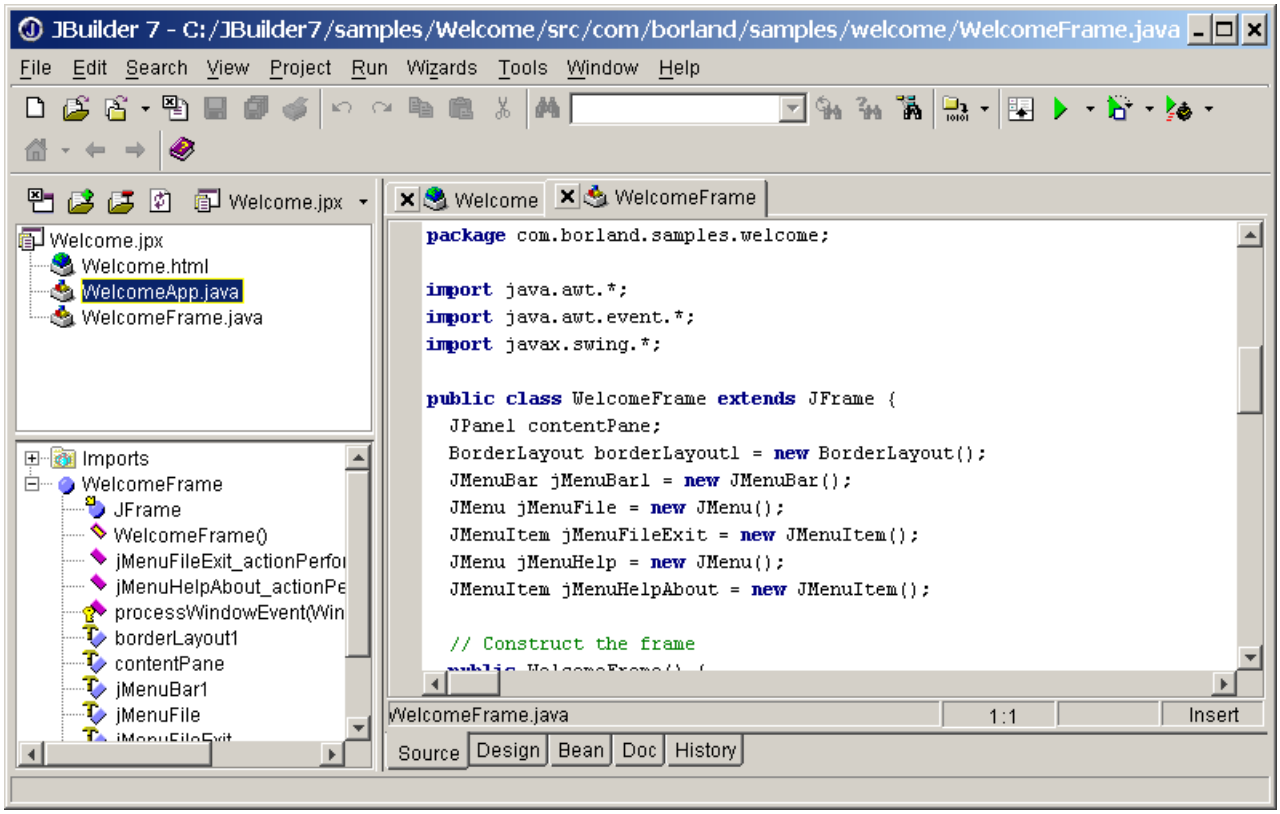
CHAPTER

2

Getting Started with JBuilder

Assume you have successfully installed JBuilder on your machine. Start JBuilder from Windows, Solaris, Linux, or Mac. The main JBuilder user interface appears, as shown in Figure 2.1. If you don't see the Welcome project, choose Welcome Project (Sample) from the Help menu.

> NOTE: The screen shots in this book are from JBuilder 7 Personal. *If you use JBuilder 7 Standard or JBuilder 7 Enterprise, your screen may look slightly different*. For example, Figure 2.2 shows the screen shot for JBuilder Enterprise.



******PD: Please leave some space on top and bottom for labels. Labels will be provided at Proof stage. Author.***

Figure 2.1

The JBuilder user interface is a single window that performs functions for editing, compiling, debugging, and running programs.

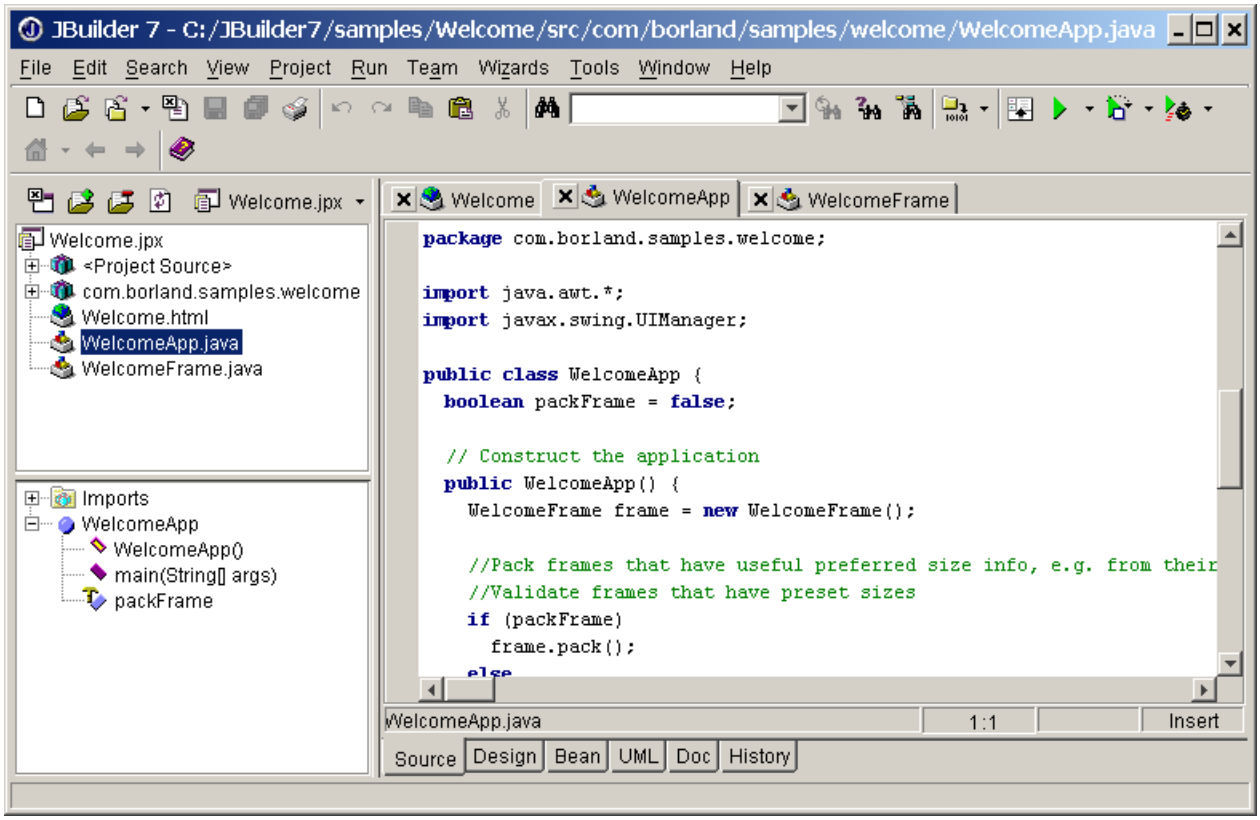


Figure 2.2

The screen of JBuilder Enterprise looks slightly different from JBuilder Personal.

The JBuilder user interface presents a single window for managing Java projects, browsing files, compiling, running, and debugging programs. This user interface is called *AppBrowser* since JBuilder 3.5. Note that the AppBrowser window and the main window are two separate windows in JBuilder 3, but are combined into one window since JBuilder 3.5.

Traditional IDE tools use many windows to accommodate various development tasks, such as editing, debugging, and browsing information. As a result, finding the window you need is often difficult. Because it is easy to get lost, beginners may be intimidated. For this reason, some new programmers prefer to use separate utilities, such as the JDK command-line tools, for developing programs.

Borland is aware of the usability problem and has made a significant effort to simply the JBuilder user interface.

JBuilder introduces the AppBrowser window, which enables you to explore, edit, design, and debug projects all in one unified window.

The AppBrowser window primarily consists of the main menu, main toolbar, status bar, project pane, structure pane, and content pane.

2.1 The Main Menu

The main menu is similar to that of other Windows applications and provides most of the commands you need to use JBuilder, including those for creating, editing, compiling, running, and debugging programs. The menu items are enabled and disabled in response to the current context.

2.2 The Toolbar

The toolbar provides buttons for several frequently used commands on the menu bar. Clicking a toolbar is faster than using the menu bar. For some commands, you also can use function keys or keyboard shortcuts. For example, you can save a file in three ways:

- Select File, Save from the menu bar.
- Click the "save" toolbar button.
- Use the keyboard shortcut Ctrl+S.

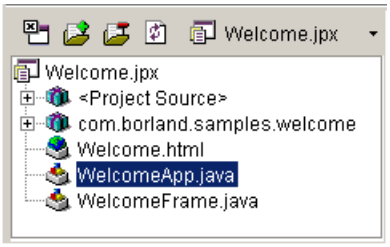
TIP: You can display a label, known as *ToolTip*, for a button by pointing the mouse to the button without clicking.

2.3 The Status Bar

The status bar displays a message that alerts the user to the operation status, such as file saved for the Save file command and compilation successful for the Compilation command.

2.4 The Project Pane

The *project pane* (known as the *navigation pane* in JBuilder 3) displays the contents of one or more projects opened in the AppBrowser. It consists of the following items, as shown in Figure 2.3.



PD: Need several labels for this figure. AU

Figure 2.3

The project pane manages JBuilder projects.

- A small toolbar with four buttons (Close Project, Add To Project, Remove From Project, and Refresh).
    - A drop-down list of all opened projects.
- A tree view of all the files that make up the active project.

