



BlueJ 導讀

第1.4版
針對1.2.X版的BlueJ

Michael Kölling
Mærsk Institute
University of Southern Denmark

戚玉樑博士審訂
張琪瑩、賴德優翻譯
中原大學資訊管理系跨平台資訊與系統整合實驗室

目錄

1 前言	3
1.1 關於BlueJ	3
1.2 本書範圍及讀者	3
1.3 版權及使用許可	3
1.4 回饋	3
2 安裝	4
2.1 Windows系統上的安裝	4
2.2 Macintosh系統上的安裝	4
2.3 Linux/Unix和其它系統上的安裝	5
2.4 安裝問題	5
3 開始—編輯/編譯/執行	6
3.1 開始BlueJ	6
3.2 開啓一個專案	7
3.3 建立物件	7
3.4 執行	9
3.5 編輯一個類別	11
3.6 編譯	12
3.7 使用編譯錯誤來協助	13
4 進一步...	14
4.1 檢閱	14
4.2 把物件作為參數傳遞	17
5 建立新專案	18
5.1 建立專案目錄	18
5.2 建立類別	18

5.3	建立類別間的相依關係	18
5.4	移除項目	19
6	除錯 (Debug)	20
6.1	設定中斷點	20
6.2	逐行執行	22
6.3	查詢變數	22
6.4	暫停與終止	23
7	建立獨立運作的應用程式	24
8	建立Applets	26
8.1	執行applet	26
8.2	建立applet	27
8.3	測試applet	27
9	其他操作	28
9.1	在BlueJ開啓非BlueJ的Packages	28
9.2	增加已存在的類別到專案中	28
9.3	呼叫主要或其他靜態方法	28
9.4	產生文件	29
9.5	與查詢系統一同作業	29
9.6	從Library類別建立物件	29
10	摘要	31

1 前言

1.1 關於 BlueJ

本文件的目的是在於介紹如何使用 BlueJ 這個程式開發環境，BlueJ 是特別為初學者而設計的 Java™ 開發環境。BlueJ 是由澳洲墨爾本 Monash 大學的 BlueJ 小組所設計及開發的。

如需要更多資訊請參考這個網址：<http://bluej.monash.edu/>

1.2 本書範圍及讀者

本文件是針對那些想讓自己熟悉如何使用這個開發環境的讀者，對於想討論這個環境設計過程中的結構組織和較深入的研究問題，則不在本文件的範圍。

本文件不會介紹 Java 的使用方法，因此讀者需已對 Java 有概念性的瞭解。當然，這份文件不是一份全面性的參考手冊，本文件重點在於簡單扼要的介紹 BlueJ。

每一小節都有一行的摘要作為開始，讀者可以根據自己對其瞭解與否來決定是否研讀或跳過。第10節是重複一遍這些摘要總結，提供讀者快速的參考。

1.3 版權及使用許可

BlueJ 系統以及這份文件對任何人任何用途都是免費的，BlueJ 系統以及它的文件檔案可以被重新發佈而不需要任何費用。

沒有經過本系統作者的授權，任何人不能銷售 BlueJ 系統或者包含該系統的套裝軟體並從中獲利。

BlueJ 版權所有© M. Kölling, J. Rosenberg

1.4 回饋

我們歡迎任何對 BlueJ 與這份導讀有關的批評、建議、問題及錯誤更正等。請 E-mail給Michael Kölling (mik@mip.sdu.dk)，中譯問題請逕洽 maxchi@cycu.edu.tw。

2 安裝

BlueJ 共有三種不同型式的安裝版本：一種是 Windows，一種是 MacOS，最後一種是其它系統的，安裝的方式十分簡單。

安裝前的準備

你必須先在你的電腦上安裝了 JDK1.3 (另稱為 J2SE) 或更高的版本。如果你還沒有安裝 JDK，你可以從 Sun 公司的網站下載：<http://java.sun.com/j2se/>。在 MacOS X 的系統上，最新的 JDK 版本是事先就安裝好的，也就是說你並不需要自行安裝。假如你找到了提供下載“JRE” (Java Runtime Environment) 和 “SDK” (Software Development Kit)的網頁，你必須下載“SDK”，只下載 JRE 是不夠的。

2.1 Windows系統上的安裝

在 Windows 系統上安裝的檔案名稱爲 *bluejsetup-xxx.exe*，其中 *xxx* 指的是版本編號。例如，BlueJ version 1.2.0的安裝檔名就是 *bluejsetup-120.exe*，你可以從光碟上得到這個檔案，或者從 BlueJ 的網站上下載：<http://www.bluej.org>。執行安裝程式。

安裝程式會讓你選擇 BlueJ 的安裝目錄，它也會在『開始』的選單和桌面上提供安裝程式的捷徑。接著在完成安裝以後，你將找到 *bluej.exe* 這個程式在 BlueJ 的安裝目錄中。

在第一次執行 BlueJ 時，它會自動尋找 Java system (JDK)，假如它找到一個以上 Java system (如你已安裝了JDK 1.3.1 and JDK 1.4)的話，那麼這時就會有一個對話視窗讓你選擇所要選用的 Java system。假如它沒有找到的話，它將會要求你自己找出來。

BlueJ 的安裝程式也會安裝一個叫做 *vmselect.exe* 的程式，使用這個程式，你可以在稍後改變 Java version BlueJ 的版本。執行 *vmselect* 可以啓動 BlueJ 使用不同版本的 Java。

JDK 的版本可以被儲存在每一版的 BlueJ 中，假如你安裝了不同版本的 BlueJ，那麼可以在其中一個版本使用 JDK 1.3.1，另一個版本使用 JDK 1.4。

2.2 Macintosh系統上的安裝

請注意 BlueJ 只能在 MacOS X 執行。

在 Mac 系統上安裝的檔案名稱爲 *BlueJ-xxx.sit*，其中 *xxx* 指的是版本編號。例如，BlueJ version 1.2.0 的安裝檔名就是 *BlueJ-120.sit*，你可以從光碟上得到這個檔案，或者從 BlueJ 的網站上下載：<http://www.bluej.org>。執行安裝程式。

我們可以使用 *StuffIt Expander* 來解壓縮這個檔案，有許多的瀏覽器都可以解壓縮這個檔案。除此之外，用滑鼠雙點這個檔案也可以進行解壓縮。

在解壓縮完成後，它會產生一個 *BlueJ-xxx* 的資料夾，接著把這個資料夾移到 Applications 目錄中(或著其它你想儲放的位置)，接著完成它的安裝。

2.3 Linux/Unix和其它系統上的安裝

一般執行安裝的檔案都是一個 jar 的檔案，它的檔名爲 *bluej-xxx.jar*，其中 *xxx* 指的是版本編號。例如，BlueJ version 1.2.0 的安裝檔名就是 *BlueJ-120.jar*，你可以從光碟上得到這個檔案，或者從 BlueJ 的網站上下載：<http://www.bluej.org>。

使用下面的命令執行安裝程式。在這個範例中，我使用的安裝檔爲 *bluej-120.jar* — 你應該使用你得到的安裝檔名稱(版本編號必須正確)。

```
<jdk-path>/bin/java -jar bluej-120.jar
```

其中<jdk-path>是 JDK 安裝的目錄。

接著會出現一個視窗，讓你選擇 BlueJ 的安裝目錄和執行 BlueJ 時用的 JDK 版本。注意：BlueJ 的安裝路徑（任何一個父目錄）中不能有任何空格。

點選 *Install*，完成之後，BlueJ 就安裝完成了。

2.4 安裝問題

假如你有任何疑問，請參考BlueJ網站上的 FAQ (<http://www.bluej.org/help/faq.html>) 並且閱讀如何取得幫助的章節 (<http://www.bluej.org/help/ask-help.html>)。

