

## Supplement H.2: JBuilder 2005 Tutorial

For Introduction to Java Programming, 5E

By Y. Daniel Liang

This supplement covers the following topics:

- Getting Started with JBuilder
- Creating a Project
- Creating, Compiling, and Running a Java Program
- Debugging in JBuilder
- Getting Help in JBuilder
- Creating and Testing Java Applets
- Adding classes to the library

NOTE: To use this supplement with the text, cover Sections 1, 2, and 3 in this supplement after Chapter 1 in the text, cover Sections 4 and 5 in this supplement after Chapter 2 in the text, and cover Section 6 in this supplement at the beginning of Chapter 14 in the text.

### 0 Introduction

This tutorial is for students who are currently taking a Java course using JBuilder with Introduction to Java Programming, 5E.

You can use the JDK command line utility to write Java programs. The JDK command line utility consists of a set of separate programs, such as compiler and interpreter, each of which is invoked from a command line. Besides the JDK command line utility, there are more than a dozen Java development tools on the market today, including Borland JBuilder, NetBeans, Sun ONE Studio (a commercial version of NetBeans), Eclipse, and WebGain Visual Café. These tools support an *integrated development environment* (IDE) for rapidly developing Java programs. Editing, compiling, building, debugging, and online help are integrated in one graphical user interface. Using these tools effectively will greatly increase your programming productivity.

This brief tutorial will help you to become familiar with JBuilder. Specifically, you will learn how to create projects, create programs, compile, and run programs.

NOTE: JBuilder can run on any platform with a Java Virtual Machine. The screen shots in the tutorial are taken from Windows using JBuilder 2005. You can download JBuilder 2005 Foundation from [http://www.borland.com/products/downloads/download\\_jbuilder.html](http://www.borland.com/products/downloads/download_jbuilder.html).

NOTE: JBuilder 2005 is bundled with JDK 1.4. To use JDK 1.5 in JBuilder, (1) download and install JDK 1.5. JDK 5.0 can be downloaded from <http://java.sun.com/j2se/1.5.0/download.jsp>. See Supplement A, "Installing JDK 5.0," on how to install JDK 5.0; (2) See Supplement H.3 on how to configure and set JDK 1.5 in JBuilder.

## 1 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 1.1. If you don't see the Welcome project, choose *Help, Learning About JBuilder, Welcome Project (Sample)* from the main menu.

NOTE: The screen shots in this book are from JBuilder 2005 Enterprise. The screen for JBuilder 2005 Foundation and JBuilder 2005 Developer may look slightly different.

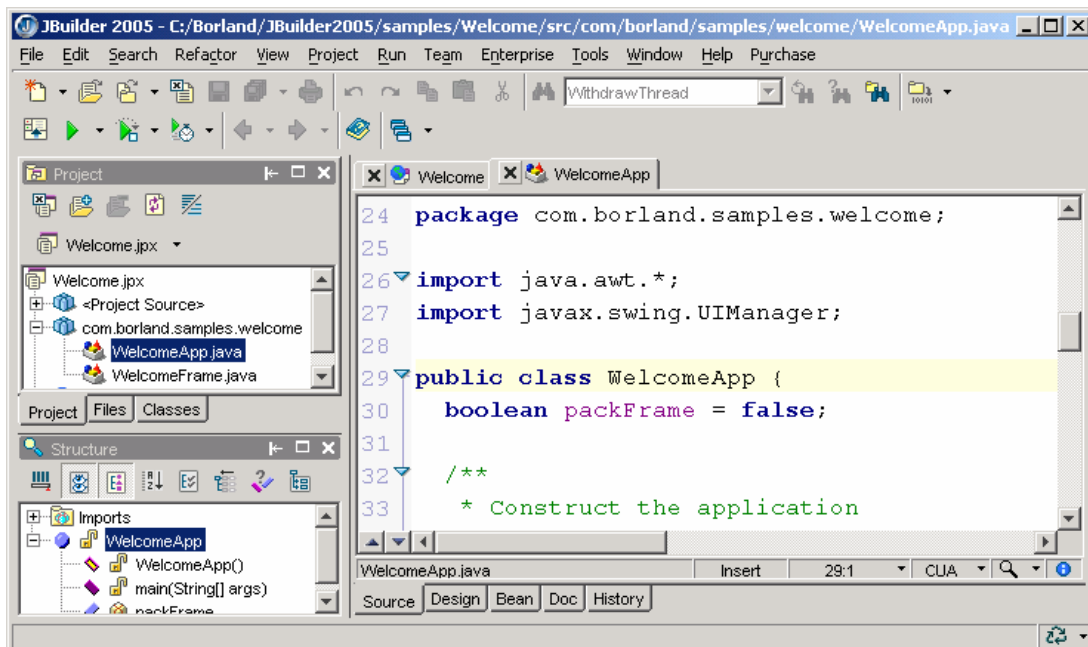


Figure 1.1

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


The JBuilder user interface presents a single window, called *AppBrowser*, for managing Java projects, browsing files, compiling, running, and debugging programs. The *AppBrowser* window primarily consists of the main menu, main toolbar, status bar, project pane, structure pane, and content pane.

### *1.1 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.

### *1.2 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.

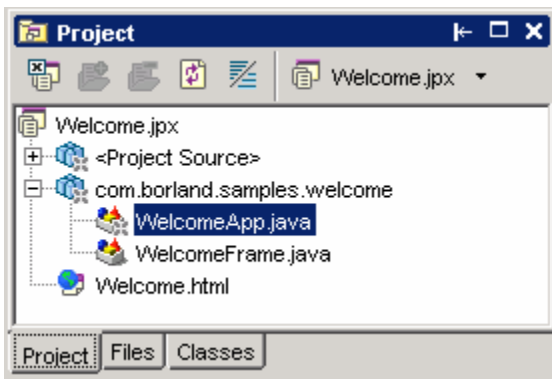
TIP: You can display a label, known as *ToolTip*, for a button by pointing the mouse to the button without clicking.

### *1.3 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.





### *1.4 The Project Pane*

The *project pane* displays the contents of one or more projects opened in the *AppBrowser*. It consists of the following items, as shown in Figure 1.2.



