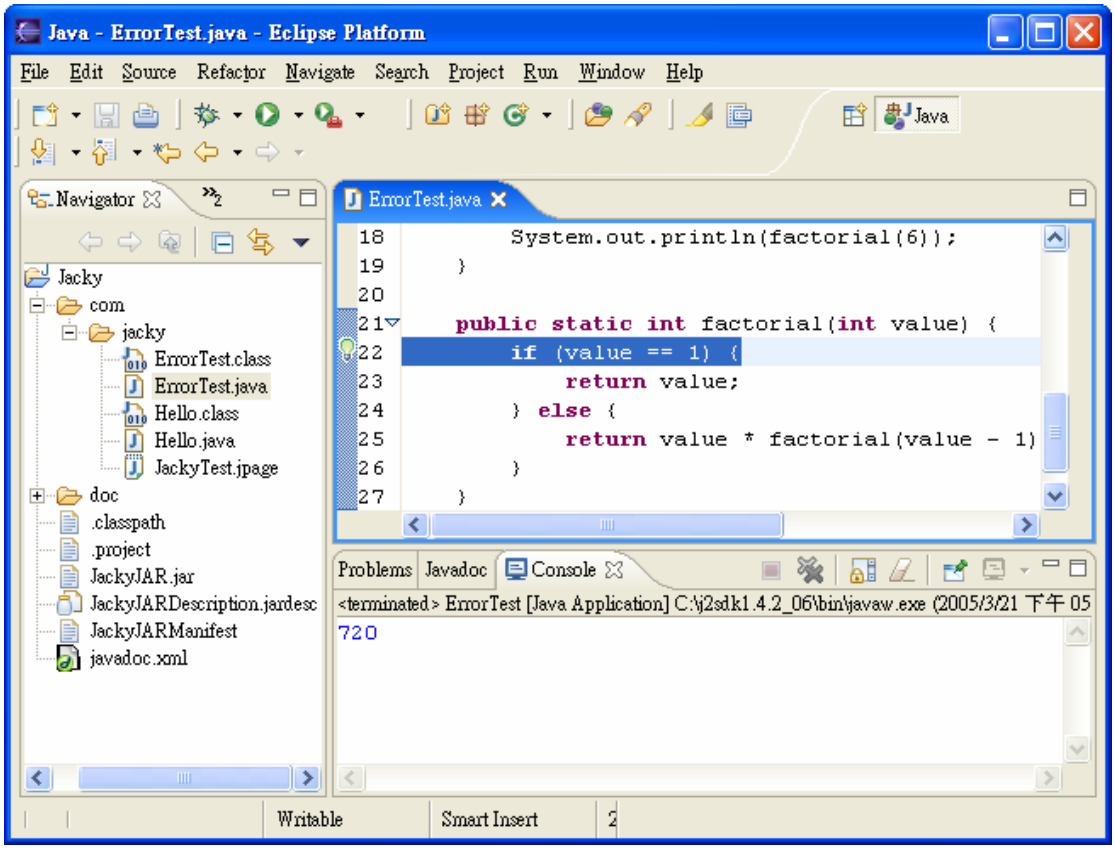


圖 5.18

5.6 岔斷點組態設定

利用 Hit Count 讓除錯方便一點，也可以用其他的做法來設定岔斷點的組態，一樣達到方便除錯的目的。在「Breakpoints」視圖對岔斷點按右鍵，選擇 Properties。

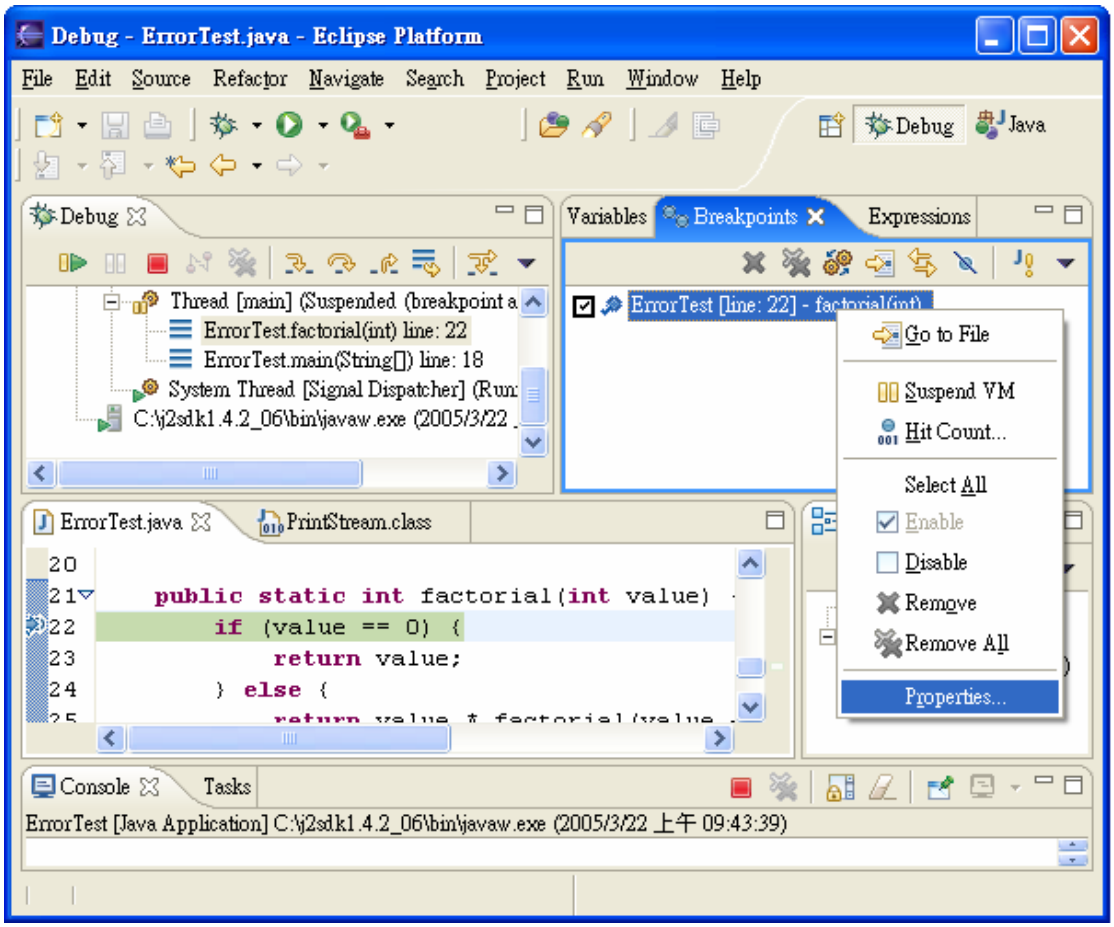


圖 5.19

開啟 Breakpoints Properties 視窗，選取 Enable Condition 的核取方塊，然後就可以輸入條件式來暫停程式。

Suspend when 的選項中：

- condition is 'true'（條件式成立）
- value of condition changes(值改變時)

Suspend Policy

- Suspend Thread 表示只暫停錯誤發生時的 Thread，其他 Thread 繼續執行。
- Suspend VM 表示暫停整個虛擬機器。

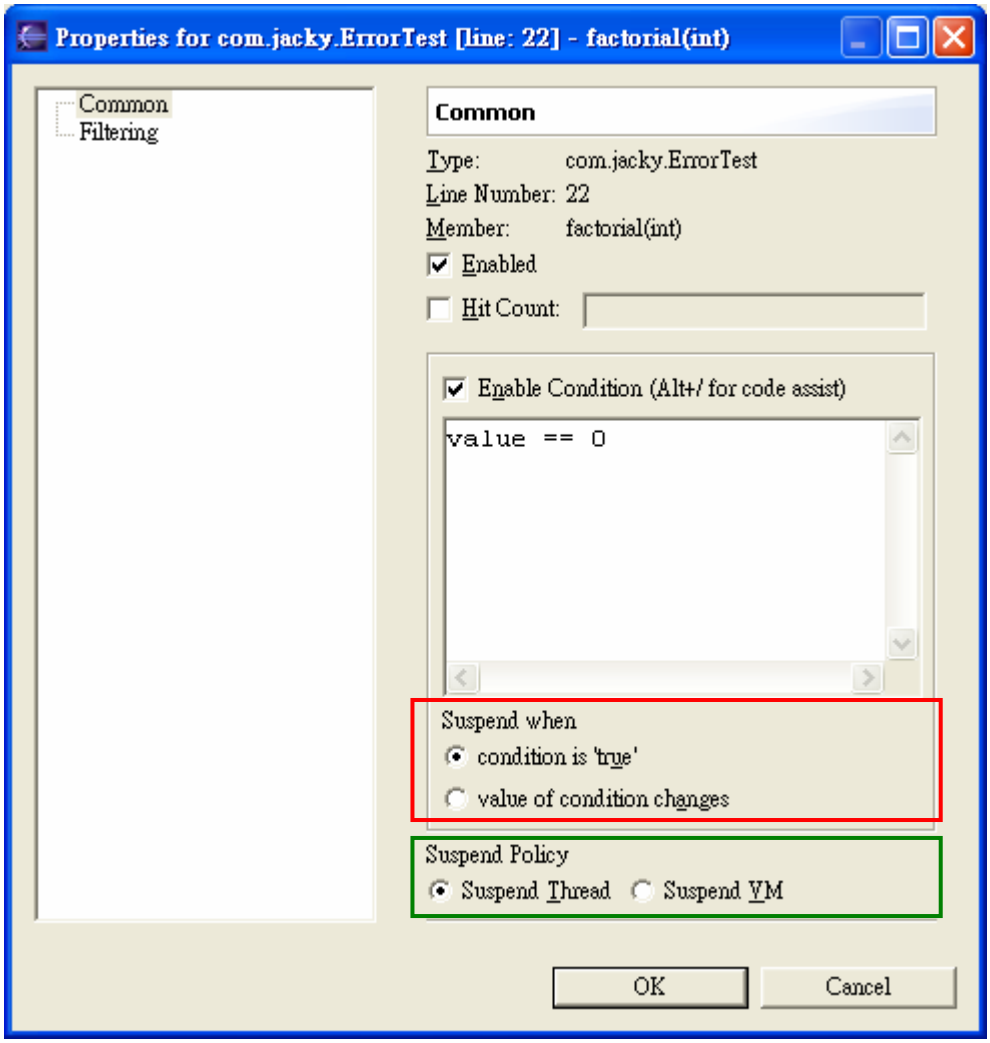


圖 5.20

例如階乘只能使用正整數，當 value == 0 時，就不符合階乘的條件，就讓程式暫停。

5.7 監視點(Watchpoint)

之前使用的岔斷點稱為 Line Breakpoint，除了 Line Breakpoint 以外，也支援監視點(Watchpoint)、方法岔斷點(Method Breakpoint)以及異常岔斷點(Exception Breakpoint)。

設定監視點，表示當程式準備去存取或修改某欄位時，就會暫停執行。監視點不能設在區域變數身上，只能在欄位身上。

設定監視點，在「Java」視景的編輯器中，選取一個欄位，然後再選「Run」→「Toggle Watchpoint」。

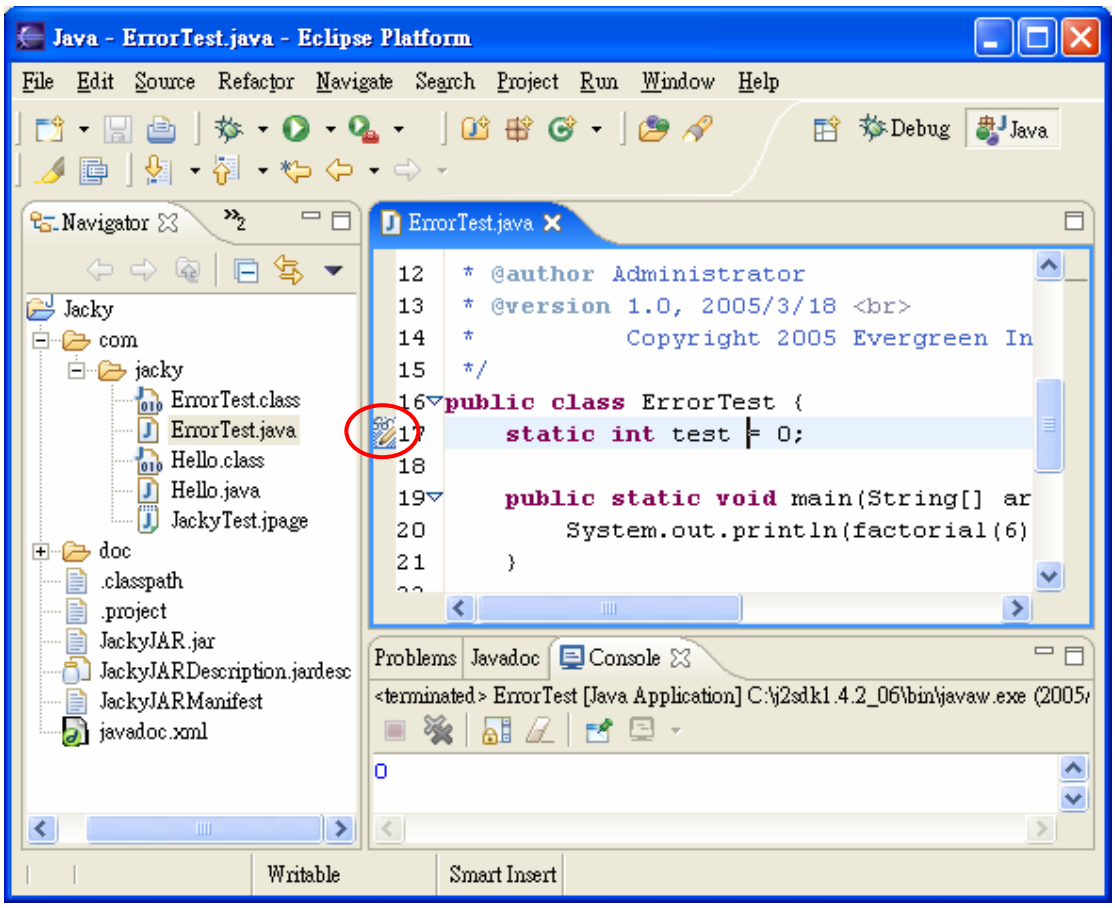


圖 5.21

設定完成後，在「Marker Bar」會出現這個圖示。

新的監視點會出現在「Debug」視景中的「Breakpoints」視圖裡，對該監視點按右鍵，選擇 Properties。

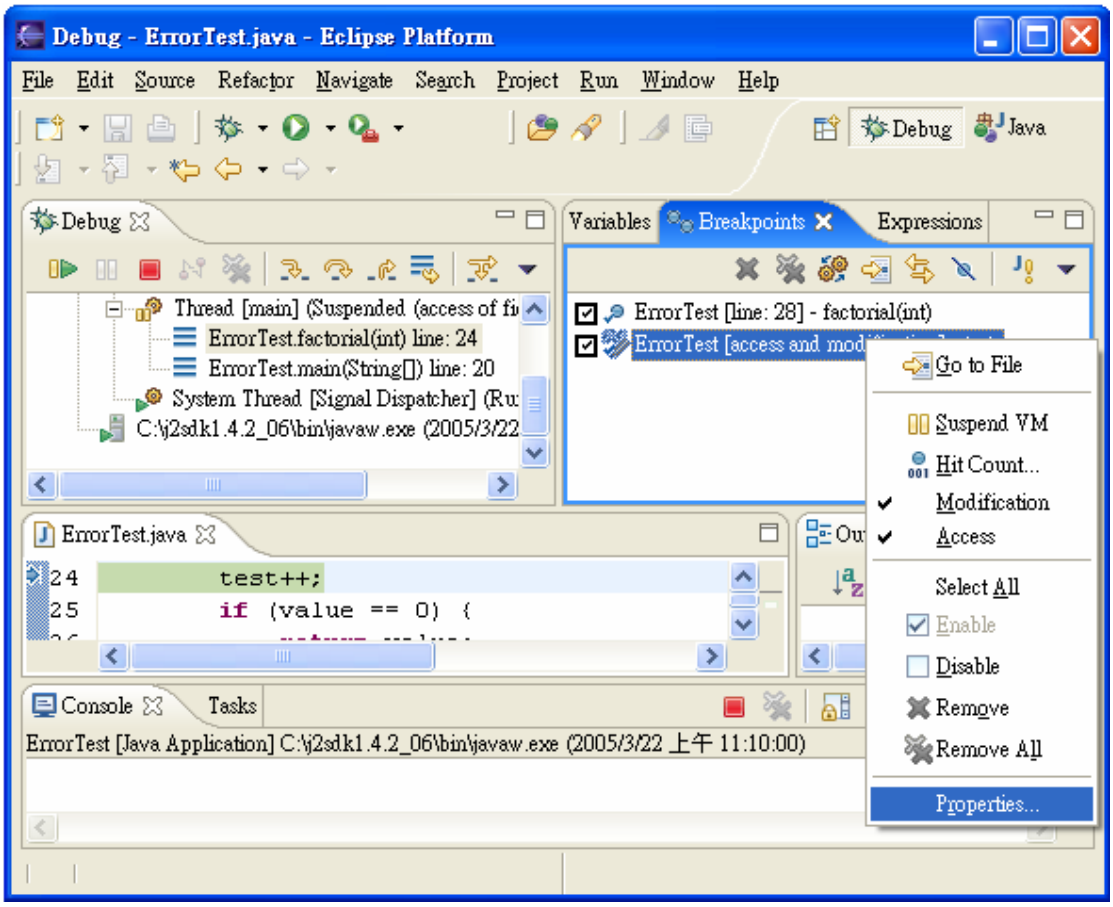


圖 5.22

開啟 Breakpoints Properties 視窗，選項設定跟之前的岔斷點差不多，特別的是 Suspend on 的選項

- Field Access 暫停程式之依據是當欄位被存取
- Field Modification 暫停程式之依據是當欄位被修改

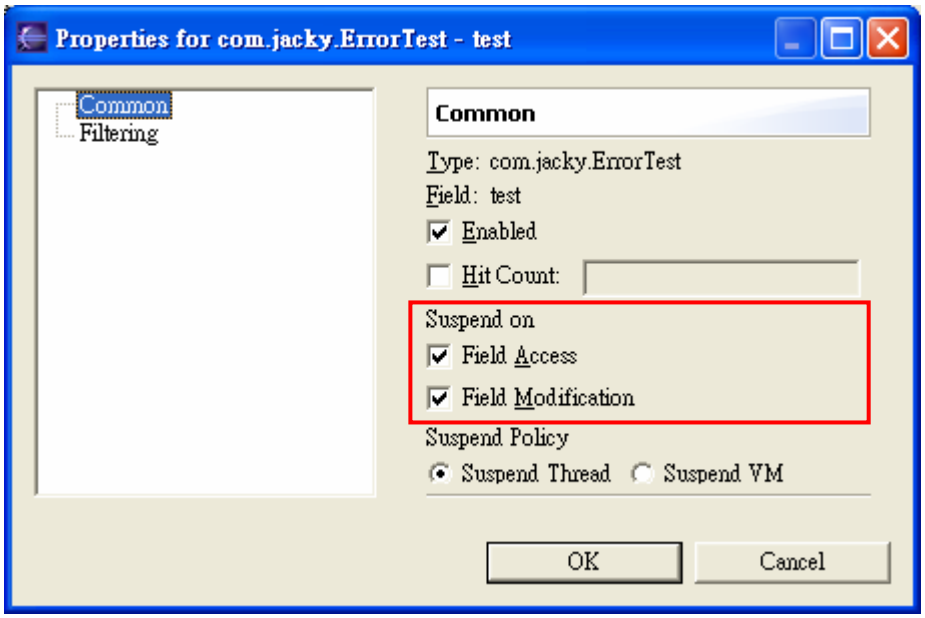


圖 5.23

5.8 方法岔斷斷點(Method Breakpoint)

進入或離開某方法時，方法岔斷點(Method Breakpoint)會暫停程式執行，至於是進入之時或是離開之時，依據組態的設定。

設定方法岔斷點，在「Java」視景的編輯器中，把游標放在要監視的方法前，然後再選「Run」→「Toggle Method Breakpoint」。

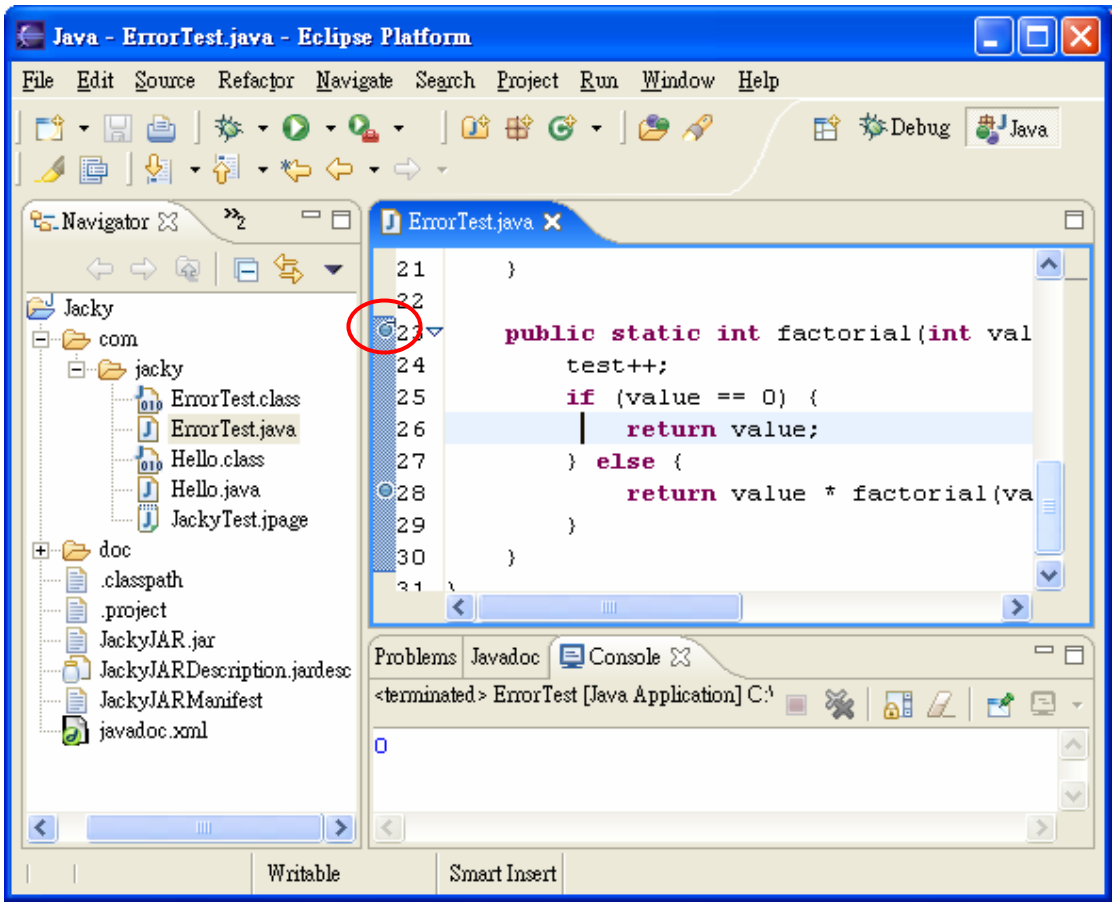


圖 5.24

設定完成後，在「Marker Bar」會出現這個圖示。

新的方法岔斷點會出現在「Debug」視景中的「Breakpoints」視圖裡，對該岔斷點按右鍵，選擇 Properties。

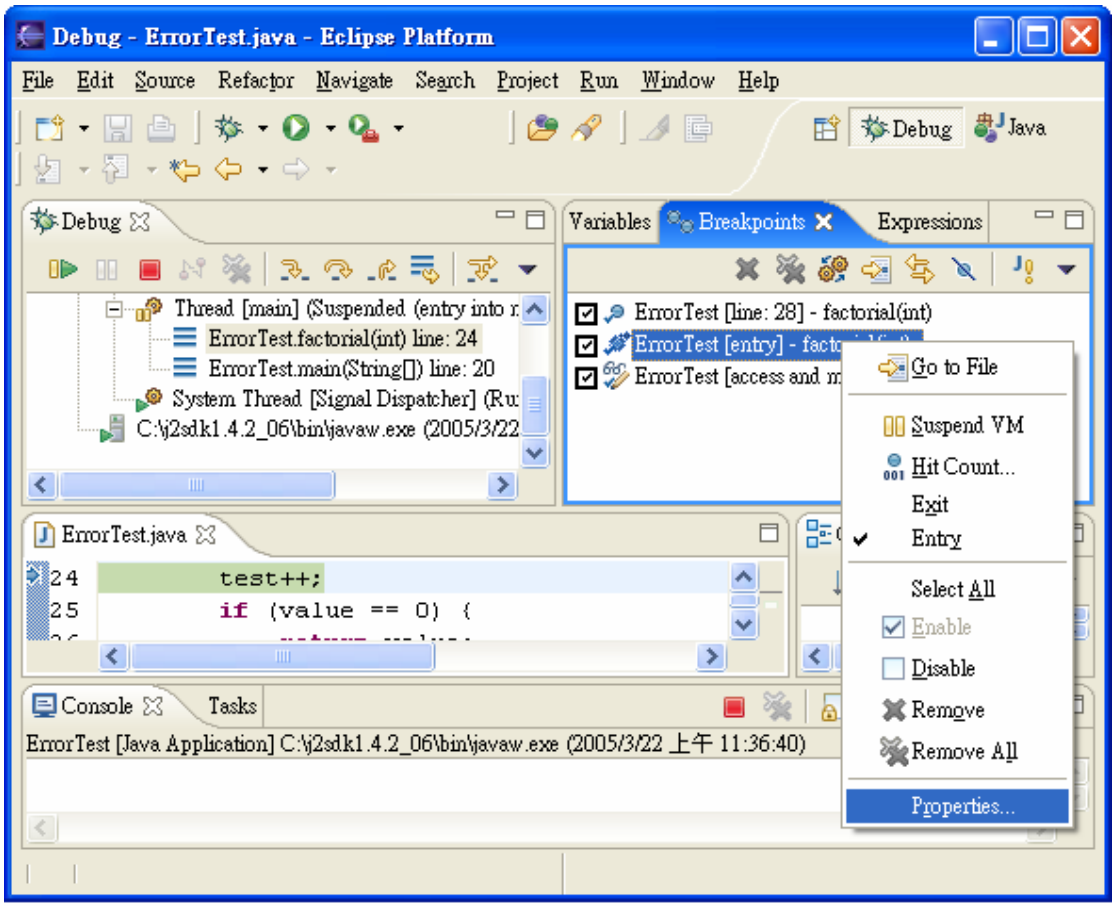


圖 5.25

開啟 Breakpoints Properties 視窗，選項設定跟之前的岔斷點差不多，特別的是 Suspend on 的選項

- Method Entry 決定岔斷點生效之時是在進入該方法
- Method Exit 決定岔斷點生效之時是在離開該方法

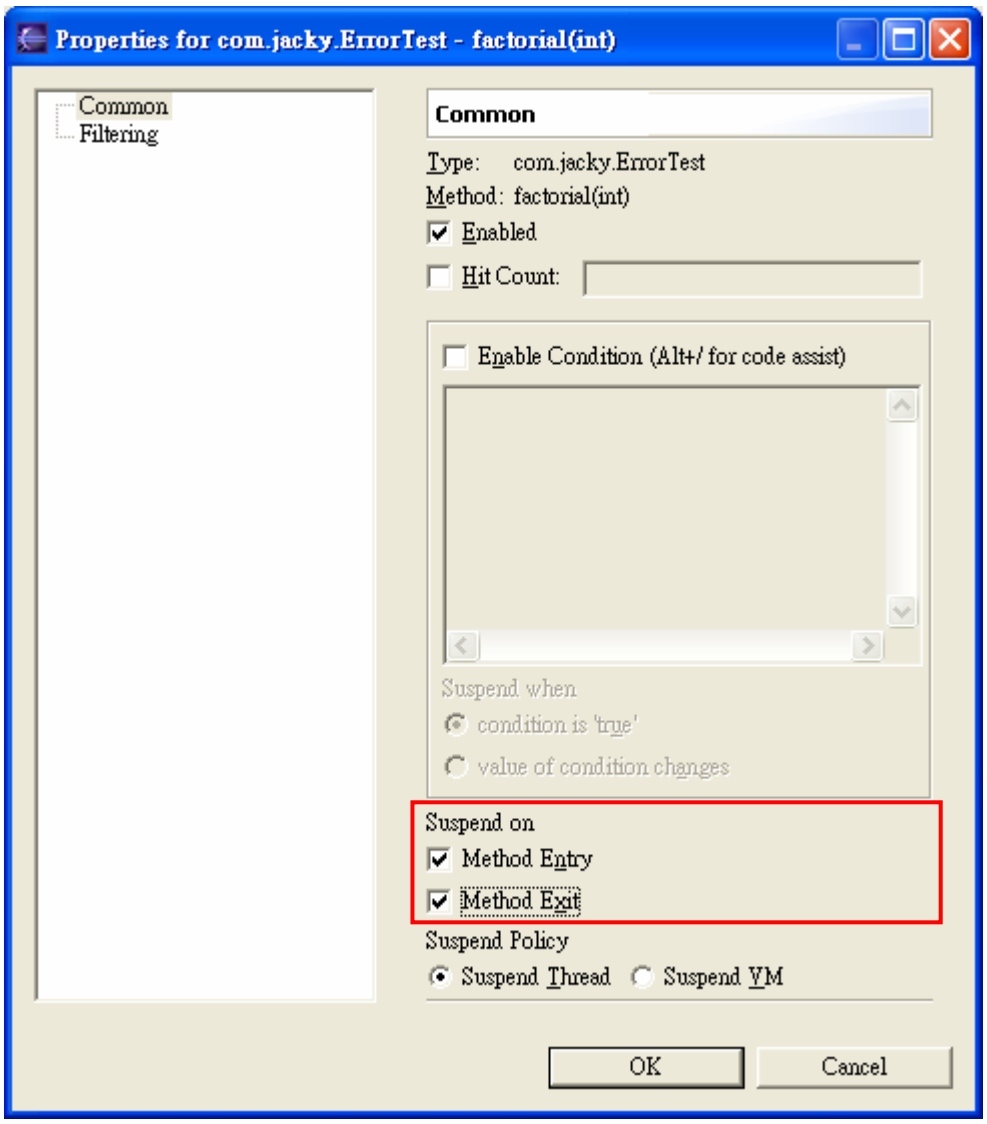


圖 5.26

5.9 異常岔斷點(Exception Breakpoint)