3 開始—編輯/編譯/執行

3.1 開始 BlueJ

在 Windows 和 MacOS 系統中，執行一個名稱爲 *BlueJ* 的程式。

在 Unix 系統中，安裝程式已經安裝了一個名稱爲 *bluej* 的 script 於安裝目錄中，你只需要在圖形介面裡執行該檔案就可以啓動 BlueJ。

在命令列裡，你可以以帶參數或不帶參數的方式啓動 BlueJ：

```
$ bluej
```

或者

```
$ bluej example/people
```

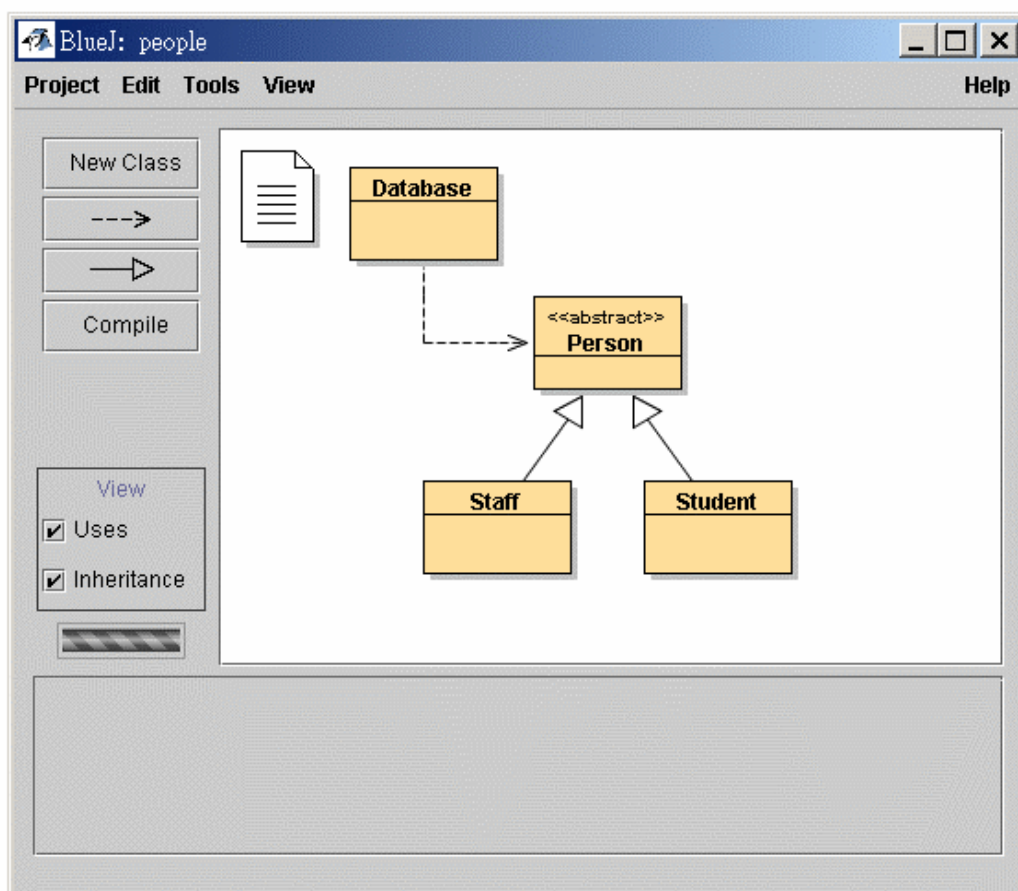


圖1：BlueJ 主視窗

3.2 開啓一個專案

摘要：要開啓一個專案，請點選功能表中 Project menu 中的 Open。

BlueJ 的專案和標準的 Java 套件一樣，是一個包含專案檔案的目錄。在啓動 BlueJ 後，點選 *Project* 功能列中的 *Open* 來開啓專案，此時一些標準的 BlueJ 範例專案已經放置在 *examples* 的目錄中。

爲了學習這份導讀，開啓專案 *people*，你可在 BlueJ 安裝目錄下的 *examples* 目錄中找到這個專案，開啓專案後你應該可以看到和圖1類似的視窗，這個視窗可能看起來和圖1不盡相同，但差別應該不大。

3.3 建立物件

摘要：要建立一個物件，就從類別功能表中選擇一個建構函數。

BlueJ 有一個很基本的特點就是你不僅能夠執行一個完整的應用程式，而且也能夠直接和單獨一個任何類別的物件互動溝通，並執行物件裡的 *public* 方法。在 BlueJ 中的可執行檔案通常需要事先建立物件，然後呼叫其中一個物件的方法來完成。這種模式在開發一個應用軟體的過程中是十分有幫助的—當你一寫完某個類別時，你就可以單獨測試這個已經完成的類別，這樣就沒有必要一開始就把整個應用程式寫完。

*註解：靜態方法不需事先建立一個類別就能夠直接呼叫，*main()* 是一個靜態方法，因此我們可以做和 Java 應用程式中相同的事情—透過執行靜態方法 *main()* 的方式啓動一個應用程式。我們會在後面繼續討論。現在，我們來做一些不能在一般的 Java 環境中所做到的有趣事情。*

你可以在下列視窗裡看到幾個圖示（分別標註了 *Database*、*Person*、*Staff* 和 *Student*），這些圖示是代表了能夠在這個應用程式中所能用到的類別。在類別的圖示上按滑鼠右鍵，你會得到一個能夠操作該類別的一個功能表 (Macintosh: ctrl-click) (如圖2所示)。圖中顯示的操作功能是由 BlueJ 作業環境所提供，並以 *new* 功能搭配類別中定義的各個建構函數所組成。

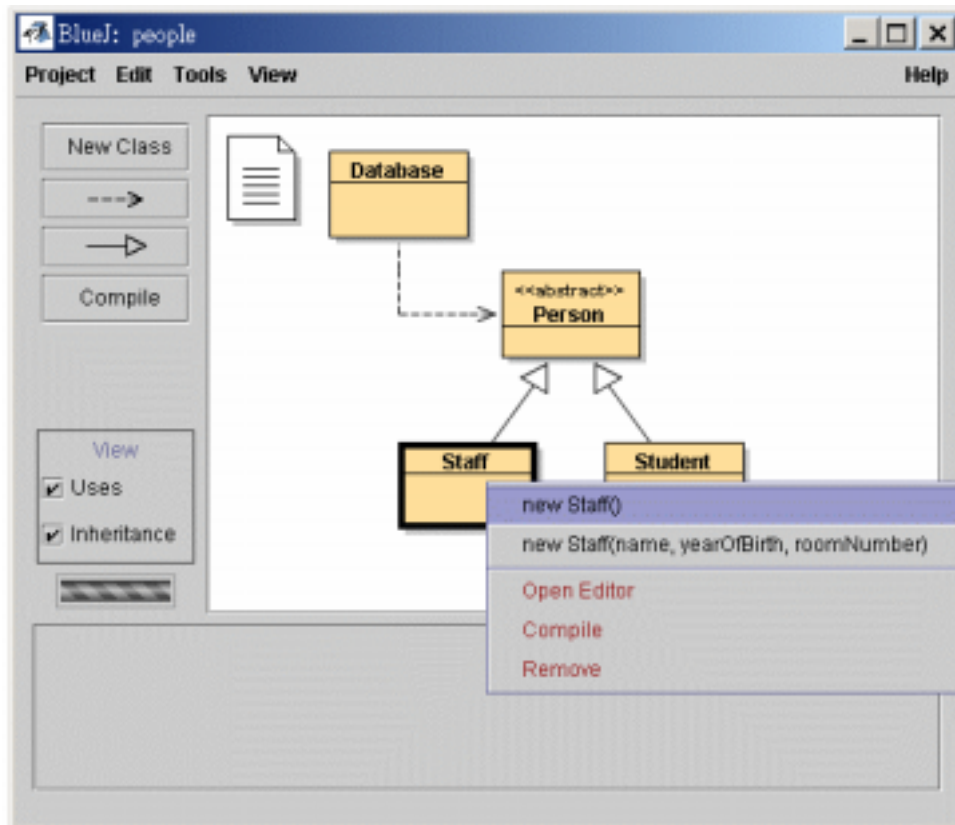


圖2：類別功能（彈出功能表）

例如我們現在希望建立一個 *Staff* 物件，所以你應該右鍵單擊 *Staff* 圖示（會彈出如圖2中顯示的功能表），這個功能表會顯示了兩個建構函數可以用來建立 *Staff* 物件：一個有參數，另一個則沒有。在此，先選擇沒有參數的建構函數，對話方塊將會如圖3所示：

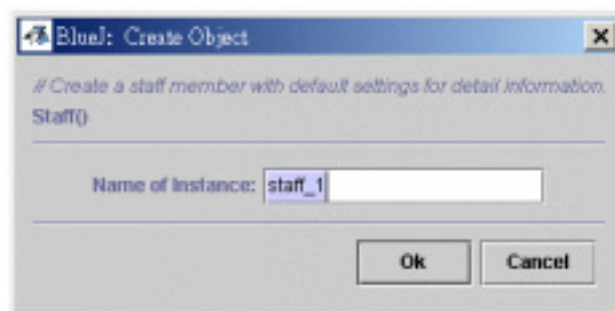


圖3：建立沒有參數的物件

這個對話方塊要求你輸入一個物件的名字，同時它提供了一個預設的名字（*staff_1*）。如果這個預設的名字對我們來說已經足夠了，就按 *OK*，一個 *Staff* 物件就建立完成了。