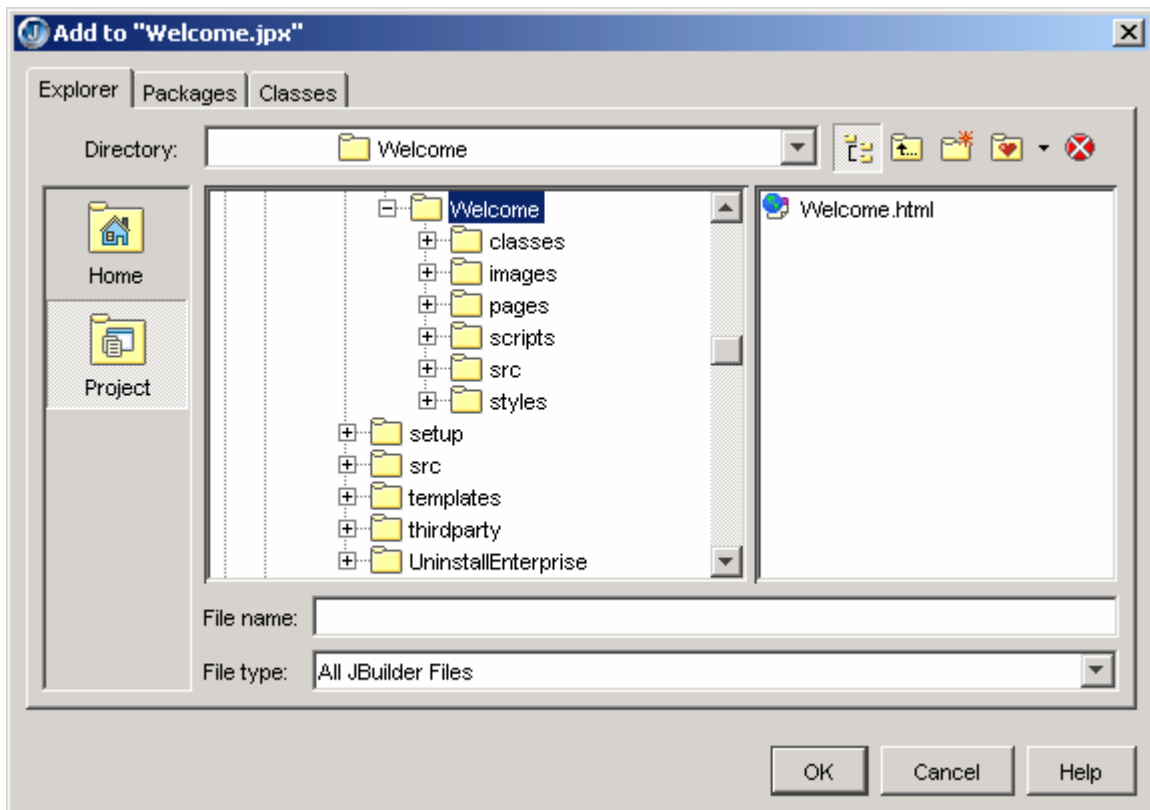
**Figure 1.2**

*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 it 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 1.2 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 1.3.
2. Open Welcome.html. You will see Welcome.html displayed in the project pane.



**Figure 1.3**

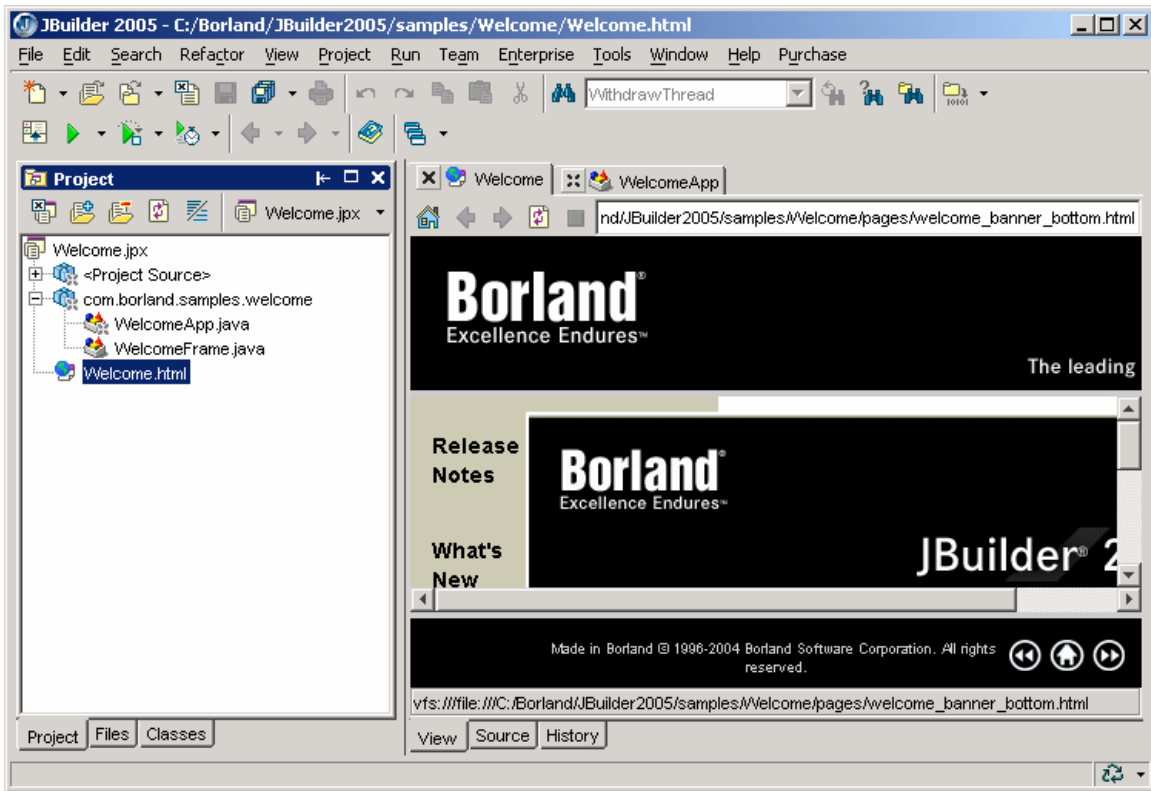
*The Open dialog box enables you to open an existing file.*