The project pane shows a list of one or more files. The project (.jpx) file appears first. Attached to that is a list of the files in the project. The list can include .java, .html, text, or image files. You select a file in the project pane by clicking it. The content pane and the structure pane display information about the selected file. As you select different files in the project pane, each one will be represented in the content and structure panes. The project pane shown in Figure 2.3 contains three files. The Add button is used to add new files to the project, and the Remove button to remove files from the project. For example, you can remove Welcome.html by selecting the file in the project pane and clicking the Remove button. You can then add the file back to the project as follows:

1. Click the Add button to display the Open dialog box shown in Figure 2.4.

2. Open Welcome.html. You will see Welcome.html displayed in the project pane.

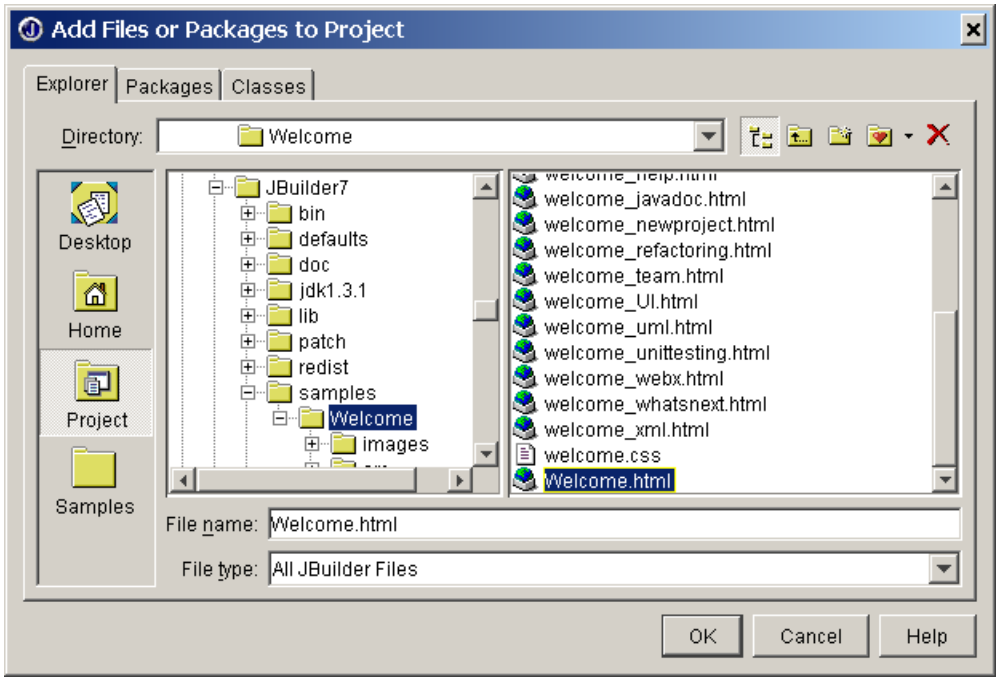


Figure 2.4

The Open dialog box enables you to open an existing file.

> TIP: You can select multiple files by clicking the files with the CTRL key pressed, or select consecutive files with the SHIFT key pressed.

2.5 The Content Pane

The content pane displays all the opened files as a set of tabs. To open a file in the content pane, double-click it in the project pane. The content pane displays the detailed content of the selected file. The editor or viewer used is determined by the file's extension. If you click the WelcomeApp.java file in the project pane, for example, you will see six tabs (Source, Design, Bean, UML, Doc, and History) at the bottom of the content pane (see Figures 1.1). If you select the Source tab, you will see the JBuilder Java source code editor. This is a full-featured, syntax-highlighted programming editor.

If you select Welcome.html in the project pane, you will see the content pane become an HTML browser, as shown in Figure 2.5. If you choose the Source tab, you can view and edit the HTML code in the content pane, as shown in Figure 2.6.

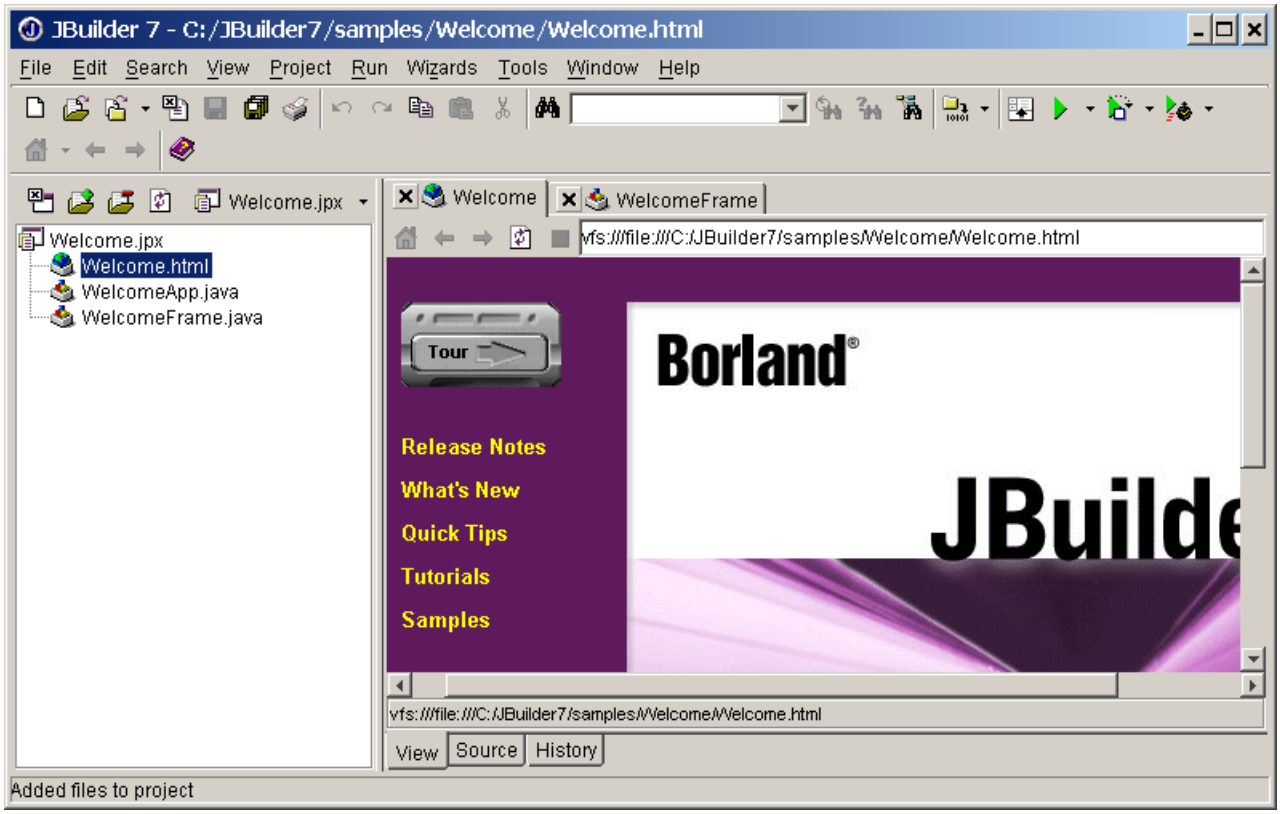


Figure 2.5

JBuilder renders HTML files in the content pane.

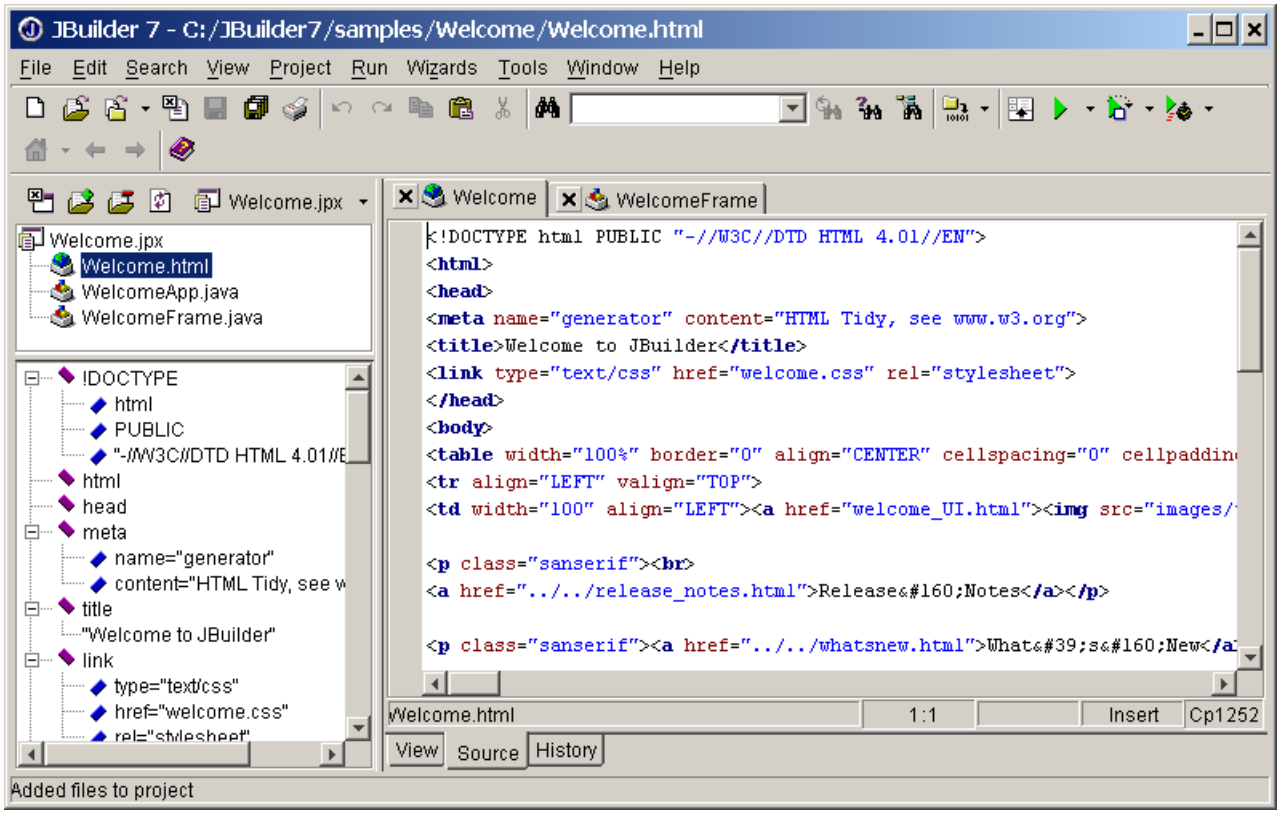


Figure 2.6

You can edit HTML files in the content pane.

2.6 The Structure Pane

The *structure pane* displays the structural information about the files you selected in the project pane. All the items displayed in the structure pane are in the form of a hierarchical indexed list. The expand symbol in front of an item indicates that it contains subitems. You can see the subitems by clicking on the expand symbol.

You can also use the structure pane as a quick navigational tool to the various structural elements in the file. If you select the WelcomeFrame.java file, for example, you will see classes, variables, and methods in the structure pane. If you then click on any of those elements in the structure pane, the content pane will move to and highlight it in the source code.