當一個物件建立完成後，它會被放在視窗底部的物件工作區裏(如圖4)。

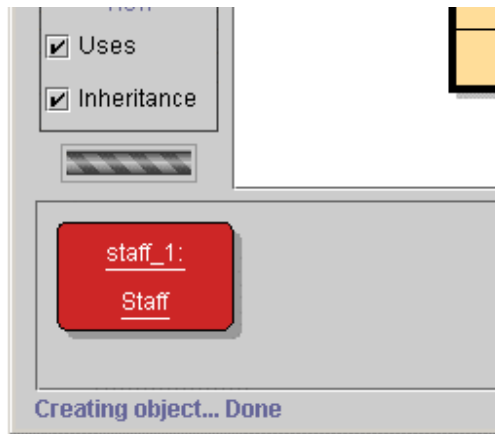


圖4：一個物件在物件工作區裏

你可能已經注意到 *Person* 類別被標註為 `<<abstract>>`，這表示他是一個抽象類別，如果你試著從 *Person* 類別建立物件的話，你將會發現這是不可能的，因為你不能夠建立一個抽象類別的物件，這就如同 Java 語言定義的規則一樣。

3.4 執行

*摘要：*要執行一個方法，從物件功能表中選擇它。

現在你已經建立了一個物件，你可以執行它的 *public* 行為 (Java稱這種行為是 *methods*)。用右鍵單擊剛剛所建立的物件圖示，就會彈出一個包含物件操作的功能表 (如圖5所示)。這個功能表顯示了該物件所有可執行的方法和兩個 BlueJ 環境提供的操作功能 (分別是 *Inspect* 和 *Remove*)，我們會在後面討論這些功能。現在，讓我們集中注意力在方法上。

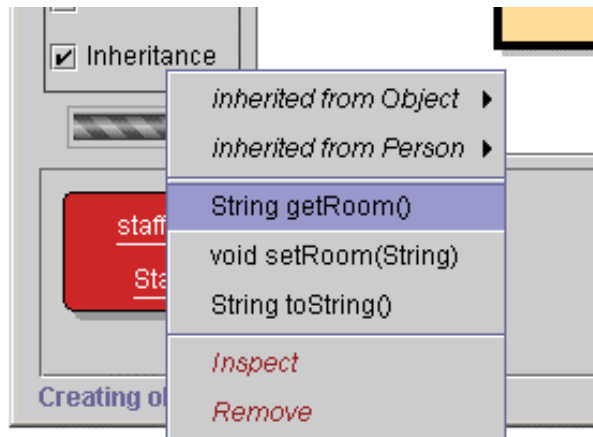


圖5：物件功能表

你可以看到功能表裏有兩個方法 *getRoom* 和 *setRoom*，分別是設定和回覆這個教職員的房間編號。試著呼叫 *getRoom*，只要簡單的從功能表裏選擇它，它就會被執行，此時一個對話方塊會顯示出執行的結果（如圖6所示）。在這個範例中，執行的結果是“(unknown room)”，這是因為我們還沒有為這個教職員指定房間編號。

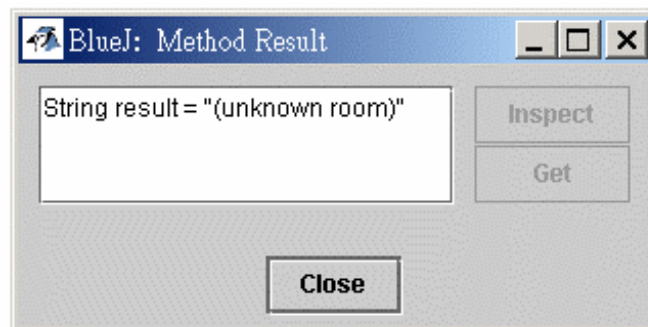


圖6：顯示一個函數的結果

從父類別繼承而來的方法我們可以在子功能表中找到，在物件彈出功能表的最上方有兩個子功能表，一個包含從 *Object* 繼承而來的方法，另一個包含從 *Person* 繼承而來的方法（如圖5），你可以透過選擇 *Person* 子功能表中的函數名稱來呼叫 *Person* 的方法（例如 *getName*），你會發現執行結果同樣的奇怪：它出現了“(unknown name)”，這是因為我們也還沒有給這個人取一個名字。

現在讓我們試著來指定一個房間號碼，以下的操作將顯示如何呼叫有參數的方法（結果的呈現是經由呼叫 *getRoom* 和 *getName* 來達成，但呼叫時不需參數），選擇功能表中的 *setRoom* 功能，一個對話方塊會出現提示你輸入參數（如圖7所示）。

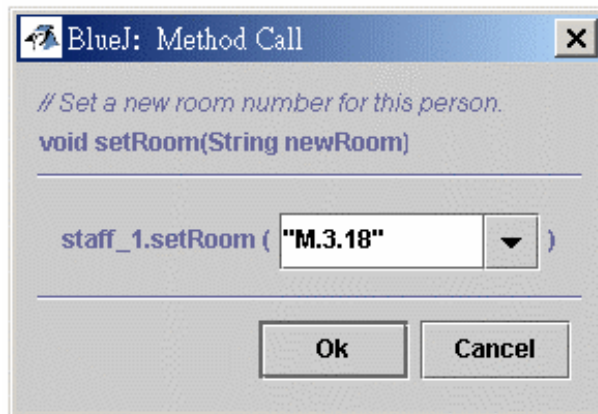


圖7：呼叫帶有參數的函數

在這個對話方塊的最上端顯示的是被呼叫方法的介面說明（包括註解和語法定義），下面是一個輸入方塊，你可以在裡面輸入參數值，對話方塊中的定義告訴我們需要一個字串類型的參數，接著在輸入方塊裡輸入新的字串類型的房間名字（須包括雙引號），然後點擊 *OK*。

以上作業到此為止－由於這個方法沒有返回值，所以沒有執行結果對話方塊。現在再一次呼叫 *getName* 來檢查房間名字是否真的改變了。

你可以開始對物件的建立和方法呼叫稍加練習一下，然後試著呼叫有參數的建構函數和更多的方法直到你對這些操作熟悉為止。

3.5 編輯一個類別

摘要：要編輯類別的程式碼，只要雙擊類別圖示即可。

到目前為止，我們只討論到了一個物件的介面，現在是探討一下物件內部的時候了，一個類別究竟是如何的實作，可透過在類別功能表裡選擇 *Open Editor* 來瞭解（提示：右鍵單擊類別圖示就會跳出類別功能表），雙擊類別圖示也可以達到同樣的功能。這份文件對 BlueJ 的編輯器沒有很深入的描述，因為使用 BlueJ 一般來說是相當簡單且直覺化的。關於編輯器的細節描述在後面章節會再說明。現在，打開實作 *Staff* 類別，找到 *getRoom* 方法，正如函數名稱所的意思，它會返回該教職員的房間編號。現在讓我們改變這個方法，在函數返回值的前面加上一個“*room*”字首（這樣這個方法就會返回，“*room G.4.24*”而不是“*G.4.24*”）。我們透過修改下面這一行來做到這一點。

```
return room;  
改爲  
return “room ” + room;
```

BlueJ 完全支援 Java，所以在類別的實作上並沒有什麼特殊的要求。

3.6 編譯

*摘要：要編譯一個類別，就在編輯器中點選 **Compile** 按鈕。如果是要編譯一個專案，就點選專案視窗裏的 **Compile** 按鈕。*

經過剛剛在編輯器內新增文字之後，試著看看專案的主視窗有什麼不同，你會發現 *Staff* 的圖示出現了條紋的改變，條紋的出現表示類別文件自從上次修改以來，還沒有被重新編譯過，請再返回編輯器。

*註解：你可能在疑惑，為什麼前面類別在專案第一次被打開的時候沒有條紋，這是因為 *people* 專案中的類別都是已經被編譯過的。在一般情況下，隨著 BlueJ 一起發佈的專案通常是沒有被編譯過的，所以新開啓的專案，類別圖示上都會出現條紋。*

在編輯器頂端的工具列包含一些經常使用的功能按鈕，其中一個是 *Compile*，你可以使用這個按鈕直接編譯目前開啓的類別文件。現在點一下 *Compile* 按鈕。如果你沒有犯任何錯誤，那麼在編輯器最下方的訊息區會出現一個訊息提示你這個類別已經被編譯完成，如果你犯了語法的錯誤，錯誤行會反白顯示，並且在訊息區會有相對應的錯誤提示。（假設你第一次進行編譯是沒有出錯，現在可試著製造一個語法錯誤－比如漏寫分號－然後再次編譯，看看會出現什麼結果）。