當例外發生時，可以暫停程式執行。如果程式會拋出例外事件，諸如 Null 例外事件，而且不知道這個例外事件是從何時(或是何處)發生的，這個岔斷點就很有用。可以暫停程式，觀看程式中拋出例外事件時，出了什麼事。

設定異常岔斷點，在「Java」視景的編輯器中，選「Run」→「Add Java Exception Breakpoint」。

新的異常岔斷點會出現在「Debug」視景中的「Breakpoints」視圖裡，對該岔斷點按右鍵，選擇 Properties。

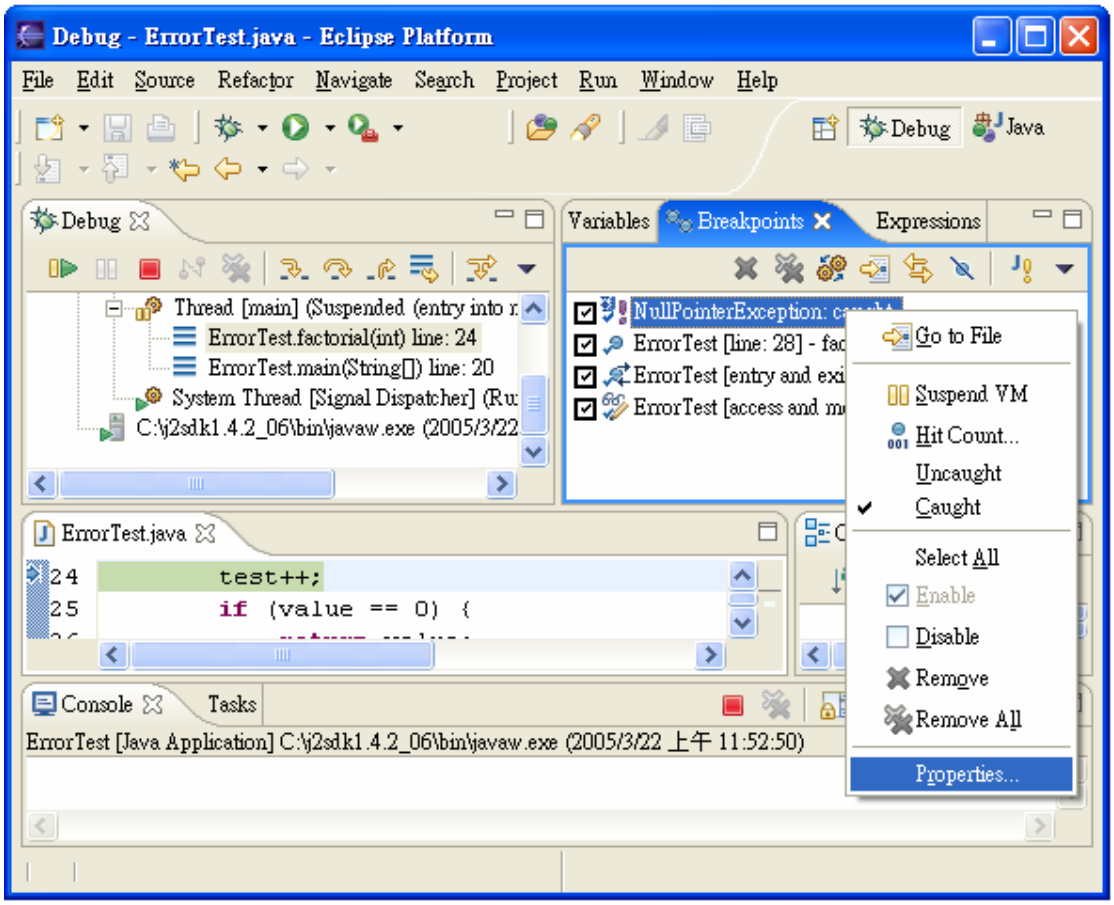


圖 5.27

開啟 Breakpoints Properties 視窗，選項設定跟之前的岔斷點差不多，特別的是 Suspend on 的選項

- Caught Exception 決定岔斷點生效之時是例外事件被捕捉
- Uncaught Exception 決定岔斷點生效之時是例外事件沒被捕捉

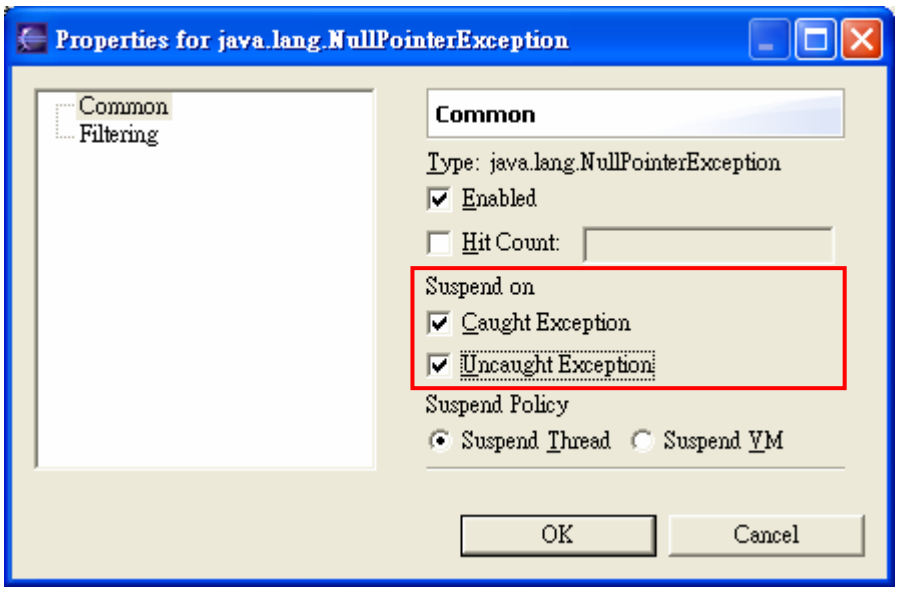


圖 5.28

5.10 Java 表示式及變更某些值

在除錯時，可以在「Expressions」視圖中的詳細資料窗格內輸入表示式，選取表示式，按右鍵選擇 Inspect。

例如現在變數值是 6，在詳細資料窗格內輸入表示式 value + 1，選取表示式，按右鍵選擇 Inspect

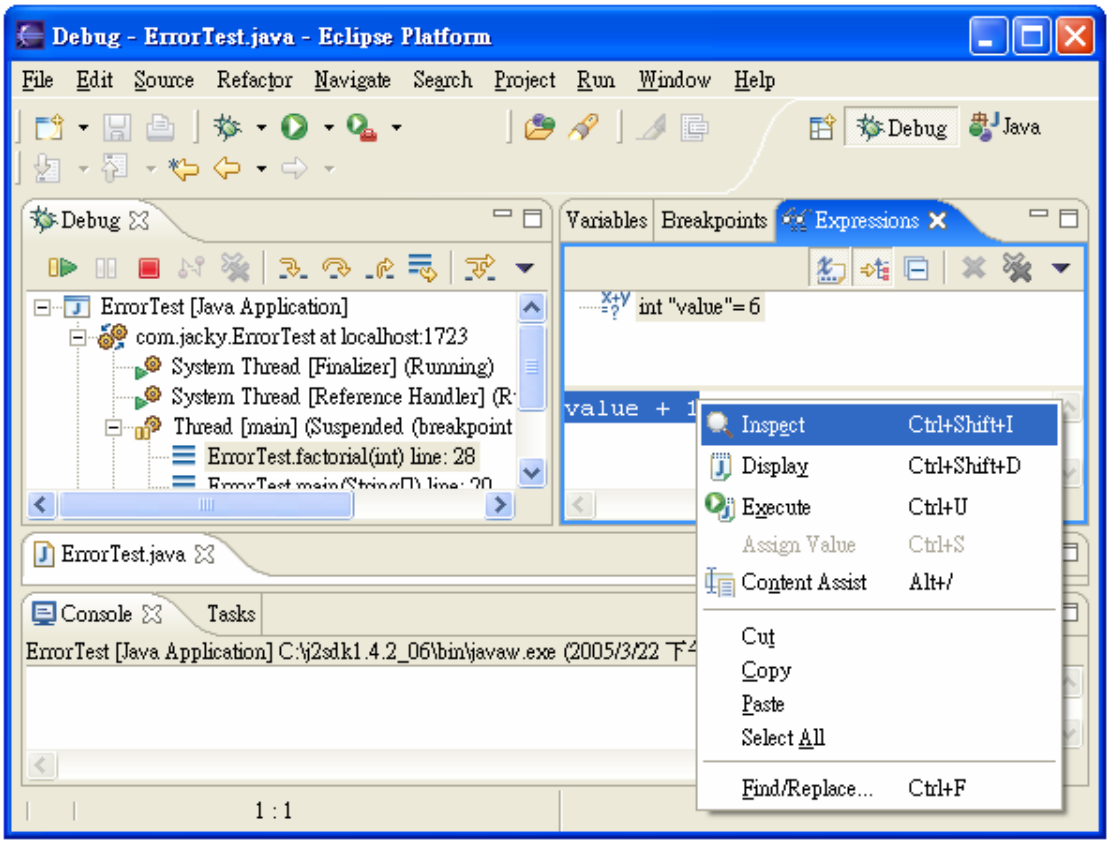


圖 5.29

這樣做會把 value + 1 加進「Expressions」視圖中的表示清單。

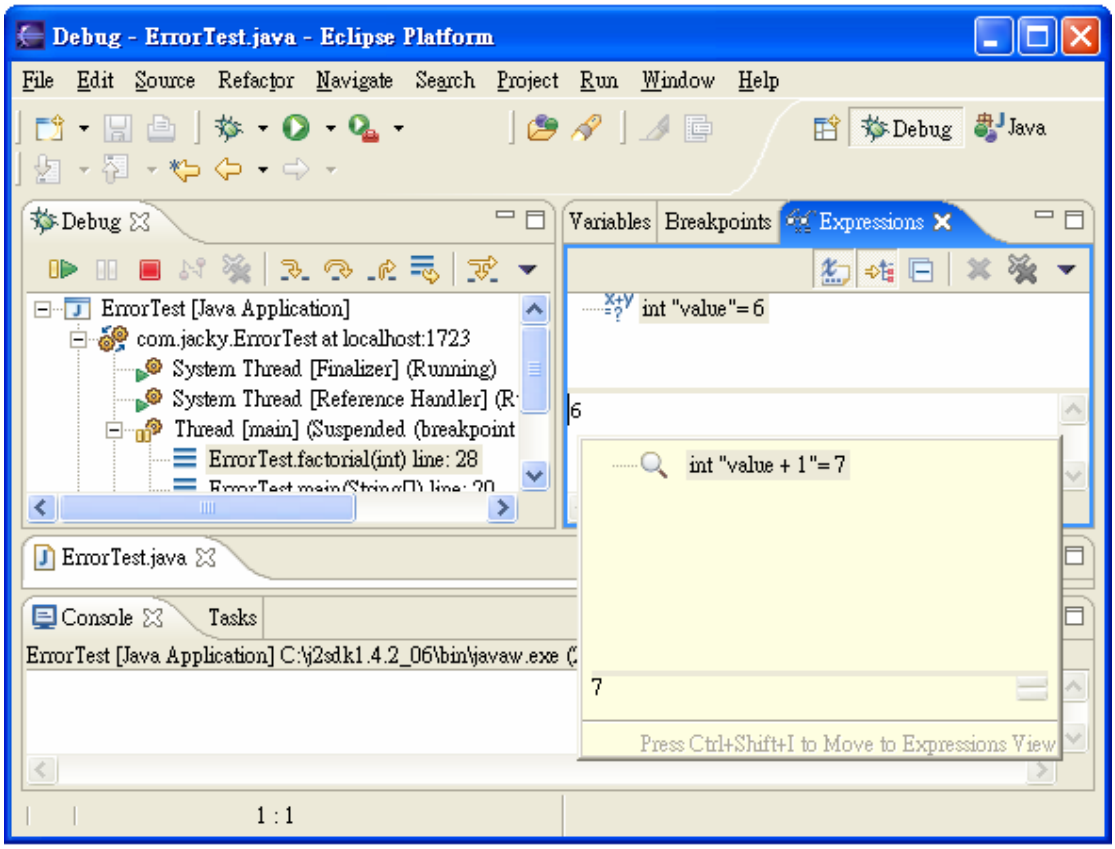


圖 5.30

在除錯時編輯欄位和變數的值，只要對「Variables」視圖中的欄位或變數點兩下，開啟 Set Value 視窗，輸入新值。例如在執行期間變更 value 的值為 5。

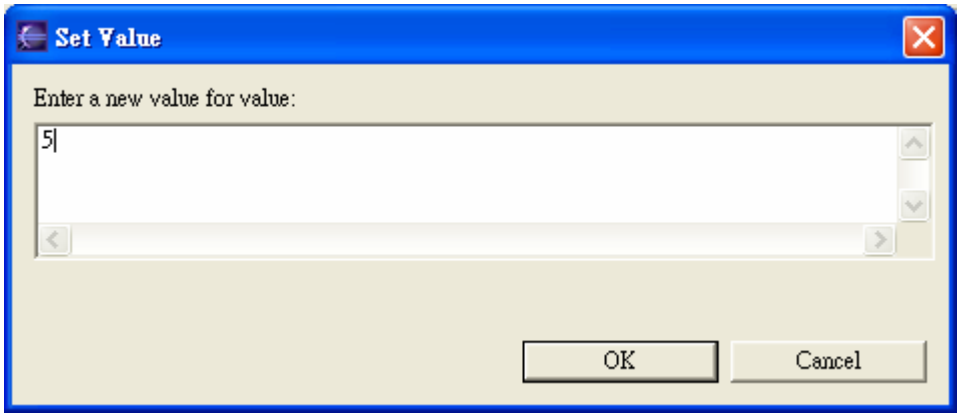


圖 5.31

想檢查程式針對不同測試值的反應，或是避開某些有問題的值，這個功能非常有用。

6.重構(Refactoring)

Java 程式重構的目標就是進行全系統程式碼變更，但不會影響程式的行為。Eclipse 提供有易於重構程式碼的協助。

重構工具支援若干在 Martin Fowler 所著的 Refactoring: Improving the Design of Existing Code, Addison Wesley 1999 一書中描述的轉換，如擷取方法、列入區域變數等。

在執行重構作業時，可以先選擇性地預覽所有因某個重構動作而發生的變更，然後再決定是否實行。當預覽重構作業時，系統將通知潛在的問題，而且將呈現一個清單，列出重構動作將執行的變更。如果未預覽重構作業，系統將完整地進行變更，而且將顯示任何產生的問題。如果偵測到不容許重構作業繼續的問題，則這個作業將會中止，並顯示問題清單。

重構指令可在一些 Java 視圖（如：套件瀏覽器、概要）與編輯器的內容功能表中找到。有許多「看似簡單」的指令，如移動和重新命名，實際上是重構作業，因為移動 Java 元素以及將它重新命名，通常都需要變更相依檔。

6.1 重新命名

6.1.1 區域變數(Local Variable)

如果要將區域變數（或方法參數）重新命名，請執行下列動作：

I. 在 Java 編輯器中選取變數（或其參照）

II. 「Refactor」→「Rename」

（或是在編輯器按右鍵，選取「Refactor」→「Rename」）

出現 Rename Local Variable 視窗

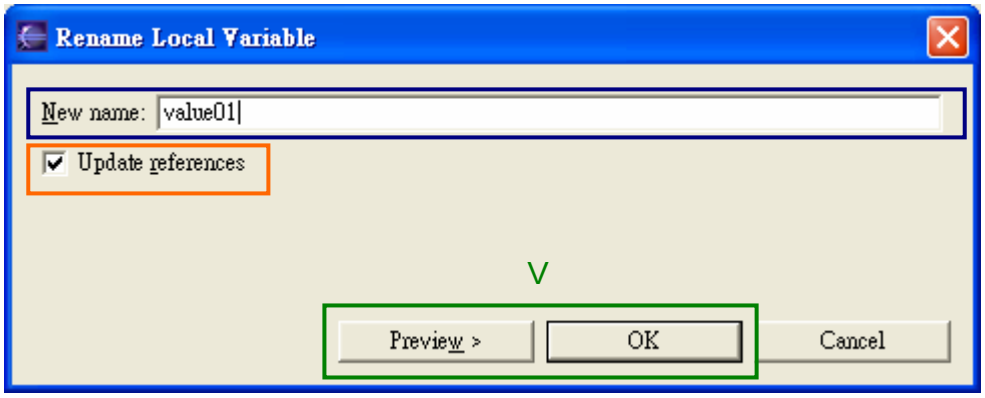


圖 6.1

III. 設定新的 Name

IV. 如果不想更新已重新命名之區域變數的參照，請取消選取更新已重新命名之元素的參照勾選框。

V. 按一下 OK 以執行快速的重構作業，或按一下 Preview 以執行受控制的重構作業。

VI. 預覽視窗會顯示重構要更動的部份

VII. 下半部的窗格顯示兩者的比較

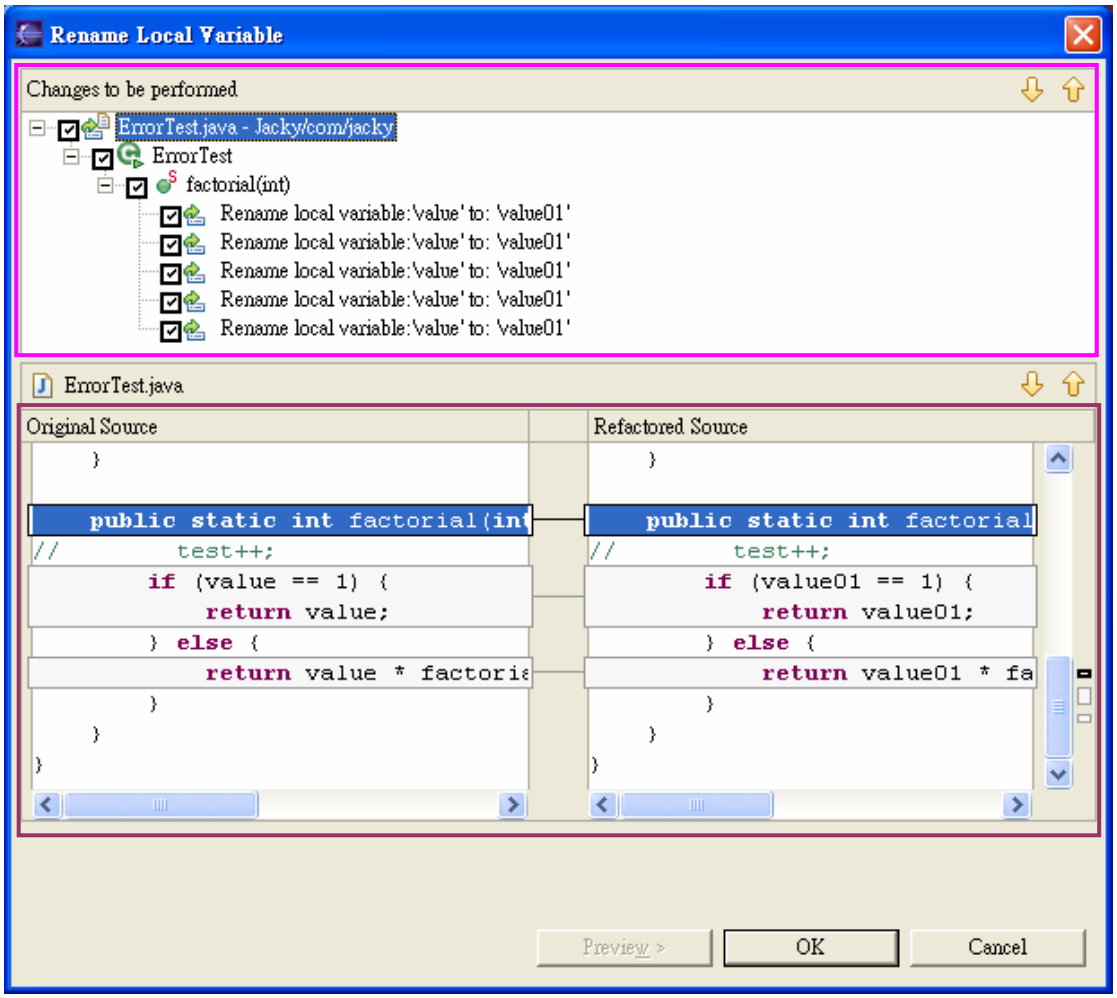


圖 6.2

6.1.2 欄位(Field)

如果要將欄位重新命名，請執行下列動作：

I. 在 Java 編輯器中選取欄位

II. 「Refactor」→「Rename」

(或是在編輯器按右鍵，選取「Refactor」→「Rename」)

出現 Rename Field 視窗

圖 6.3

III. 設定新的 Name

IV. 如果不想更新已重新命名之欄位的參照，請取消選取更新已重新命名之元素的參照勾選框。

V. 如果想更新字串文字中的參照，請選取更新字串文字中的參照勾選框。

VI. 如果重構作業找到要重新命名之欄位的存取元(getter/setter)方法，則重構作業會建議亦重新命名這些方法（並更新其所有參照）：

- 如果想要重新命名 Getter，請選取重新命名 Getter 勾選框
- 如果想要重新命名 Setter，請選取重新命名 Setter 勾選框

VII. 按一下 OK 以執行快速的重構作業，或按一下 Preview 以執行受控制的重構作業。

VIII. 預覽視窗會顯示重構要更動的部份

IX. 下半部的窗格顯示兩者的比較

VIII

IX

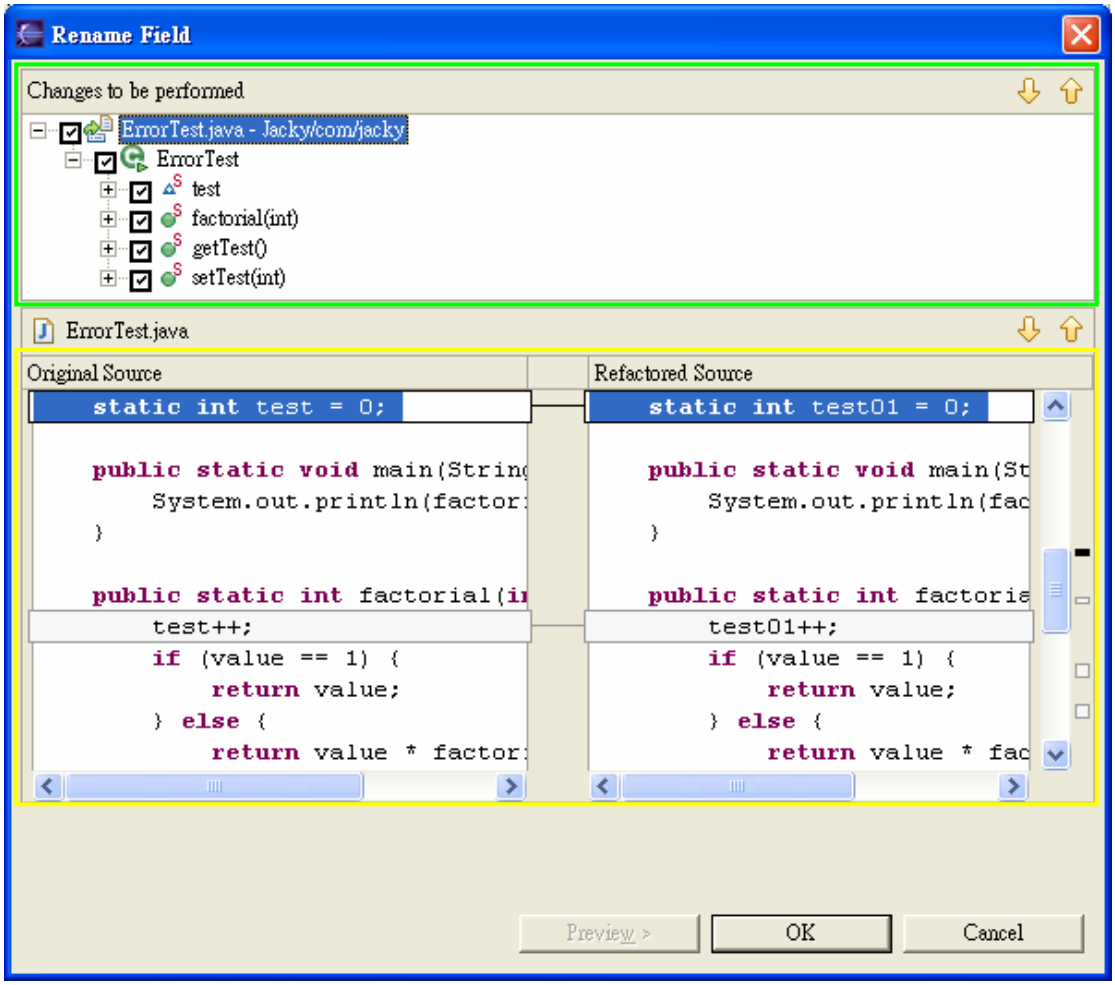


圖 6.4

6.1.3 方法(Method)

如果要將方法重新命名，請執行下列動作：

I. 在 Java 編輯器中選取方法的名稱

II. 「Refactor」→「Rename」

（或是在編輯器按右鍵，選取「Refactor」→「Rename」）

出現 Rename Method 視窗

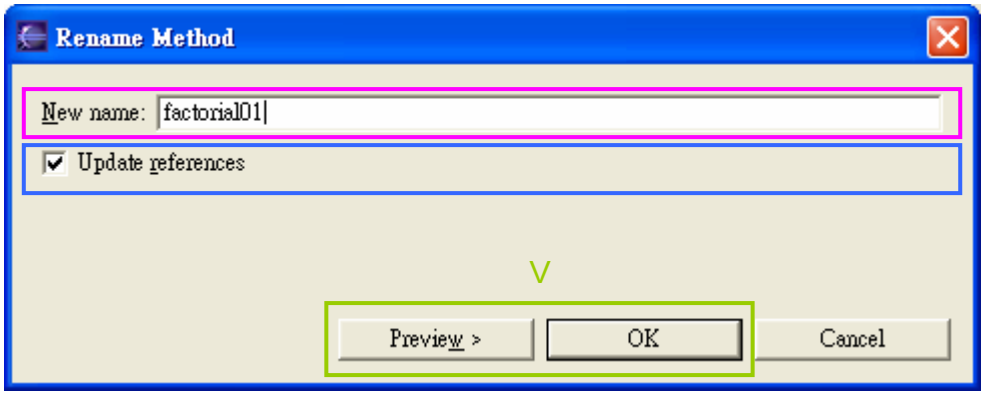
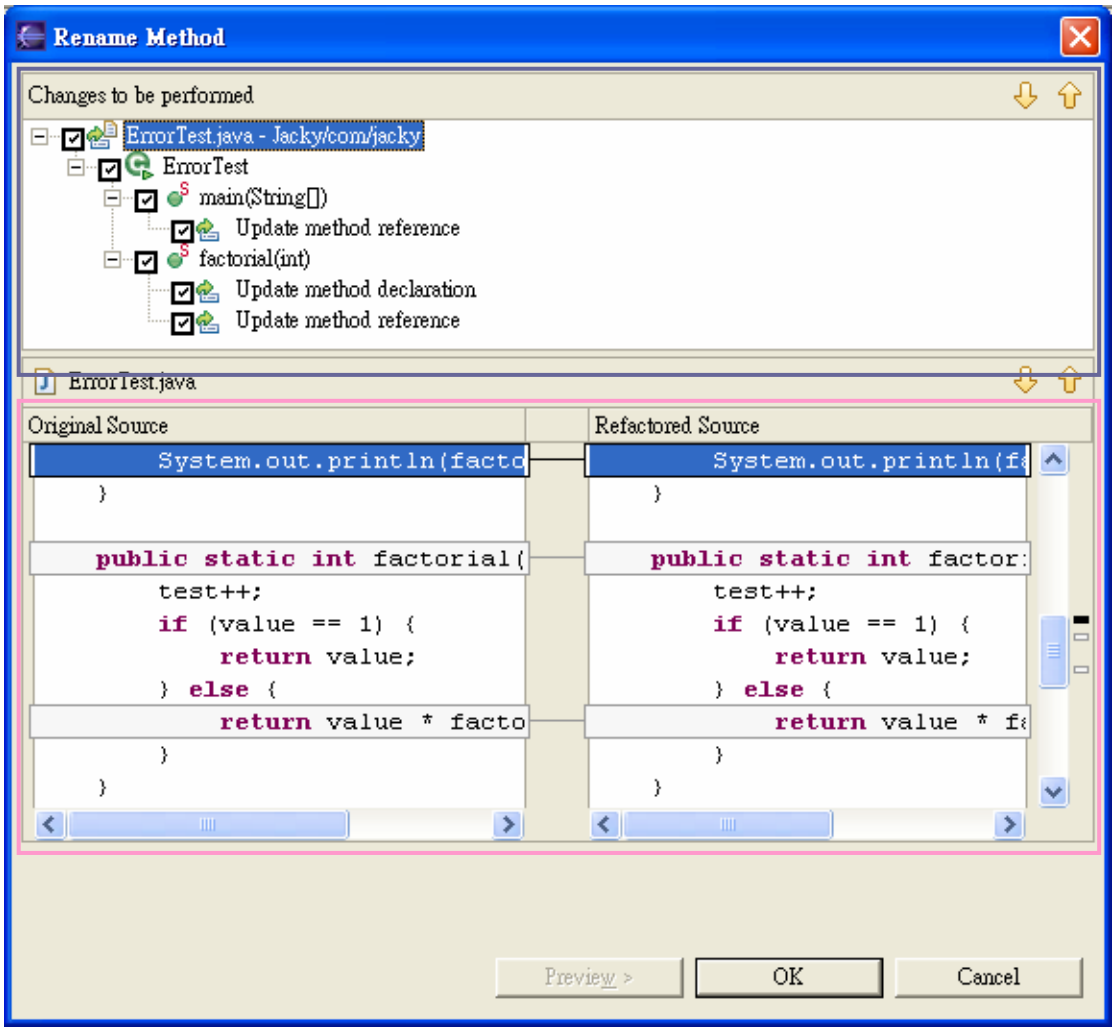