If you click on the jMenuFile item in the structure pane, as shown in Figure 2.7, the content pane moves to and highlights the statement that defines the jMenuFile data field. This provides a much faster way to browse and find the elements of a file than scrolling through it.

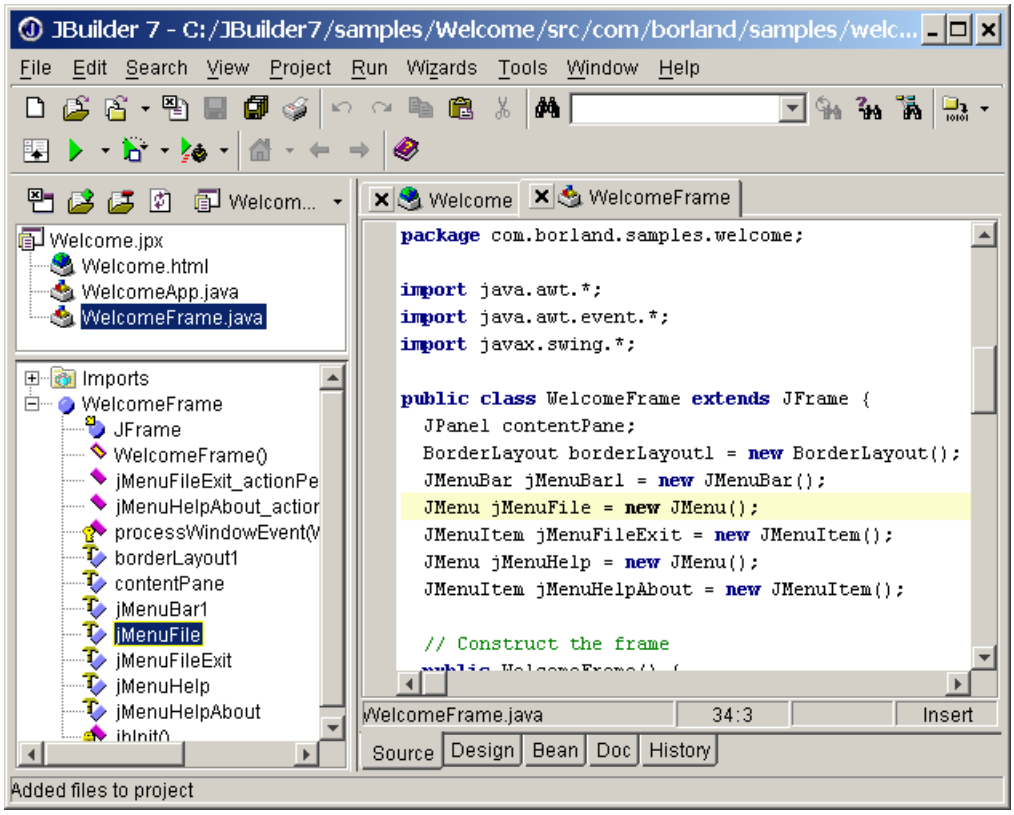


Figure 2.7

You can cruise through the source code from the structure pane.

CHAPTER

3

Creating and Managing Projects

A project is like a holder that ties all the files together. The information about each JBuilder project is stored in a project file with a .jpx file extension. (Prior to JBuilder 7, the project file extension is either .jpx or .jpr. The .jpr project files are still compatible in JBuilder 7.) The project file contains a list of all the files and project settings and properties. JBuilder uses this information to load and save all the files in the project and compile and run the programs.

3.1 Creating a Project

To avoid frustrating mistakes, it is important to create project in a consistent and uniformed way for all your programs. I recommend new users to create a project as follows:

1. Choose *File*, *New Project* to bring up the Project wizard dialog box, as shown in Figure 3.1.

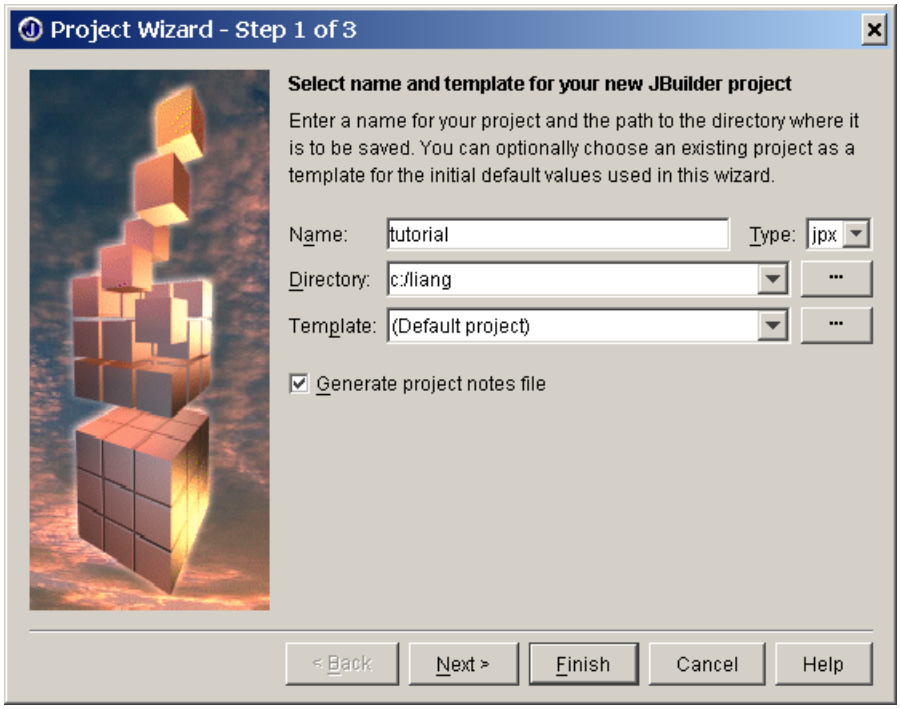


Figure 3.1

The Project wizard dialog box enables you to specify the project file with other optional information.

2. Type tutorial in the Name field and c:/liang in the Directory field. Check the *Generate project notes file* option box. Click *Next* to display Project Wizard - Step 2 of 3, as shown in Figure 3.2.

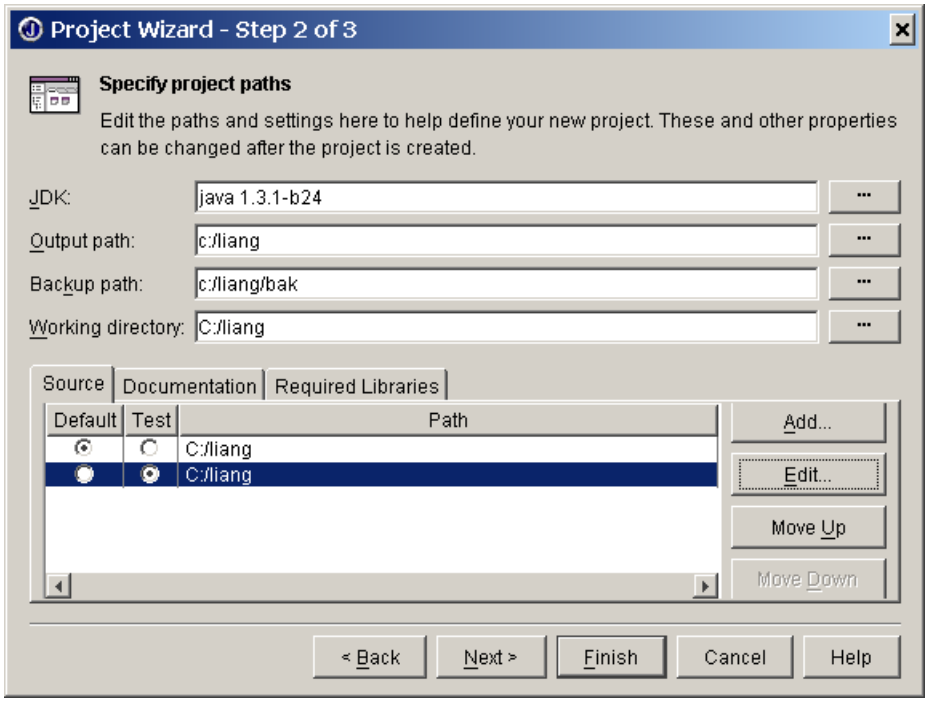


Figure 3.2

The Project wizard Step 2 of 3 enables you to modify project settings.

3. Type c:/liang in the Output path field, c:/liang/bak in the Backup path field, and c:/liang in the Working directory field. In the Source tab, change the Default path and Test path to c:/liang. To do so, click the Edit button to display the Select Directory dialog box, as shown in Figure 3.3. Type c:/liang in the Directory field. If the directory does not exist, you will see the No Such Directory dialog box, as shown in Figure 3.4. Click *Yes* to return to Project Wizard - Step 2 of 3.

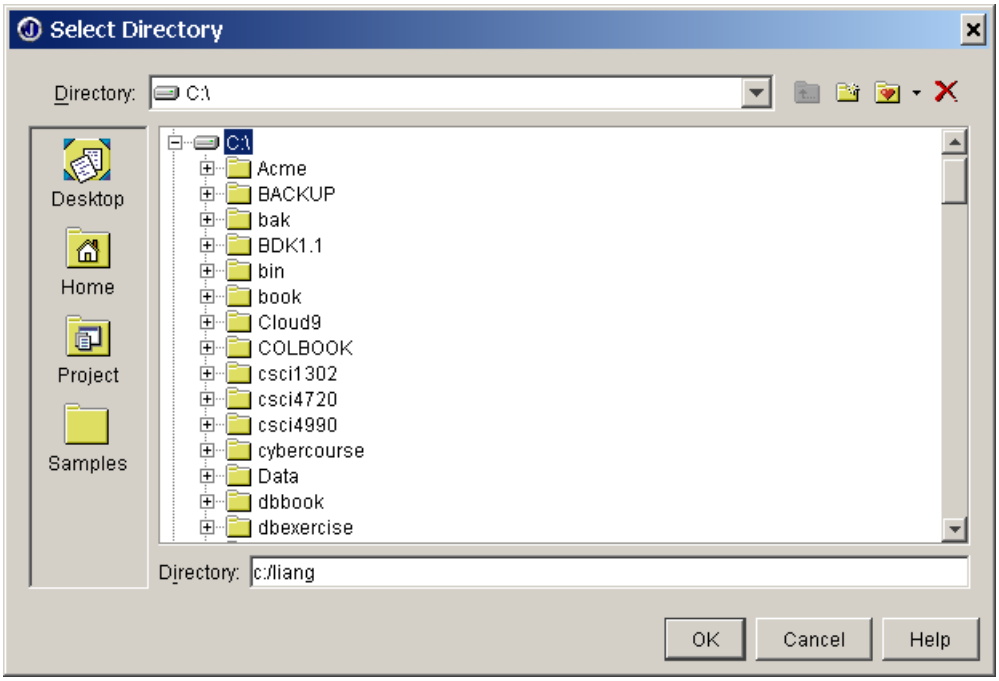


Figure 3.3

You can select or specify a directory in the Select Directory dialog box.