在你成功的編譯完成後，關閉編輯器。

註解：你沒有必要特地去儲存一個類別文件，原始檔案在正常操作下會被自動地儲存（比如當編輯器關閉時或者類文件被編譯時）。當然，你也可以刻意的儲存文件（在編輯器的工具列上有相對應的按鈕），但是這種操作真的沒什麼必要，除非你的系統相當不穩定，隨時都有當掉的危險，而你又很擔心你會失去你的工作成果。

專案視窗的工具列上同樣也有 *Compile* 按鈕，這個編譯功能會編譯整個專案（實際上它判斷哪些類別需要重新編譯然後按照正確的順序重新編譯這些類別），你可以試著修改二個或更多的類別（這樣會有二個或多個類別的圖示出現條紋），然後點選 *Compile* 按鈕，如果在被編譯類別的過程中出現錯誤，編輯器就會被打開，錯誤位置和錯誤資訊會被顯示出來。

你可能注意到了物件工作區又被清空了，當每次實作改變之後舊的物件就會被刪除。

3.7 使用編譯錯誤來協助

摘要：要由編譯錯誤訊息得到協助，用滑鼠點一下錯誤訊息旁邊的問號圖示。

有一個很普遍的現象就是初學者對編譯錯誤訊息總是不太容易理解，以下我們試著提供一些幫助。

請再次打開編輯器，在原始檔案裏製造一個錯誤，然後編譯它。一個錯誤資訊便會出現在編輯器的訊息區裡，在它的右端有一個問號，你可以點一下它來得到更多的關於這個類型錯誤的資訊（如圖8所示）。

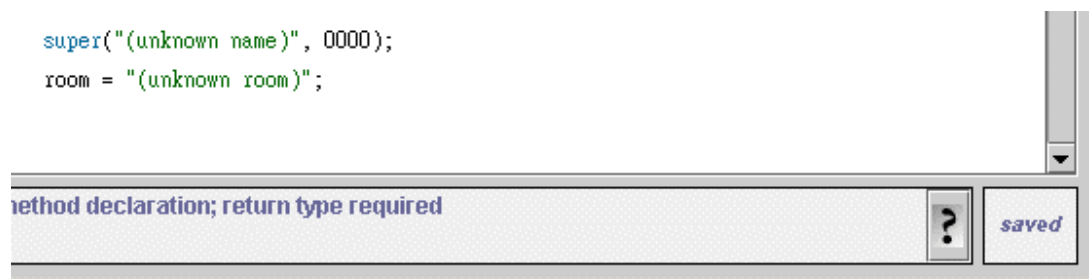


圖8：編譯器錯誤和幫助按鈕

在現階段中並不是所有的錯誤訊息都有相對應的說明檔，一些說明檔目前仍然持續編寫增加，但是許多錯誤訊息都已經有了相對應的解釋，剩下的那些錯誤訊息將會被編寫並包含在 BlueJ將來的版本中。

4 進一步

在本章節中，我們將教導你在這個撰寫環境的一些額外的情況，雖然這些小情況雖然不是必須的，但卻是經常用到的。

4.1 檢閱

摘要：利用顯示物件的內部狀態，物件檢閱可以作一些簡單的除錯。

當你執行一個物件的方法時，你可能會注意到 *Inspect* 這項功能，它可以應用於除了使用者定義方法之外的所有物件（如圖5）。檢閱操作可以檢閱物件的實體變數（“fields”）的狀態，試著用一些使用者自行定義的值來建立一個物件（例如：以帶參數的建構函數建立的一個 *Staff* 物件）。然後從物件功能表中選擇 *Inspect*，就會彈出一個對話方塊顯示該物件的欄位，其類型和值如圖9所示。

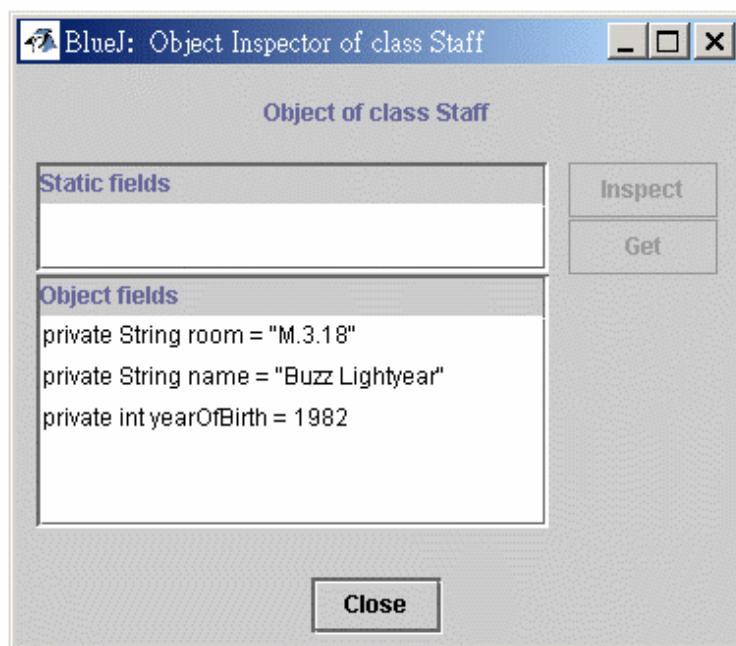


圖9：檢閱對話方塊

檢閱物件對於快速檢查那些物件局部改變(如物件狀態)是否被正確的執行是很有用的。因此“檢閱物件”功能是一個簡單的除錯工具。

在 *Staff* 這個例子中，所有的欄位都是簡單資料類型，這些類型的值可以直接顯示出來，這樣你可以及時查看建構函數是否正確的完成任務。

在更複雜的情況下，欄位的值可能參照使用者自行定義的物件，我們將用下一個專案來展示這樣的例子：開啓 *people2* 專案，此專案同樣包含在標準的 BlueJ 版本中。*people2* 的介面如圖10所示，在這個例子中，除了有前面看到的那些類別之外還有一個 *Address* 類別，*Person* 類別中的其中一個欄位就是使用者自行定義的類型 *Address*。

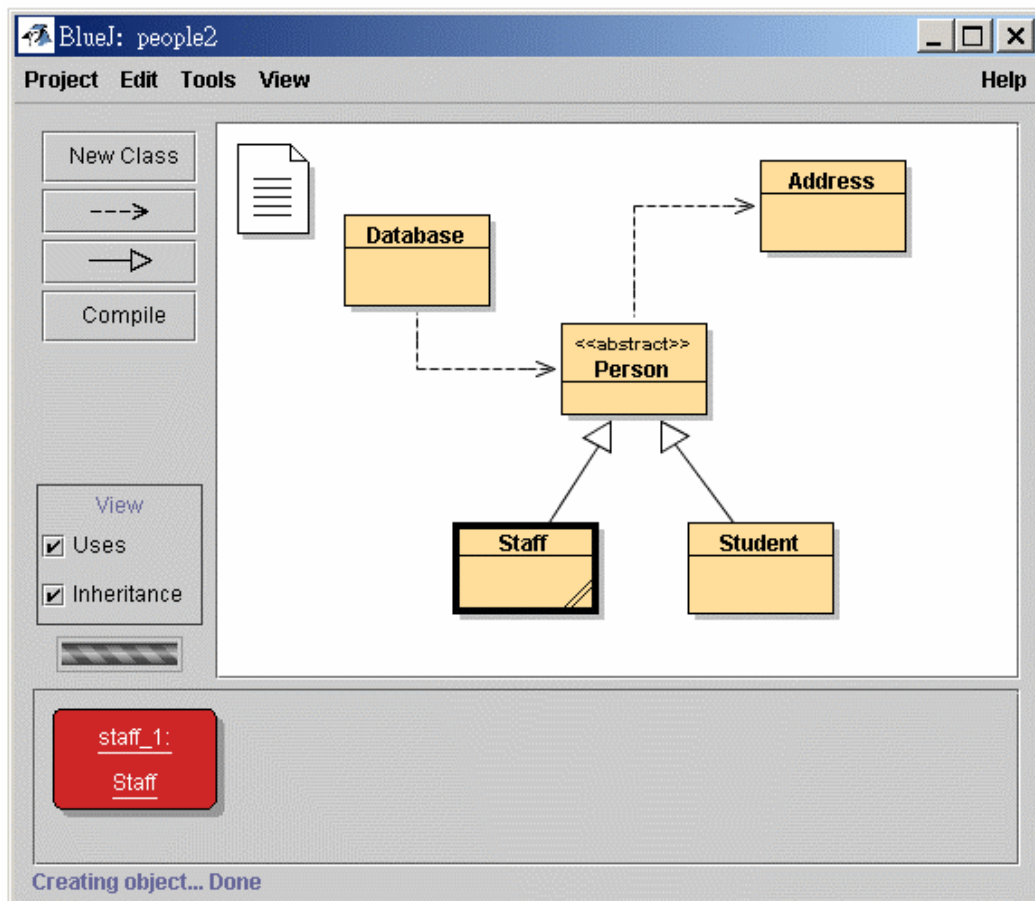


圖10：People2專案視窗

接下來我們要測試的是檢閱物件的欄位—試著建立一個 *Staff* 物件並且呼叫它的 *setAddress* 方法（你可以在 *Person* 子功能表中找到），然後輸入一個位址，在內部 *Staff* 的程式碼建立了 *Address* 類別的一個物件並且儲存在 *address* 欄位裏。