圖 6.5

III. 設定新的 Name

IV. 如果不想更新已重新命名之欄位的參照，請取消選取更新已重新命名之元素的參照勾選框。

V. 按一下 OK 以執行快速的重構作業，或按一下 Preview 以執行受控制的重構作業。



VI

VII

圖 6.6

VI. 預覽視窗會顯示重構要更動的部份

VII. 下半部的窗格顯示兩者的比較

6.1.4 類別(Class)或是介面(Interface)

如果要將類別重新命名，請執行下列動作：

I. 在 Java 編輯器中選取類別的名稱

II. 「Refactor」→「Rename」

（或是在編輯器按右鍵，選取「Refactor」→「Rename」）

出現 Rename Type 視窗

圖 6.7

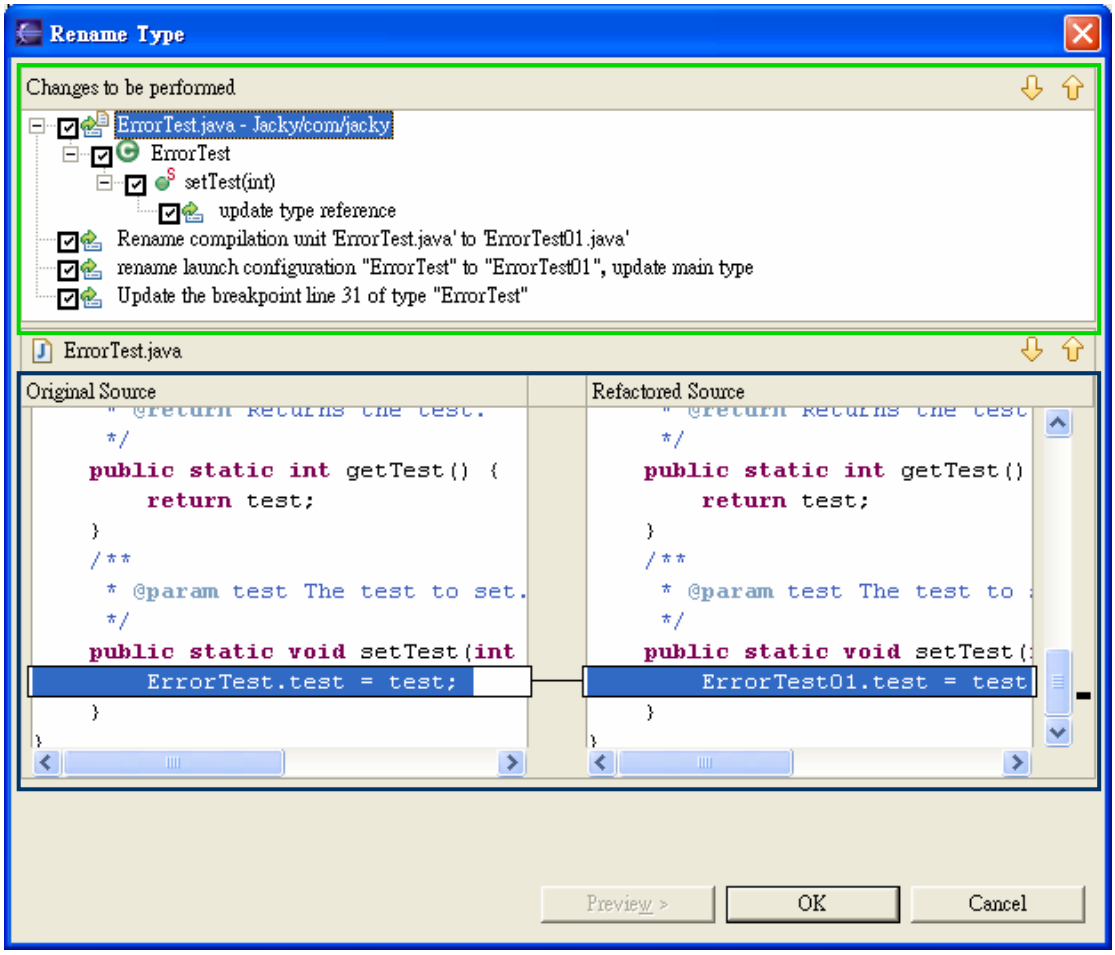
III. 設定新的 Name

IV. 如果不想更新已重新命名之類別或介面的參照，請取消選取更新已重新命名之元素的參照勾選框。

V. 如果想更新字串文字中的參照，請選取更新字串文字中的參照勾選框。

VI. 如果想更新一般（非 Javadoc）註解中的參照，請選取更新一般註解中的參照勾選框。

VII. 按一下確定以執行快速的重構作業，或按一下預覽以執行受控制的重構作業。



VIII

IX

圖 6.8

VIII. 預覽視窗會顯示重構要更動的部份

IX. 下半部的窗格顯示兩者的比較

6.1.5 套件(Package)

如果要將套件重新命名，請執行下列動作：

I. 在 Java 編輯器中選取套件的名稱

II. 「Refactor」→「Rename」

(或是在編輯器按右鍵，選取「Refactor」→「Rename」)

出現 Rename Package 視窗

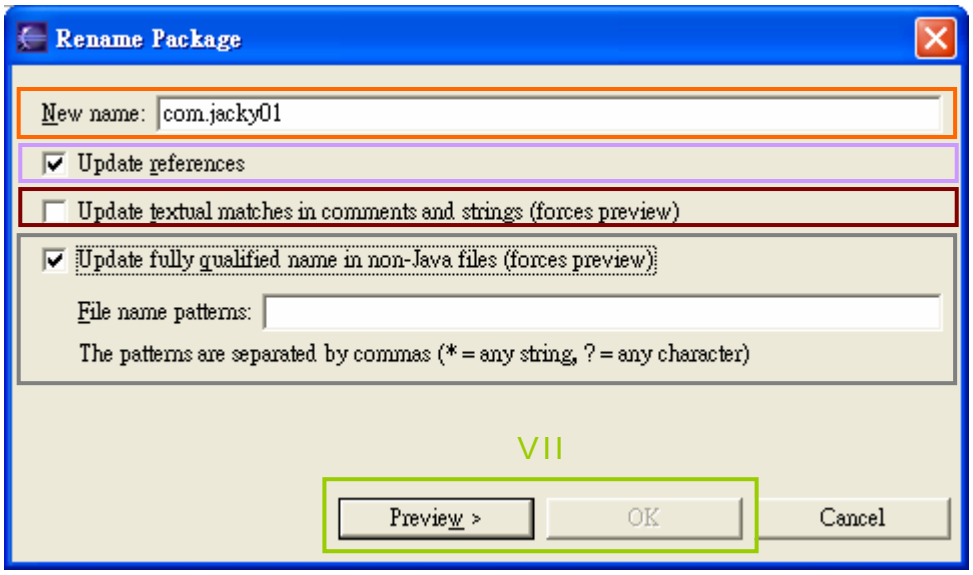


圖 6.9

III. 設定新的 Name

IV. 如果不想更新已重新命名之類別或介面的參照，請取消選取更新已重新命名之元素的參照勾選框。

V. 如果想更新字串文字中的參照，請選取更新字串文字中的參照勾選框。

VI. 如果想更新一般（非 Javadoc）註解中的參照，請選取更新一般註解中的參照勾選框。

VII. 按下預覽，以查看變更的預覽，或按下確定，執行重構作業，不查看預覽。

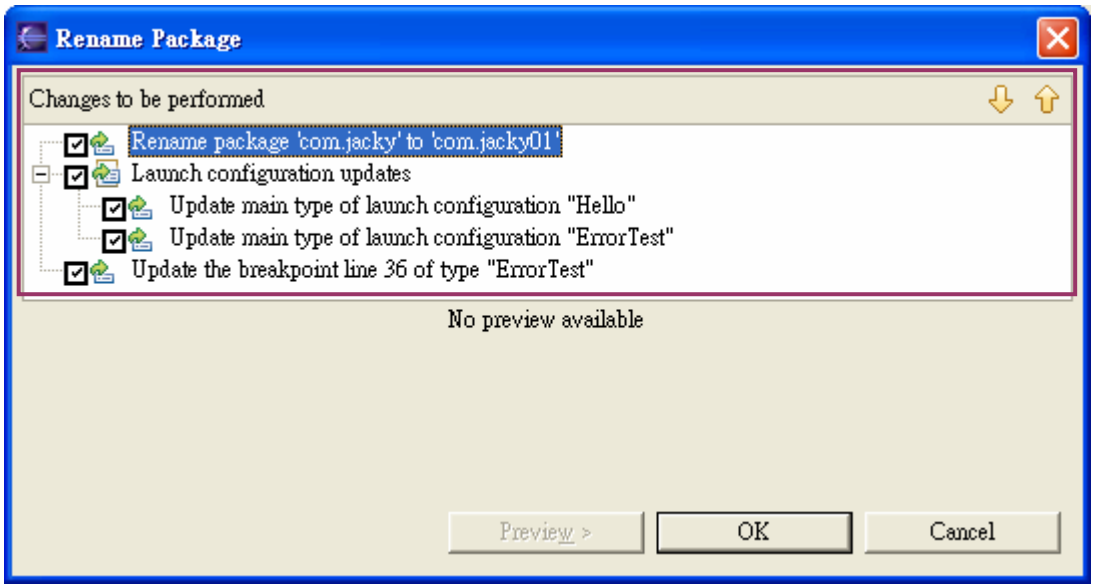


圖 6.10

VIII. 預覽視窗會顯示重構要更動的部份

6.2 擷取(Extracting)

6.2.1 擷取常數(Extracting a Constant)

如果要擷取常數，請執行下列動作：

I. 在 Java 編輯器中選取常數

II. 「Refactor」→「Extract Constant...」（或是在編輯器按右鍵，選取「Refactor」→「Extract Constant...」）

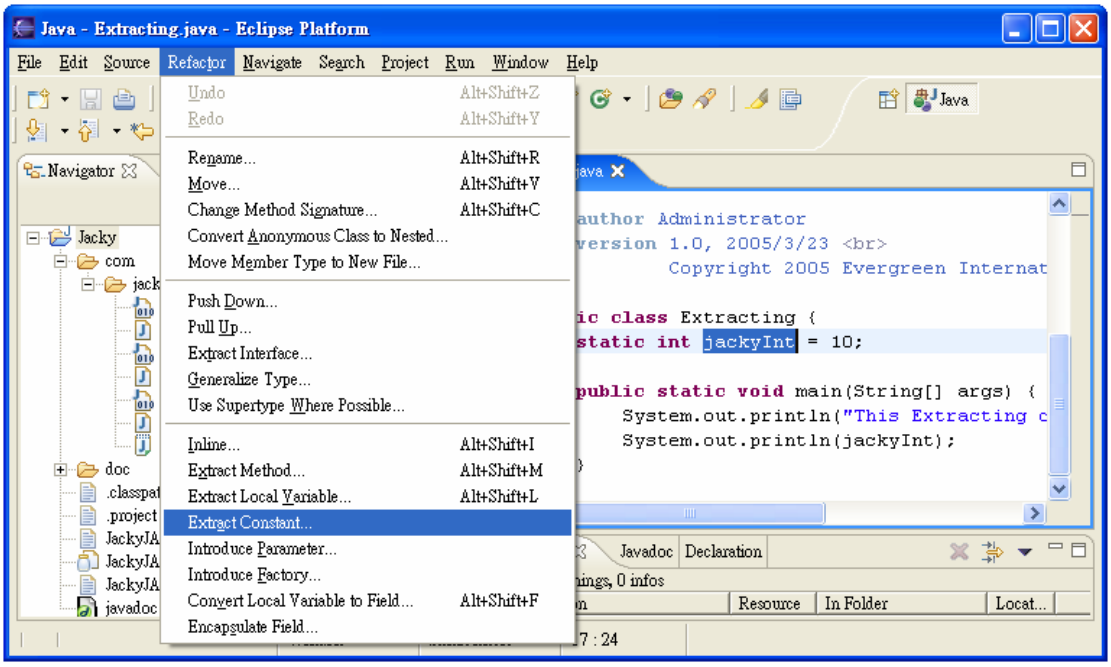


圖 6.11

出現 Extract Constan 視窗

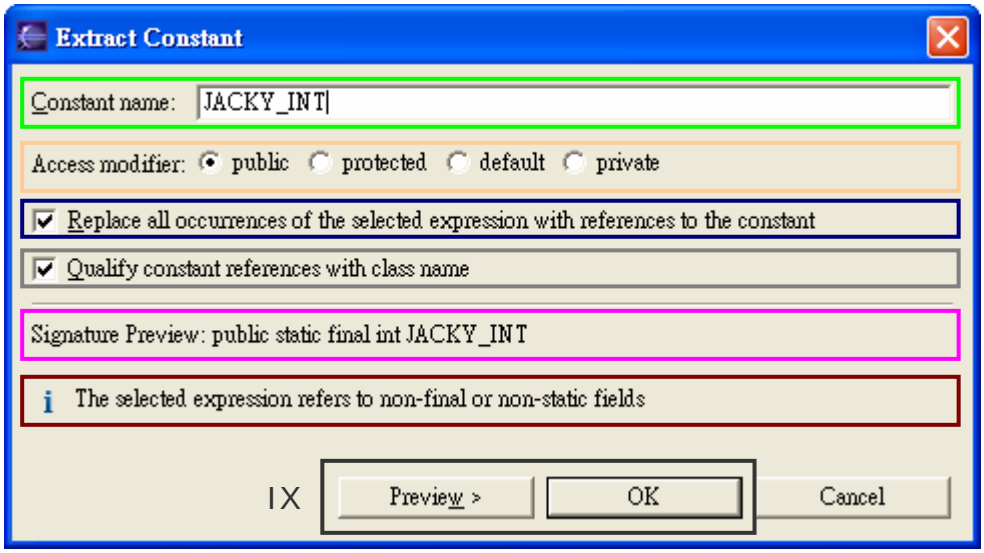


圖 6.12

III. 設定新的 Name

IV. 在存取修飾元清單中，指定方法的可見性（public、default、protected 或 private）。

V. 在呼叫重構作業時，如果只想更換所選的表式示，可選擇性地清除將所有出現所選表示式之處換成常數的參照勾選框

VI. 變數名稱前是否要帶限定者的名稱。

- 若選取 Qualify constant reference with class name 勾選框

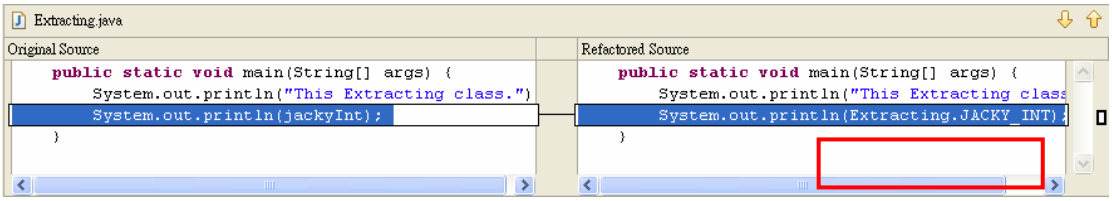


圖 6.13

- 若沒有選取 Qualify constant reference with class name 勾選框

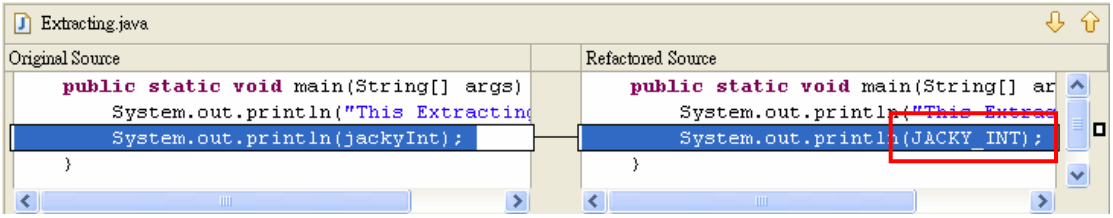


圖 6.14

VII. 立即顯示設定後的狀態。

VIII. 訊息。

若沒有符合 Java 命名規則，會出現警告的訊息

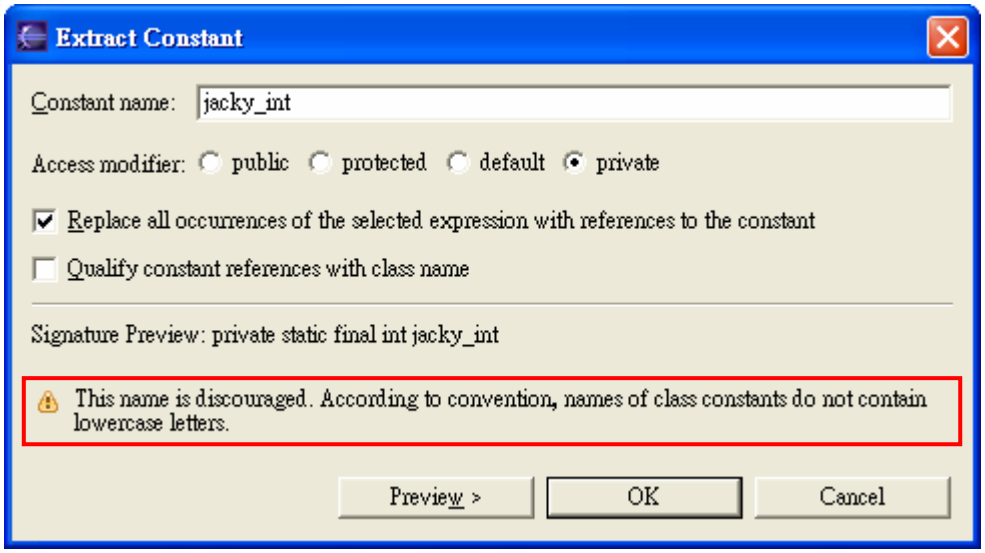


圖 6.15

IX. 按一下確定以執行快速的重構作業，或按一下預覽以執行受控制的重構作業。

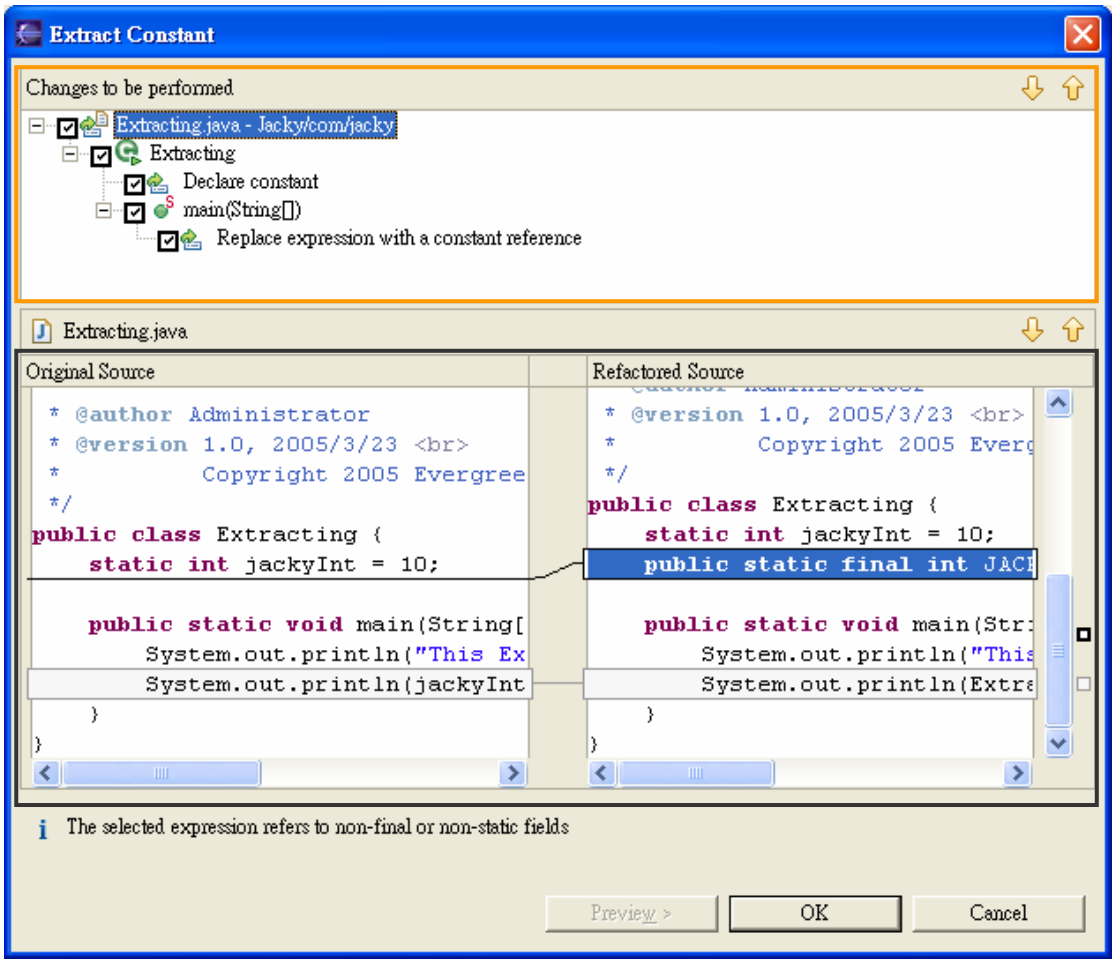


圖 6.16

X. 預覽視窗會顯示重構要更動的部份

XI. 下半部的窗格顯示兩者的比較

程式碼如下：

```
public class Extracting {
    static int jackyInt = 10;

    public static void main(String[] args) {
        System.out.println("This's Extracting class.");
        System.out.println(jackyInt);
    }
}
```

6.2.2 擷取區域變數(Extracting a Local Variable)

如果要擷取區域變數，請執行下列動作：

I. 在 Java 編輯器中選取區域變數

II. 「Refactor」→「Extract Local Variable...」（或是在編輯器按右鍵，選取「Refactor」→「Extract Local Variable...」）

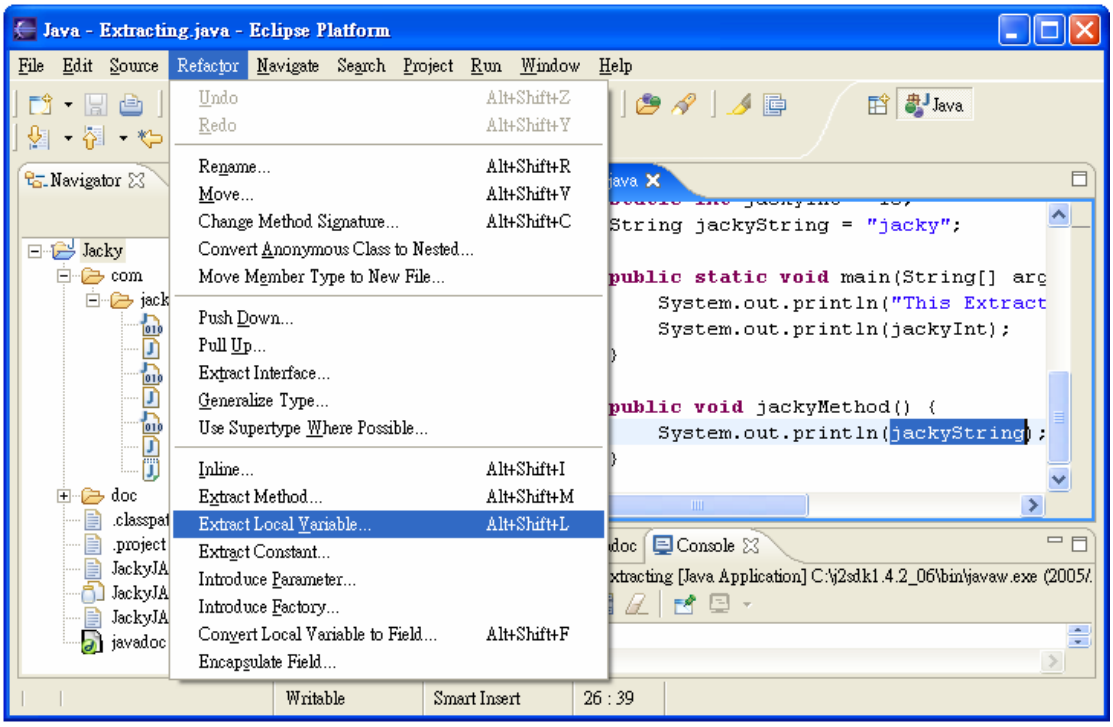


圖 6.17

出現 Extract Local Variable 視窗

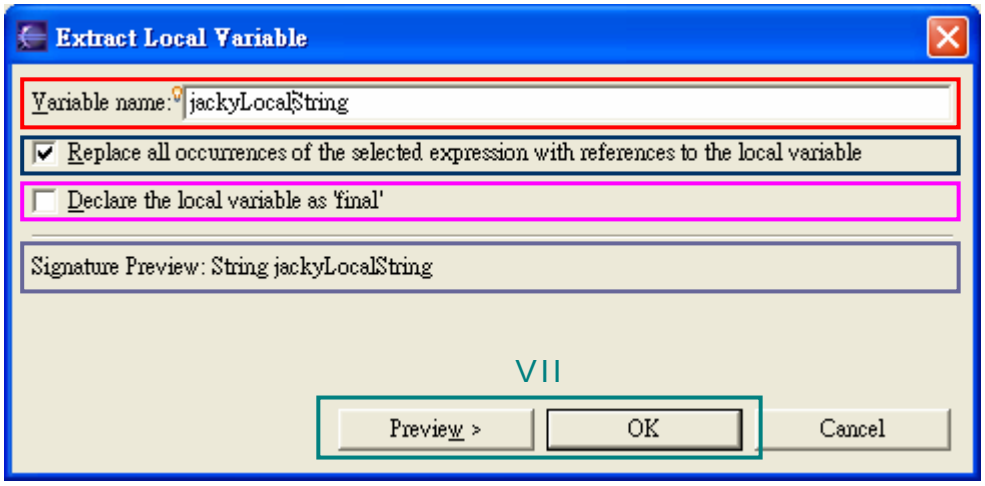


圖 6.18

III. 設定新的 Name

IV. 在呼叫重構作業時，如果只想更換所選的表式示，可選擇性地清除將所有出現所選表示式之處換成區域變數的參照勾選框。

V. 可選擇性地選取將區域變數定義成'final'。

VI. 立即顯示設定後的狀態。

若沒有符合 Java 命名規則，會出現警告的訊息

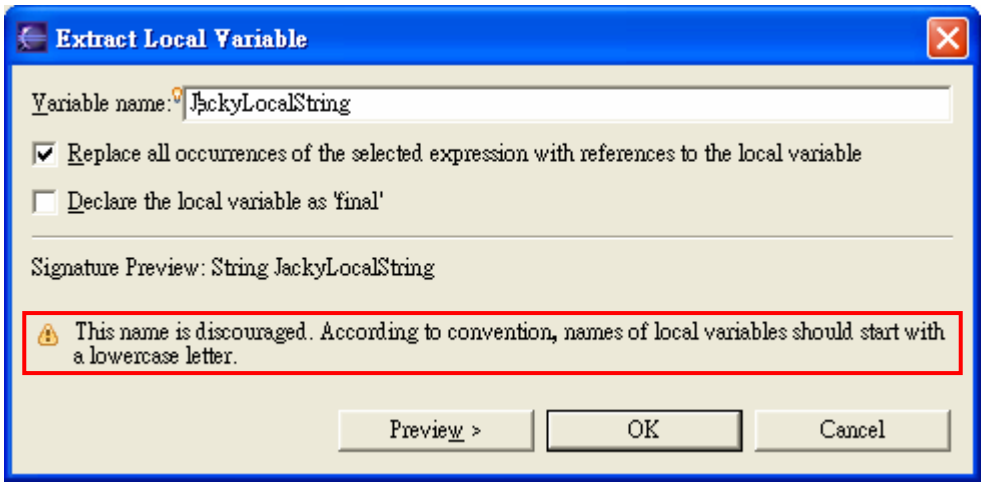
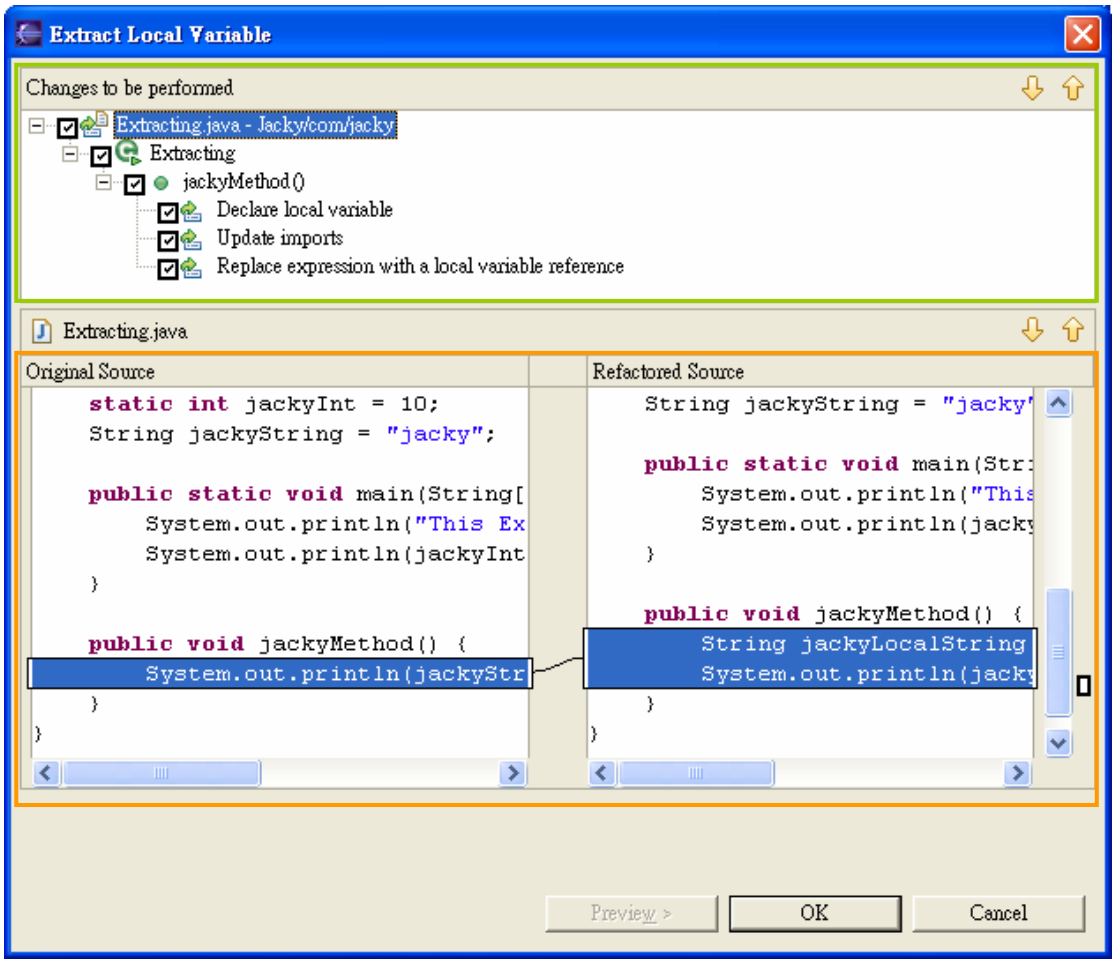