Figure 3.4

You can select or specify a directory in the Select Directory dialog box.

4. Click Next in Project Wizard - Step 2 of 3 to display Project Wizard - Step 3 of 3, as shown in Figure 3.5. Fill in the title, author, company, and description fields. These optional fields provide a description for the project.

5. Click Finish. The new project is displayed in the project pane, as shown in Figure 3.6. The Project wizard created the project file (tutorial.jpx) and an HTML file (tutorial.html), and placed them in c:\liang. The project file stores the information

about the project, and the HTML file is used to describe the project.

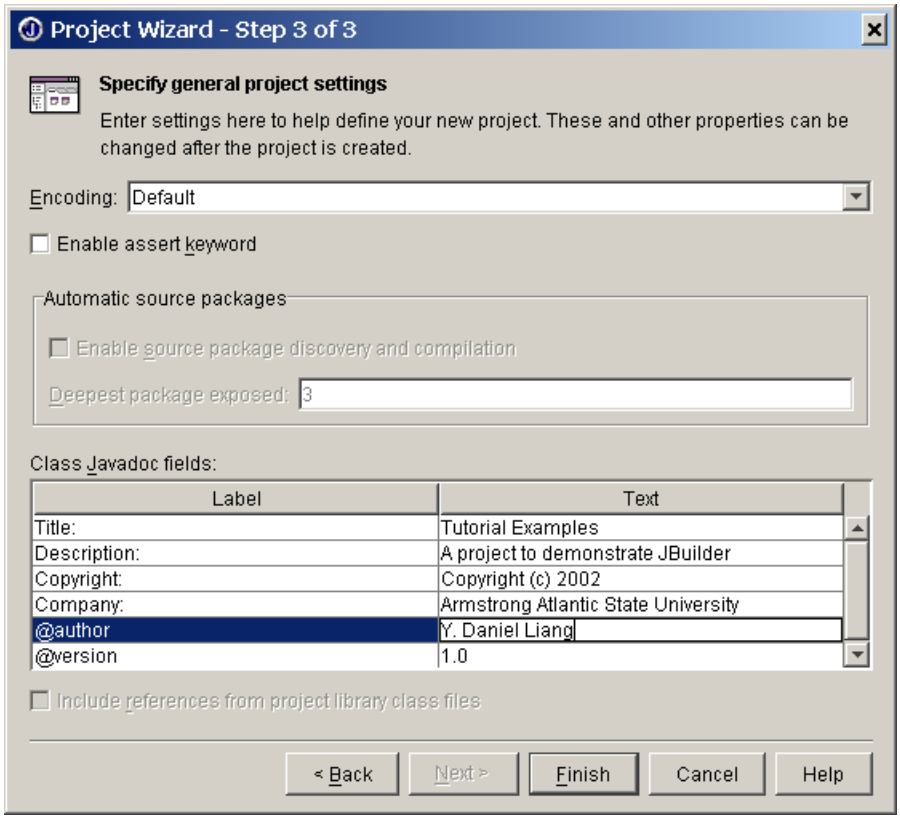


Figure 3.5

The Project wizard Step 3 of 3 collects optional information for the project.

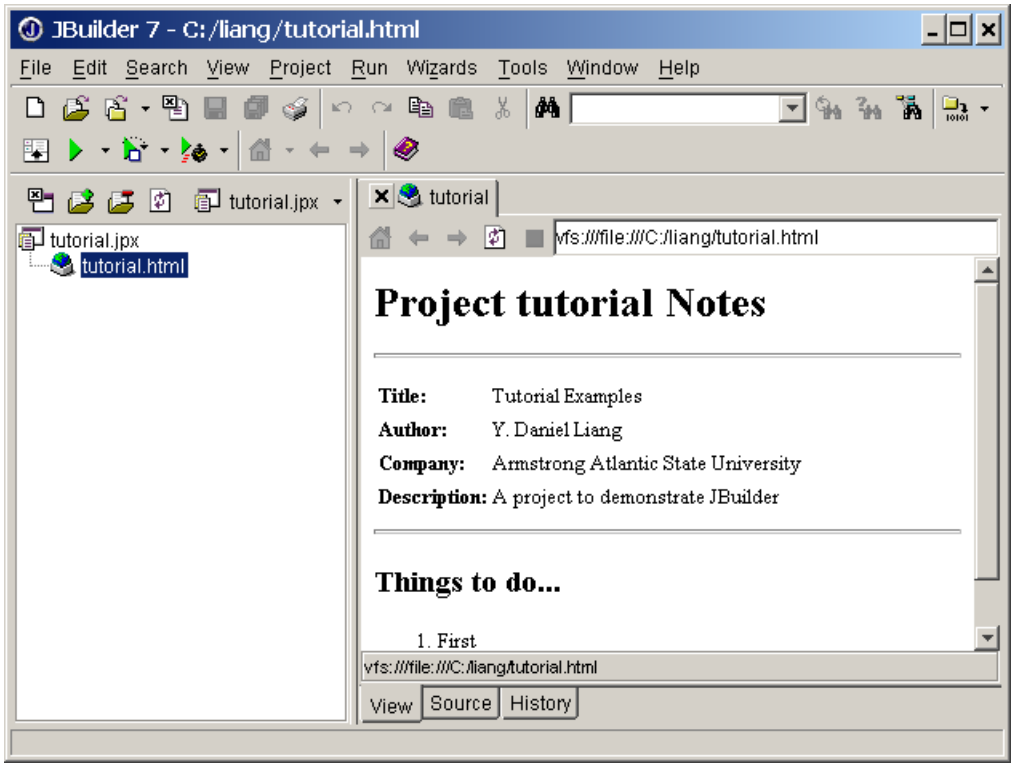


Figure 3.6

A new project is created with the .jpx file and .html file.

> NOTE: JBuilder automatically generates many backup files. I use **bak** as the root directory for all these backup files so they can be easily located and removed.
>
> CAUTION: Creating a project is a preliminary step before developing Java programs. Creating projects incorrectly is a common problem for new JBuilder users, and can lead to frustrating mistakes. You may create your project exactly as shown in this section and change *liang* to your name.

3.2 Managing Projects in JBuilder (Optional)

JBuilder uses a project file to store project information. You cannot edit the project file manually; it is modified automatically, however, whenever you add or remove files from the project or set project options. You can see the project file as a node at the top of the project tree in the project pane (see Figure 3.6). JBuilder uses a Project Properties dialog box for setting project properties and provides a Project wizard to facilitate creating projects.

> NOTE: The project properties can be modified in the Project Properties dialog box after a

project file is created. However, there is no need to change any properties for using this book if you have set the path properties correctly in the Project wizard in JBuilder 7. That is why this section is marked optional.

3.2.1 Setting Project Properties

JBuilder uses the Default Project Properties dialog box to set default environment properties for all the projects. An individual project has its own Project Properties dialog box, which can be used to set project-specific properties. To display the Default Project Properties dialog box, select Project, Default Properties, as shown in Figure 3.7. To display the Project Properties dialog box, select Project, Properties, as shown in Figure 3.8. You can also right-click the project file in the project pane and choose the Properties command to display the Project Properties dialog box.

The Default Project Properties dialog box (Figure 3.7) and the Project Properties dialog box (Figure 3.8) look the same but have different titles. Both dialog boxes contain Paths, General, Run, Debug, and Code Style. The Team tab is only available in JBuilder Professional and Enterprise. You can set options in these pages for the current project or the default project, depending on whether the dialog box is for the current project or for the default project. JBuilder Professional and JBuilder Enterprise Edition have more options in the Project Properties dialog box.

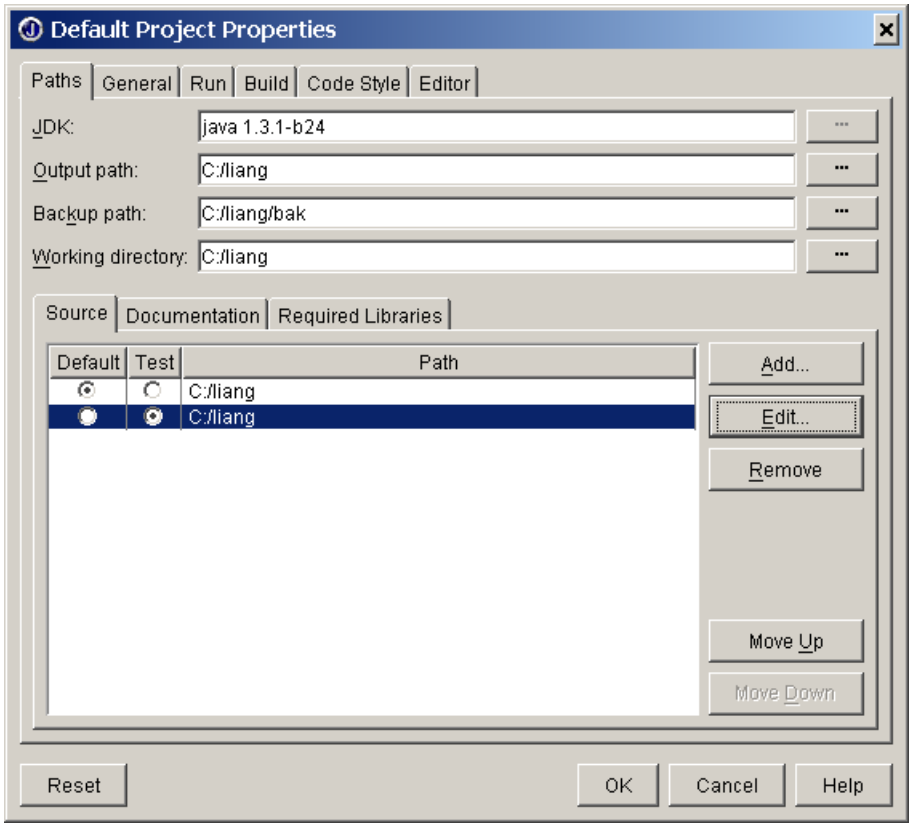


Figure 3.7

The Default Project Properties dialog box enables you to set default properties to cover all the projects.