現在檢閱 *Staff* 物件，檢閱結果的對話方塊如圖11所示，此時 *Staff* 類別內部的欄位包含 *address*，它的值顯示為 *<object reference>*，因為這是一個由使用者自行定義的物件，所以它的值不能直接被顯示在列表中。為了進一步檢查 *address*，在列表中選擇 *address* 欄位並且點一下對話方塊中的檢閱物件按鈕（你同樣也可以雙擊 *address* 欄位），這時就會彈出另一個檢閱視窗，這時候就會顯示 *Address* 物件的詳細內容（如圖12所示）。

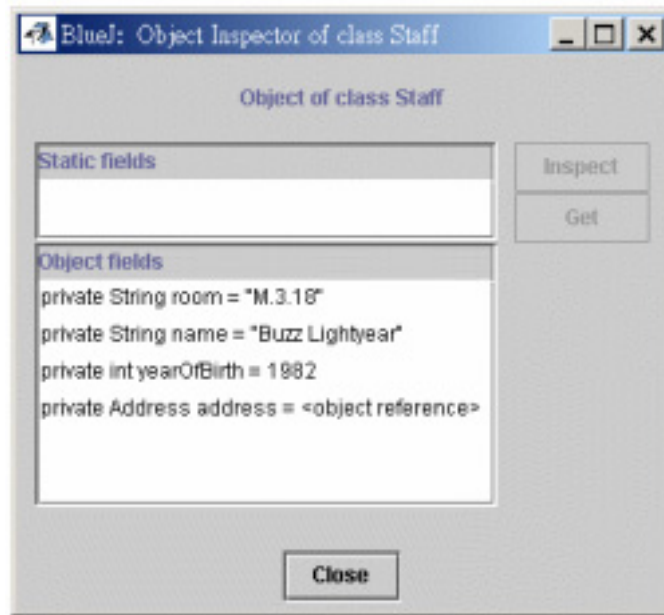


圖11：物件參照的檢閱

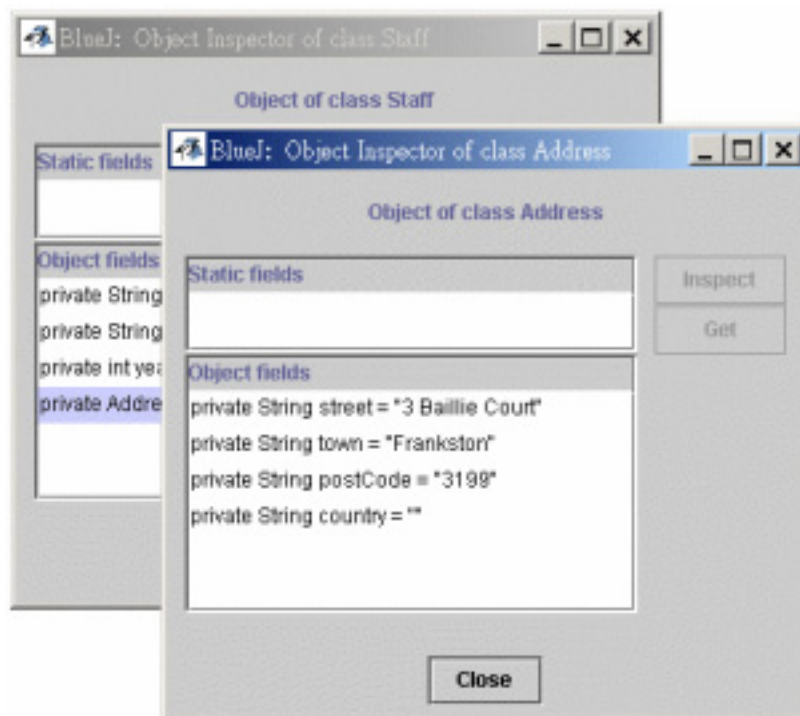


圖12：檢閱內部物件

如果選擇的欄位是 `public`，你同樣可以選擇 `address` 欄位並且點選 `Get` 按鈕來替代 `Inspect` 按鈕，此項操作會將選取的物件放入物件工作區中，你可以進一步的透過呼叫它的方法來測試它。

4.2 把物件作為參數傳遞

摘要：透過點擊一個物件的圖示可以把一個物件作為參數傳給另一個方法使用。

一個物件可以被作為傳遞的參數，進而傳送到另一個物件的方法中，讓我們試著執行下面的例子，首先建立一個 *Database* 類別的物件（你會注意到 *Database* 類別只有一個不帶參數的建構函數，所以建立一個物件只有這樣一種方式），*Database* 物件可含有一群 *person* 的能力，它有相對應的一些操作來加入 *person* 物件和顯示所有當前儲存的 *person*。（把這個物件稱為資料庫是有點超過了！）

如果你還沒有在物件工作區中建立一個 *Staff* 或者 *Student* 物件，請先建立它們之中的一個，接下來，在物件工作區中同時需要一個 *Database* 物件和一個 *Staff* 或者 *Student* 物件。

現在呼叫 *Database* 物件的 *adperson* 方法，語法提示將會告訴你需要一個 *Person* 類型的參數，（提醒你，*Person* 類別是一個抽象類別，因此沒有任何一個物件是 *Person* 類型；但由於 *subTyping*，*Student* 和 *Staff* 物件則可以作為 *Person* 的替代物件，因此在需要 *Person* 類型的參數時，傳送 *Staff* 或者 *Student* 是合法的。）為了將你的物件工作區中的物件作為一個參數傳送給你呼叫的方法，你可以在參數欄位中輸入它的名字或者利用另一種快速的方式，只需要點擊你需要的物件，這將會把它的名字輸入到方法呼叫對話方塊中去，點擊 *OK* 之後呼叫就生效了。因為這個方法沒有任何的返回值，我們不能立刻看到結果。你可以呼叫 *Database* 的 *listAll* 方法查看此操作是否確實被執行了。*ListAll* 這個方法會把個人資訊寫入到標準輸出，你將會發現一個文字畫面會自動開啓並顯示這些訊息。

請練習再輸入更多的個人資料到“資料庫”中。

5 建立新專案

本章節將讓您瞭解如何迅速的建立一個新專案。

5.1 建立專案目錄

摘要：要建立專案，就選擇 Project 功能表中的 New Project。

在 Project 功能表中選擇 *New Project* 就能夠建立新專案，此時你需要設定新專案的名稱以及目錄存放的位置，在設定完成後就能發現你所設定的目錄名稱已經建立，而 BlueJ 主畫面也會顯示一個新的空白專案。

5.2 建立類別



摘要：要建立類別，按下 New Class 按鈕並設定類別名稱。

在專案功能列中選擇 *New Class* 按鈕就可以建立類別，此時需要給此類別一個合法的名稱，你可以選擇四種不同的類別型態：*Class*、*Abstract Class*、*Interface* 或 *Applet*。這個選擇將會決定此類別的語法架構，你也可以在之後編輯程式碼時改變此類別型態，例如在程式碼中加入修飾字 “*abstract*”。



在建立完成類別之後，畫面上就會顯示出一個代表此類別的圖示。如果此類別不是標準的 *Class* 型態，就會在圖示上有著註解，例如 <<*abstract*>>代表 *Abstract Class*，<<*interface*>>代表 *Interface*，<<*applet*>>代表 *Applet*。當你雙擊此類別圖示或是在圖示上按右鍵開始選單，選擇 *Open Editor* 時，你可以發現一個類別的基本語法已經建立完成了！這將會有助於你開始撰寫程式，這些已經建立的程式碼在語句構造上是合法的，所以你亦可以立即編譯而不會產生錯誤。

5.3 建立類別間的相依關係

摘要：類別間的相依關係，可以用拖曳箭頭的方式建立，或是在編輯器中撰寫程式碼達成。

類別之間的相依性是以不同型式的箭頭呈現，繼承關係是以  表示（如 *extends* 或 *implements*），使用關係則是以  表示。

增加類別間的相依關係可以用兩種方式達成：直接在圖形上設定，以及在程式碼中撰寫程式。如果是以圖形方式增加關連，程式碼亦會自動更新，相反地如果是撰寫程式語法表達關係，圖形亦會產生相對應的箭頭符號。