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

### 1.5 The Content Pane

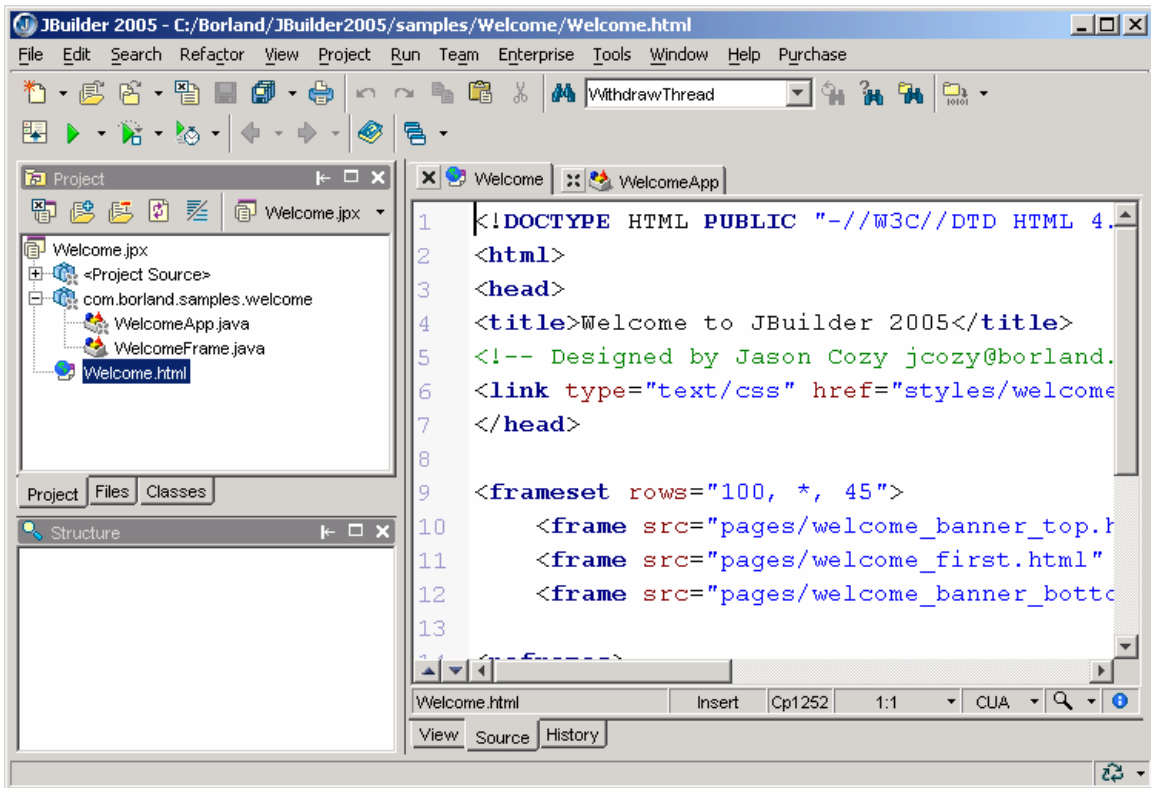
The content pane displays all the opened files as a set of tabs. To open a file in the content pane, double-click it in the project pane. The content pane displays the detailed content of the selected file. The editor or viewer used is determined by the file's extension. If you click the `WelcomeApp.java` 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 Figure 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 1.4. If you choose the Source tab, you can view and edit the HTML code in the content pane, as shown in Figure 1.5.



**Figure 1.4**

*JBuilder renders HTML files in the content pane.*



**Figure 1.5**

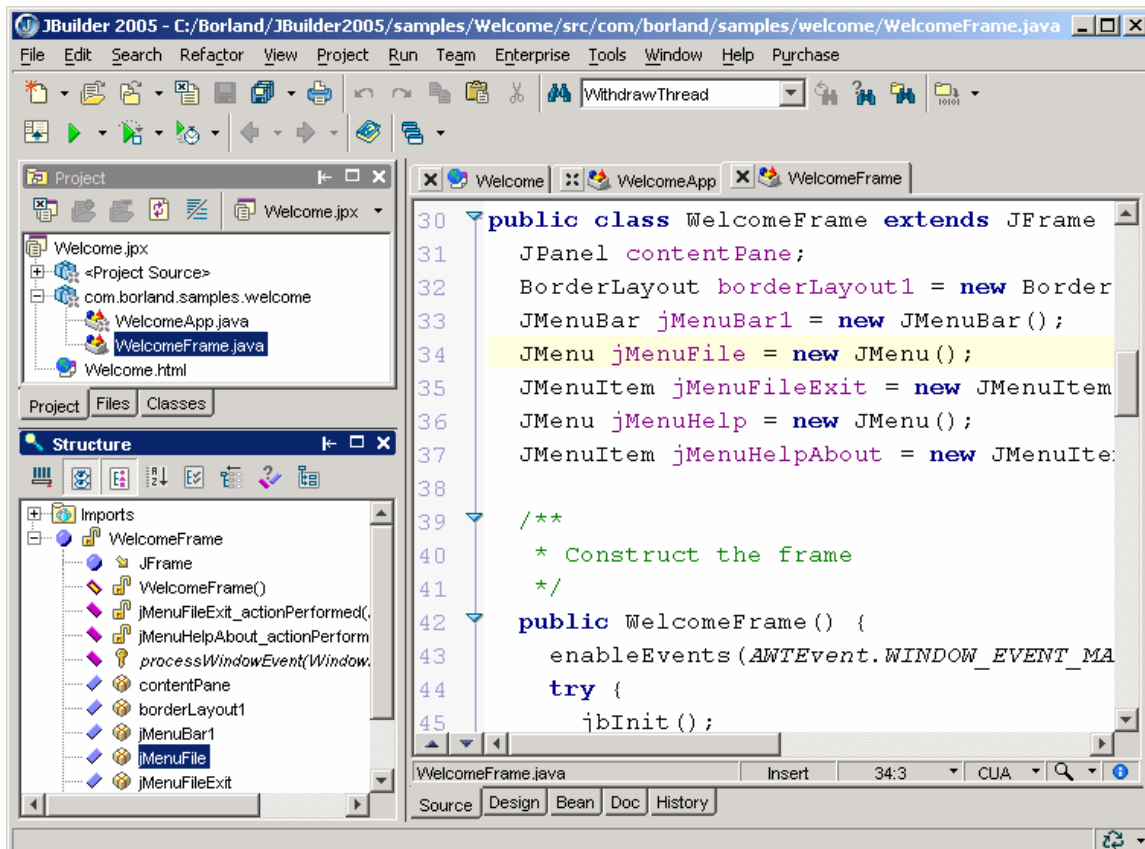
*You can edit HTML files in the content pane.*