圖 6.19

VII. 按下預覽，以查看變更的預覽，或按下確定，執行重構作業，不查看預覽。



VIII

IX

圖 6.20

VIII. 預覽視窗會顯示重構要更動的部份

IX. 下半部的窗格顯示兩者的比較

程式碼如下：

```
public class Extracting {
    static int jackyInt = 10;
    String jackyString = "jacky";

    public static void main(String[] args) {
        System.out.println("This's Extracting class.");
        System.out.println(jackyInt);
    }
```

```
    public void jackyMethod() {
        System.out.println(jackyString);
    }
}
```

6.2.3 擷取方法(Extracting a Method)

如果要擷取方法，請執行下列動作：

I. 在 Java 編輯器中選取方法

II. 「Refactor」→「Extract Method...」

(或是在編輯器按右鍵，選取「Refactor」→「Extract Method...」)

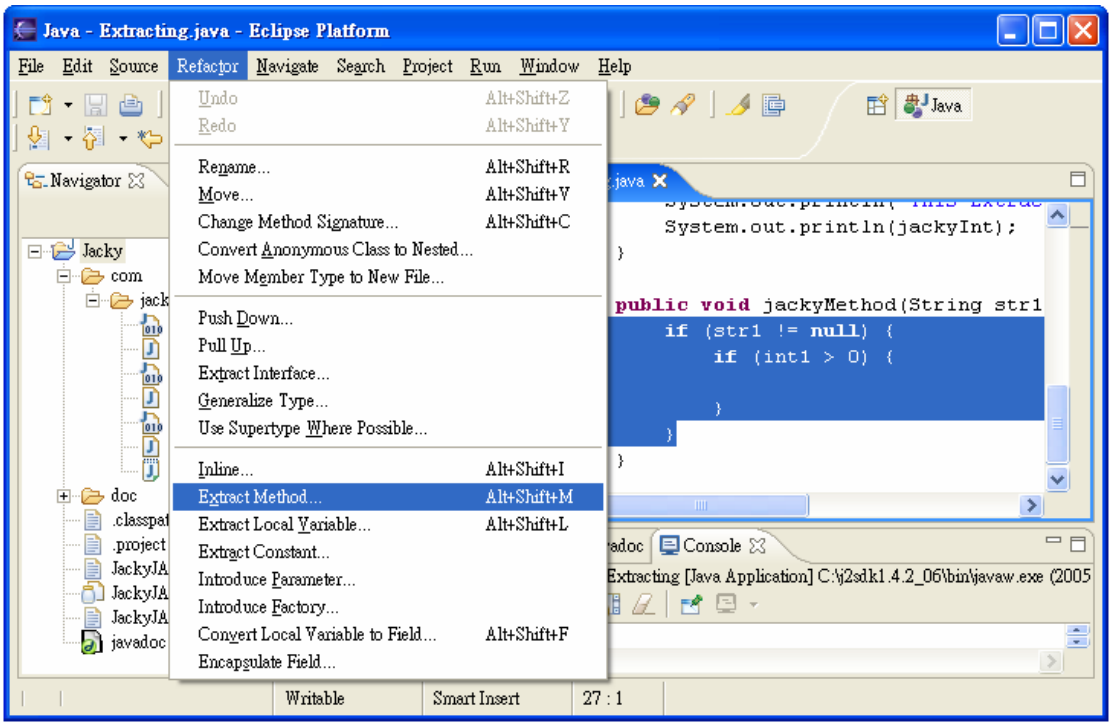


圖 6.21

出現 Extract Methode 視窗

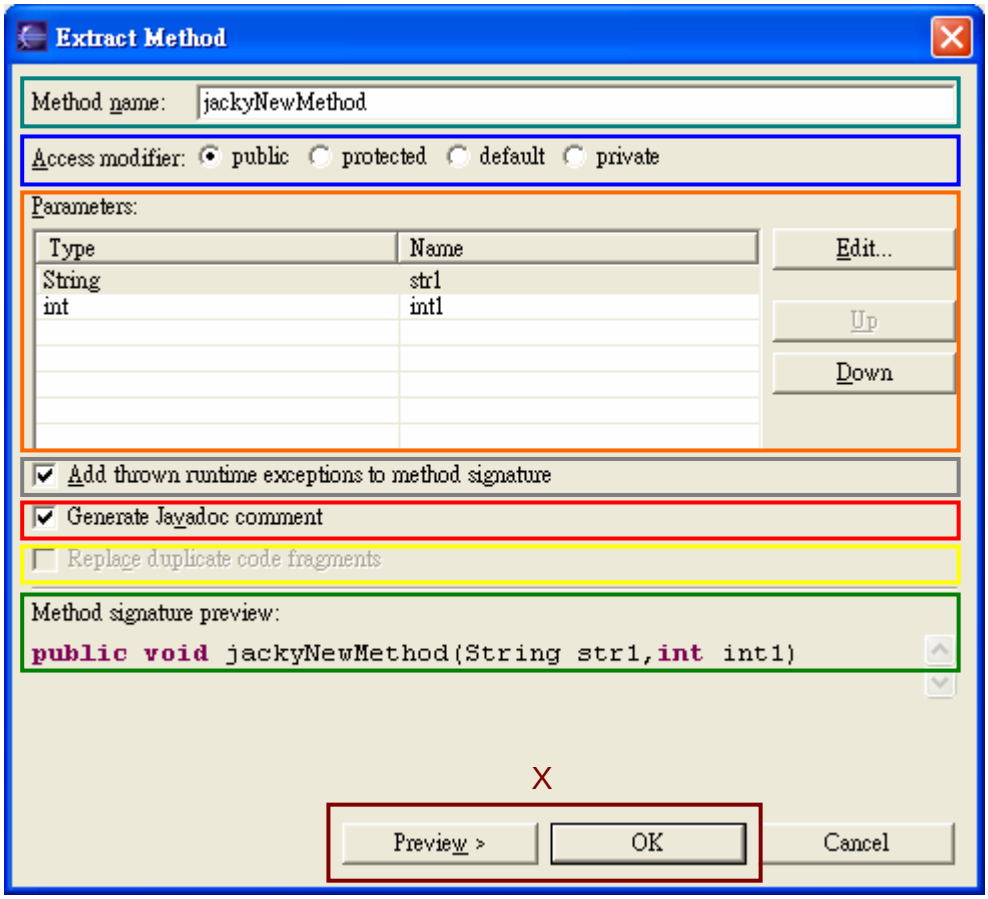


圖 6.22

III. 設定新的 Name

IV. 在存取修飾元清單中，指定方法的可見性（public、default、protected 或 private）。

V. 可以重新排列新方法的參數，並且將它重新命名。

重新命名

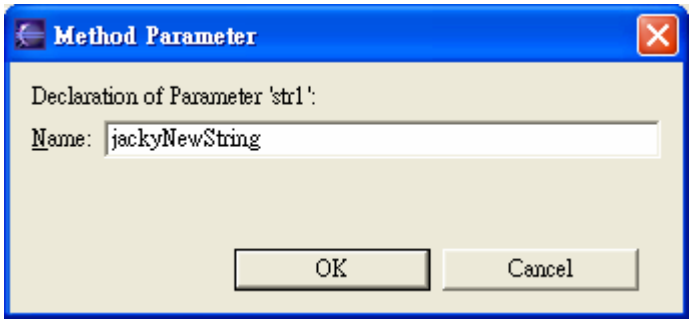


圖 6.23

重新排列

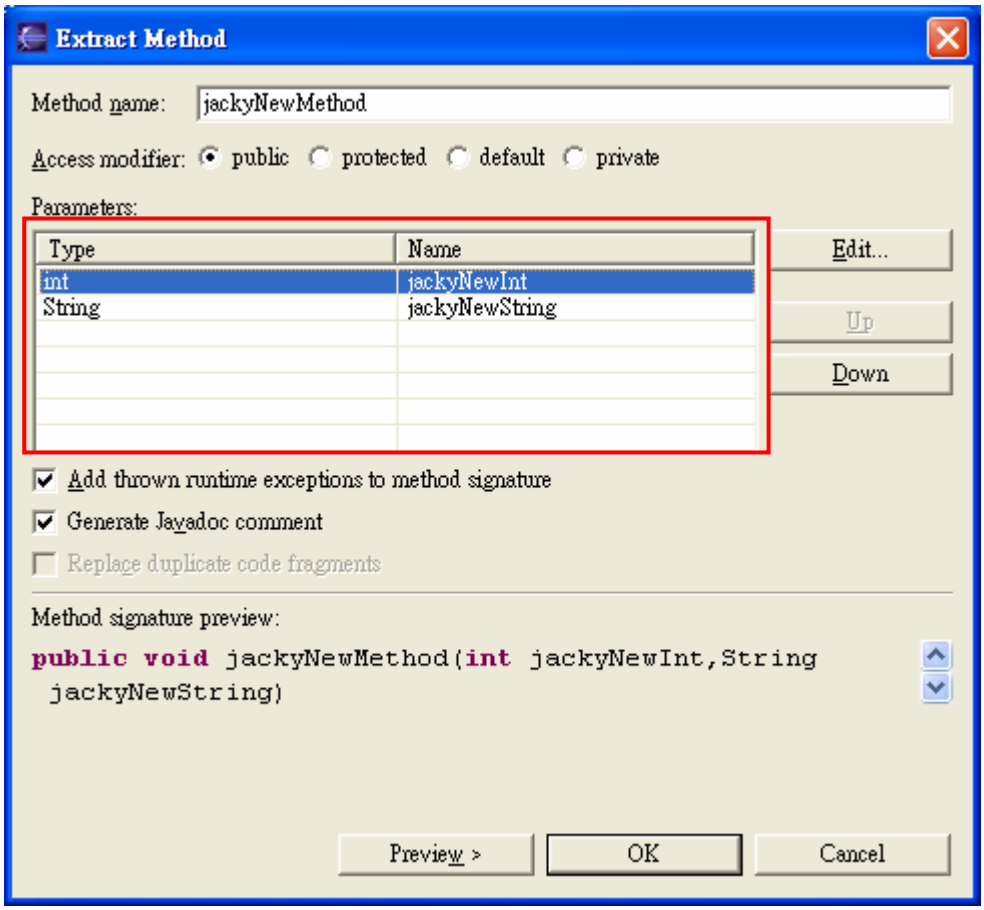


圖 6.24

VI. 可以新增擲出執行時期異常狀況到方法簽章中，方法是選取對應的勾選框。

VII. 產生 Javadoc 註解

VIII. 取代重複的程式碼片段

IX. 立即顯示設定後的狀態。

若沒有符合 Java 命名規則，會出現警告的訊息

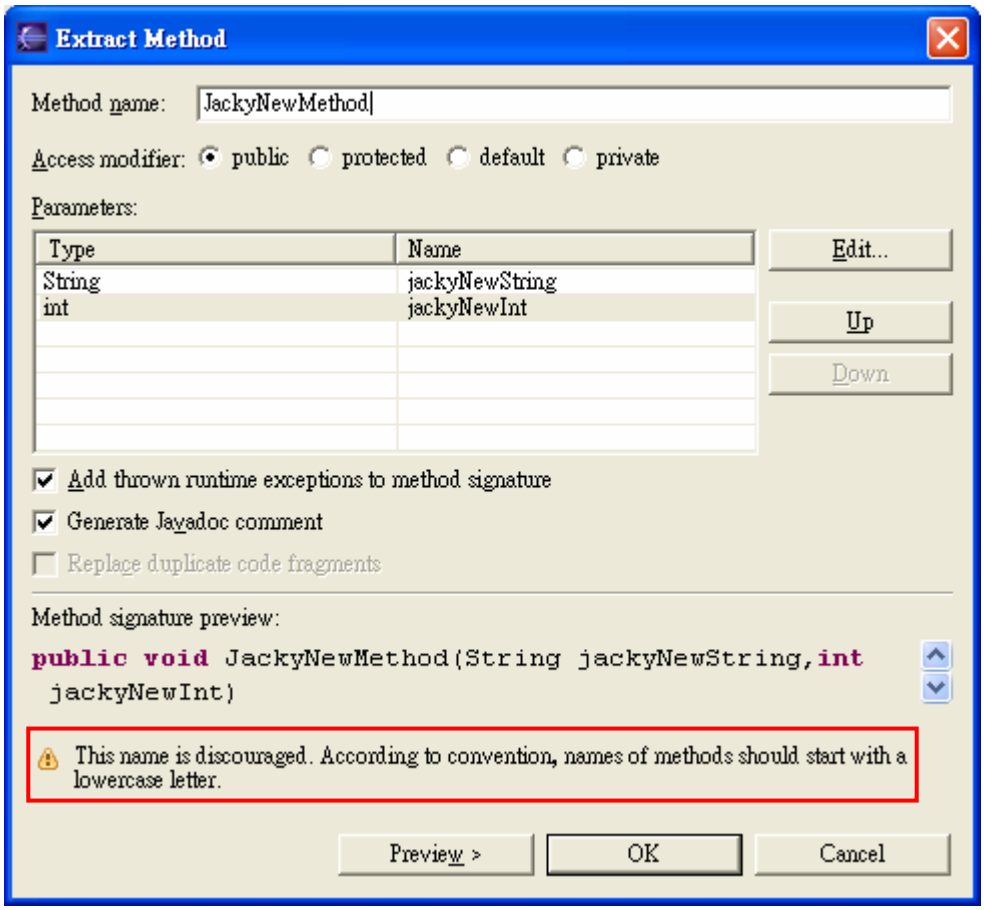
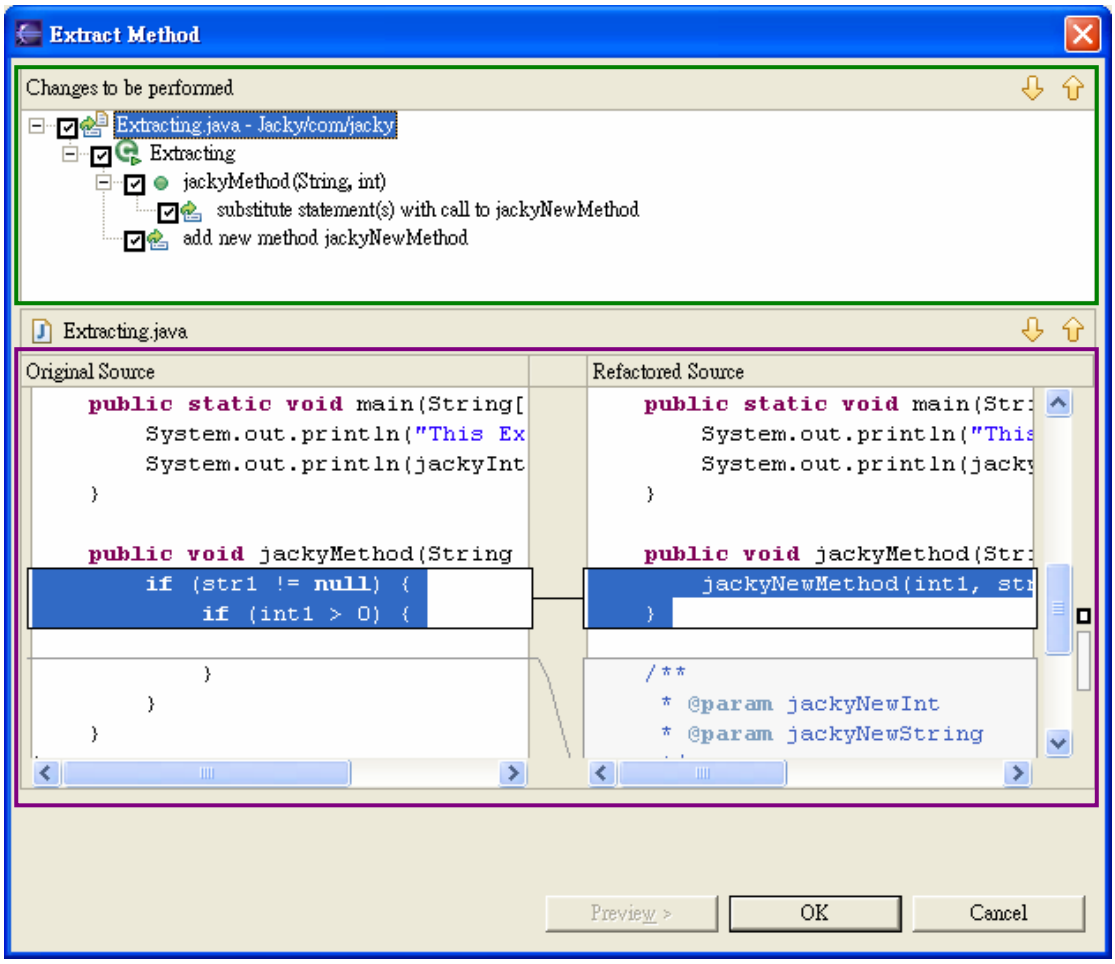


圖 6.25

X. 按一下確定以執行快速的重構作業，或按一下預覽以執行受控制的重構作業。



XI

XII

圖 6.26

XI. 預覽視窗會顯示重構要更動的部份

XII. 下半部的窗格顯示兩者的比較

程式碼如下：

```
public class Extracting {
    static int jackyInt = 10;

    String jackyString = "jacky";

    public static void main(String[] args) {
        System.out.println("This's Extracting class.");
        System.out.println(jackyInt);
```

```
    }

    public void jackyMethod(String str1, int int1) {
        if (str1 != null) {
            if (int1 > 0) {

            }
        }
    }
}
```

6.3 列入(Inlining)

程式碼如下：

```
public class Inlining {
    static final int jackyInt = 10;

    public static void main(String[] args) {
        System.out.println("This's Inlining class.");
        System.out.println(jackyInt);
    }

    public void jackyMethod() {
        jackyMethod("jacky", 5);
    }

    public void jackyMethod(String str1, int int1) {
```

```
        String jackyString = "Jacky";
        if (jackyString != null) {
            System.out.println(jackyString);
        }
    }
}
```

6.3.1 列入常數(Inlining a Constant)

如果要列入常數，請執行下列動作：

I. 在 Java 編輯器中選取常數

II. 「Refactor」→「Extract Inlining...」

(或是在編輯器按右鍵，選取「Refactor」→「Extract Inlining...」)

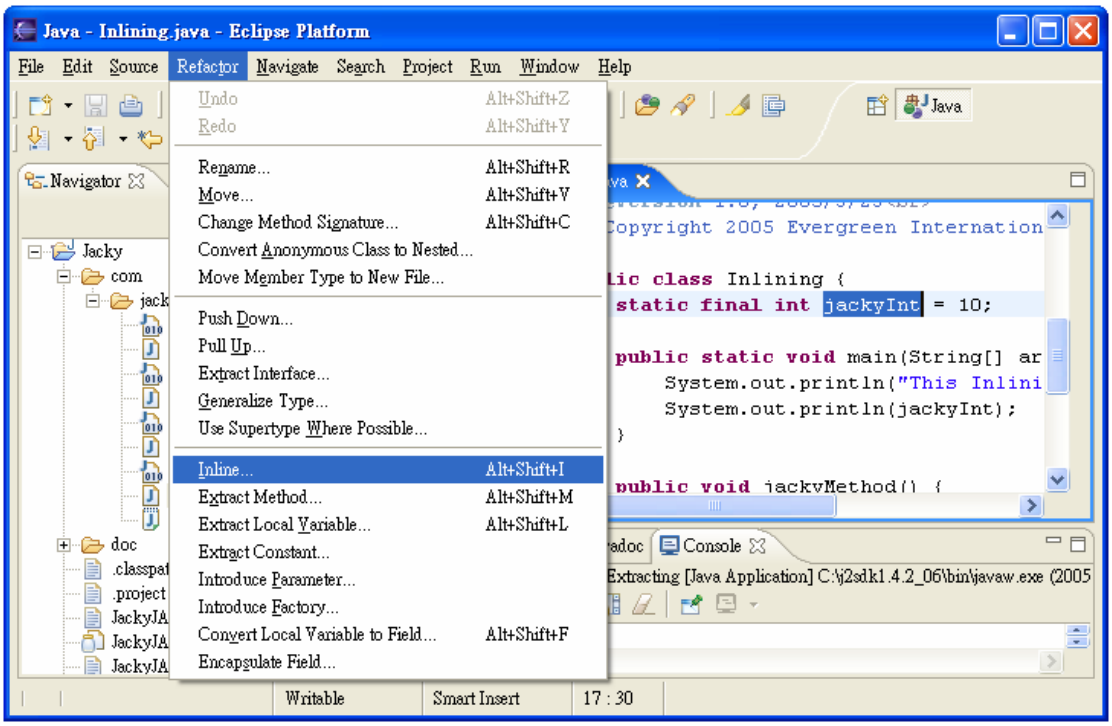


圖 6.27

出現 Inline Constant 視窗

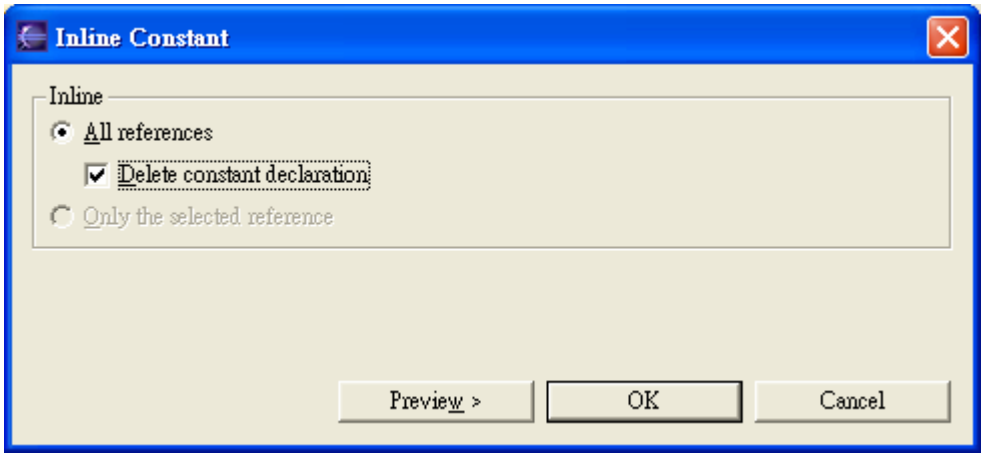


圖 6.28

有選 Delete constant declaration，常數會被刪除

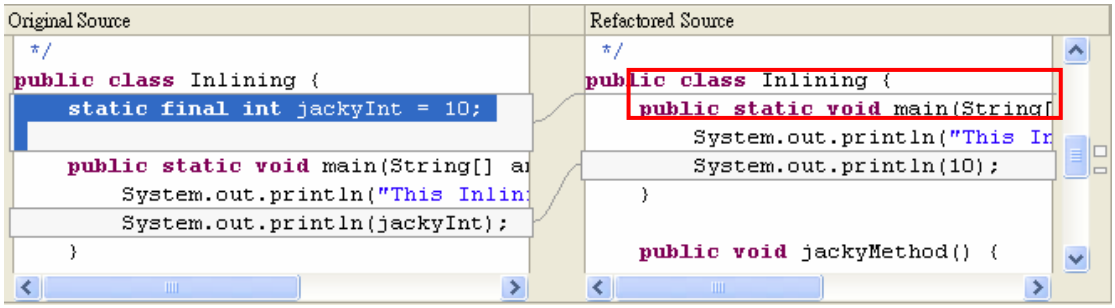


圖 6.29

沒有選 Delete constant declaration，常數不會被刪除

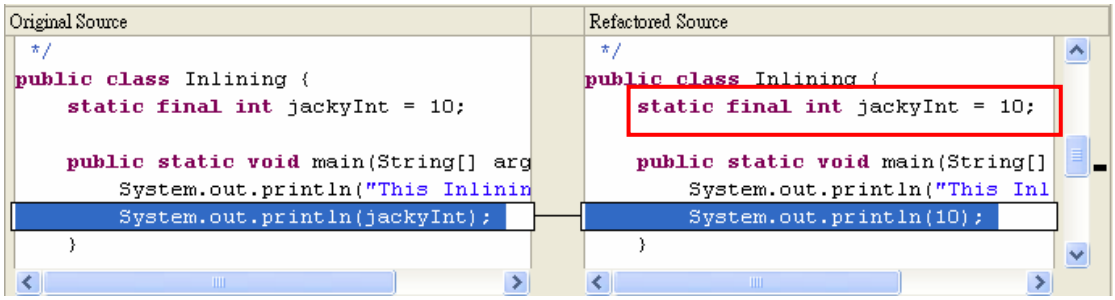


圖 6.30

預覽視窗會顯示重構要更動的部份，下半部的窗格顯示兩者的比較

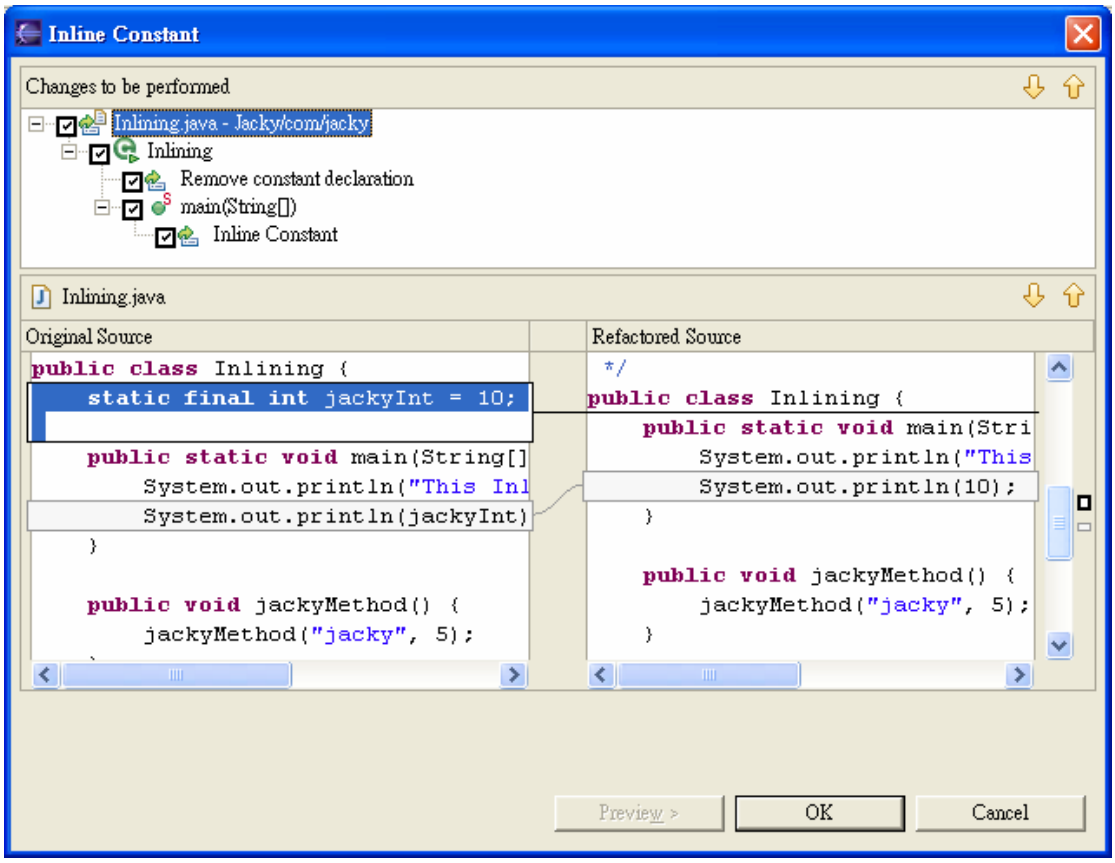


圖 6.31

6.3.2 列入區域變數(Inlining a Local Variable)

如果要列入區域變數，請執行下列動作：

I. 在 Java 編輯器中選取區域變數

II. 「Refactor」→「Extract Inlining...」

(或是在編輯器按右鍵，選取「Refactor」→「Extract Inlining...」)