在使用圖形方式增加關連時，必須注意要選擇適當的箭頭符號，表示 *extends* 或 *implements* 關係，而  表示使用關係。使用方式很簡單，只要從一個類別拖曳出箭頭圖示到另一個類別即可。

增加繼承箭頭將會在類別程式碼中插入 *extends* 或 *implements* 定義，至於是哪一個定義則由箭頭的目的地是類別或介面而定。

增加使用箭頭並不會立即改變程式碼，除非箭頭的終端是另一個 *Package* 的類別，如果發生上述的情形，在程式碼中就會產生 *import* 的陳述。如果使用箭頭指到某一個類別，但是卻沒有在程式碼中使用這個類別，將會產生一個警告訊息顯示：這個類別已經宣告了他的關係，但是卻沒有被使用過！

以文字去增加箭頭也是很容易的，只要跟往常一樣撰寫程式碼即可，當類別儲存時，圖形也就會跟著更新。（注意：當你關掉編輯器時，就會自動儲存！）

5.4 移除項目

*摘要：要移除類別，只需在類別上按右鍵選取功能表中的 *Remove* 則可，要移除箭頭，則需先選取功能列上的 *Edit* 中的 *Remove Arrow* 後，再選取欲移除的箭頭。*

要移除類別時，先選取欲移除的類別，再選取功能列上 *Edit* 中的 *Remove* 即可，或是直接在欲移除類別上按右鍵選擇 *Remove* 亦可；若是要移除箭頭，則先選取功能列上 *Edit* 中的 *Remove Arrow* 之後，再選取欲移除的箭頭即可。

6 除錯 (Debug)

本章節將介紹 BlueJ 在除錯方面的幾個重要功能，在與資訊相關科目老師的會談中，我們最常聽到他們認為教導初學者使用程式除錯器的重要性，但往往因時間的限制而無法遂行，而初學者也因忙於程式的撰寫、編譯、與執行，所以也似乎沒有剩餘的時間去學另一個複雜的工具。

正因為上述的理由，我們決定做一個非常簡單的程式除錯器，希望能藉由大約 15 分鐘的講授後，初學者便可自行使用這個程式除錯器，讓我們一同看看是如何達成此目標。

首先，我們簡化了傳統除錯機制，進而成為下列三項功能：

- 設定中斷點 (Breakpoints)
- 逐行執行
- 監看變數

現在就執行 BlueJ，開啓 BlueJ 安裝目錄 *examples* 中的 *debugdemo* 專案，經由這個專案你能夠瞭解除錯的一些基本範例。

6.1 設定中斷點

摘要：要設定中斷點，只需在編輯器左方的中斷區域點選即可。

設定中斷點能夠讓你在程式執行時中斷在程式碼中的特定地方，當程式中斷時，你就可以觀察程式執行至中斷點時物件的狀態，這有助於瞭解程式的執行狀況與結果，在編輯器的最左邊就是中斷點區域（如圖 13 所示），你只需在欲中斷的那列程式碼左方按一下滑鼠即可設定中斷點，此時一個小的中斷記號就會顯示。現在就實做一次！開啓 *Demo* 這個類別，找到 *loop* 這個方法，設定一個中斷點在 *loop* 方法的某處，你就已經完成了中斷點的設定！

```
public int loop(int count)
{
    int sum = 17;

    for (int i=0; i<count; i++) {
        sum = sum + i;
        sum = sum - 2;
    }
    return sum;
}
```

圖 13：中斷點

當程式執行到這個中斷點時就會自動中斷，現在就實做一次！首先建立一個 *Demo* 類別的物件，接著呼叫方法 *loop*，輸入 *count* 數值為 10，當執行到中斷點時，就會顯示出目前執行到哪一程式、狀態如何，如圖 14 所示。

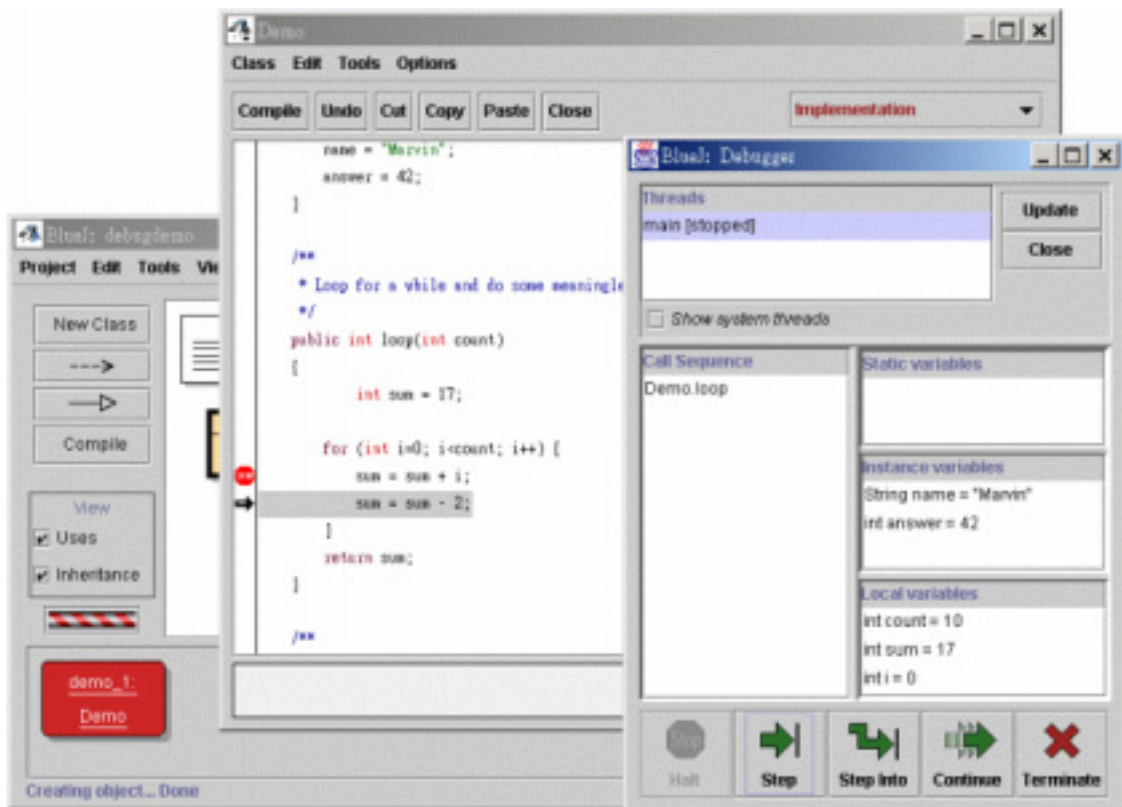


圖 14：除錯器視窗

編輯器所顯示的反白列表示中斷後將被開始執行的地方（亦即程式在前一列執行完畢後停止）！

6.2 逐行執行

摘要：要一步一步執行程式，就使用除錯器 (debugger) 中的 Step 和 Step Into 按鈕。

設定中斷點可以停止執行中的程式，但是如果想要瞭解程式是如何執行的，就可以用逐行執行的方式來達成。要如何做逐行執行呢？只需重複按除錯器上的 Step 按鈕即可！你就可以看到編輯器上的較明顯那行程式碼不斷的改變，那行就是即將被執行的程式碼，每當你按一次 Step 按鈕，就會執行一行程式碼並中斷一次，此時在除錯器上的變數也會隨著程式執行而有所變化，所以你可以一步一步執行並觀察變化，一旦你厭煩了重複按 Step 按鈕後，只需去除中斷點並在除錯器上按 Continue 按鈕就能夠重新執行此程式。

現在讓我們以另一個方法來測試，將 Demo 類別中的 carTest 方法設定一個中斷點。

```
places = myCar.seats();
```

然後呼叫這個方法，當程式執行到中斷點時，程式正要呼叫包含在 Car 類別中的 seats() 方法，如果這時候選的是 Step 按鈕，將會跑完在 Car 類別裡的 seat() 方法，但是你卻不清楚在 seat() 方法裡是如何運作的，因為畫面顯示的只是執行過那行程式。如果你是選 Step Into 按鈕的話，則會另外出現屬於 Car 類別的除錯器，此時你就能夠更清楚的瞭解到整個程式是如何運作的，你就能夠一步一步的在 seats() 方法裡運行程式，直到到達 seat() 方法的結尾時，才會回到 Demo 主類別。如果除錯的該行程式沒有包含一個呼叫方法的話，Step 與 Step Into 運作起來是相同的！