### 1.6 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 1.6, 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 1.6**

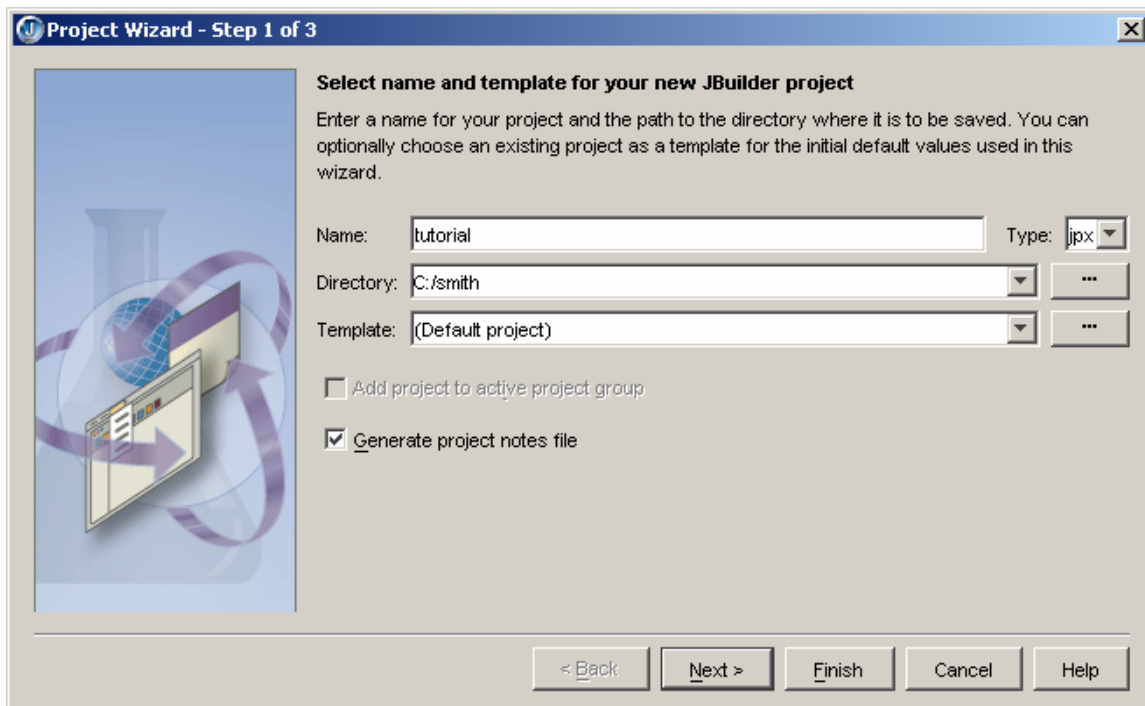
*You can cruise through the source code from the structure pane.*

## 2 Creating a Project

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 was either .jpx or .jpr. The .jpr project files are still compatible in JBuilder X.) 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. To create and run a program, you have to first create a project.

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

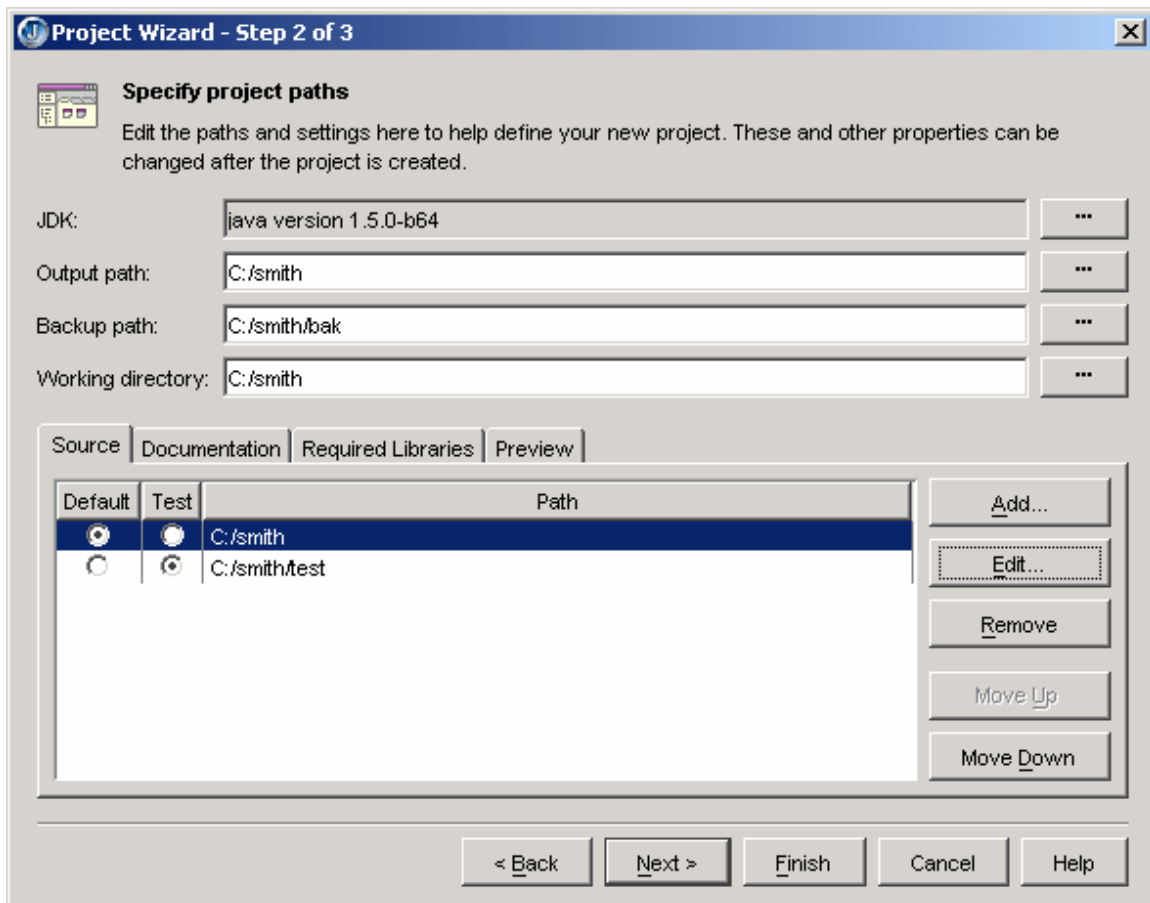
1. Choose *File, New Project* to bring up the Project wizard dialog box, as shown in Figure 2.1.