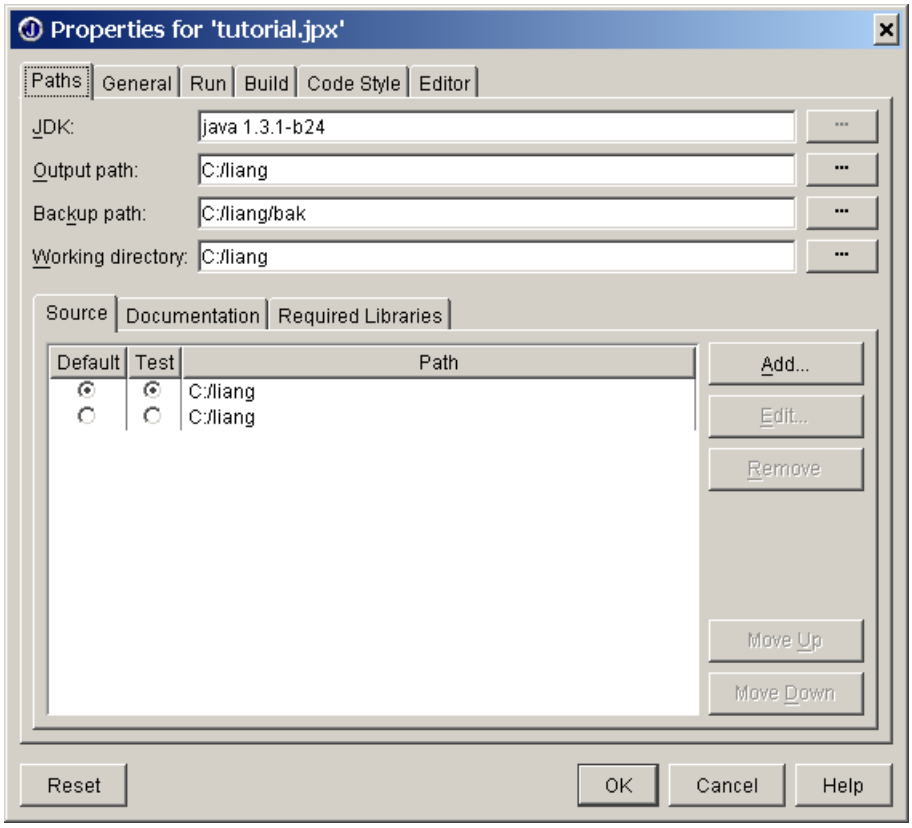


Figure 3.8

Each project keeps its own project properties.

3.2.2 The Paths Page

The Paths page of the Project Properties dialog box sets the following options:

[BL] JDK version.

[BL] Output path where the compilation output is stored.

[BL] Backup path where the backup files are stored.

[BL] Working directory where the temporary files are stored.

[BL] Source page where the source files are stored.

[BL] Documentation page where the source files are stored.

[BL] Required Libraries page where the libraries are searched for compiling and running.

JBuilder can compile and run against JDK 1.2, JDK 1.3, or JDK 1.4. To set up the list of available JDKs, click the Ellipsis button to display the Available JDK Versions dialog box for adding new JDK compilers. This feature is available

only in JBuilder 7 Standard and Enterprise.

Setting proper paths is necessary for JBuilder to locate the associated files in the right directory for compiling and running the programs in the project. The Source page specifies one or more paths for the source file. The Output path specifies the path where the compiler places the .class files. Files are placed in a directory tree that is based on the Output path and the package name. For example, if the Output path is **c:\liang** and the package name is **tutorial** in the source code, the .class file is placed in **c:\liang\tutorial**. The Backup path specifies a directory where the backup source files are stored. JBuilder automatically backs up the files before committing any change to the current file. Note that the .java files are backed up in the backup directory, but the other files, like .html and .jpx, are backed up in their original directory. All the standard JDK libraries are already preinstalled. If you need to add the custom libraries to the project, click the *Add* button to display a list of available libraries. To remove a library, click the *Remove* button. To switch the order of libraries, use the *Move Up* and *Move Down* buttons.

3.2.3 The General Page

The General page (Figure 3.9) allows you to specify an encoding scheme used in JBuilder, enable or disable automatic source package discovery, and modify javadoc fields.

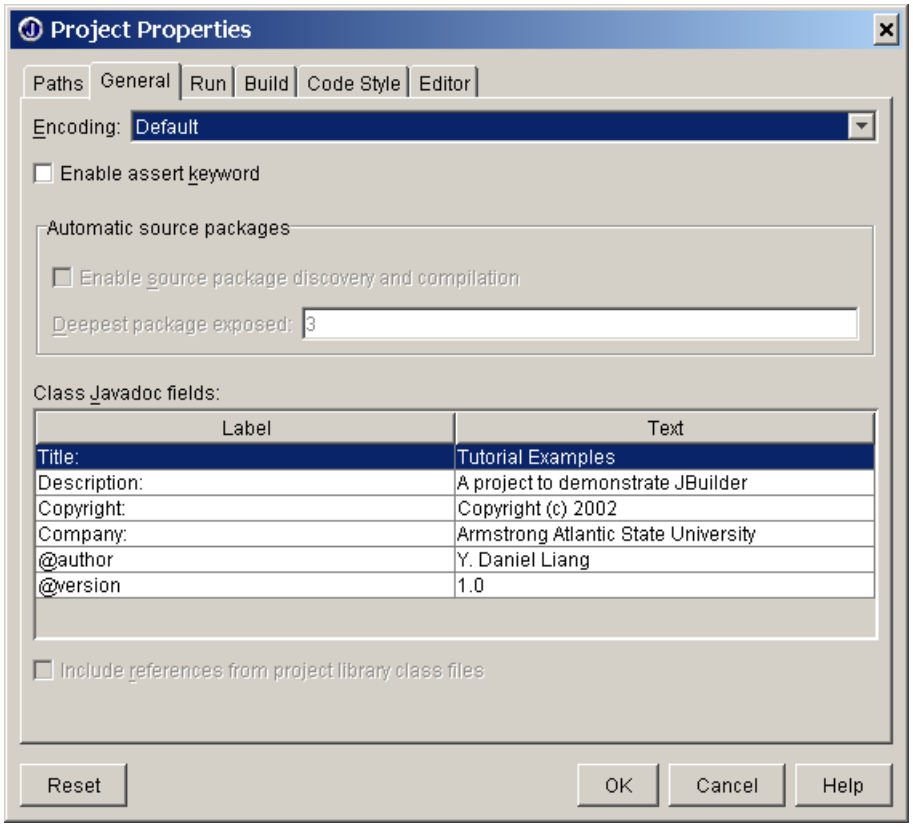


Figure 3.9

The General page has options for selecting encoding type, enabling/disabling automatic source package discovery, and modifying javadoc fields.

The "Encoding" choice menu specifies the encoding that controls how the compiler interprets characters beyond the ASCII character set. If no setting is specified, the default native-encoding converter for the platform is used.
You can enable or disable the automatic source package. This is a useful feature available only in JBuilder Standard and Enterprise. With automatic source package enabled, all the packages in the project's source path automatically appear in the project pane, so you can navigate through the files without switching projects.

The Class Javadoc fields section specifies the javadoc tags generated in the class files when Generate Header Comments is enabled in a wizard.

3.2.4 The Run and Build Page

The Run page sets runtime configuration. The Build page (Figure 3.10) sets compiler options. The options are applied to all the files in the project, as well as to files referenced by these files, stopping at packages that are marked "stable."

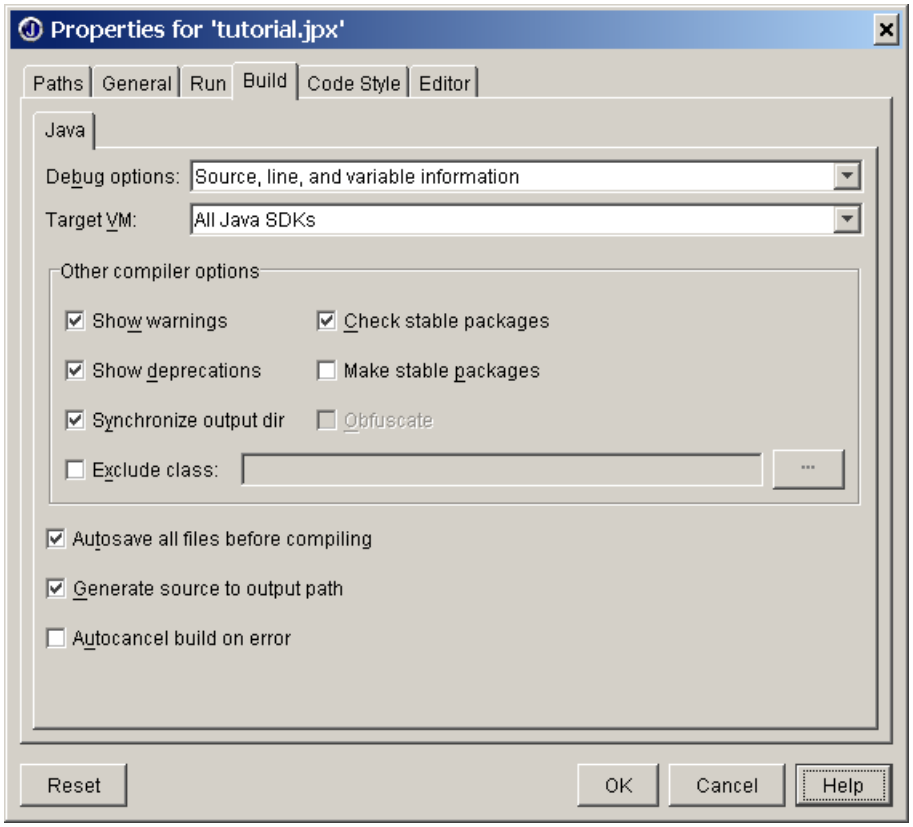


Figure 3.10

The Build page sets compiler options.

The option "Include debug info" includes symbolic debug information in the .class file when you compile, make, or rebuild a node. The option "Show warnings" displays compiler warning messages. The option "Show deprecations" displays all deprecated classes, methods, properties, events, and variables used in the API. The option "Synchronize output dir" deletes class files on the output path for which you do not have source files before compiling.

The Option "Check stable packages" checks files in the packages marked "stable" to see whether they and their imported classes need to be recompiled. This option shortens the edit/recompile cycle by not rechecking stable packages.

The option "Make stable package" checks all the classes of a package on the first build and marks the package "stable." If this option is unchecked, only the referenced classes of this package will be compiled, and the package will not be marked "stable." This option should be unchecked when working with partial projects. It is especially useful for working with a library of classes with no source code.

The option "Obfuscate" makes your programs less vulnerable to decompiling. Decompiling means to translate the Java bytecode to Java source code. After decompiling your obfuscated code, the generated source code contains altered symbol names for private symbols. This feature is available only in JBuilder Professional and Enterprise.