6.3 監看變數

摘要：監看變數是很容易的，因為變數會自動出現在除錯器中。

當你在除錯時，隨時瞭解你目前物件的區域變數與物件變數的狀態的很重要的，做這件事情是很容易的，你不需要使用特別的指令就能達成查詢變數的狀態，包括目前物件的靜態變數與物件變數，以及目前方法的區域變數都會自動的產生呈現在你眼前。你也可以選擇在呼叫序列中的其他方法來查詢目前其他使用中物件、方法的變數值為何。例如同樣的在 carTest() 方法中設定中斷點，當運作時你就可以發現在除錯器的左邊的 Call Sequence 有兩個呼叫序列。

Car.seats

Demo.carTest

這說明了 *Demo.carTest* 呼叫 *Car.seats*，你可以選擇 *Demo.carTest* 去觀察原始與目前的變數值。如果你逐行執行到包含 *new Car(...)* 的這行程式，你可以觀察到區域變數 *myCar* 的值會以 *<object reference>* 的方式呈現，所有物件類別的值都是以這種方式呈現，除了字串之外。你可以用雙點滑鼠來觀察變數。

6.4 暫停與終止

摘要：暫停和終止可以用來停止正在執行程式暫時地或永久地。

有時候程式運作很久還沒有回應，你就會懷疑說是不是某個部分出了問題，或許是程式陷入了一個無限迴圈之中，或許程式只是需要長一點的時間去運作，所以我們需要去檢查！例如開啓 *Demo* 類別裡的 *longloop()* 方法，這個方法就會需要運作一段時間。現在我們需要知道程式運作得如何了？此時打開除錯器，按 *Halt* 按鈕，程式就會自動停止，效果就如同中斷點一般，你現在就可以一步一步的觀察每個步驟、變數，看看一切是否沒有問題，你也可以重複使用 *Continue* 與 *Halt* 許多次，來試看看程式能夠計算的多快，如果你不想繼續下去，例如你發現程式已經陷入了無限迴圈之中，你可以按 *Terminate* 去終止整個程式，*Terminate* 應該要盡量避免使用，因為使用 *Terminate* 可能會導致原本撰寫完美的程式產生前後不一致的狀況，所以我建議在緊急情況時才使用 *Terminate*。

7 建立獨立運作的應用程式

摘要：要建立獨立運作的應用程式，選擇功能表 Project 裡的 Export。

BlueJ 可以建立一個可執行的 jar 檔案，可執行的 jar 檔案可以用雙點滑鼠開啓，或是運用特殊指令開啓，例如 `java -jar <file-name>.jar`。

現在就舉一個 *Hello* 的實做範例來說明，開啓在 *example* 目錄下的 *hello* 專案，確定此專案已經被編譯完成，這時候選擇功能表 *Project* 裡的 *Export*。一個對話視窗就會出現讓你選擇儲存格式(如圖 15 所示)，選擇 *jar file* 以建立可執行的 jar 檔案，爲了讓 jar 檔案是可執行的，這時候需要指定一個主要的類別，這個主要類別有著合法的主要方法定義 (Main Method)，有著 `public static void main(String[] args)` 特徵。

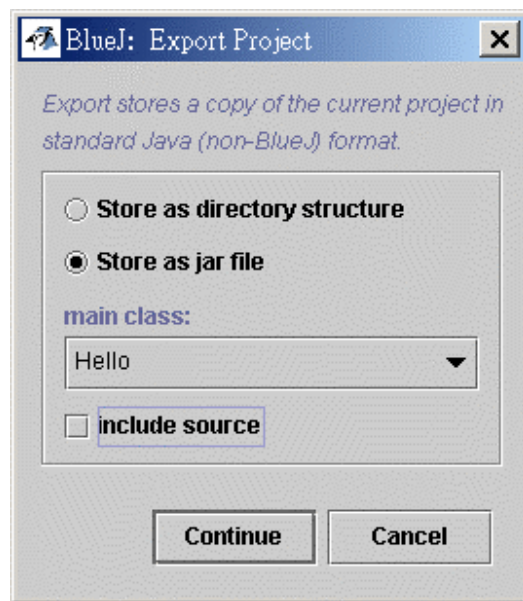


圖 15：匯出對話視窗

在這個例子中，選擇主要類別是相當容易的，因爲我們只有一個類別，所以就選擇 *Hello* 吧！如果同時還有別的專案，就選擇有 `main` 方法的類別即可。

通常是不需要在可執行檔中包含原始碼的，但是如果你願意也可附上，接著按 *Continue* 按鈕，會跳出一個對話視窗要你選擇 jar 檔案的名稱與存放位置，這時候打 *Hello*，按 *OK* 後一個可執行的 jar 檔就建立完成了！

如果應用程式是一個 GUI 介面，你可以用滑鼠雙點來開啓他，因爲我們的範

例是使用文字 I/O，所以我們就從文字終端機來啓動他！

要怎麼啓動呢？打開 DOS Windows，到你存放剛剛指定 jar 的目錄下，會看見 Hello.jar 這個檔案，假設你的 Java 系統已經正確的安裝完成在你的電腦上，這時候你就可以鍵入

```
java -jar hello.jar
```

程式應該就可以順利運作囉～

8 建立 Applets

8.1 執行 applet

摘要：要執行 applet，就從 applet 彈出的選單中選擇 Run Applet。

BlueJ 允許建立並執行 applet 就如同應用程式一般，在 *example* 目錄裡就包括一個 applet。首先，要去執行一個 applet 就先去打開 *example* 目錄下的 *appletdemo*。

你會看到這個專案只有一個類別— *CaseConverter*，這個類別的圖示有被註記為 <<applet>>，在經過編譯之後，在圖示上按右鍵會彈出選單，選擇 *Run Applet*，一個對話視窗就會顯示，如同圖 16 所示。

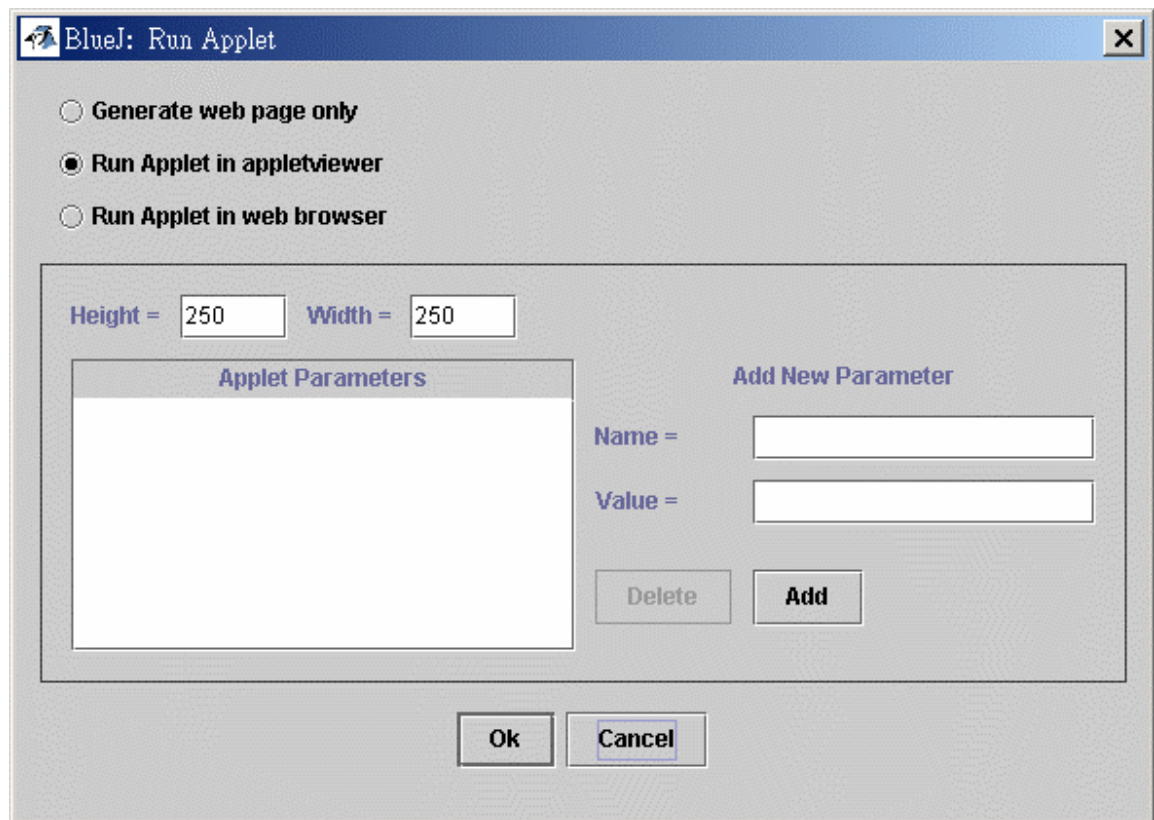


圖 16：執行 applet 對話視窗

Applet 瀏覽器 (Applet Viewer) 已經和你的 JDK 一起裝在電腦中，而且版本是與 Java 編譯器一致的，因為這樣子比較不會有版本上的問題，不像網頁瀏覽器在跑不同版本的 Java 時都是依據網頁瀏覽器內定的版本而比較會產生問題！