**Figure 2.1**

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

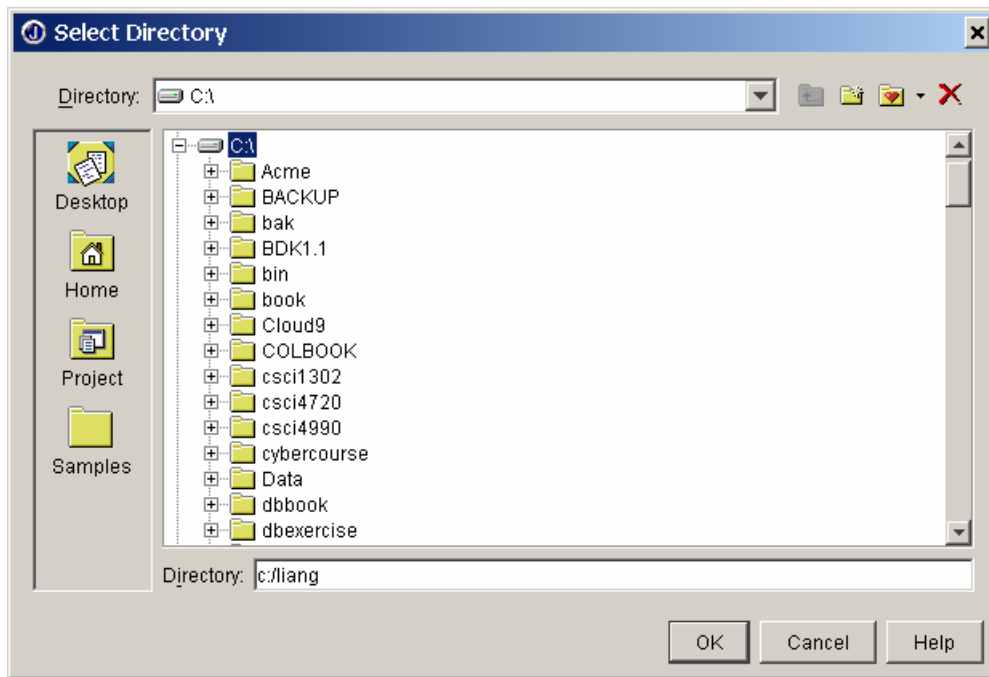
2. Type `tutorial` in the Name field and `c:/smith` in the Directory field. Check the *Generate project notes file* option box. Click Next to display Project Wizard Step 2 of 3, as shown in Figure 2.2.



**Figure 2.2**

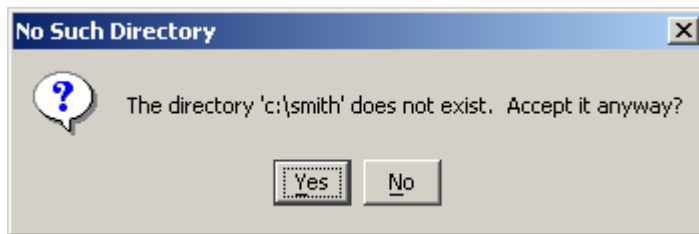
*Project Wizard Step 2 of 3 enables you to modify project settings.*

3. Type `c:/smith` in the Output path field, `c:/smith/bak` in the Backup path field, and `c:/smith` in the Working directory field. In the Source tab, change the Default path and Test path to `c:/smith`. To do so, click the Edit button to display the Select Directory dialog box, as shown in Figure 2.3. Type `c:/smith` in the Directory field. If the directory does not exist, you will see the No Such Directory dialog box, as shown in Figure 2.4. Click Yes to return to Project Wizard - Step 2 of 3.



**Figure 2.3**

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



**Figure 2.4**

*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 2.5. Fill in the title, author, company, and description fields. These optional fields provide a description for the project. (Uncheck *Enable source package discovery and compilation* to shorten the contents displayed in the project pane)
5. Click *Finish*. The new project is displayed in the project pane, as shown in Figure 2.6. The Project wizard created the project file (tutorial.jpx) and an HTML file (tutorial.html), and placed them in c:\smith. The project file stores the information

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

**Project Wizard - Step 3 of 3**

**Specify general project settings**  
Enter settings here to help define your new project. These and other properties can be changed after the project is created.

Encoding:

Automatic source packages

Enable source package discovery and compilation

Deepest package exposed:

Class Javadoc fields:

Label	Text
Title:	JBuilder Tutorial
Description:	First Project
Copyright:	Copyright (c) 2004
Company:	ABC University
@author	Smith
@version	1.0