The "Exclude Class" choice lets you specify that a .class file is excluded from being compiled. This feature is available only in JBuilder Standard and Enterprise.

The option "Autosave all files before compiling" automatically saves all files in the project before each compile.

The option "Generate source to output path" is applicable only to RMI (Remote Method Invocation) and IDL (Interface Definition Language) files. These files are used in multi-tier Java applications.

3.2.5 The Code Style Page

The Code Style page (Figure 3.11) enables you to specify the code style of the program generated by JBuilder.

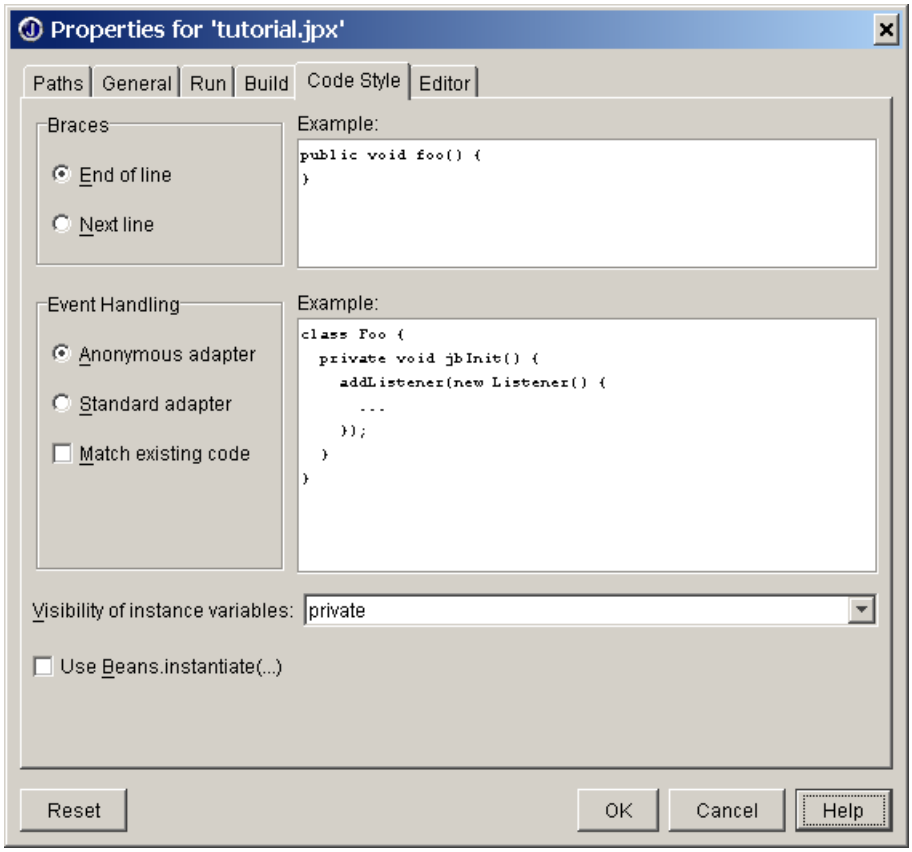


Figure 3.11

The Code Style page of the Project Properties dialog box sets the options for the code style of the program generated by JBuilder.

With the option "End of line" selected, JBuilder will generate the code with opening braces inserted at the end of the line; otherwise, the opening braces are inserted at the beginning of the new line.

In the Event Handling section, you can choose one of the options to tell JBuilder to generate event-handling code using an anonymous adapter, a standard adapter, or matching the existing style of event-handling code. The adapters are used in the UI designer in JBuilder. For more information on adapters, please refer to Chapter ?, "UI Designer.''

The option "Use Beans.instantiate" tells JBuilder to instantiate objects using Beans.instantiate() instead of the new operator. This feature is related to JavaBeans and is discussed in my book *Rapid Java Application Development Using JBuilder 4/5/6/7, Second Edition.*

3.2.6 The Editor Page

The Editor page (Figure 3.12) sets the end of line characters in the editor. The end-of-line characters are

platform-dependent.

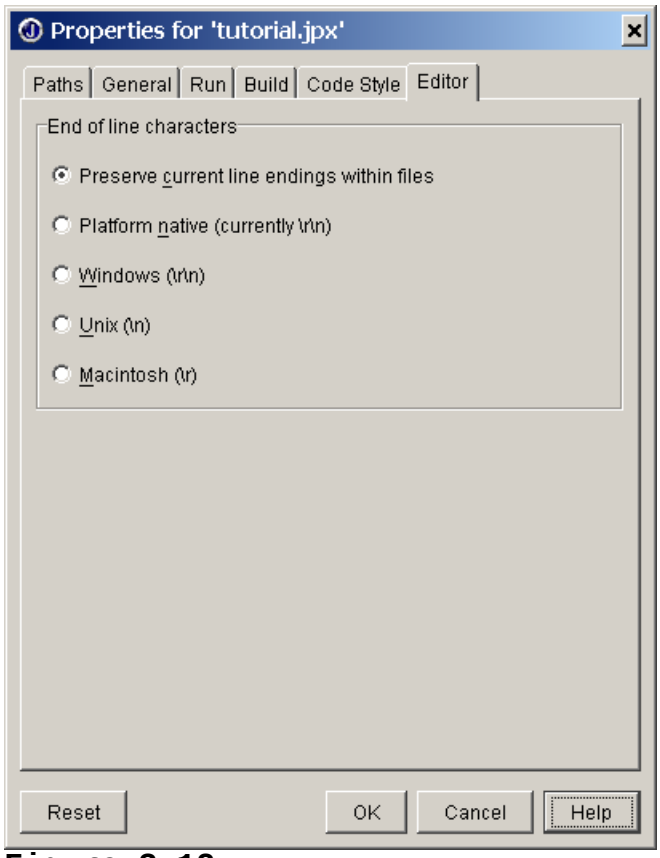


Figure 3.12

The Editor page of the Project Properties dialog box sets the end of line characters in the editor.

CHAPTER

4

Creating, Compiling, and Running Java Programs

This chapter shows you how to create, compile and run a Java program.

4.1 Creating a Java Program

There are many ways to create a Java program in JBuilder. This book will show you how to use various wizards to create certain types of Java programs. In this section, you will learn how to create Java programs using the Class wizard.

The following are the steps to create a Java program:

1. Open the tutorial.jpx project if it is not in the project pane. To open it, choose File, Reopen to display a submenu consisting of the most recently opened projects and files, as shown in Figure 4.1. Select the project if it is in the menu. Otherwise, choose File, Open to locate and open tutorial.jpx. The project file is the one with the ( )icon.

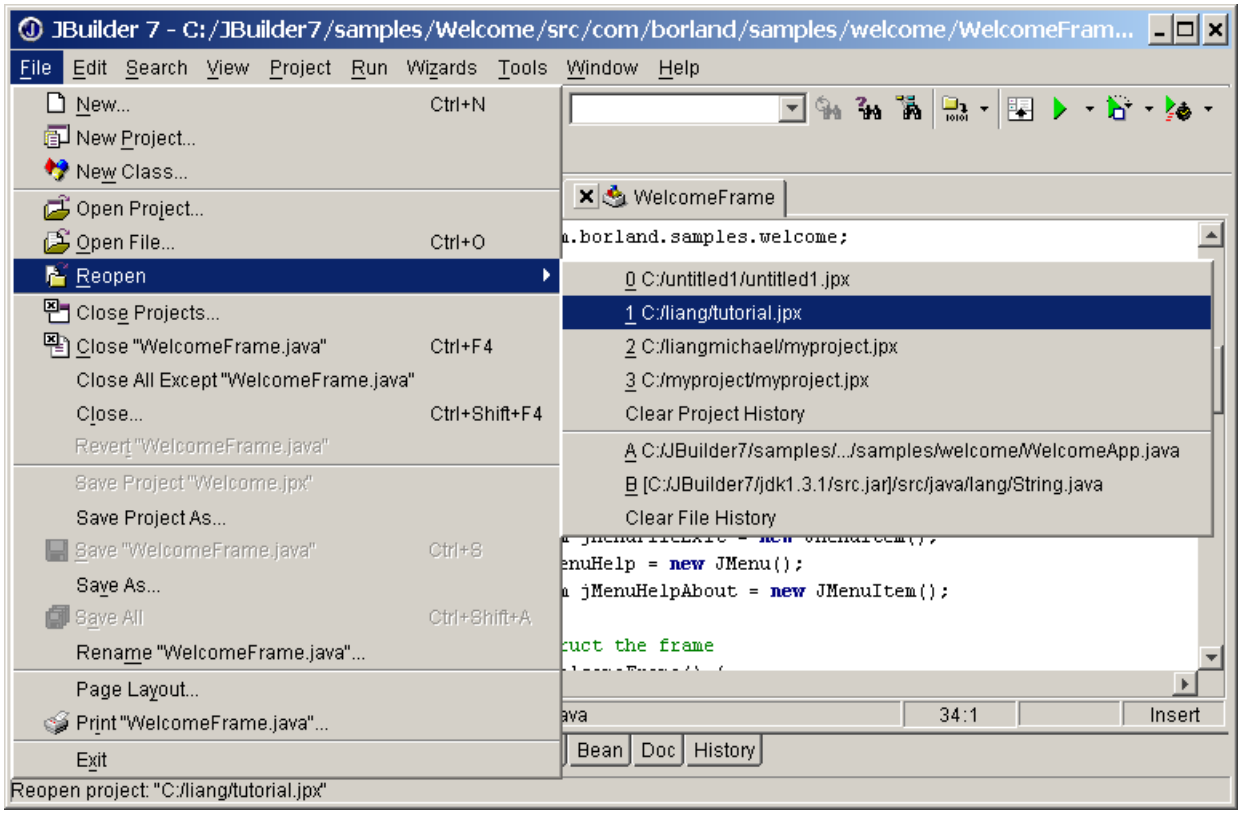


Figure 4.1

Recently used projects can be reopened by choosing File, Reopen.