在 Microsoft Windows 和 MacOS 系統上，BlueJ 會使用你預設的瀏覽器，在 Unix 系統則會將瀏覽器定義為 BlueJ 的設定。

8.2 建立 applet

摘要：要建立 applet，選擇 New Class 按鈕，類別型態選 Applet 即可。

在看過如何執行 applet 之後，現在就來建立自己的 applet 吧！

建立一個以 Applet 類別型態的 Class，然後編譯他，接著執行他！看！就是這麼簡單！很容易吧～

Applet 產生時就已經包含了基本的類別架構以及一些合法的程式碼，這些程式碼只用了兩行就能夠顯示出一個 applet，你現在可以開啓編輯器並加入你自己的程式碼。

在這個範例裡面你可以看到許多常用的 applet 方法，每一個方法都加註了註解來解釋其目的，所有的範例程式碼都在 *Paint* 方法中。

8.3 測試 applets

在某些情況下經由物件區來建立 applet 是相當方便的，applet 的 *constructor* 就顯示在 applet 的彈出選單中，但是從物件區中你無法執行整個 applet，你可以呼叫一些方法來達成。這用來測試你 applet 中已經寫好的單一方法是很有用的！

9 其他操作

9.1 在 BlueJ 開啓非 BlueJ 的 Packages

摘要：非 BlueJ 的 packages 可以用功能表 Project 裡的 Open Non BlueJ 來開啓。

BlueJ 可以讓你開啓其他非 BlueJ 所製作的 packages，只要選擇功能表 Project 裡的 *Open Non BlueJ* 就能開啓，之後選擇包含 Java 來源檔的目錄，按 *Open in BlueJ* 即可。

9.2 增加已存在的類別到專案中

摘要：使用 Add Class from File 就可以將類別從外部複製到專案中。

通常你會在你自己的 BlueJ 專案中使用到從外面得到的類別，例如老師可能會將一個 Java 類別給學生，學生選擇功能表中 *Edit* 裡的 *Add Class from File* 就能夠很容易的將老師給的類別整合到自己的專案中，這樣子你可以選擇一個 Java 來源檔（附檔名為.java）來匯入。

當類別已經被匯入專案之後，類別將會自動備份到目前專案的目錄之下，效果就如同你自己用程式碼建立類別一樣。

另一種方法是直接將來源檔加入到你的專案目錄下，下次啓動專案時，此類別就會自動包含在專案裡。

9.3 呼叫主要或其他靜態方法

摘要：靜態方法可以從類別的彈出選單呼叫。

從 *example* 目錄裡開啓 *hello* 專案，這個專案只有一個類別（Hello），此類別定義一個標準的主要方法。

用滑鼠右鍵點選此類別，你將會從選單上看到類別建構子和靜態主要方法，你可以從選單中直接呼叫主要方法，所有的靜態方法都可以用這種方式呼叫。標準的主要方法需要一個字串陣列來當作參數，你也可以使用標準 Java 語法組成陣列常數來傳遞字串陣列，例如你可以傳遞

```
{"one", "two", "three"}
```

(包括括號) 傳遞給方法。試看看！

註解：在標準 Java 中，陣列常數不可以用來當作呼叫方法時的實際參數，只能夠用在初始值，但是在 BlueJ 中，為了使標準主要方法能夠互相運用，這是允許的。

9.4 產生文件

摘要：要產生文件，從工具列 Tool 中選擇 Project Documentation 即可。

你可以從 BlueJ 產生出符合標準 Java 文件格式的專案文件，只要選取工具列中 Tool 裡的 *Project Documentation* 即可，這個功能將會從類別的來源程式碼產生出關於專案內所有類別的文件，並開啓網頁瀏覽器方式呈現。

你也可以在 BlueJ 的編輯器中只產生並檢視單一類別的文件，只要打開編輯器並使用編輯器功能表中的下拉式選單，將 *Implementation* 改成 *Interface* 即可，這將會在編輯器上顯示出 Java 文件風格的文件模式。

9.5 與查詢系統一同作業

摘要：Java 標準類別的 API 可以用選擇功能表 Help 中的 Java Standard Libraries 來檢視。

通常當你寫 Java 程式時，會參照 Java 標準的 library，當你在連線時，你可以選擇功能表 Help 中的 *Java Standard Libraries* 來開啓網頁瀏覽器顯示出 JDK API 的文件。在安裝完成 JDK 文件後，你也可以在離線狀態使用，詳細細節會在 BlueJ 網站的求助網頁解釋。

9.6 從 Library 類別建立物件

摘要：要從 library 類別建立物件，就選擇功能表中 Tools 裡的 Use Library Class。

BlueJ 也提供了從其他專案的類別建立物件的功能，但是必須定義在 library 裡，例如你可以建立字串或陣列類別的物件，當你在快速實驗這些 library 物件時是非常有用的，你也可以用選擇功能表中 Tools 裡的 *Use Library Class* 來建立 library

物件。一個對話視窗會提示你輸入完整合法的類別名稱，例如 `java.lang.String`。（注意：你必須輸入完整合法名稱，這個名稱包括了包含此類別的 `package` 名稱）

文字輸入欄位可下拉得到最近使用過的類別名稱，一旦有類別名稱輸入，按下 *Enter*，就會顯示出在欄位內類別的所有建構子與靜態方法，任何建構子或靜態方法此時就可以經由從清單中選取而匯入。

10 摘要

基礎篇

01. 開啓一個專案，點選 *Project menu* 中的 *Open*。
02. 要建立一個物件，從類別功能表中選擇一個建構函數。
03. 要執行一個方法，從物件功能表中選擇它。
04. 要編輯類別的原始碼，只要雙擊類別圖示即可。
05. 要編譯一個類別，點一下編輯器裏編譯按鈕。要編譯一個專案，點一下專案視窗裏的編譯按鈕。
06. 要得到編譯錯誤訊息的幫助，點一下錯誤資訊旁邊的問號圖示。

進階篇

07. 利用顯示物件的內部狀態，物件檢閱可以作一些簡單的除錯。
08. 透過點擊一個物件的圖示可以把一個物件作為參數傳給另一個方法使用。

建立新專案

09. 要建立專案，就選擇 *Project* 功能表中的 *New Project*。
10. 要建立類別，就選擇 *New Class* 按鈕並設定類別名稱。
11. 要建立關連，可以用選擇箭頭按鈕並以圖形方式畫出，或是在編輯器（*Editor*）中撰寫程式碼達成。
12. 要移除類別，只需在類別上按右鍵選取功能表中的 *Remove* 則可。
13. 要移除箭頭，則需先選取功能列上的 *Edit* 中的 *Remove Arrow* 後，再選取欲移除的箭頭即可。

除錯

14. 要設定中斷點，只需在編輯器左方的中斷區域點選即可。
15. 要一步一步執行程式，就使用除錯器（*debugger*）中的 *Step* 和 *Step Into*。
16. 查詢變數是很容易的，因為變數會自動出現在除錯器中。
17. 暫停和終止可以用來停止正在執行程式暫時地或永久地。

建立獨立運作的應用程式

18. 要建立獨立運作的應用程式，選擇功能表 *Project* 裡的 *Export*。

建立 applets

19. 要執行 applet，就從 applet 彈出的選單中選擇 *Run Applet*。

20. 要建立 applet，選擇 *New Class* 按鈕，類別型態選 *Applet* 即可。

其他操作

21. 非 BlueJ 的 packages 可以用功能表 *Project* 裡的 *Open Non BlueJ* 來開啓。
22. 使用 *Add Class from File* 就可以將類別從外部複製到專案中。
23. 靜態方法可以從類別的彈出選單呼叫。
24. 要產生文件，從工具列 *Tool* 中選擇 *Project Documentation* 即可。
25. 要檢視 Java 標準類別的 API 可以用選擇功能表 *Help* 中的 *Java Standard Libraries*。
26. 要從 library 類別建立物件，就選擇功能表中 *Tools* 裡的 *Use Library Class*。