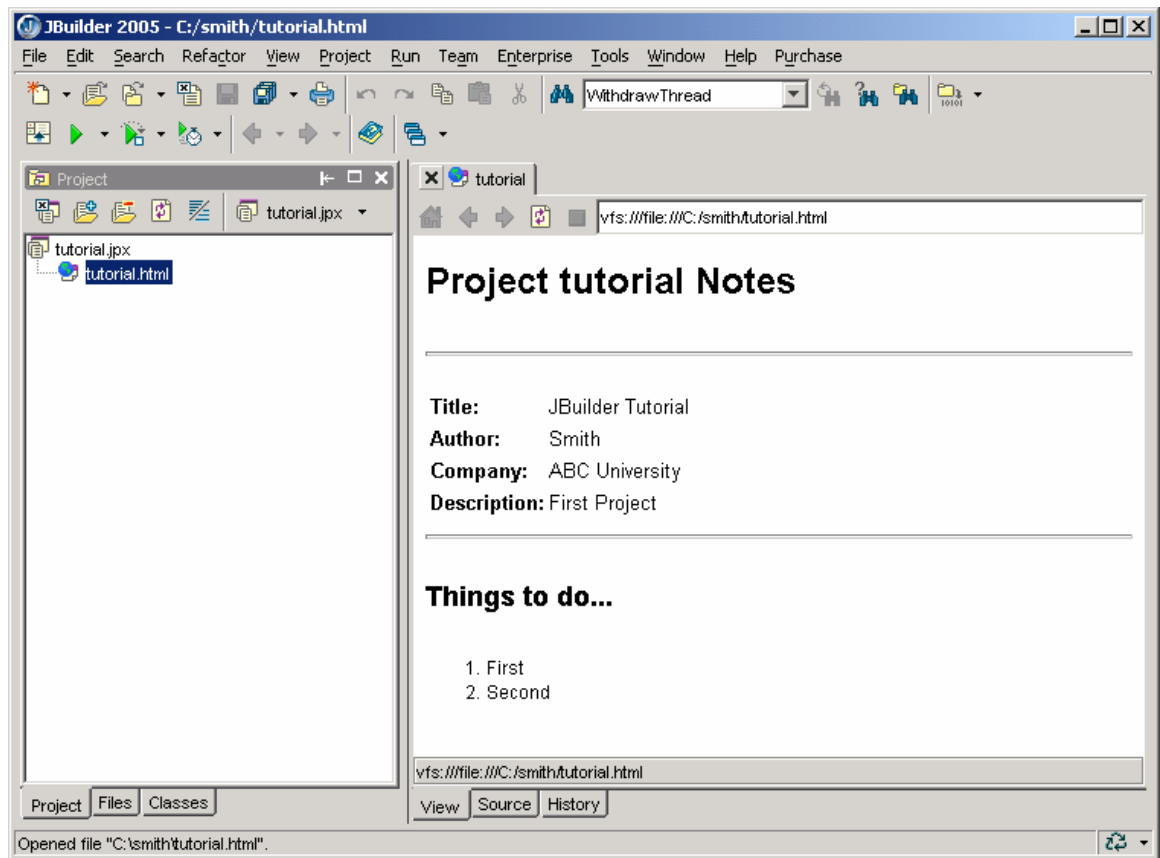
Include references from project library class files

Diagram references from generated source

< Back   Next >   Finish   Cancel   Help

**Figure 2.5**

*Project Wizard Step 3 of 3 collects optional information for the project.*



**Figure 2.6**

*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 that they can be easily located and removed.

**IMPORTANT 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. To avoid these, you may create your project exactly as shown in this section and change *smith* to your name, or follow the instructions from your instructors.

### **3 Creating, Compiling, and Running a Java Program**

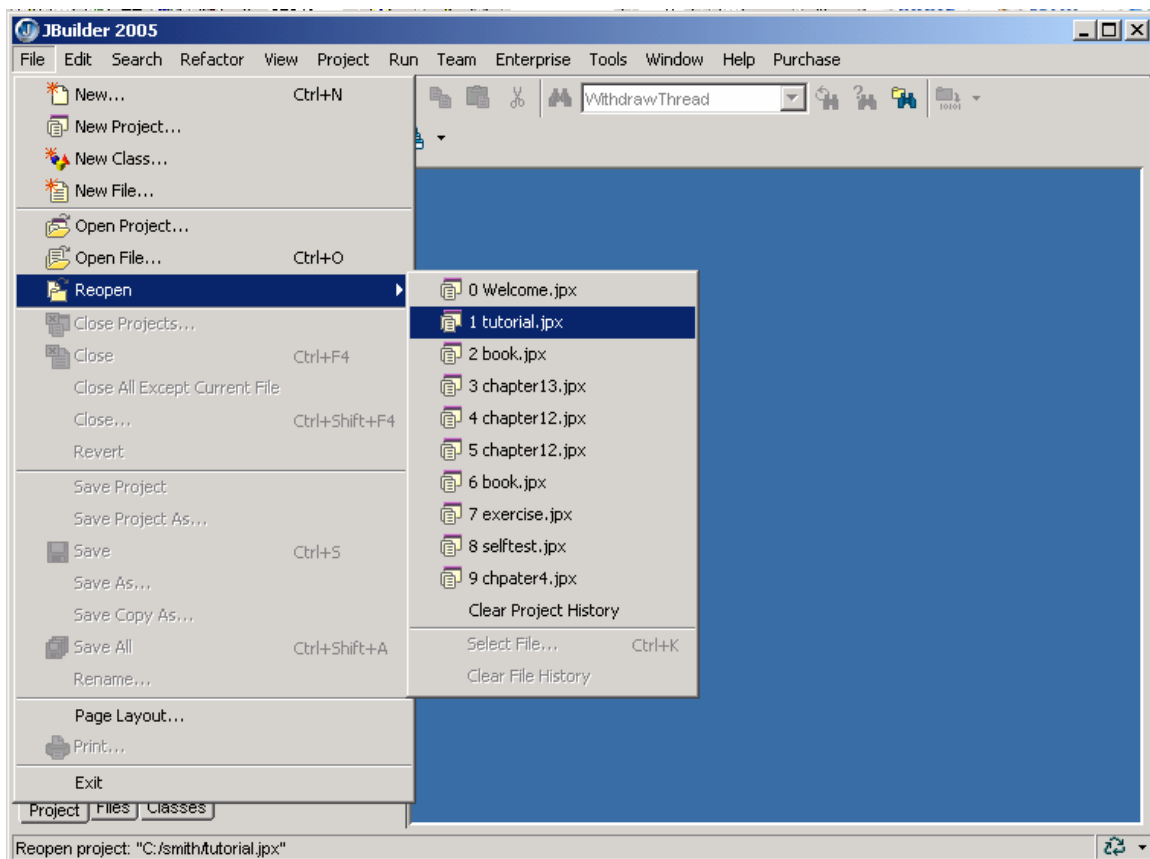
After you created a project, you can create programs in the project.