2. Choose File, New Class to display the Class wizard, as shown in Figure 4.2.

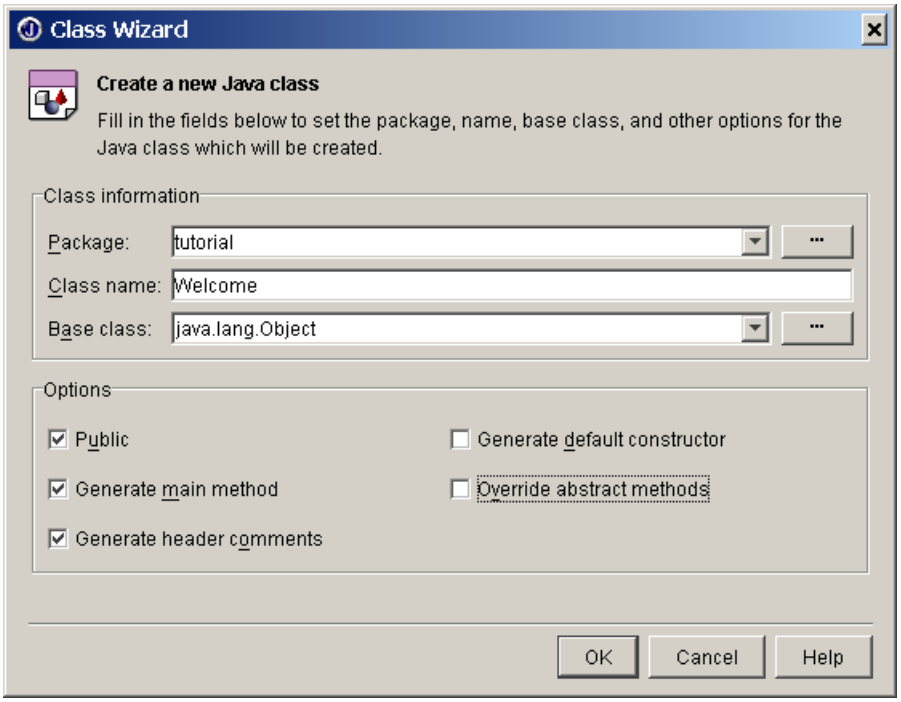


Figure 4.2

You can use the Class wizard to create a template for a new class.

3. In the Class wizard, type tutorial in the Package field and Welcome in the Class name field, and check the options Public, Generate main method, and Generate header comments in the Options section, as shown in Figure 4.2. Click *OK* to generate Welcome.java, as shown in Figure 4.3.

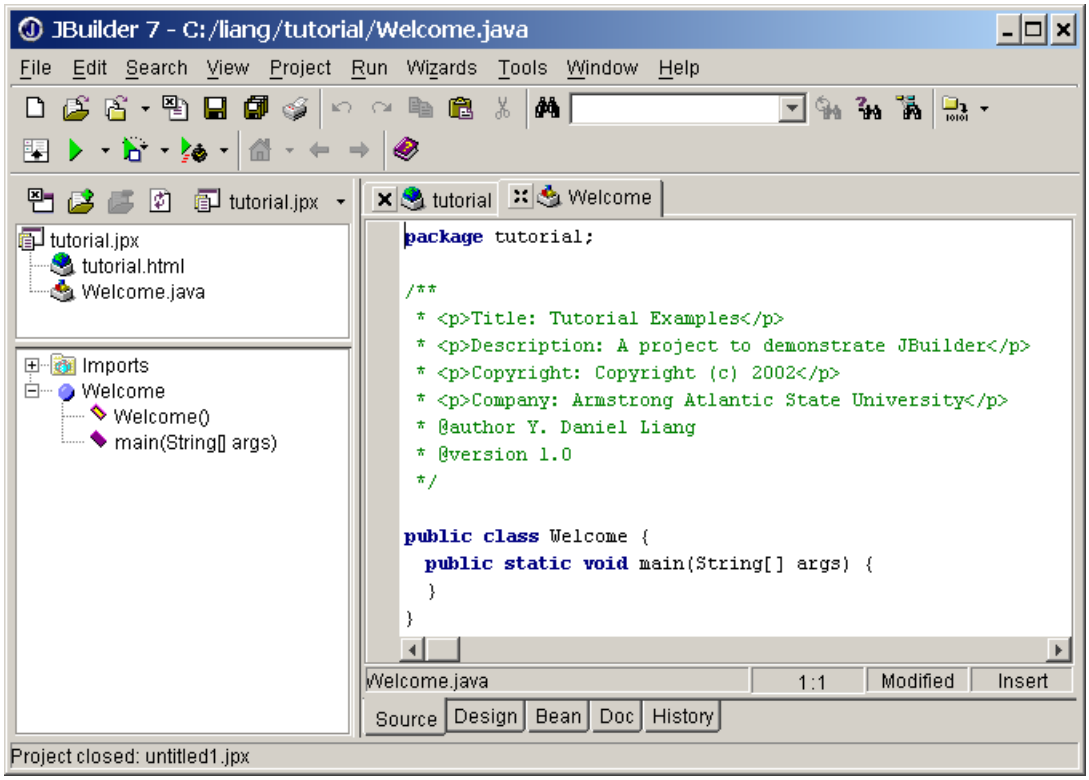


Figure 4.3

The program Welcome.java is generated by the Class wizard.

4. Add the following line in the main method,

```
System.out.println("Welcome to Java");
```

5. Select File, Save All to save all your work. You should see a confirmation message in the status bar indicating that the files are saved.

NOTE: As you type, the code completion assistance may automatically come up to give you suggestions for completing the code. For instance, when you type a dot (.) after System and pause for a second, JBuilder displays a popup menu with suggestions to complete the code, as shown in Figure 4.4. You can then select from the menu to complete the code.

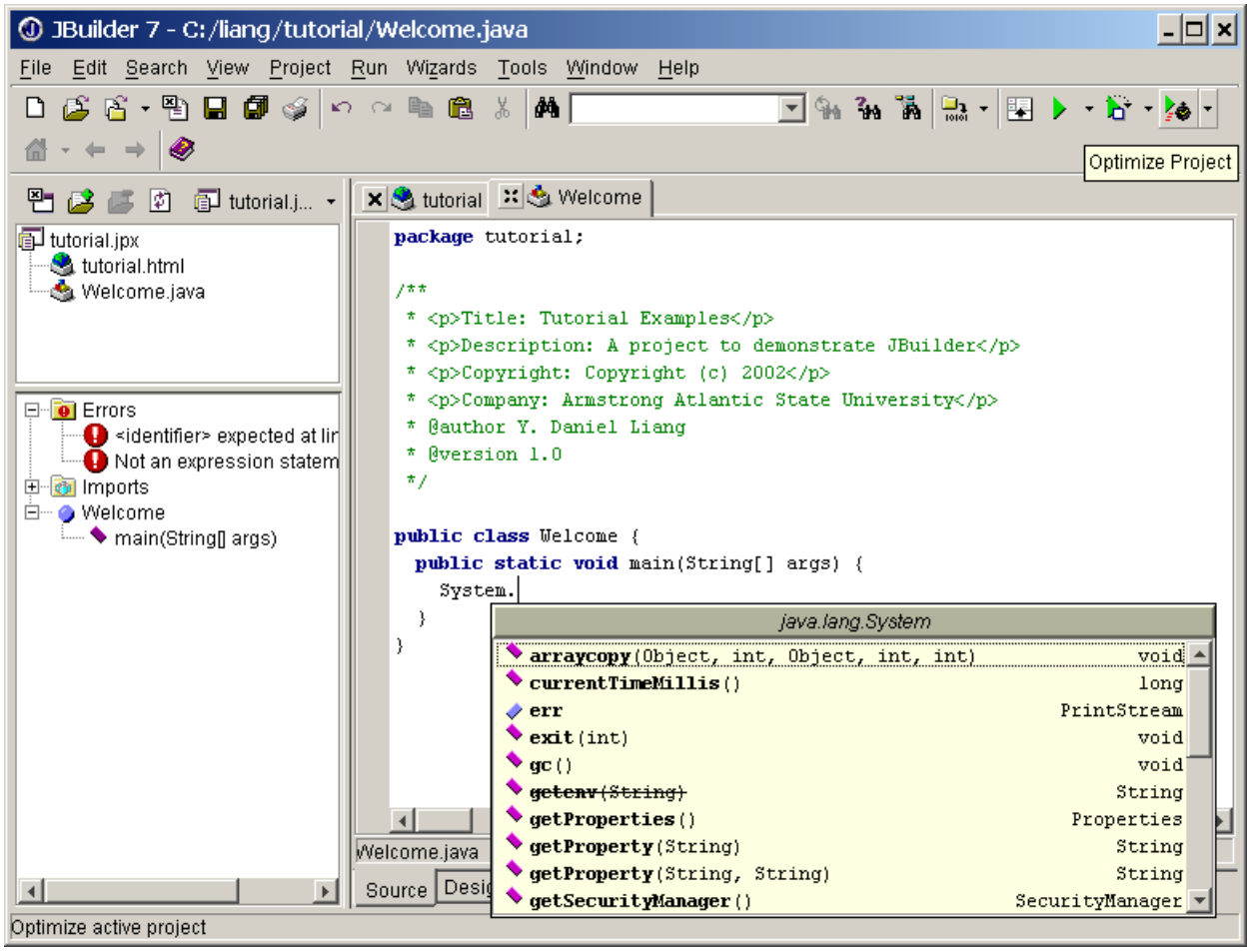


Figure 4.4

The Code Insight popup menu is automatically displayed to help you complete the code.

CAUTION: Java source programs are case-sensitive. It would be wrong, for example, to replace main in the program with Main. Program file names are case-sensitive on UNIX and generally not case-sensitive on PCs, but file names are case-sensitive in JBuilder.

TIP: The public class name must match the file name. To change the file name, right-click the file in the project pane to display the context menu. Choose *Rename* in the context menu to change the file name, as shown in Figure 4.5. You can also delete the file from the context menu.

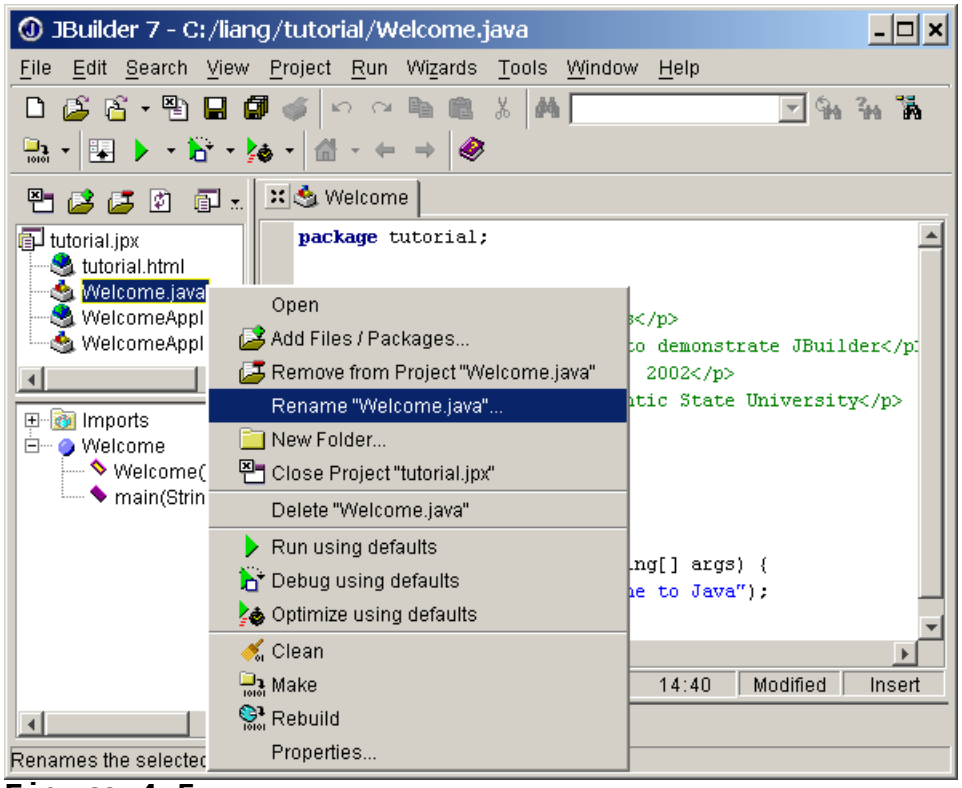


Figure 4.5

The context menu of the file in the project pane has many useful commands.