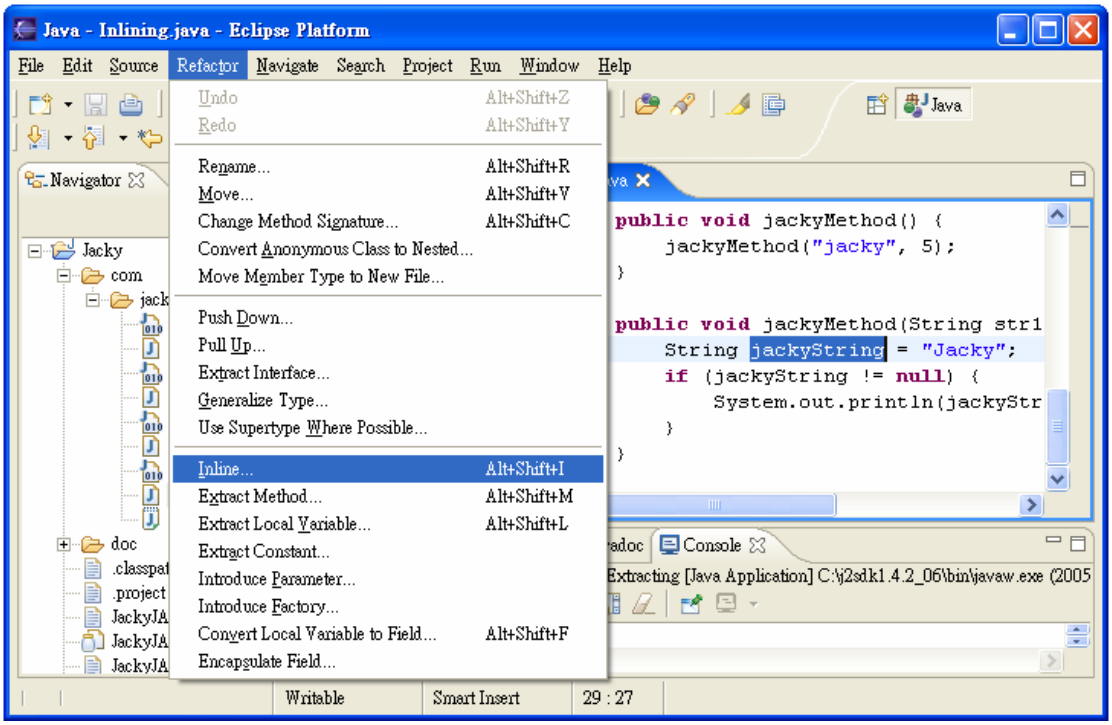


圖 6.32

出現 Inline Local Variable 視窗

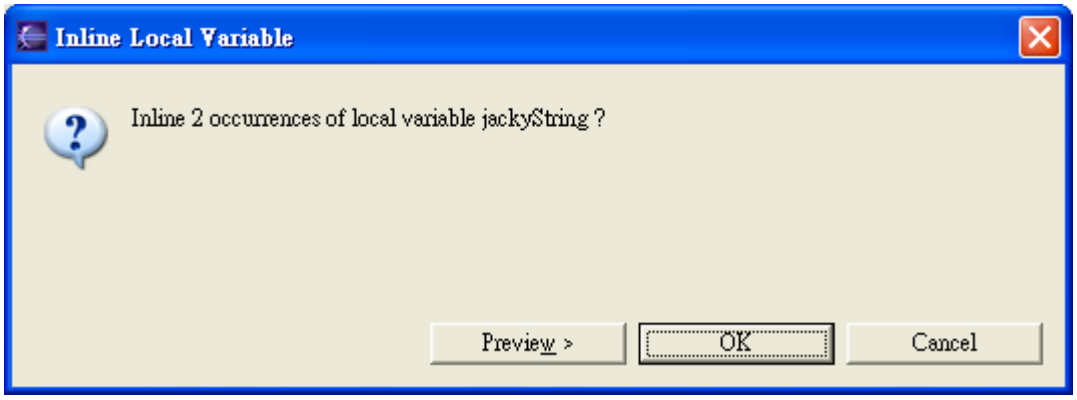


圖 6.33

預覽視窗會顯示重構要更動的部份，下半部的窗格顯示兩者的比較

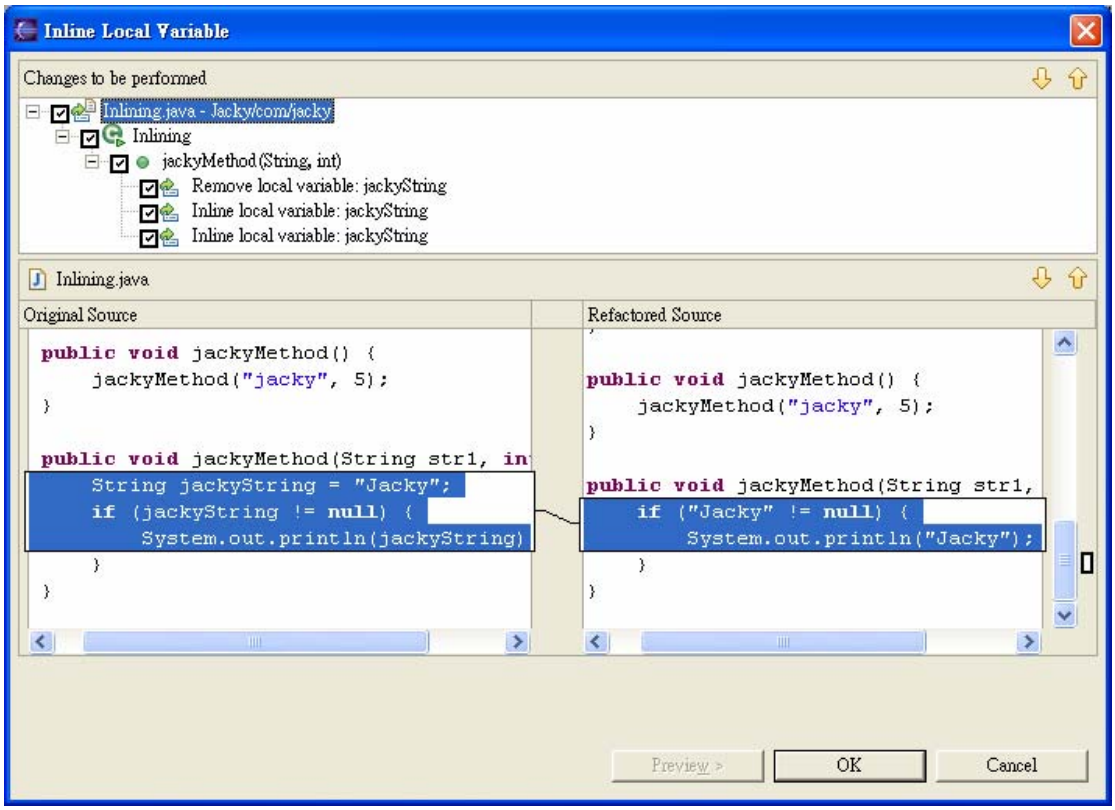


圖 6.34

6.3.3 列入方法(Inlining a Method)

如果要列入方法，請執行下列動作：

I. 在 Java 編輯器中選取方法

II. 「Refactor」→「Extract Inlining...」

(或是在編輯器按右鍵，選取「Refactor」→「Extract Inlining...」)

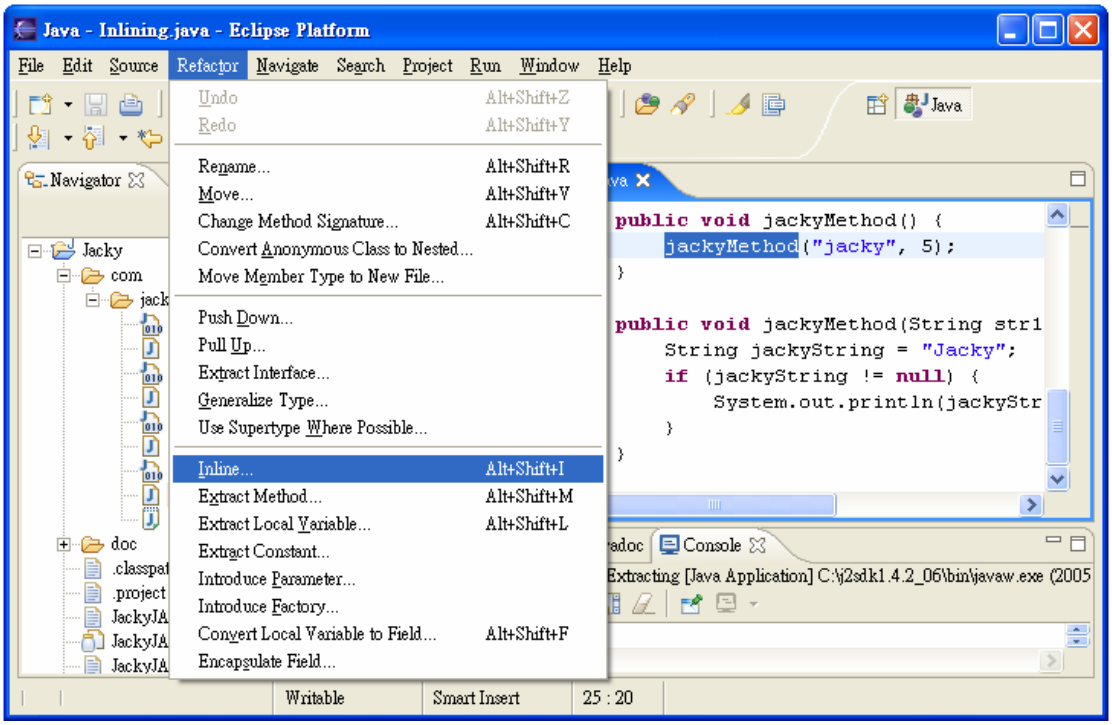


圖 6.35

出現 Inline Constant 視窗

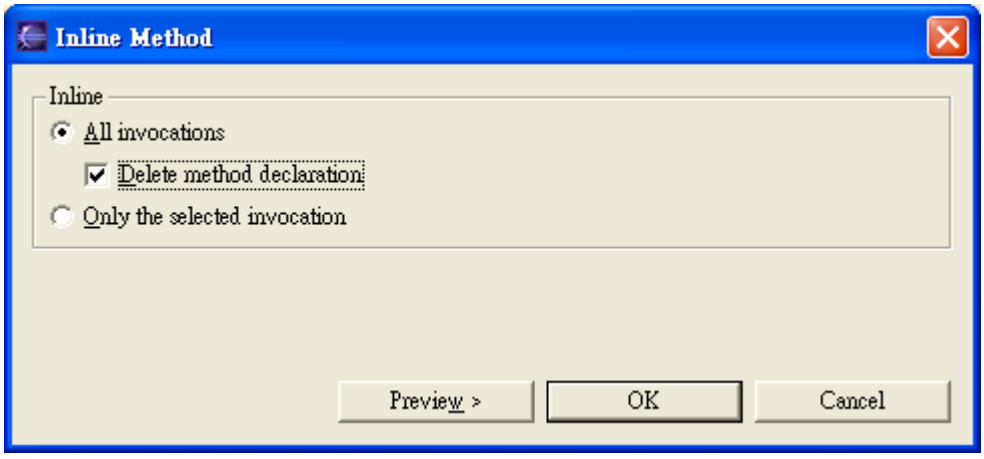


圖 6.36

有選 Delete method declaration，方法會被刪除

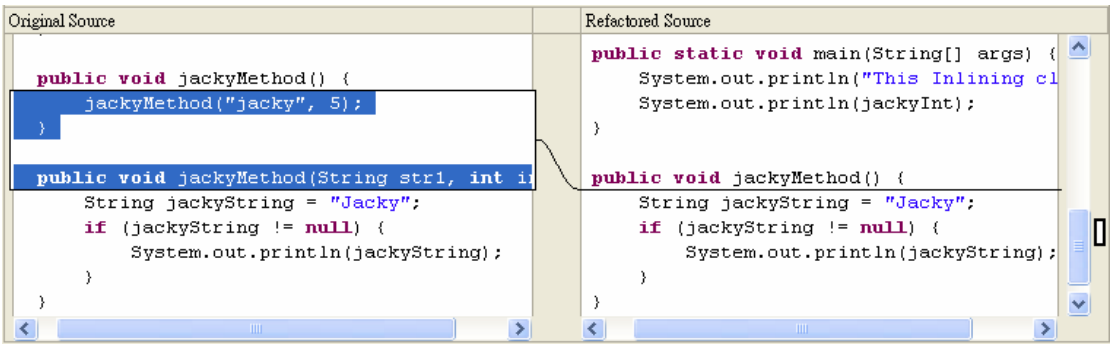


圖 6.37

沒有選 Delete method declaration，方法不會被刪除

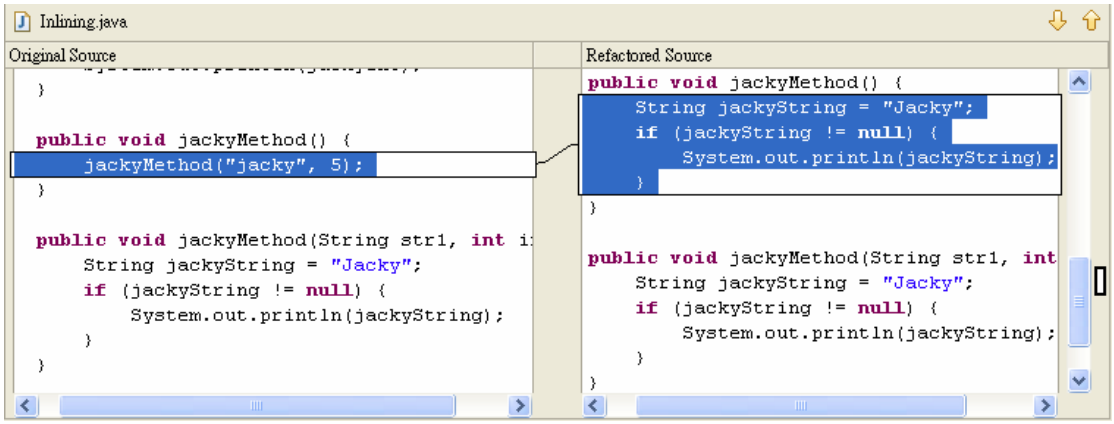


圖 6.38

這裡可以選擇要重構的動作是針對該方法全部的呼叫或是只針對選擇的部份

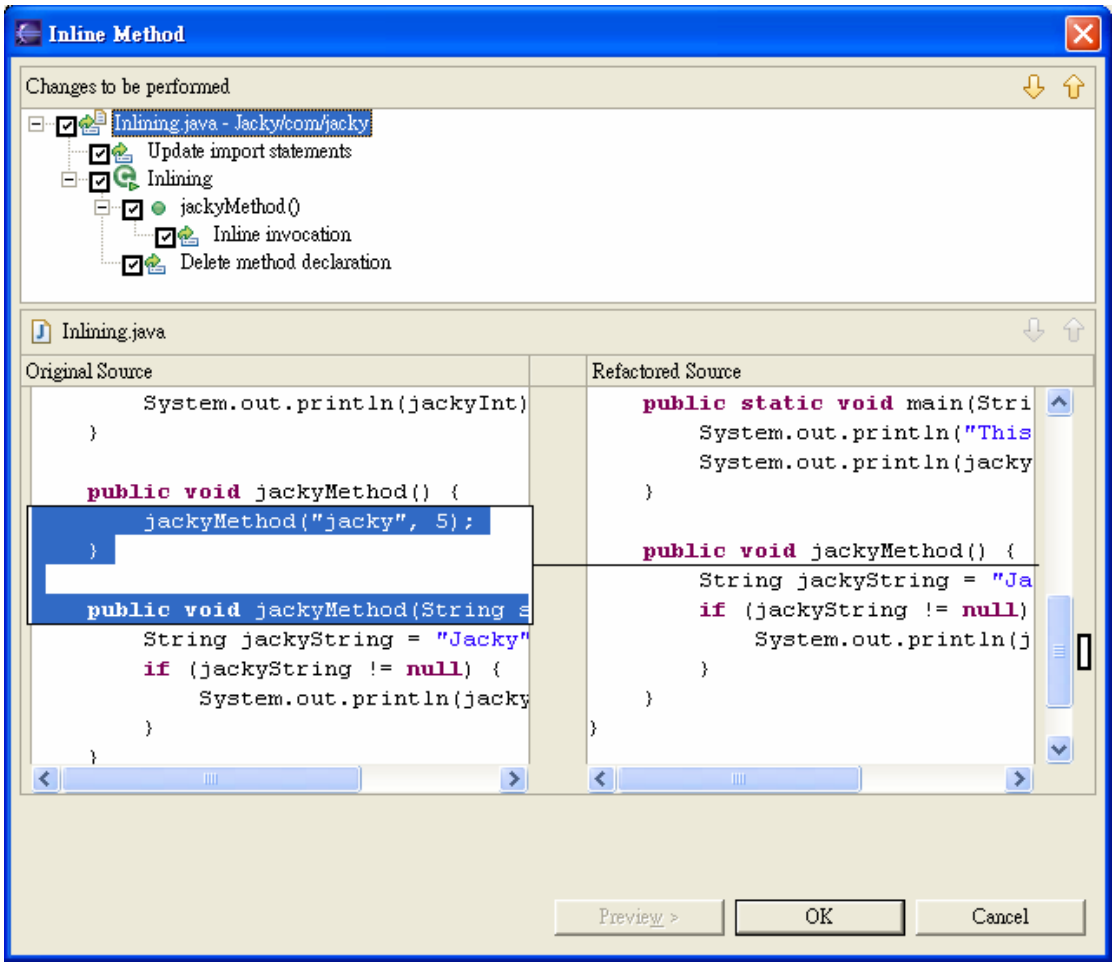


圖 6.39

預覽視窗會顯示重構要更動的部份，下半部的窗格顯示兩者的比較

6.4 變更方法簽章(Signature)

如果要變更方法簽章，請執行下列動作：

I. 在 Java 編輯器中選取方法

II. 「Refactor」→「Change Method Signature...」(或是在編輯器按右鍵，選取「Refactor」→「Change Method Signature...」)

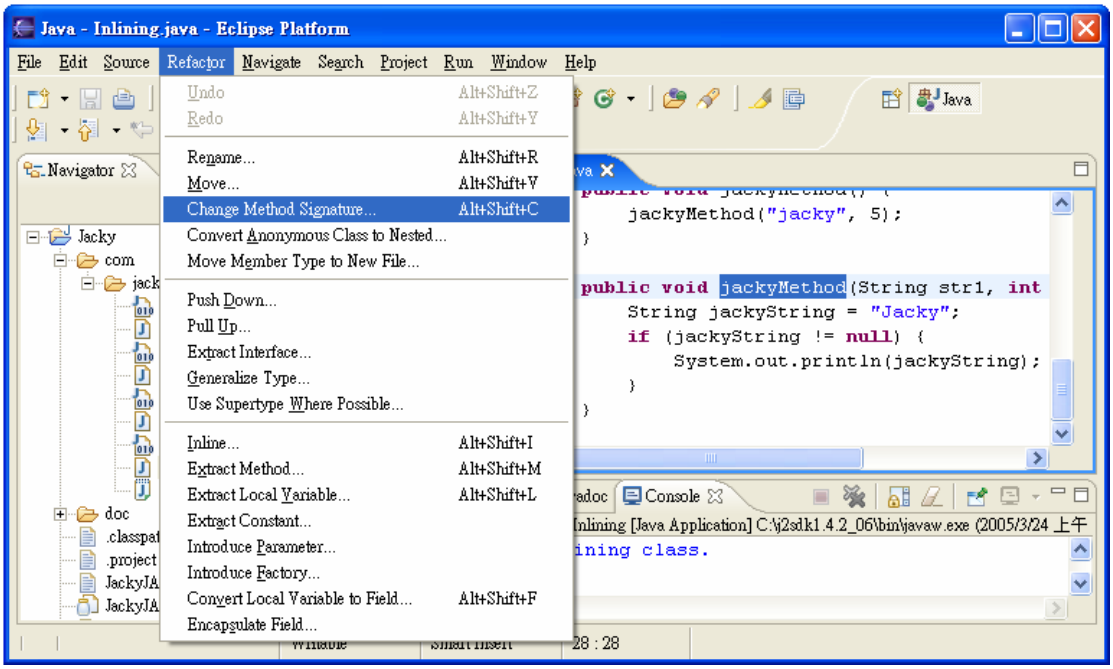


圖 6.40

出現 Inline Constant 視窗

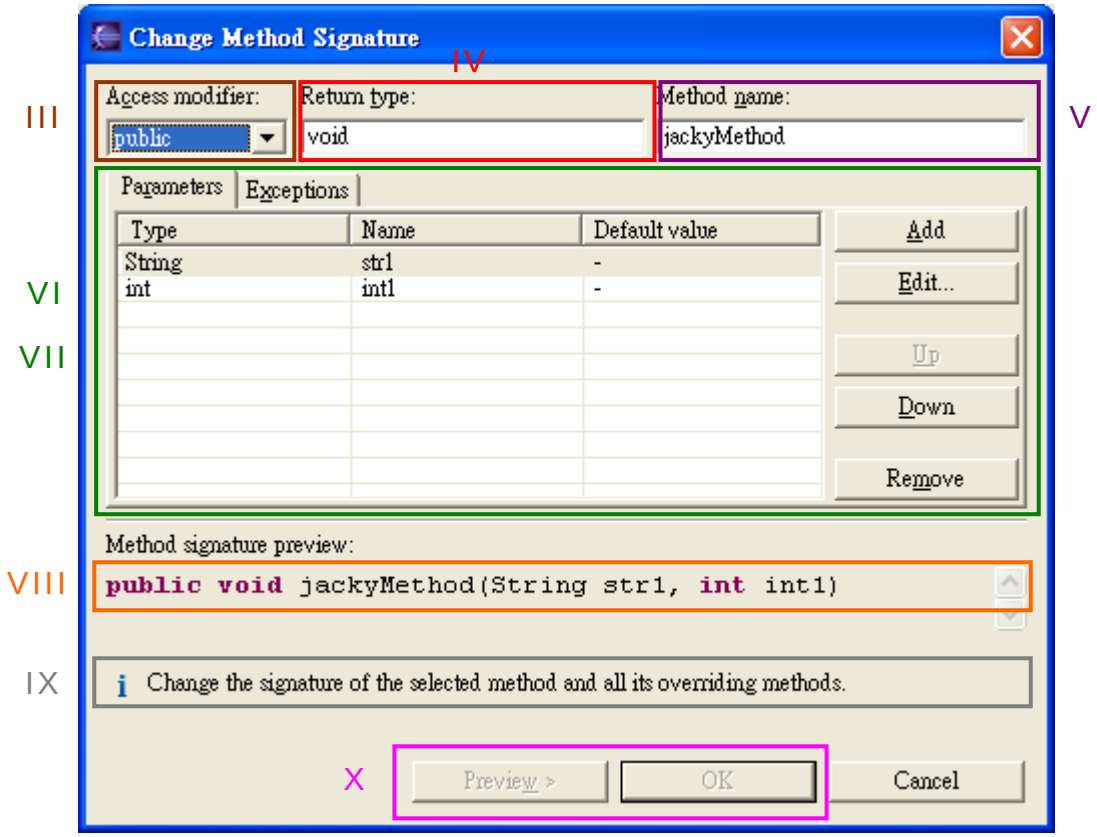


圖 6.41

III. 設定 Access modifier

IV. 設定 Return type

V. 設定新的 Method Name

VI. Parameters 頁籤

- 使用新增按鈕來新增參數；然後，可以在表格中編輯它的類型、名稱和預設值

圖 6.42

- 選取一個或多個參數，並使用上和下按鈕，以重新排序參數(可

查看參數清單下的簽章預覽）

VII. Exceptions 頁籤

可以新增或是刪除 Exception

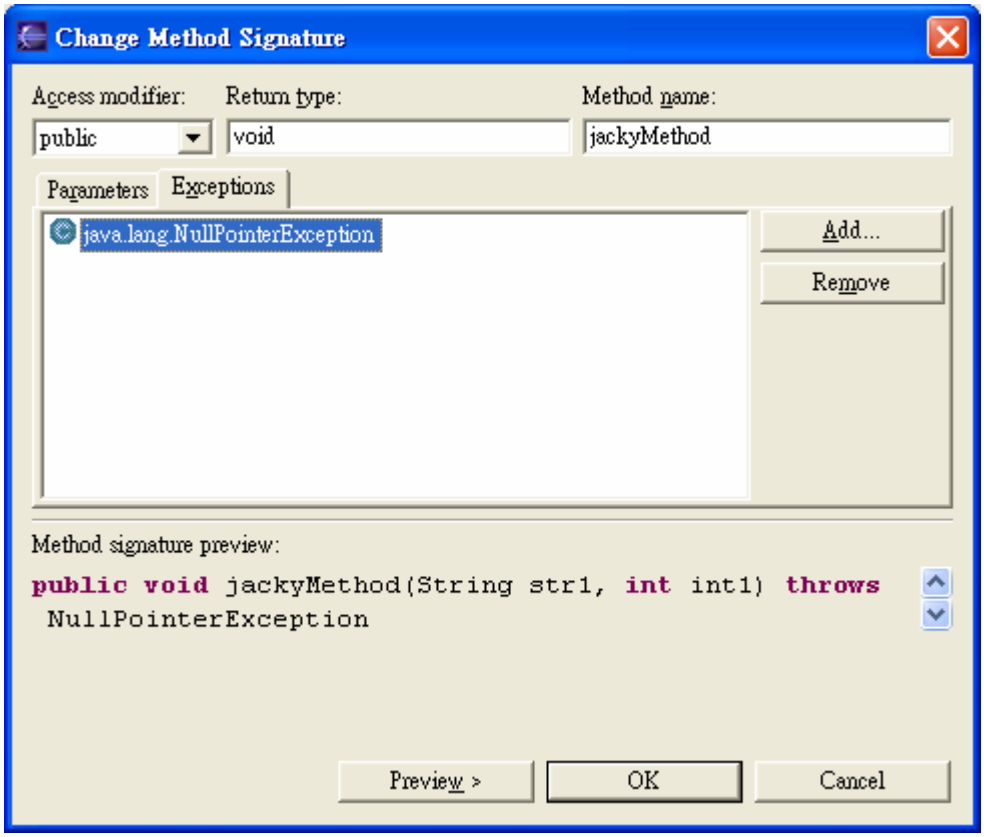


圖 6.43

VIII. 立即顯示設定後的狀態。

IX. 訊息。

X. 按預覽，以查看預覽，或按確定，執行重構作業，不查看預覽

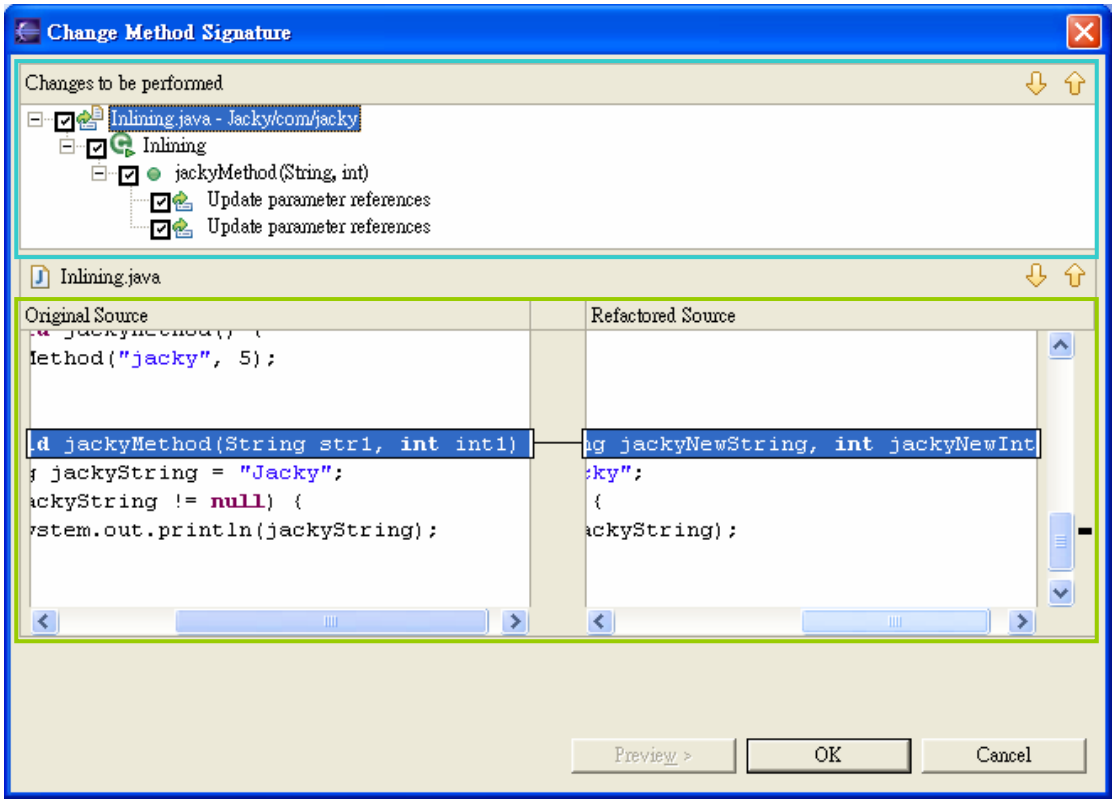


圖 6.44

XI. 預覽視窗會顯示重構要更動的部份

XII. 下半部的窗格顯示兩者的比較

附註：這項重構作業會變更所選方法和所有置換它之方法的簽章。此外，將更新所有參照以使用簽章。

6.5 移動 Java 元素(Moving Java Elements)

程式碼如下：

```
public class Moving {
    static int jackyInt;
    String jackyString = "Jacky";

    public static void main(String[] args) {
        System.out.println("This's Moving class.");
```

```
        System.out.println(jackyInt);
    }

    public void jackyMethod() {
        System.out.println(jackyString);
    }
}
```

6.5.1 欄位(Field)

如果要移動欄位，請執行下列動作：

I. 在 Java 編輯器中選取欄位

II. 「Refactor」→「Change Move...」

(或是在編輯器按右鍵，選取「Refactor」→「Move...」)