#### *3.1 Creating a Java Program*

There are many ways to create a Java program in JBuilder. This tutorial 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 in creating a Java program:

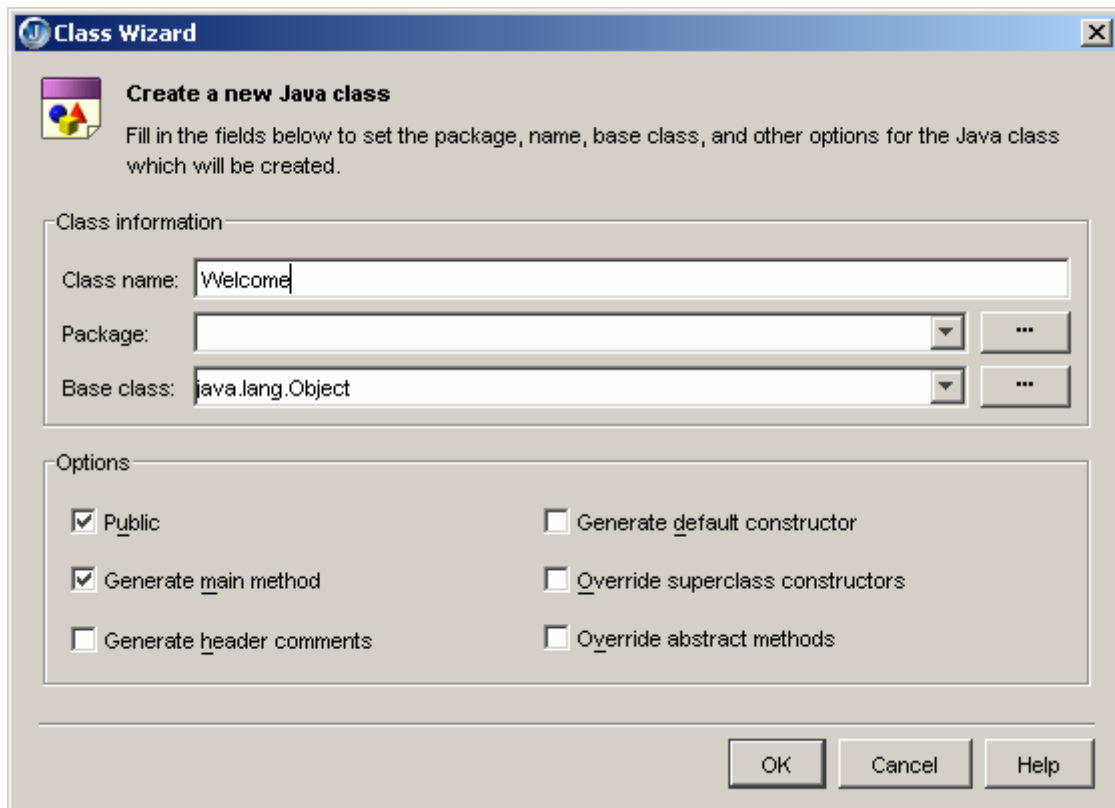
1. Open the tutorial.jpx project if it is not in the project pane. To open it, choose *File, Reopen* to display a submenu consisting of the most recently opened projects and files, as shown in Figure 3.1. Select the project if it is in the menu. Otherwise, choose *File, Open* to locate and open tutorial.jpx. The project file is the one with the (📁) icon.



**Figure 3.1**

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

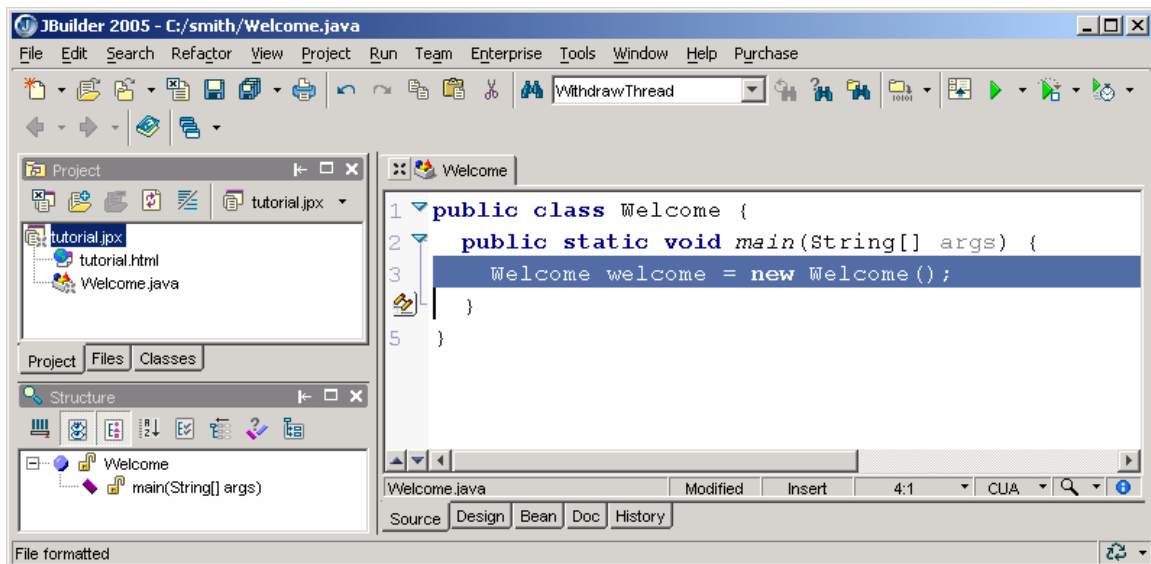
2. Choose *File, New Class* to display the Class wizard, as shown in Figure 3.2.



**Figure 3.2**

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

3. In the Class wizard, type Welcome in the Class name field, leave the Package field blank, and check the options Public, Generate main method, and Generate header comments in the Options section, as shown in Figure 3.2. Click OK to generate Welcome.java, as shown in Figure 3.3.