> NOTE: You could type any package name (e.g. com.yourcompany.hostname) in the Package field in Figure 4.2. If you don't the package statement in the program, you may leave the Package filed blank in Figure 4.2.

4.2 Compiling a Java Program

To compile **Welcome.java**, use one of the following methods. (Be sure that **Welcome.java** is selected in the project pane.)

[BL] Select Project, Make "Welcome.java" from the menu bar.

[BL] Click the Make toolbar button ( ).

[BX] Point to Welcome.java in the project pane, right-click the mouse button to display a popup menu (see Figure 4.5), and choose Make from the menu. (I find this method most useful.)

The compilation status is displayed on the status bar. If there are no syntax errors, the *compiler* generates a file named **Welcome.class**. This file is not an object file as generated by other high-level language compilers. This file is called the *bytecode*. The bytecode is similar to machine

instructions, but is architecture-neutral and can run on any platform that has the Java interpreter and runtime environment. This is one of Java's primary advantages: Java bytecode can run on a variety of hardware platforms and operating systems.

> NOTE: The bytecode is stored in OutputPath\PackageName. Therefore, Welcome.class is stored in **c:\liang\tutorial**, since the Output path is set to **c:\liang** (see Figure 3.2) and the package name is **tutorial**. The file structures for the examples in this book are shown in Figure 4.6.

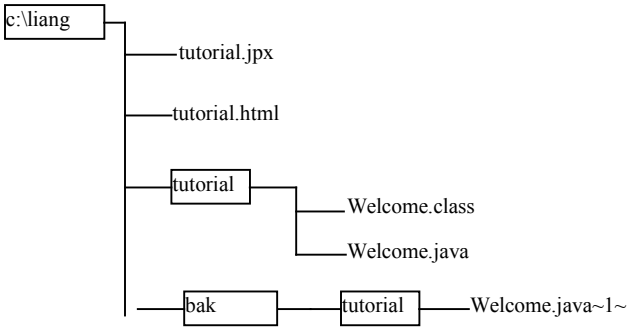


Figure 4.6

Welcome.java and Welcome.class are placed in c:\liang\tutorial.

> TIP: The public class name must match the file name. To change the file name, right-click the file in the project pane to display the context menu. Choose *Rename* in the context menu to change the file name, as shown in Figure 4.5. You can also delete the file from the context menu.

> TIP: As you type, the source code in the editor is dynamically parsed. The errors are displayed in the structure pane as shown in Figure 4.7.

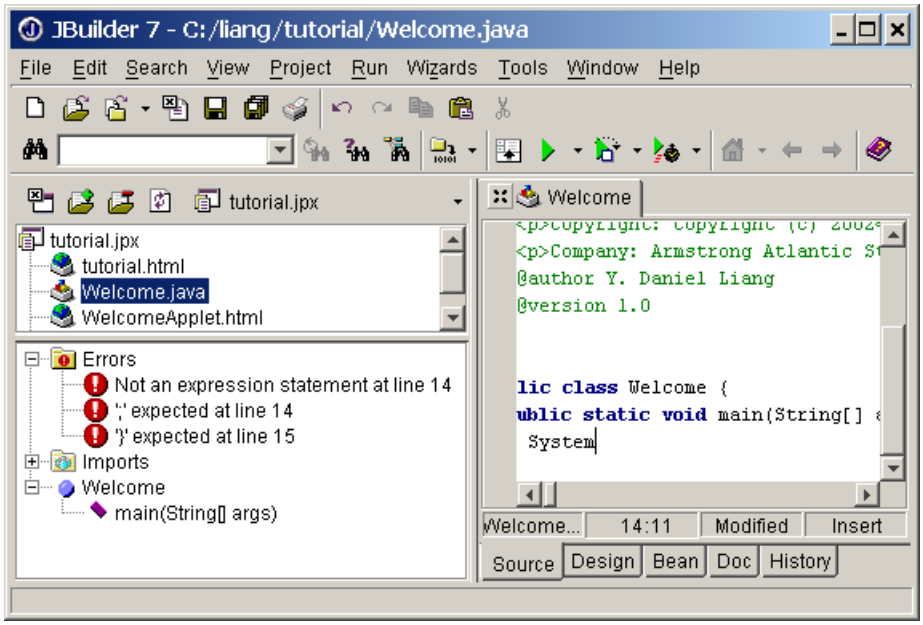


Figure 4.7

JBuilder dynamically parses the source code and displays the syntax errors in the structure pane.

4.3 Executing a Java Application

To run Welcome.class, point to Welcome.java in the project pane and right-click the mouse button to display a popup menu. Choose *Run Using Defaults* from the popup menu.

> NOTE: The Run command invokes the Compile command if the program is not compiled or was modified after the last compilation.
>
> NOTE: You could run a program by selecting Run, Run ''Welcome.java'' from the main menu, or by clicking the Run toolbar button ( ), but then you have to specify a main class in the Runtime Properties dialog box. So it is more convenient to run a program from the project pane.

When this program executes, JBuilder displays the output in the message pane, as shown in Figure 4.8. The execution status is displayed below the message pane.

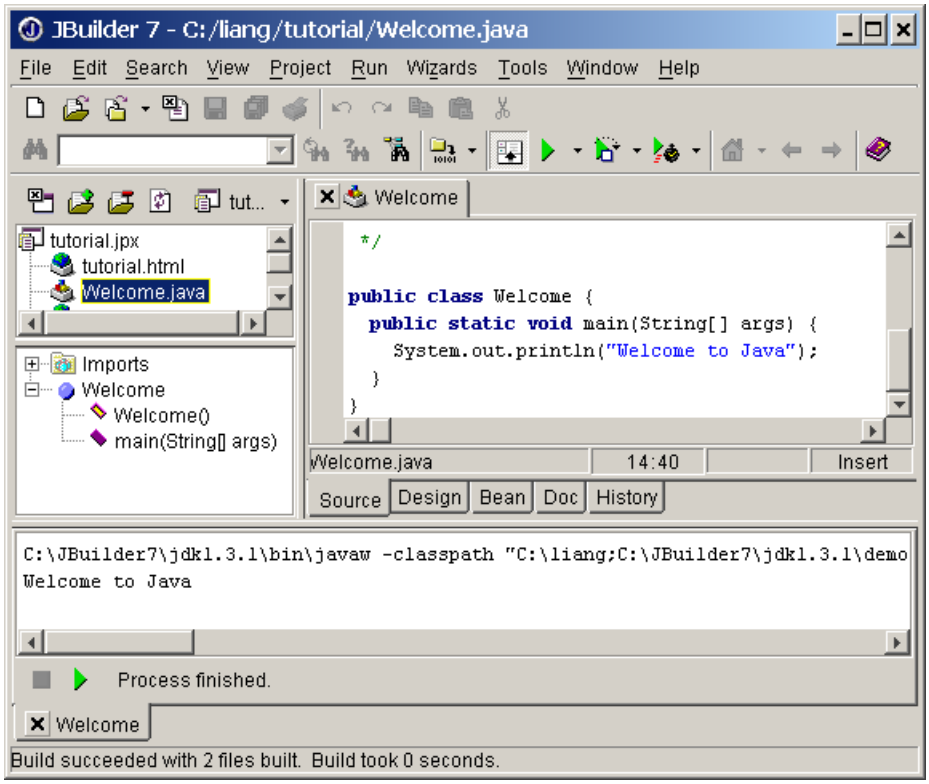


Figure 4.8

The execution result is shown in the message pane.

TIP: If the message pane is not displayed, choose View, Messages to display it.

4.4 Run Java Applications from the Command Line

So far you have run the programs in JBuilder IDE. You also can run the program standalone directly from the operating system. Here are the steps in running the **Welcome** application from the DOS prompt.

1. Start a DOS window by clicking the Window's Start button, Programs, MS-DOS Prompt in Windows.

2. Type the following commands to set up proper environment variables for running Java programs in the DOS environment in Windows:

set path=%path%;c:\JBuilder7\jdk1.3.1\bin

set classpath=.;%classpath%

3. Type **cd c:\liang** to change the directory to **c:\liang**.

4. Type **java tutorial.Welcome** to run the program. A sample run of the output is shown in Figure 4.9.

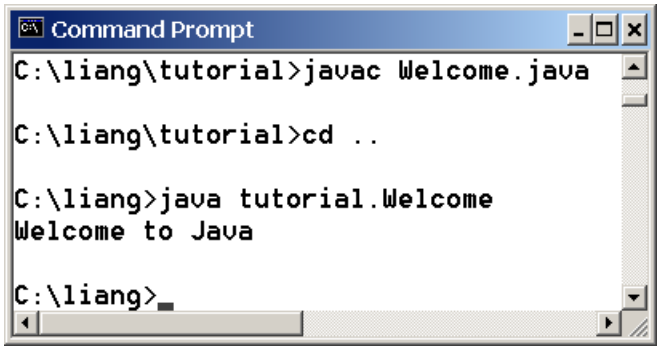


Figure 4.9

You can run the Java program from the DOS prompt using the java command.

Insert the following two lines

set path=%path%;c:\jBuilder4\jdk1.3.1\bin

set classpath=.;%classpath%

in the autoexec.bat file on Windows 95 or Windows 98 to avoid setting the environment variables in Step 2 for every DOS session. On Windows NT or Windows 2000, select System from the Control Panel to set the environment variables.

Setting environment variables enables you to use the JDK command-line utilities. The java command invokes the Java interpreter to run the Java bytecode.

NOTE: You can also compile the program using the javac command at the DOS prompt as shown in Figure 4.9.