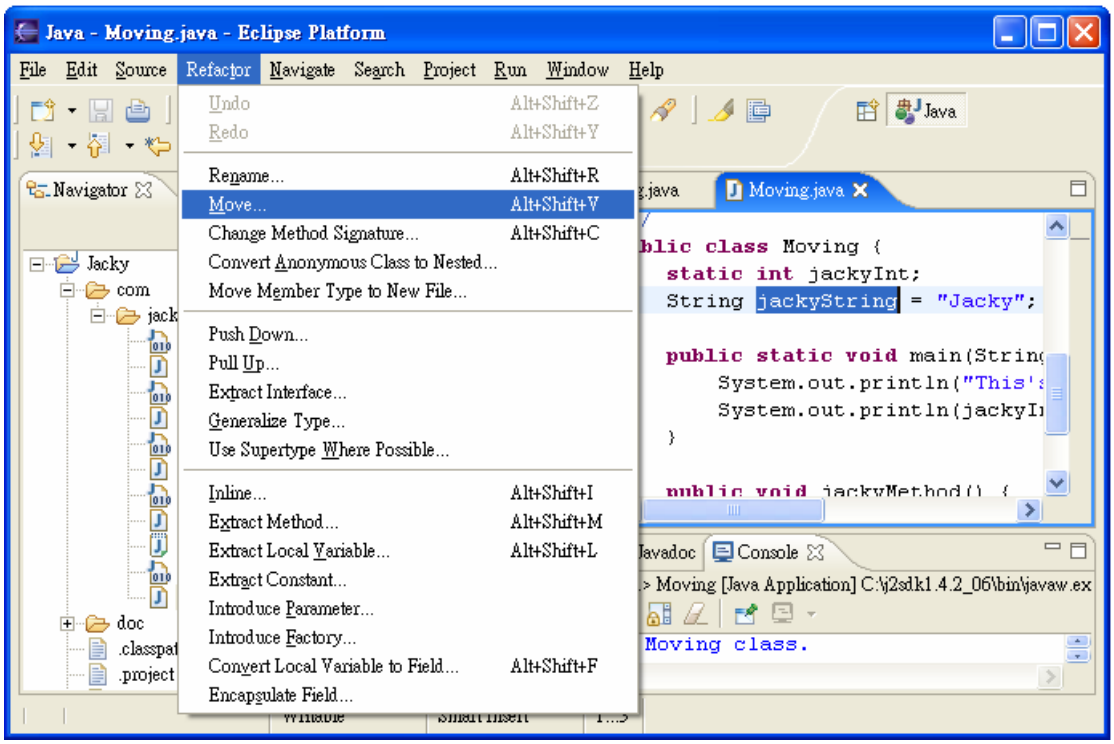


圖 6.45

出現 Textual Move 視窗

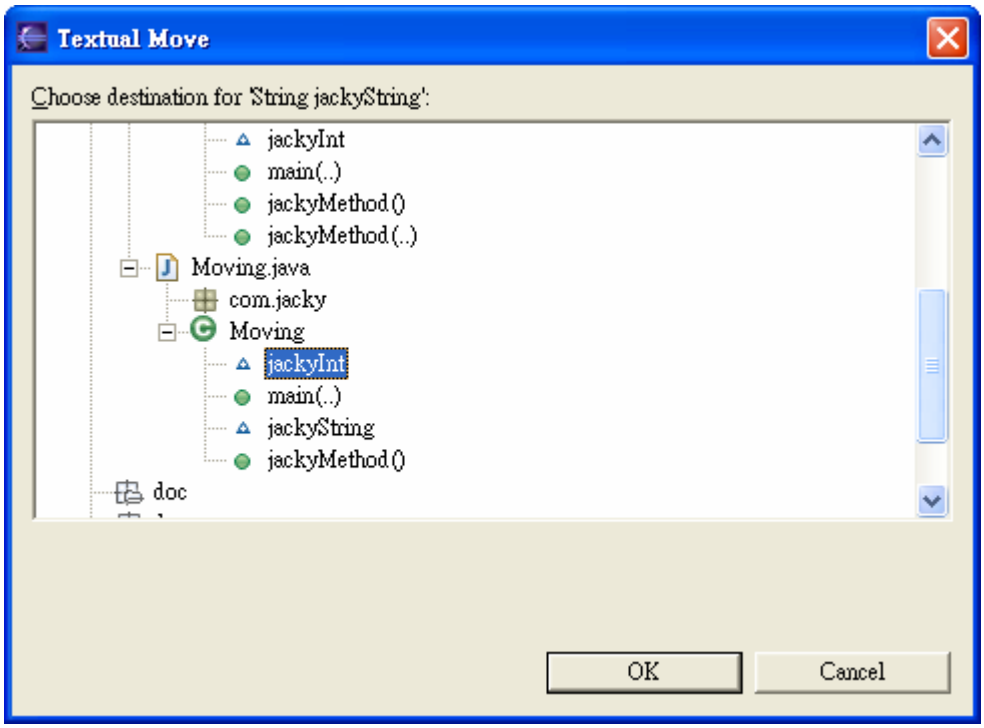


圖 6.46

III. 選擇要移動的目的地，按確定即可

6.5.2 Static Members

如果要移動 Static Members，請執行下列動作：

I. 在 Java 編輯器中選取 Static Members

II. 「Refactor」→「Change Move...」

(或是在編輯器按右鍵，選取「Refactor」→「Move...」)

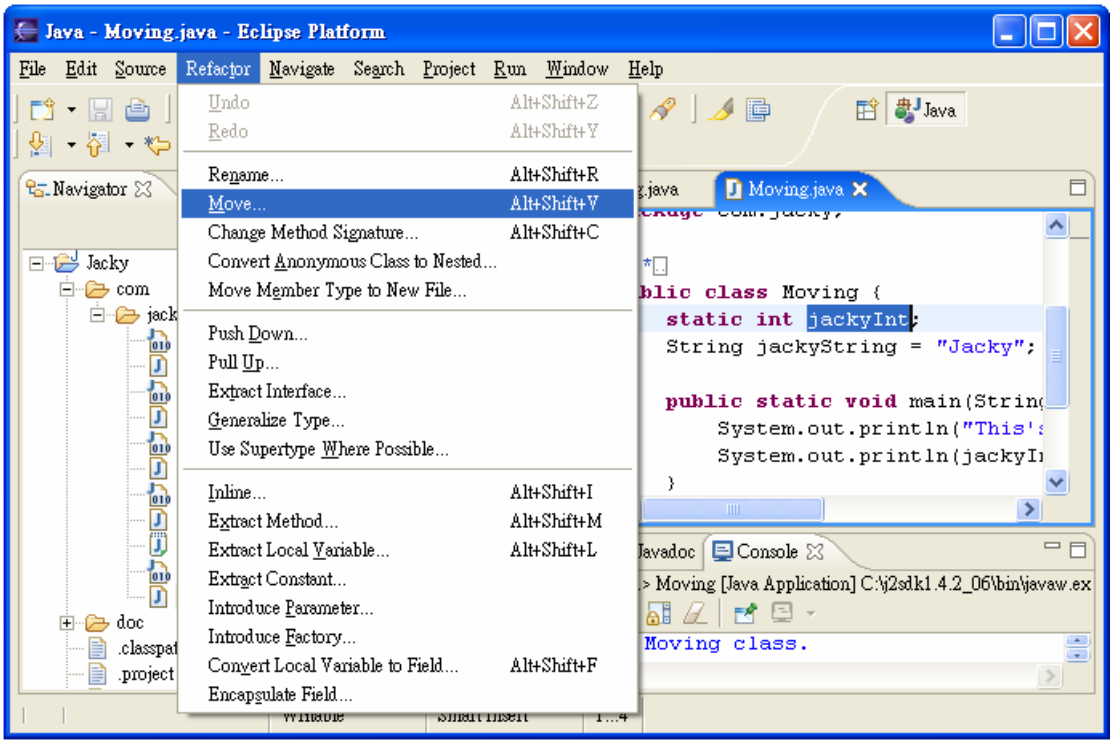


圖 6.47

出現 Move Static Members 視窗

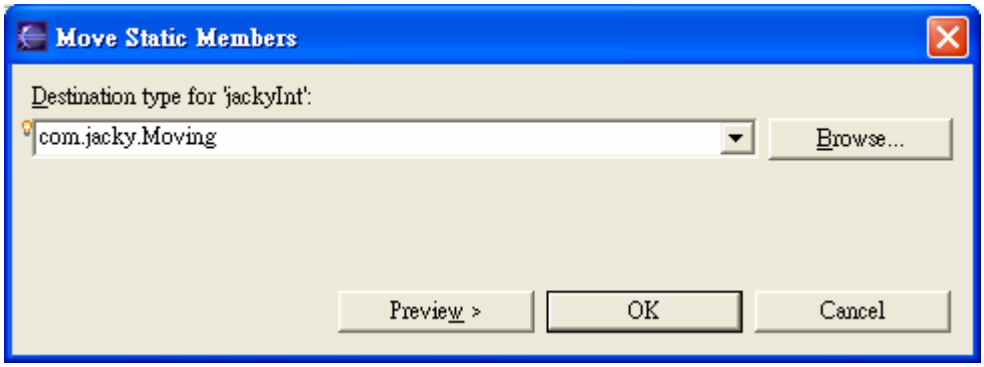


圖 6.48

III. 按瀏覽按鈕，開啟 Choose Type 視窗，選擇要移動的目的地

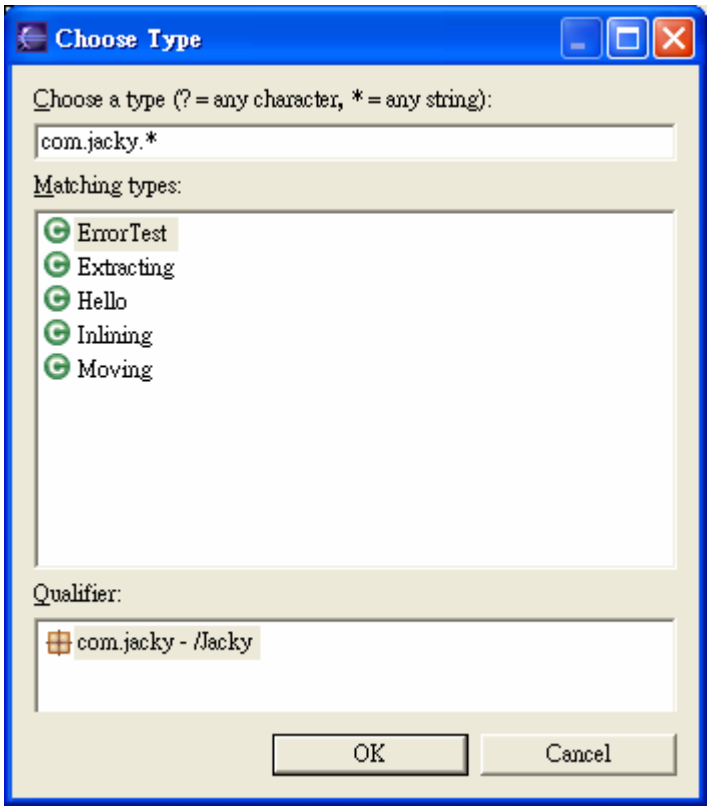


圖 6.49

IV. 按預覽，以查看預覽，或按確定，執行重構作業，不查看預覽

V. 預覽視窗會顯示重構要更動的部份，下半部的窗格顯示兩者的比較

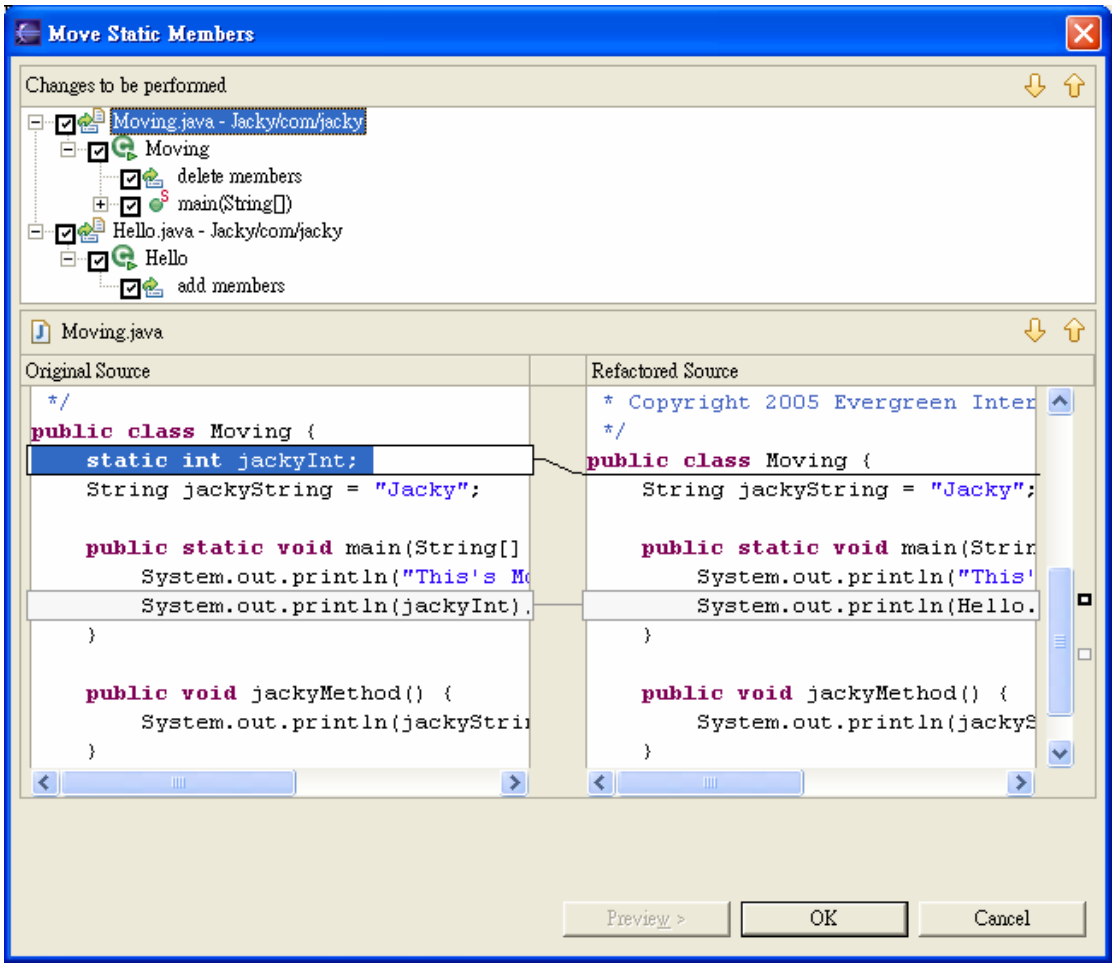


圖 6.50

6.6 自行封裝欄位(Self Encapsulating a Field)

如果要自行封裝欄位，請執行下列動作：

I. 在 Java 編輯器中選取欄位

II. 「Refactor」→「Encapsulate Field...」

(或是在編輯器按右鍵，選取「Refactor」→「Encapsulate Field...」)

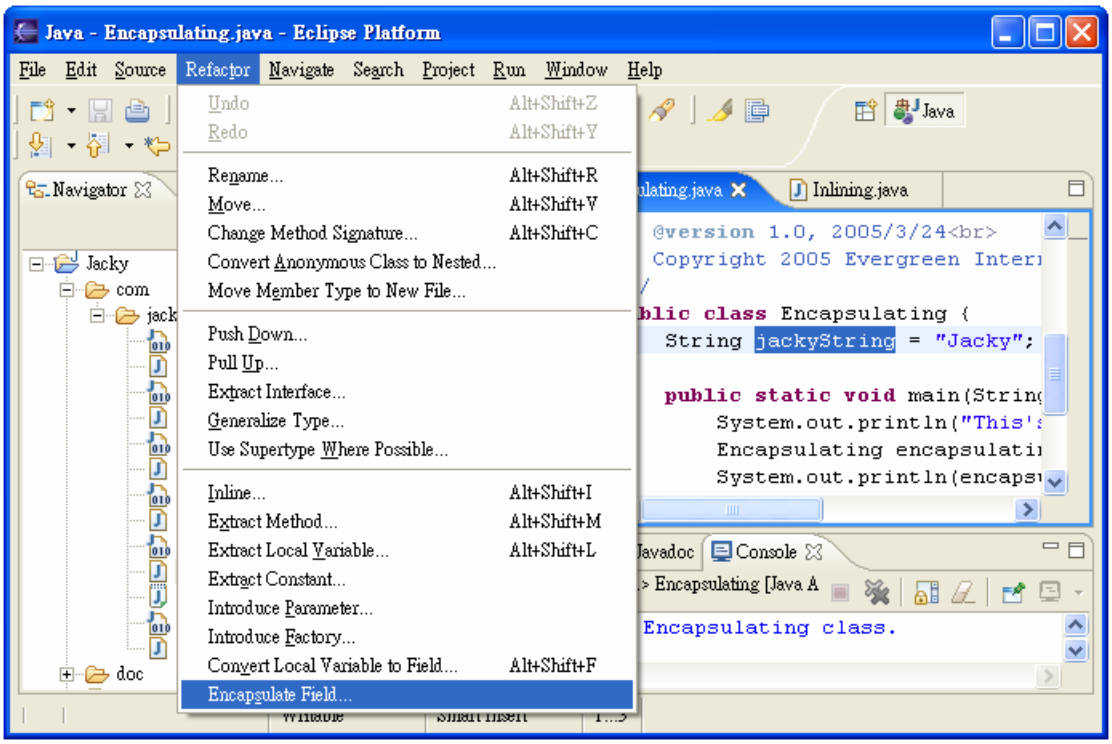


圖 6.51

出現 Encapsulate Field 視窗

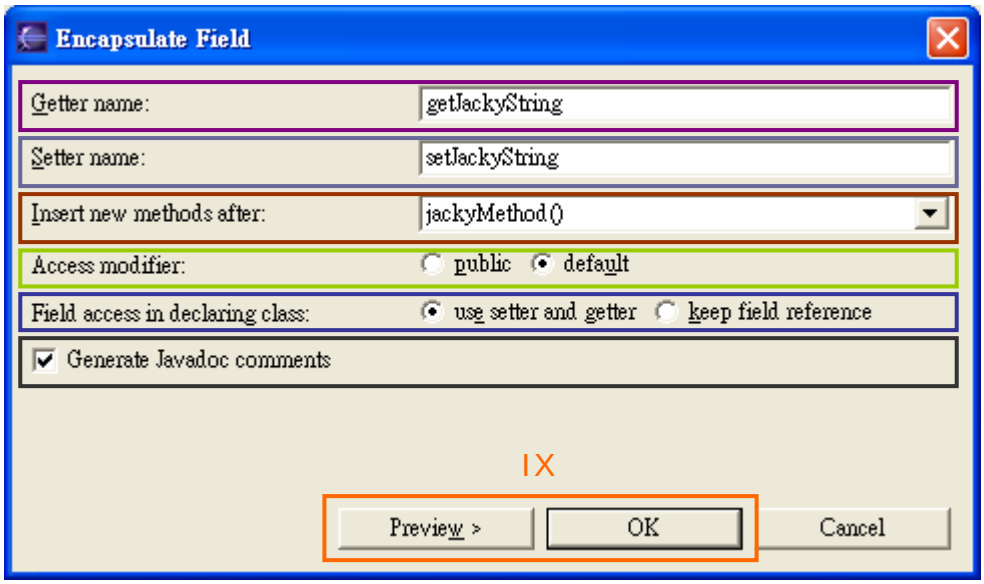


圖 6.52

III. 在 Getter 名稱欄位中輸入 Getter 的名稱。

IV. 在 Setter 名稱欄位中輸入 Setter 的名稱。

V. 使用在下列後面插入新方法組合框，指出 Getter 與(或)Setter 方法的位置。

VI. 從存取修飾元群組中選取一個圓鈕，以指定新方法的可見性。

VII. 在宣告欄位所在的類別中，讀取權和寫入權可為直接的，或者可以使用 Getter 和 Setter。

- 如果想要重構作業將所有這些存取權轉換成使用 Getter 和 Setter，請選取使用 Getter 和 Setter 圓鈕。
- 如果不想讓重構作業修改宣告欄位所在之類別中的現行欄位存取權，請選取保留欄位參照圓鈕。

VIII. 若要產生 Javadoc 註解，則選取勾選框

IX. 如果要先預覽再進行重構作業，請按預覽，或如果要直接進行重構作業而不預覽，請按確定。

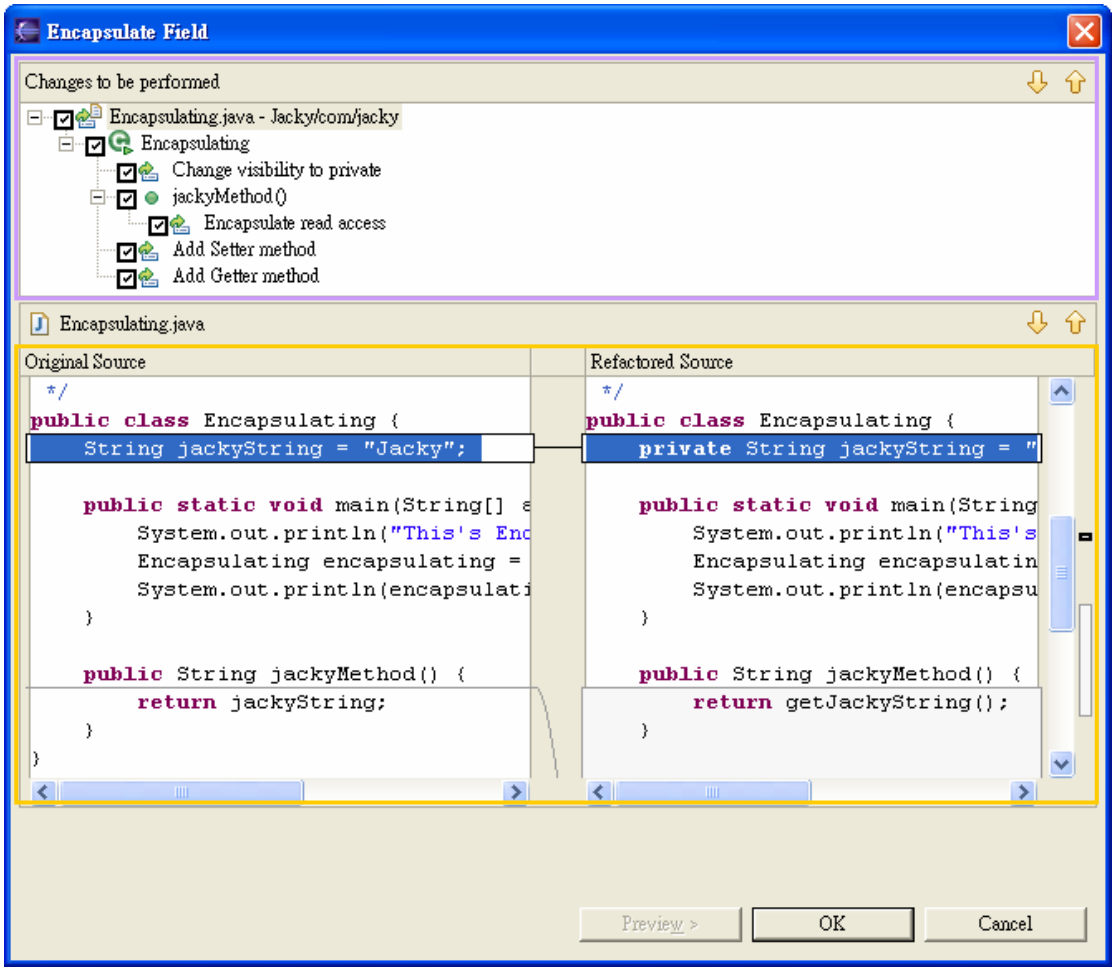


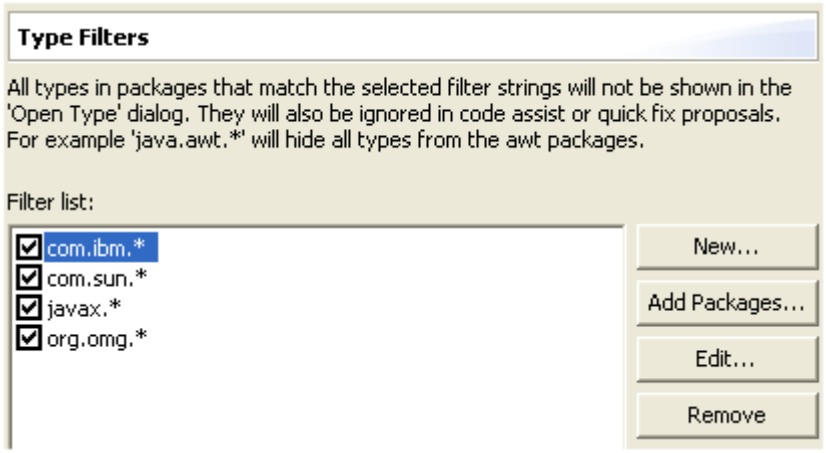
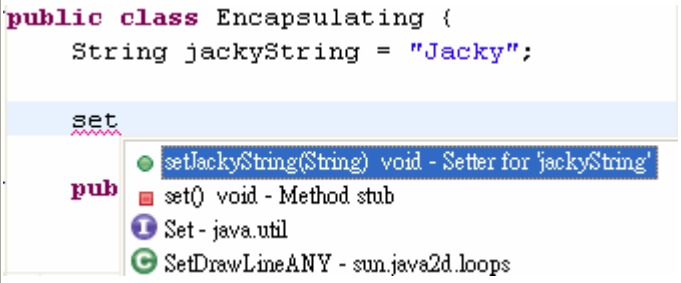
圖 6.53

X. 預覽視窗會顯示重構要更動的部份

XI. 下半部的窗格顯示兩者的比較

7.要訣和技巧(Tips and Tricks)

7.1 編輯程式檔(Editing Source)

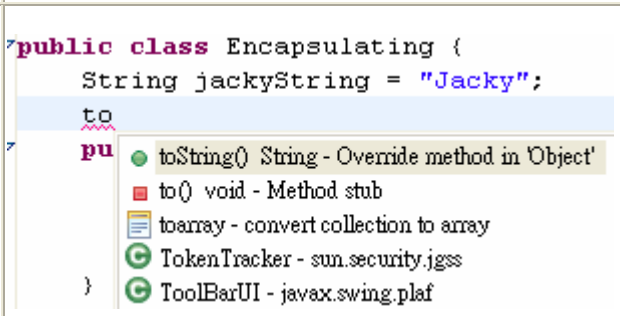
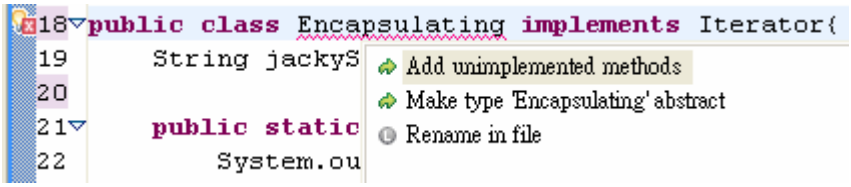
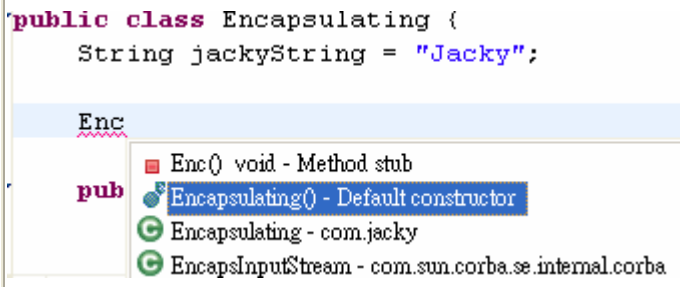
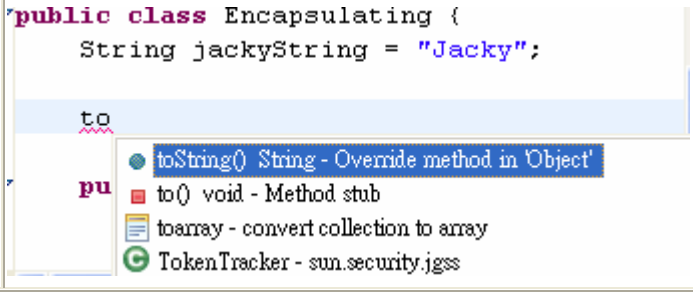
內容	說明
參數提示(Parameter Hints)	當游標位在方法引數時，可以看到參數提示的清單。在「Java編輯器」中，按下 Ctrl+Shift+空白鍵或呼叫「Edit」→「Parameter Hints」。
抑制程式碼輔助中的類型(Suppress types in code assist)	如果不要讓某些類型出現在內容輔助中，請使用在「Java」→「Type Filters」喜好設定頁面配置的類型過濾器功能。只要符合其中一項過濾器型樣的類型，就不會出現在「開啟類型」對話框中，且不供程式碼輔助、快速修正和組織匯入使用。這些過濾器型樣不會影響「Package Explorer」和「Type Hierarchy」視圖。 
使用內容輔助來建立 Getter 和 Setter(Use content assist to create Getter and Setters)	建立 Getter 和 Setter 的另一個方法，就是使用內容輔助。可以把游標停在成員之間的類型主體，然後按 Alt+/，取得建立 Getter 或 Setter 方法 Stub 的提議。 

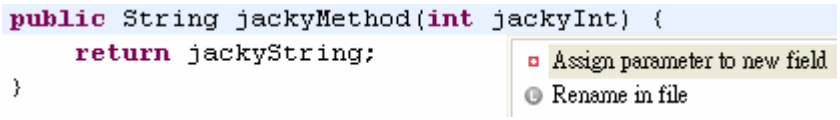
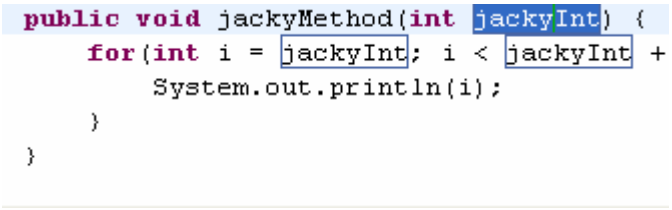
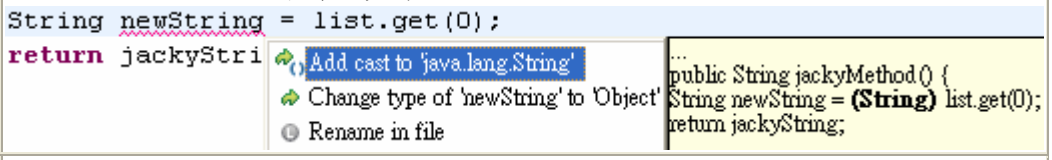
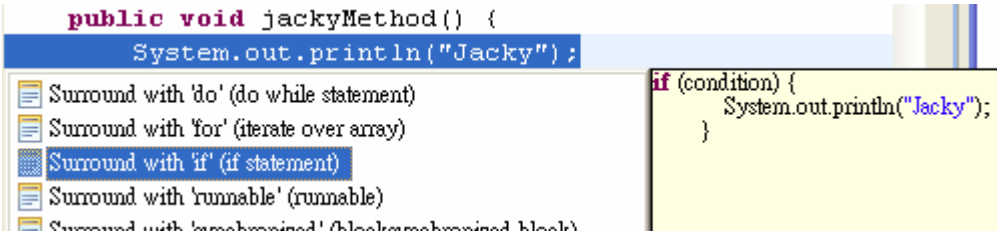
內容	說明
連同欄位一起刪除 Getters 和 Setter(Delete Getters and Setters together with a field)	當從視圖刪除欄位時，Eclipse 可能會提議連同其 Getter 和 Setter 方法一起刪除。如果在欄位使用名稱字首或字尾，請務在「Code Style」喜好設定頁面指定它(「Window」→「Preferences」→「Java」→「Code Style」)。

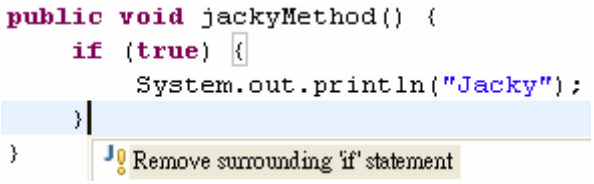
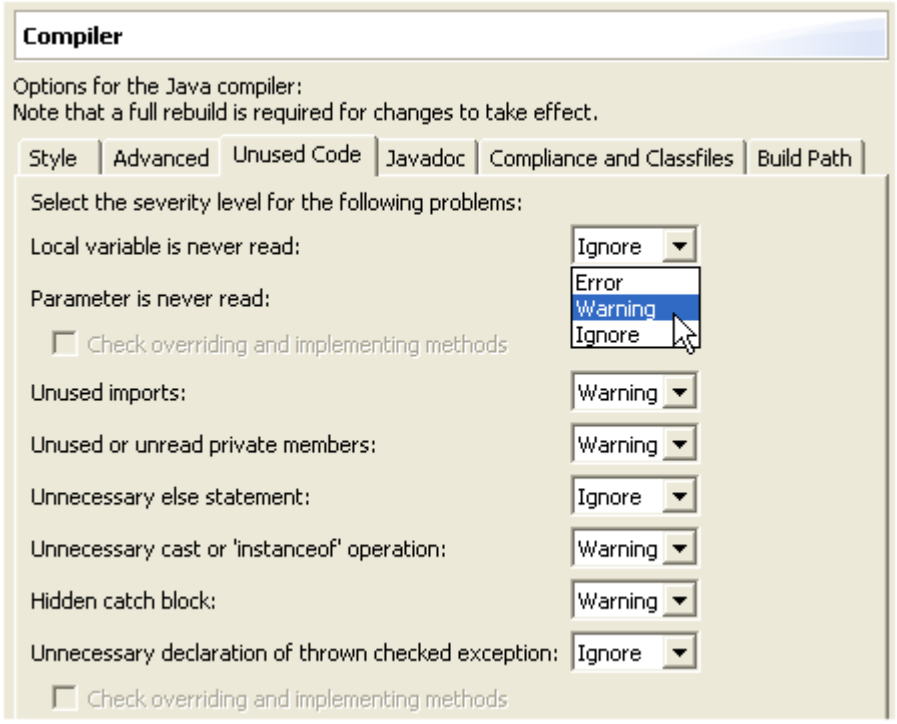
建立委派方法(Create Delegate Methods)	如果要對欄位建立委派方法， 請選取欄位的宣告，並呼叫「Source」→「Generate Delegate Methods」。 這將新增所選方法至含有至委派的方法之轉遞呼叫的類型。
	`public class Encapsulating {`
	`  String jackyString = "Jacky";`
	`  public String toLowerCase() {`
	`  return jackyString.toLowerCase();`
	`  }`

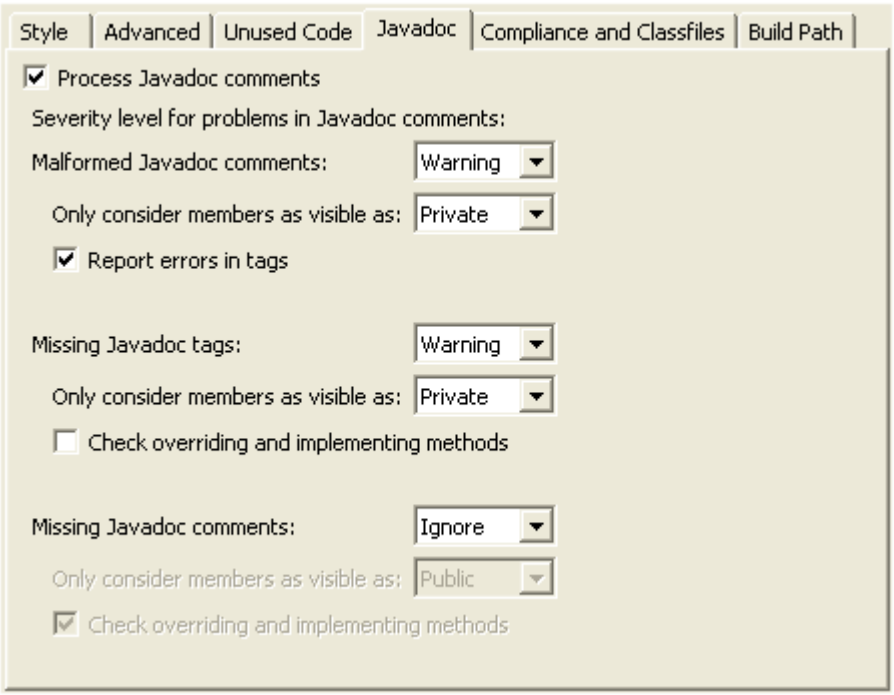
使用「範本」來建立方法(Use Templates to create a method)	可以定義新的範本 (「Window」→「Preferences」→「Java」→「Editor」→「Templates」) 來含有方法 Stub。範本會與內容輔助(Alt+/)提議一起顯示。
	也有現有的範本，如'private_method'、'public_method'、'protected_method'，以及其他等等。
	使用 Tab 鍵，在要輸入的值 (傳回類型、名稱和引數) 之間導覽。
	`public class Encapsulating {`
	`  String jackyString = "Jacky";`
	`  private`
	`  public`
	`  Sys`
	`  Enc`
	`  Sys`
	`  }`
	private
	private_method - private method
	private_static_method - private static method
	PRIVATE_MEMBER - org.omg.CORBA
	PrivateCredentialPermission - javax.security.auth

使用「內容輔助」來置換方法(Use Quick Fix to create a new method)	在類型主體中應該加入方法的位置，呼叫內容輔助(Alt+/)。內容輔助將提供所有可以被置換的方法。將建立所選方法的方法主體。

內容	說明
	
使用「快速修正」來新增未實作的方法(Use Quick Fix to change a method signature)	如果要實作新的介面，請先新增'implements'宣告至類型。即使沒有儲存或建置，Java 編輯器也會強調以訊號顯示該類型，表示該方法遺漏，並且顯示快速修正燈泡。可以按一下燈炮，或按下 Ctrl+1 (「Edit」→「Quick Fix」)，選擇要新增未實作的方法，或者讓類別成為 abstract。 
利用內容輔助來建立建構子 Stub(Use Content Assist to create a constructor stub)	在要新增建構子的位置，輸入該建構子名稱的第一個字母之後，再使用程式碼輔助。 
使用「內容輔助」來置換方法(Use Content Assist to override a method)	在類型主體中應該加入方法的位置，呼叫內容輔助 (Alt + /)。內容輔助將提供所有可以被置換的方法。將建立所選方法的方法主體。 

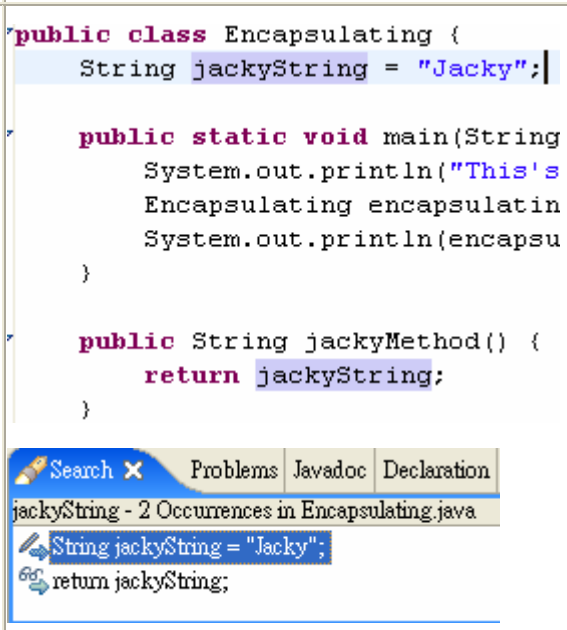
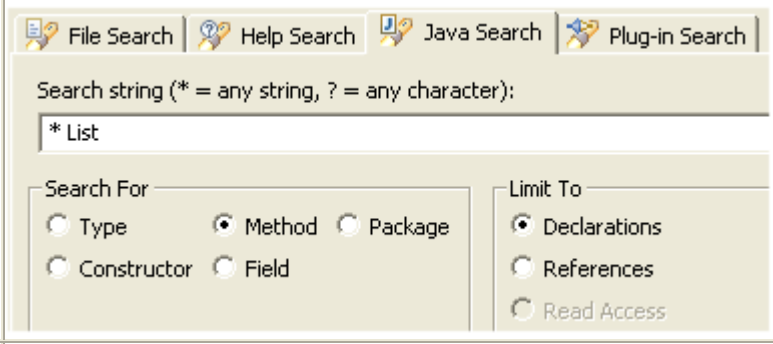
內容	說明
從參數建立新欄位(Create new fields from parameters)	需要建立新欄位來存放傳入建構子中的引數嗎？請在參數使用快速輔助(Ctrl+1)，來建立指派作業和欄位宣告，並且讓 Eclipse 根據「程式碼樣式」喜好設定來提議名稱。 
在檔案中重新命名(Rename in File)	如果要快速進行重新命名作業，可以使用「在檔案中重新命名」快速輔助，而不必在其他檔案完整分析其相依關係。在 Java 編輯器中，將游標定位在變數、方法或類型的識別碼，然後按下 Ctrl+1 (「Edit」→「Quick Fix」) 編輯器會切換至鏈結的編輯模式（如同範本），而且變更識別碼將同時變更該變數、方法或類型的所有其他參照。 
減少強制轉型表示式的作業時間(Less work with cast expressions)	別花太多時間在輸入強制轉型。先略過它們，等完成陳述式之後，再利用快速輔助加入它們。 比方說，以指派作業為例： 
包覆字行(Surround Lines)	如果要以 if/while/for 陳述式或區塊包覆陳述式，請選取要包覆的字行，然後按下 Ctrl+1 (「Edit」→「Quick Fix」)。這會列出所有含有${line_selection}變數的範本。 
移除包覆的陳述式(Remove Surrounding	如果要移除包覆的陳述式或區塊， 請將游標定位在右方括弧，然後按下 Ctrl+1 (「Edit」→「Quick Fix」)。

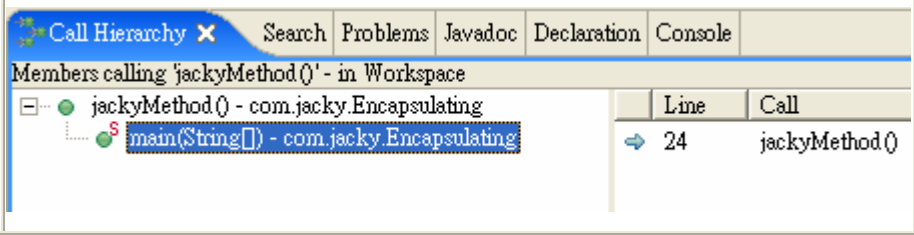
內容	說明
Statement)	
尋找對稱的括弧(Find the Matching Bracket)	如果要尋找對稱的括弧，請選取左或右括弧，然後按下 Ctrl+Shift+P(「Navigate」→「Go To」→「Matching Bracket」)。也可以在左方括弧之前，或是右方括弧之後按兩下滑鼠，選取這兩個方括弧之間的文字。
排序成員(Sort Members)	可以根據「Window」→「Preferences」→「Java」→「Appearance」→「Members Sort Order」所定義的種類順序，針對 Java 編譯單元來排序成員。 將在「Source」→「Sort Members」下找到動作
尋找未用的程式碼(Find Unused Code)	Java 編譯器會偵測無法呼叫到的程式碼、未使用的變數、參數、匯入項目和未使用的私密類型、方法和欄位。這個設定位在「Window」→「Preferences」→「Java 」→「Compiler」。  當輸入時也會偵測這些設定，並提供一個快速修正以移除不必要的程式碼。
處理 Javadoc 註解(Javadoc	Java 編譯器可以處理 Javadoc 註解。搜尋報告會參照 doc 註解，而重構會更新這些參照。這項特性是從「Window」→

內容	說明
Comment Handling)	「Preferences」→「Java 」→「Compiler」Javadoc 標籤加以控制(或是利用「Project」→「Properties」→「Java Compiler」→「Javadoc」，針對個別專案加以設定)。  當它開啟時，形態異常的 Javadoc 註解會在 Java 編輯器中標示出來，也可以利用「Edit」→「Quick Fix」(Ctrl+1)加以修正

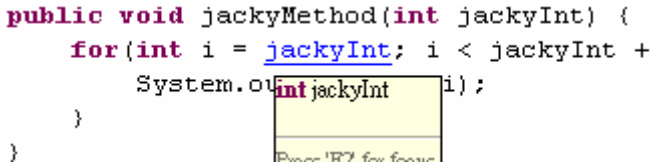
7.2 搜尋(Searching)

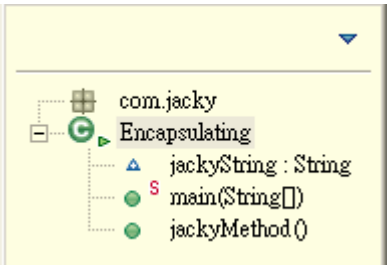
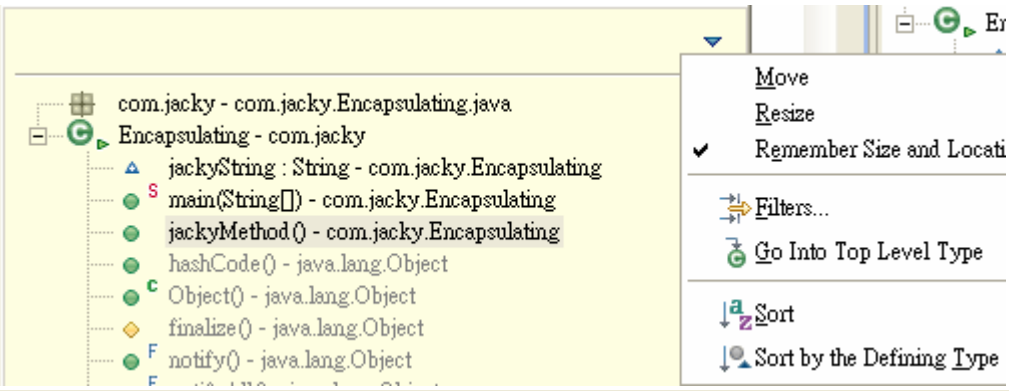
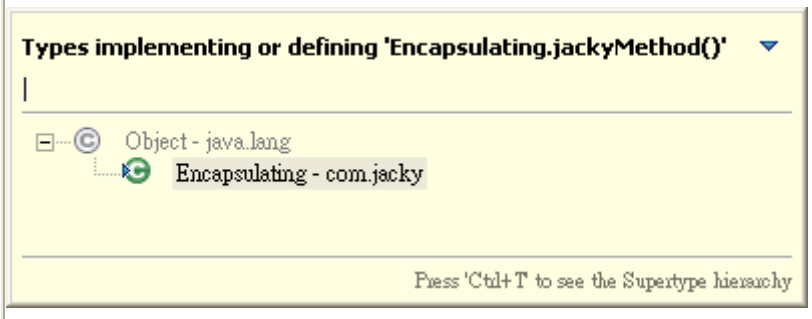
內容	說明
尋找變數及其讀寫權(Locate variables and their read/write access)	可以選取一個 ID(變數、方法、類型參照或宣告)，呼叫「Search」→「Occurrences in File」，來尋找變數，看看它們的讀寫狀態。此舉會標示同一檔案中該識別碼的所有參照。結果也會顯示在搜尋視圖中，以及有圖示會顯示變數的讀取權或寫入權。

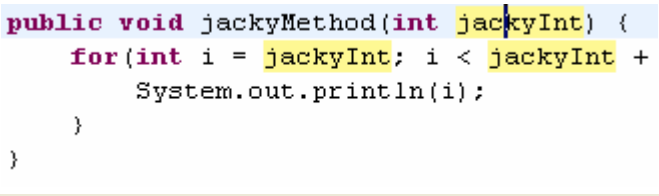
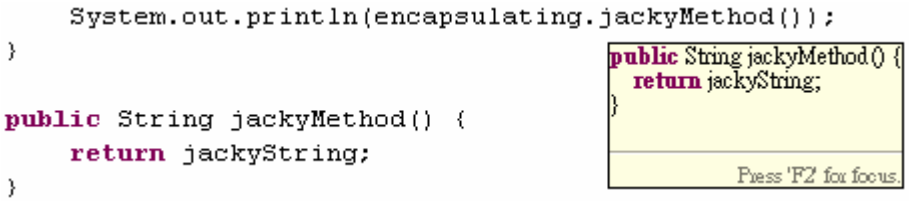
內容	說明
	 或者，也可以使用新的標示搜尋結果特性，動態強調搜尋結果。可以使用一般搜尋特性來搜尋數個檔案（「Search」→「References」）。
搜尋具有特定傳回類型的方法 (Search for methods with a specific return type)	如果要搜尋具有特定傳回類型的方法，請使用"*<return type>"，如下所示： ■ 開啟搜尋對話框，再按一下 Java Search 標籤。 ■ 在 Search string 中輸入以空格隔開的'*'和傳回類型。 ■ 選取 Case sensitive 勾選框。 ■ 選取 Method 和 Declarations，然後按一下 Search。 
從 Java 搜尋移除 Javadoc 結果 (Remove Javadoc results from Java search)	依預設，Java 搜尋會在 Java 程式碼和 Javadoc 中尋找參照。如果不要在 Javadoc 中尋找參照，可以取消勾選「Window」→「Preferences」→「Java」→「Compiler」→「Javadoc」的 Process Javadoc comments，停用這項行為。

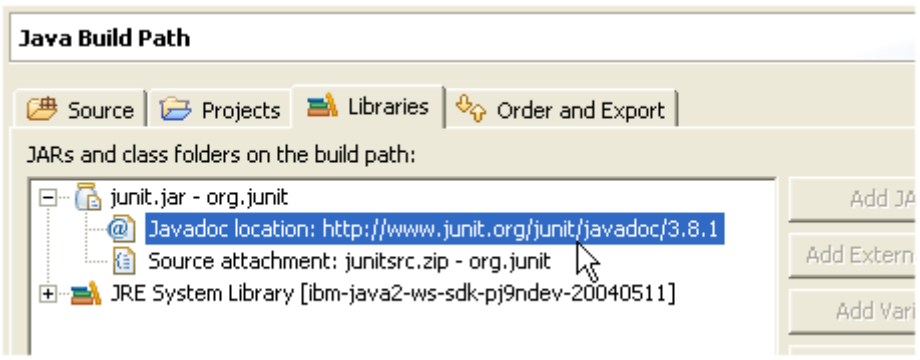
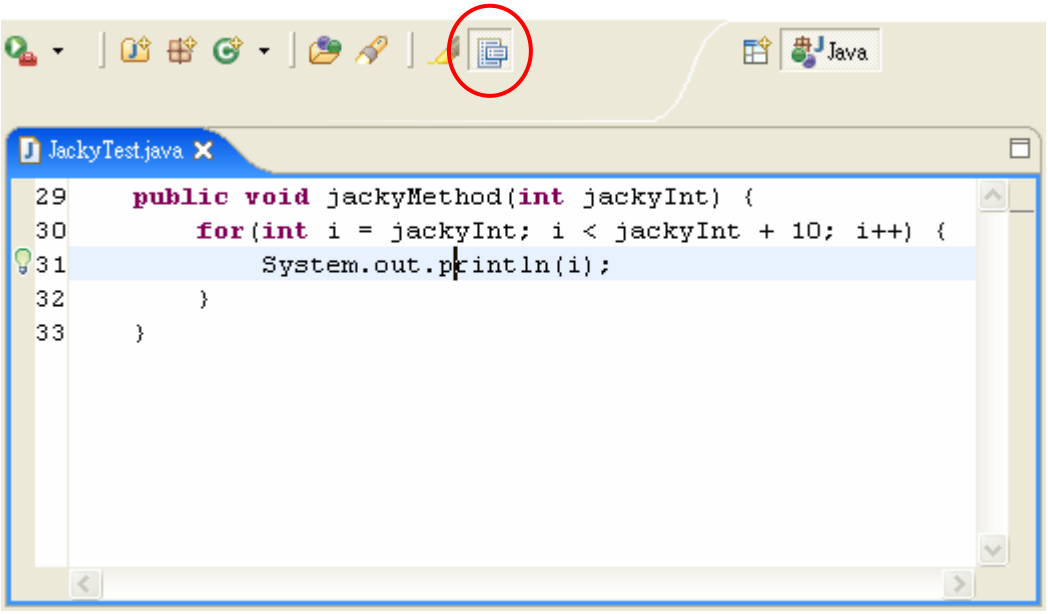
內容	說明
以呼叫階層來追蹤方法呼叫鏈(Trace method call chains with the Call Hierarchy)	有沒有發覺自己曾經不斷的搜尋方法參照？現在有一種新的呼叫階層，可以跟著又長又複雜的呼叫鏈，又不會遺漏原始的環境定義：只要選取一個方法，然後呼叫「Navigate」→「Open Call Hierarchy」(Ctrl+Alt+H)即可。 

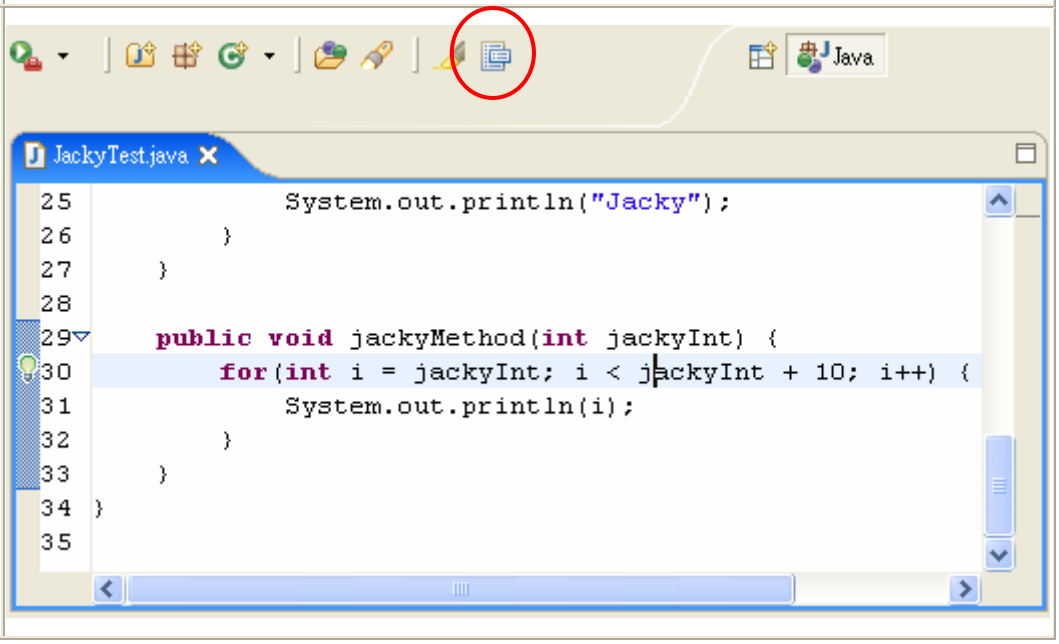
7.3 程式碼導覽和讀取(Code navigation and reading)

內容	說明
在 Java 編輯器中依據選項開啟(Open on a selection in the Java editor)	在 Java 編輯器中，有兩種方法，可讓從元素的參照中開啟元素。 ■ 選取程式碼中的參照，並按下 F3 (「Navigate」→「Open Declaration」) ■ 按住 Ctrl 鍵，將滑鼠指標移到參照之上 
原位概要(In-place outlines)	在 Java 編輯器中按下 Ctrl+F3，以在現行游標位置蹦現元素的原位概要。或者按 Ctrl+O (「Navigate」→「Quick Outline」)，以蹦現現行程式檔的原位概要。

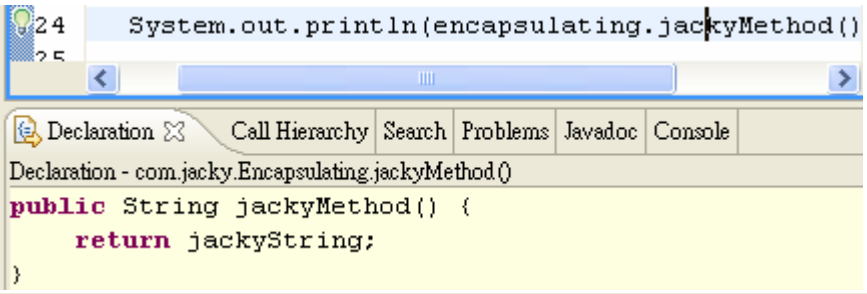
內容	說明
	
原位概要會顯示繼承的成員(In-place outlines show inherited members)	再按一次 Ctrl+O 或 Ctrl+F3，把繼承的成員加到一個開啟的原位概要中。繼承的成員附有一個灰色標籤。可以利用右上角的功能表來過濾和排列概要。 
原位階層(In-place hierarchy)	利用「Quick Hierarchy」，找出哪些是虛擬呼叫可能的接收端。然後把游標放在方法呼叫裡面，按 Ctrl+T(「Navigate」→「Quick Outline」)。這個視圖會以完整的圖示，顯示所有實作方法的類型。  再按一次 Ctrl+T，切換至「Supertype hierarchy」階層。
標示搜尋結果(Mark Occurrences)	當在編輯器工作時，請開啟工具列上的標示搜尋結果（）或者按(Alt+Shift+O)。會在檔案中，看到被參照的變數、方法或類型。

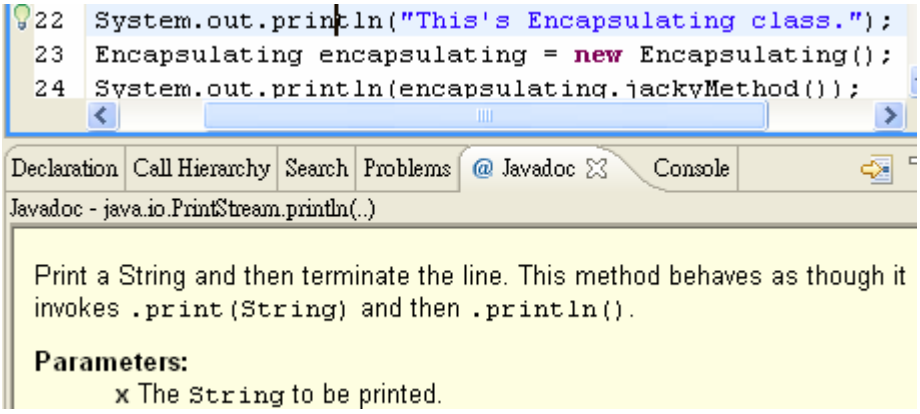
內容	說明
	
移至下一個 / 上一個方法(Go to next / previous method)	如果要快速導覽至下一個或上一個方法或欄位，請使用 Ctrl+Shift+上移鍵 (「Navigate」→「Go To」→「Previous Member」)，或 Ctrl+Shift+下移鍵 (「Navigate」→「Go To」→「Next Member」)
Java 編輯器中的浮動說明 (Hovers in the Java editor)	可以使用修正鍵 (Shift、Ctrl、Alt)，在 Java 編輯器看到不同的浮動說明。 當將滑鼠在 Java 編輯器中的識別碼上移動時，依預設，將顯示一個浮動說明，其中含有從這個元素的對應程式檔擷取的 Javadoc。按住 Ctrl 鍵將顯示程式碼。  可以在「Window」→「Preferences」→「Java」→「Editor」→「Hovers」中，變更這個行為，以及定義其他修正鍵的浮動說明。
開啟和配置外部 Javadoc 文件 (Open and configure external Javadoc documentation)	如果想要利用 Shift+F2 (「Navigate」→「Open External Javadoc」)，來開啟類型、方法或欄位的 Javadoc 文件，必須先指定元素母項程式庫 (JAR、類別資料夾) 或專案 (來源資料夾) 的文件位置。 對於程式庫，請開啟建置路徑頁面 (「Project」→「Properties」→「Java Build Path」)、移至程式庫，展開可以在其中編輯 'Javadoc 位置' 節點的程式庫節點。文件可以放在本端檔案系統上的一個資料夾中，也可以放在保存檔或 Web 伺服器中。

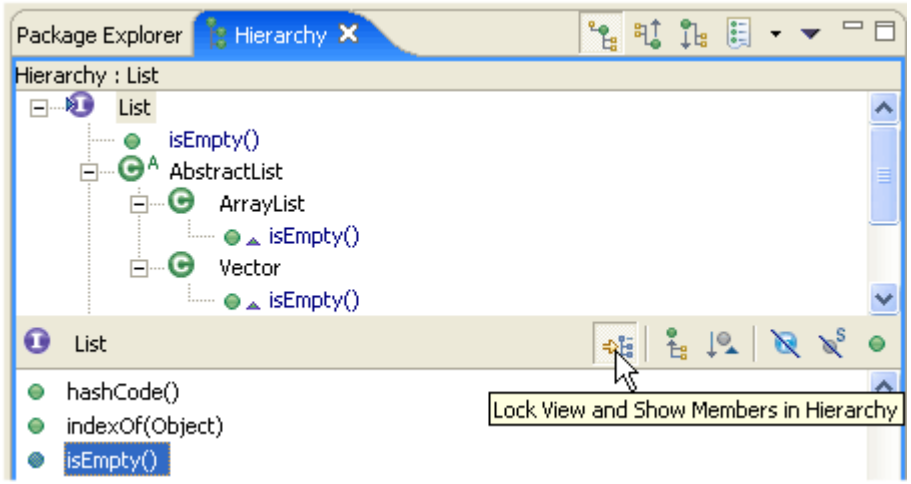
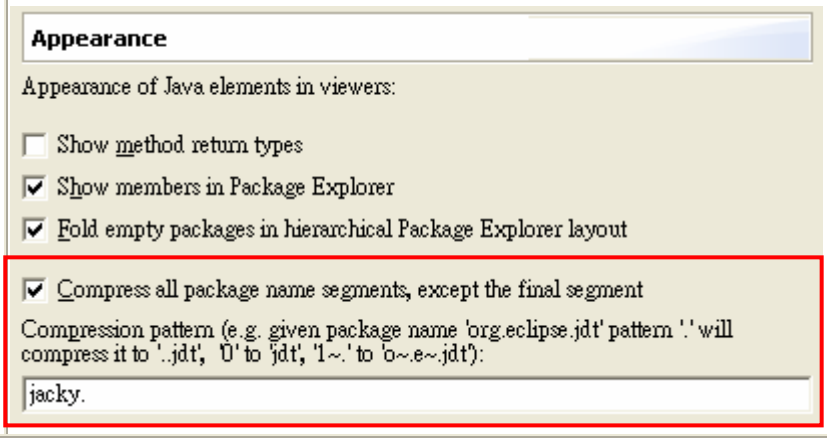
內容	說明
	 對於來源資料夾中的類型、方法或欄位，請移至（「Project」→「Properties」→「Javadoc Location」）。
僅顯示所選元素的程式碼 Show Source of Selected Element Only）	工具列包括僅顯示所選元素的程式檔按鈕（  ），這個按鈕讓只有所選概要元素的程式碼會顯示在 Java 編輯器中。在底下範例中，僅顯示 jackyMethod()方法。  再按一下僅顯示所選元素的程式檔按鈕，以重新查看整個 Java 檔。

內容	說明
	

7.4 Java 視圖(Java views)

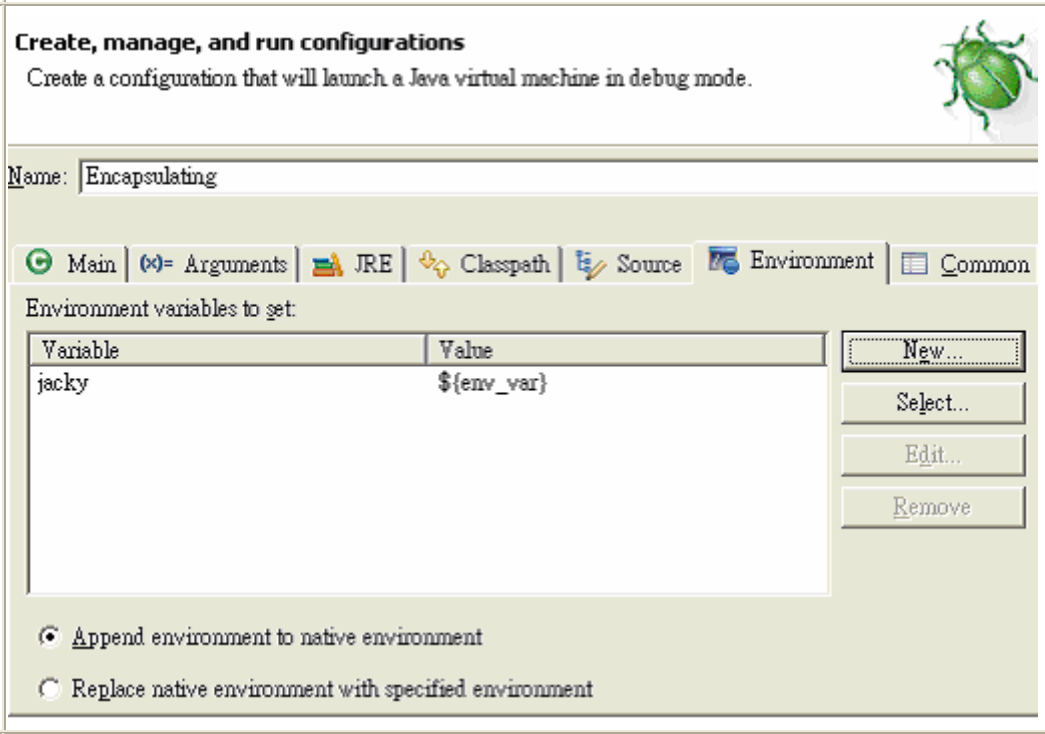
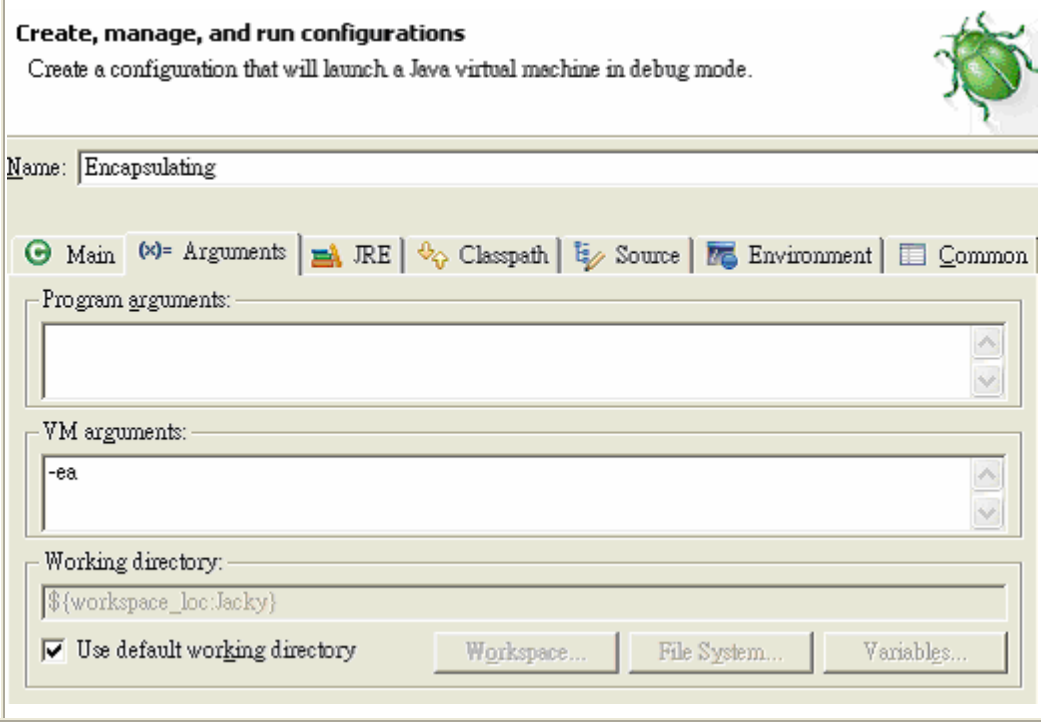
內容	說明
宣告視圖(Declaration view)	新的「宣告」視圖（「Window」→「Show View」→「Other...」→「Java」→「Declaration」）會顯示在 Java 編輯器或 Java 視圖中所選元素的程式碼。 
Javadoc 視圖(Javadoc view)	Javadoc 視圖（「Window」→「Show View」→「Other...」→「Java」→「Javadoc」）會顯示在 Java 編輯器或 Java 視圖所選元素的 Javadoc。Javadoc 視圖會使用 SWT 瀏覽器小組件，在支援它的平台上顯示 HTML。

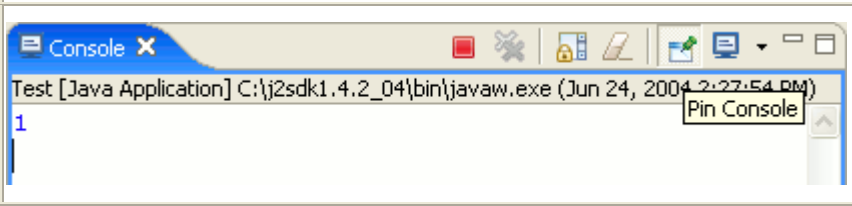
內容	說明
	
類型階層中的秘訣(Tricks in the type hierarchy)	■ 在元素或所選名稱上按 F4 (「Navigate」→「Open Type Hierarchy」)，將類型階層的焦點放在新類型。 ■ 不單可以開啟「Hierarchy」視圖以顯示類型，也可以顯示套件、來源資料夾、JAR 保存檔與 Java 專案。 ■ 可以將元素拖放到「Hierarchy」視圖中，以便將它的焦點放在該元素上。 ■ 可以從視圖的工具列，變更「Hierarchy」視圖的擺放方式(從預設的垂直方向到水平方向)。
找出在階層中哪一個位置實作方法(Find out where a method is implemented in the hierarchy)	如果要瞭解階層中哪些類型會置換方法，請使用「顯示階層中的成員」功能。 ■ 選取要查看的方法，然後按下 F4 (「Navigate」→「Open Type Hierarchy」)。這將根據方法的宣告類型開啟類型階層視圖。 ■ 如果「階層」視圖已經選取該方法，請按「Lock View and Show Members in Hierarchy」工具列按鈕。 ■ 階層視圖現在僅顯示實作或定義'已鎖定'方法的類型。舉例來說，可以看到'isEmpty()'在'List'中定義，且在'ArrayList'和'Vector'，但不在'AbstractList'中實作。

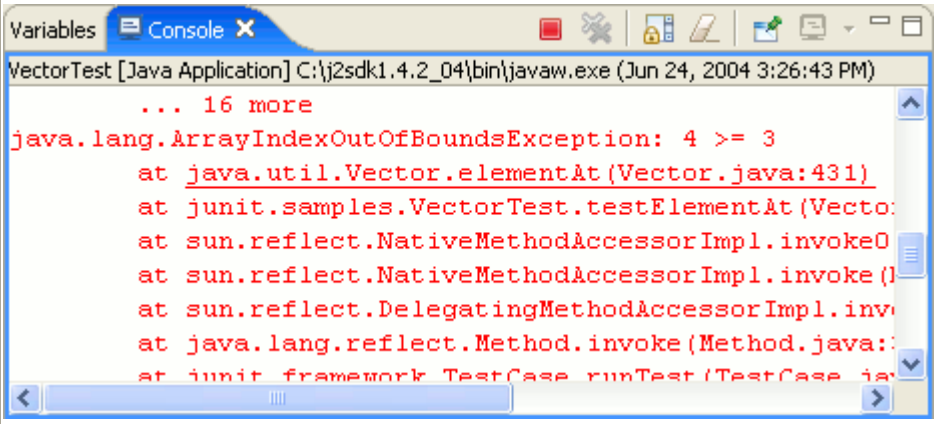
內容	說明
	
壓縮套件名稱(Compress package names)	如果套件名稱很長，可以配置一個在檢視器顯示的壓縮名稱。壓縮型樣的配置是在「Window」→「Preferences」→「Java」→「Appearance」中完成。 

7.5 除錯(Debugging)

內容	說明
環境變數(Environment Variables)	可以透過環境標籤，指定啟動 Java 應用程式所用的環境。

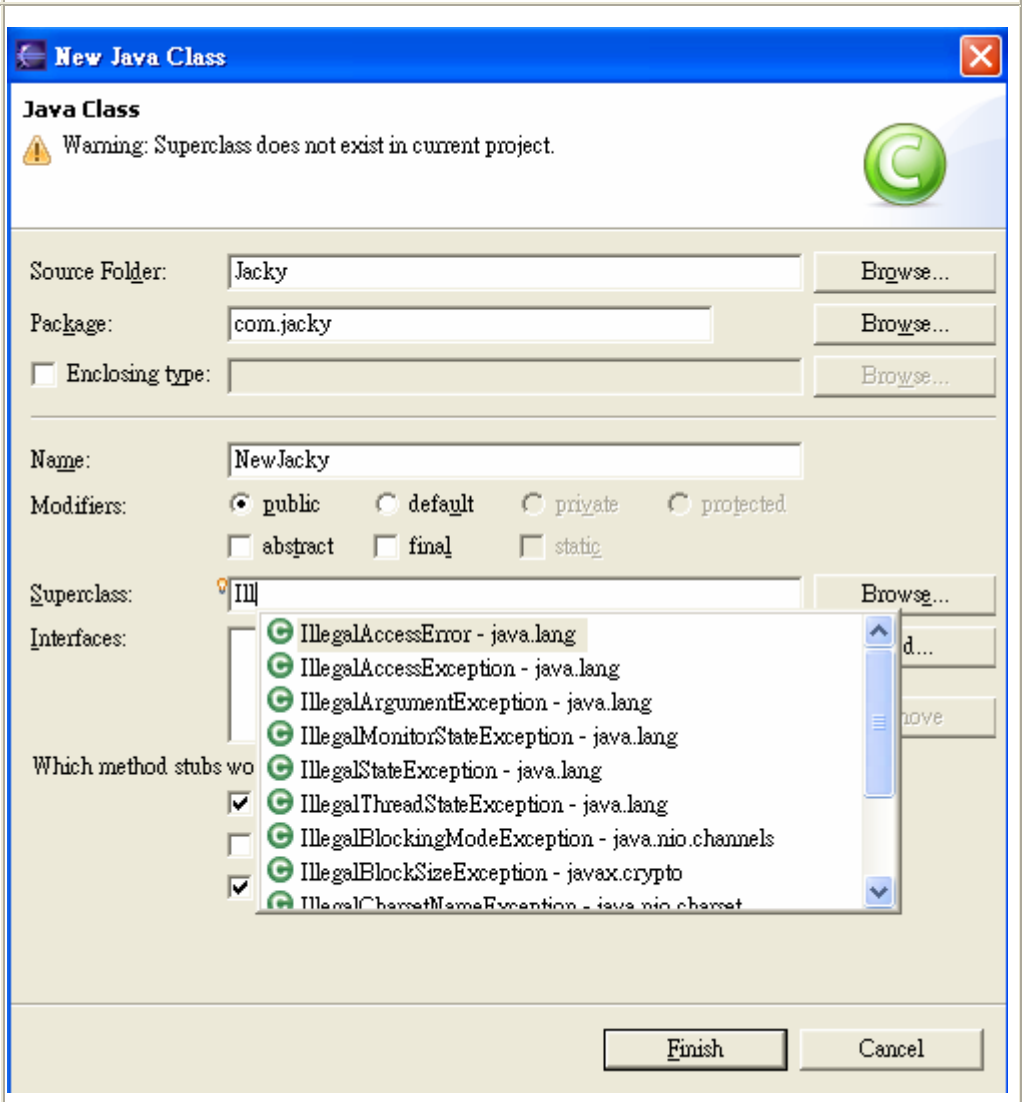
內容	說明
	
預設的 VM 引數 (Default VM Arguments)	如果經常指定同樣的引數給特定的 VM，則可以在安裝的 JRE 喜好設定頁面，配置預設的 VM 引數。比起每採用一個啟動配置都要指定一次引數，這樣要方便得多。 
控制主控台 (Controlling your console)	主控台所顯示的輸出，可以透過「主控台」視圖工具列中的 Pin 主控台動作，鎖定至特定程序。另外還有一個捲動鎖定動作，可以停止主控台在附加新輸出時自動捲動。

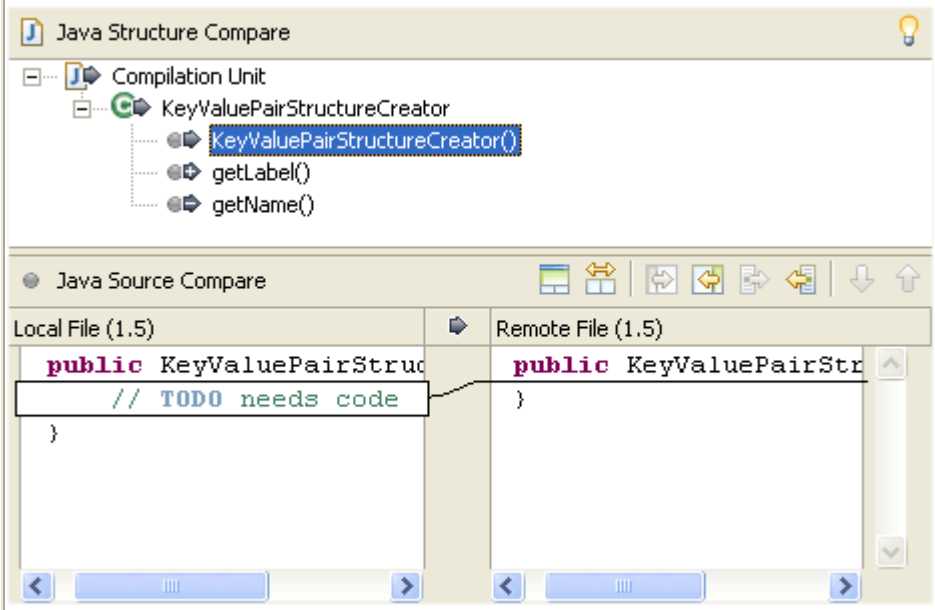
內容	說明
	
執行緒和監視器視圖 (Threads and Monitors view)	除錯器的執行緒和監視器視圖顯示哪些執行緒正保有鎖定，以及哪些正等待取得鎖定。 
逐行過濾器 (Step filters)	逐行過濾器可以避免除錯器在進入程式碼進行副程序除錯時，在指定的類別和套件中暫停執行。逐行過濾器是在「Window」→「Preferences」→「Java」→「Debug」→「Step Filtering」中建立的。當使用逐行過濾器切換為開啟時（在除錯工具列和功能表），逐行過濾器會套用到所有的逐行動作。在「除錯」視圖中，所選堆疊框的套件或宣告類型可以迅速地新增至過濾器清單，方法為從堆疊框的快速功能表選取 Filter Type 或 Filter Package。
執行有編譯錯誤的程式碼 (Running code with compile errors)	可以執行和除錯並未清楚地編譯的程式碼。執行有和沒有編譯錯誤的程式碼之間的唯一差異，就是如果執行一行有編譯錯誤的程式碼，將發生下列兩種情況之一： ■ 如果在「Java」→「Debug」喜好設定頁面上設定了'Suspend execution on compilation errors'，且正在進行除錯，則除錯階段作業將暫停，如似遇到岔斷點一般。請注意，如果 VM 支援「Hot Code Replace」，將可以修正編譯錯誤並回復除錯 ■ 否則，執行將終止，並出現'unresolved compilation'錯誤 有一點必須注意，只要執行路徑避開有編譯錯誤的程式碼行，就可以執行並除錯，正如同平常所做一般。

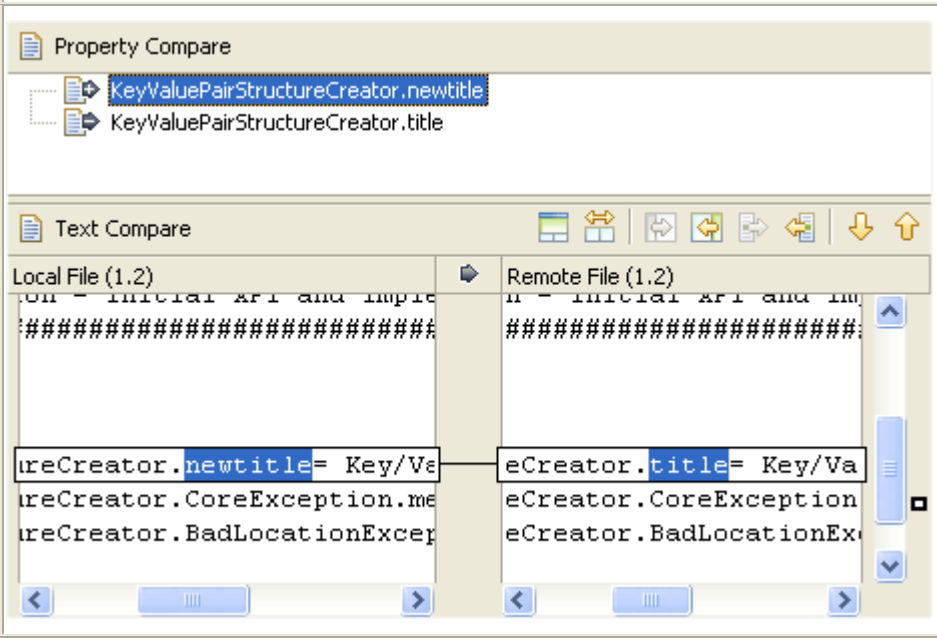
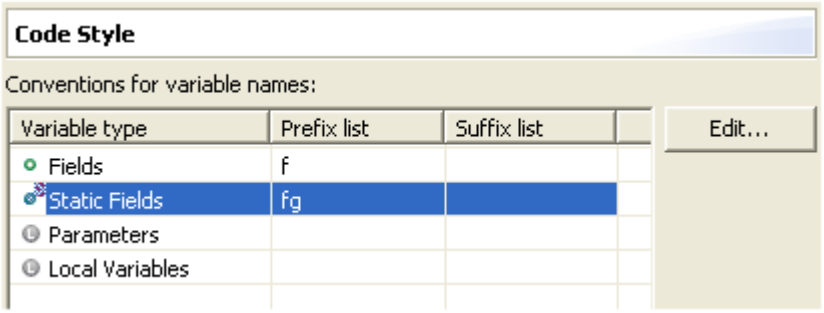
內容	說明
堆疊追蹤超鏈結(Stack trace hyperlinks)	主控台中會出現含超鏈結的 Java 堆疊追蹤。當滑鼠放在堆疊追蹤中的某一行上時，游標就會變成手狀，而且堆疊追蹤之下會有一條底線出現。按下滑鼠按鈕將開啟相關聯的 Java 程式檔，並且會將游標定位在對應行中。如果在堆疊追蹤頂端的異常狀況名稱上按一下滑鼠按鈕，就會建立一個異常狀況岔斷點。 

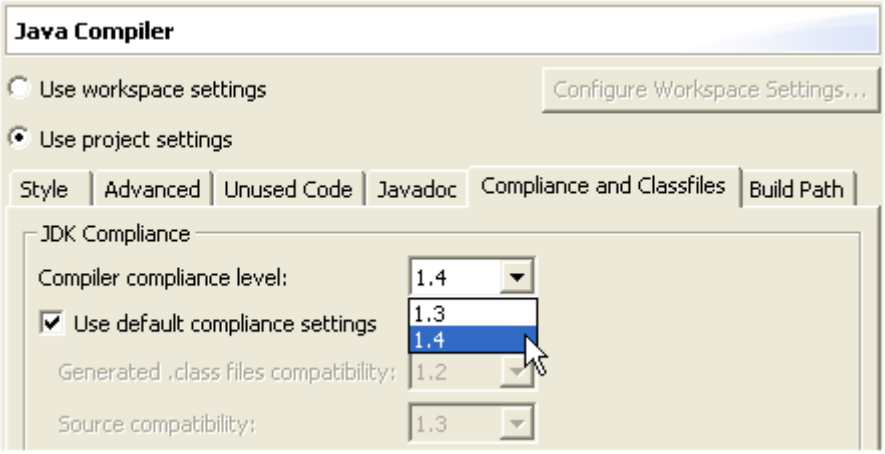
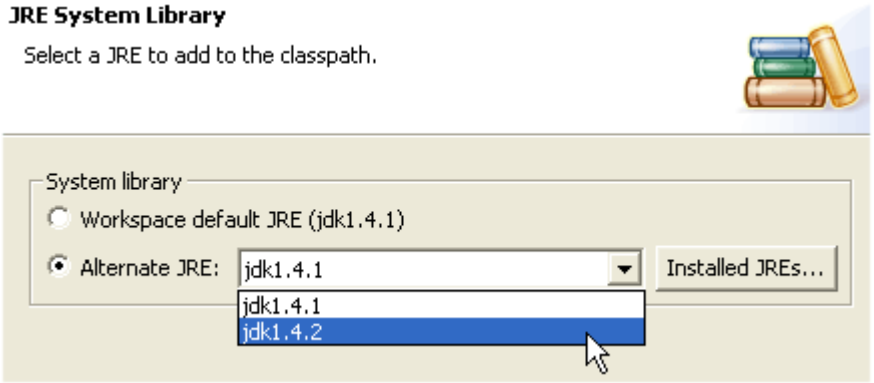
7.6 各種(Various)

內容	說明
JUnit	在視圖選取 JUnit 測試方法，然後從快速功能表選取「Run」→「JUnit Test」；或者從主功能表選取「Run」→「Run As」→「JUnit Test」。這會建立一個啟動配置來執行所選測試。
隱藏 JUnit 視圖直到發生錯誤或失敗(Hide JUnit view until errors or failures occur)	可以使用 JUnit 視圖僅在發生錯誤或失敗時才開啟。 如此一來，可以讓這個視圖設成快速視圖，且在沒有失敗測試時從不查看它。如果測試沒有問題，那麼在執行它們時，會看到這個圖示 （小綠色方塊的數目會隨之增加，指出進度），而且在完成它們之後，也會看到這個圖示 。不過，如果發生錯誤或失敗，圖示就會變成 （如果已經完成測試，則會變成 ），而且會出現 JUnit 視圖。可以透過「Java」→「JUnit」喜好設定頁面來配置這項行為。
對話框欄位中的內容輔助(Content assist in dialog	內容輔助(Alt+/)現在也可以在各種 Java 對話框的輸入欄位中使用。請尋找焦點所在的欄位旁的小燈泡圖示。

內容	說明
fields)	 舉個例說，內容輔助是在「New Java Class」、「New Java Interface」和「New JUnit Test」精靈，以及在變更方法簽章和移動 Static 成員的重構對話框中實作。 「Extract Local Variable」、「Convert Local Variable to Field」以及「Introduce Parameter」重構，會提供內容輔助提議給新元素名稱使用。
Java 程式檔的結構比較 (Structural compare of Java source)	Java 程式檔的結構比較會忽略 Java 元素（如方法和欄位）的文字次序，而更清楚的顯示哪些元素已經變更、新增或是移除。可以選擇下列一種方法，來起始 Java 檔的結構比較： ■ 選取兩個 Java 編譯單元(compilation units)，然後從視圖的快速功能表選取「Compare With」→「Each Other」。如果檔案不一樣，將以「Compare Editor」開啟它們。窗

內容	說明
	格頂端會顯示不同的 Java 元素；如果按兩下其中一個元素，便會在窗格底端顯示該元素的程式檔。 ■ 在任何包含檔案比較的環境定義中（如 CVS 同步化）按兩下 Java 檔，不僅可以文字比較檢視器顯示檔案內容，還會執行結構比較，並且開啟新窗格來顯示結果。  在執行結構比較時，甚至可以忽略註解和格式化變更：透過「Compare Editor」的工具列按鈕，或「CVS Synchronization」視圖的下拉功能表，來開啟忽略空格選項(Ignore Whitespace)。
內容檔的結構比較(Structural compare of property files)	Java 內容檔（副檔名：.properties）的結構比較會忽略內容的文字次序，而更清楚的顯示哪些內容已經變更、新增或是移除。 可以選擇下列一種方法，來起始內容檔的結構比較： ■ 在「Package Explorer」或「Navigator」中選取兩個檔案，然後從視圖的快速功能表選取「Compare With」→「Each Other」。如果檔案具有差異，將以「Compare Editor」開啟它們。窗格頂端顯示受影響的內容；按兩下它們之一將在窗格底端顯示內容的程式檔。 ■ 在任何包含檔案比較的環境定義中（如 CVS 同步化），按兩下內容檔不僅可以在文字比較檢視器顯示檔案內容，還會執行結構比較，並且開啟新窗格來顯示結果。

內容	說明
	
定義欄位、參數和區域變數的字首或字尾(Define prefixes or suffixes for fields, parameters and local variables)	除了配置欄位的字首或字尾之外，還可以指定 Static 欄位、參數和區域變數的字首或字尾。每當變數名稱需要重新計算時，「Java」→「Code Style」喜好設定頁面上的這些設定，就會用在內容輔助、快速修正以及重構作業。 
使用專案特有的編譯器設定(Use project specific compiler settings)	每一個專案都可以決定要使用廣域編譯器設定，還是定義專案特有的設定。選取專案並在專案內容中開啟 Java 編譯器頁面（「Project」→「Properties」→「Java Compiler」）

內容	說明
	
對專案使用特定的 JRE(Use a specific JRE for a project)	在建立新專案時預設會加入的 JRE，就是在「Preferences」→「Java」→「Installed JRE's」所選的 JRE。如果要設定專案特定的 JRE，請開啟專案的「Java Build Path」內容頁面(「Project」→「Properties」→「Java Build Path」)、開啟程式庫頁面、選取「Libraries」，然後按編輯。可以在「Edit Library」對話框中，選取要採用預設的 JRE，還是在新專案加入專案特有的 JRE。 
從異常不一致回復(Recovering from abnormal inconsistencies)	在發生罕見的功能不良事件時，JDT 可能會發生一些不一致的情況，例如： ■ Java Search 或 Open Type 漏掉結果 ■ 套件瀏覽器中的項目無效 如果要讓它們回復一致，必須完全依照下列順序，執行下列動作： 1. 利用導覽器關閉專案功能表動作，關閉所有的專案

內容	說明
	2. 結束和重新啟動 Eclipse 3. 利用導覽器開啟專案功能表動作，開啟所有的專案 4. 以手動方式觸發重新建置整個工作區 (「Project」→「Clean...」)