**Figure 3.3**

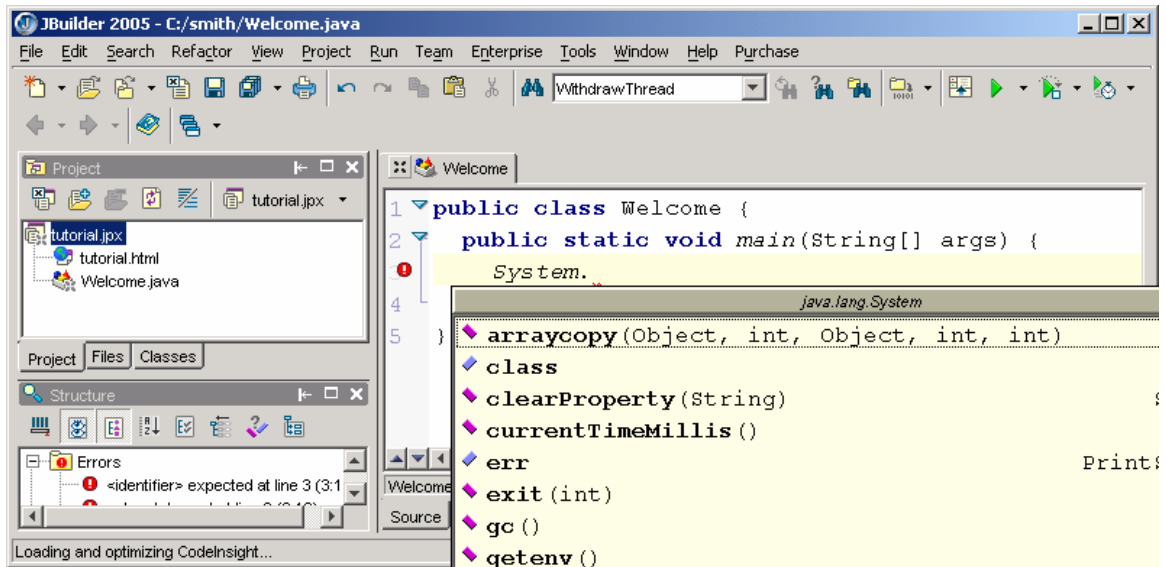
*The program Welcome.java is generated by the Class wizard.*

4. Replace Line 3 with 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 hints for completing the code, as shown in Figure 3.4. You can then select from the menu to complete the code.

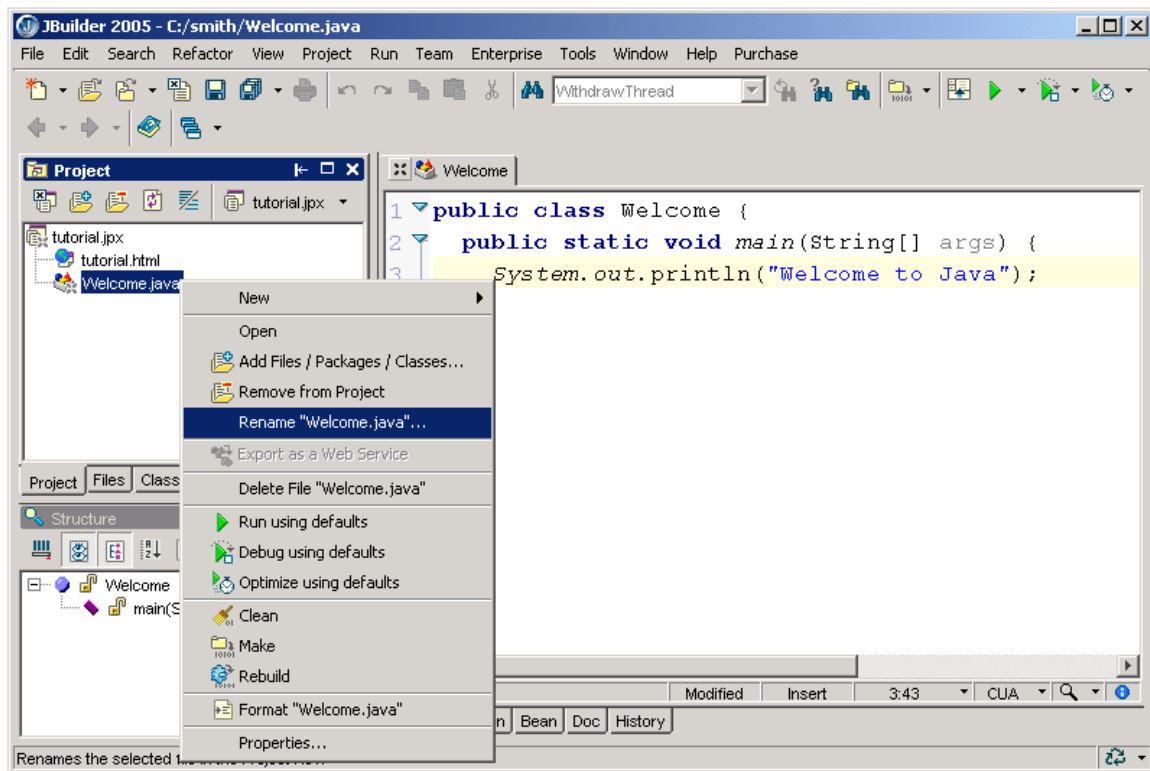


**Figure 3.4**

*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 3.5. You can also delete the file from the context menu.



**Figure 3.5**


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

NOTE: You could type any package name (e.g., chapter1 or com.yourcompany.hostname) in the Package field in Figure 3.2. To match the examples in the book, don't use package and leave the Package field blank in Figure 3.2.

### 3.2 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.

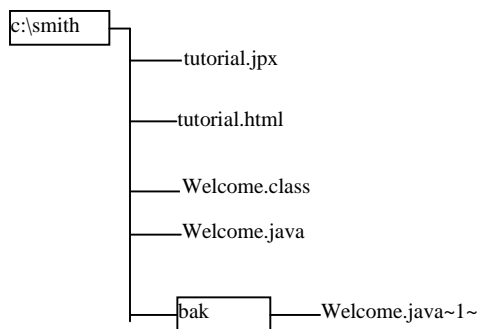
[BL] Click the Make toolbar button ()

[BX] Point to Welcome.java in the project pane, right-click the mouse button to display a popup menu (see Figure 3.5), 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:\smith`, since the Output path is set to `c:\smith` (see Figure 3.2) and the package name is blank. The file structures for the examples in this book are shown in Figure 3.6.



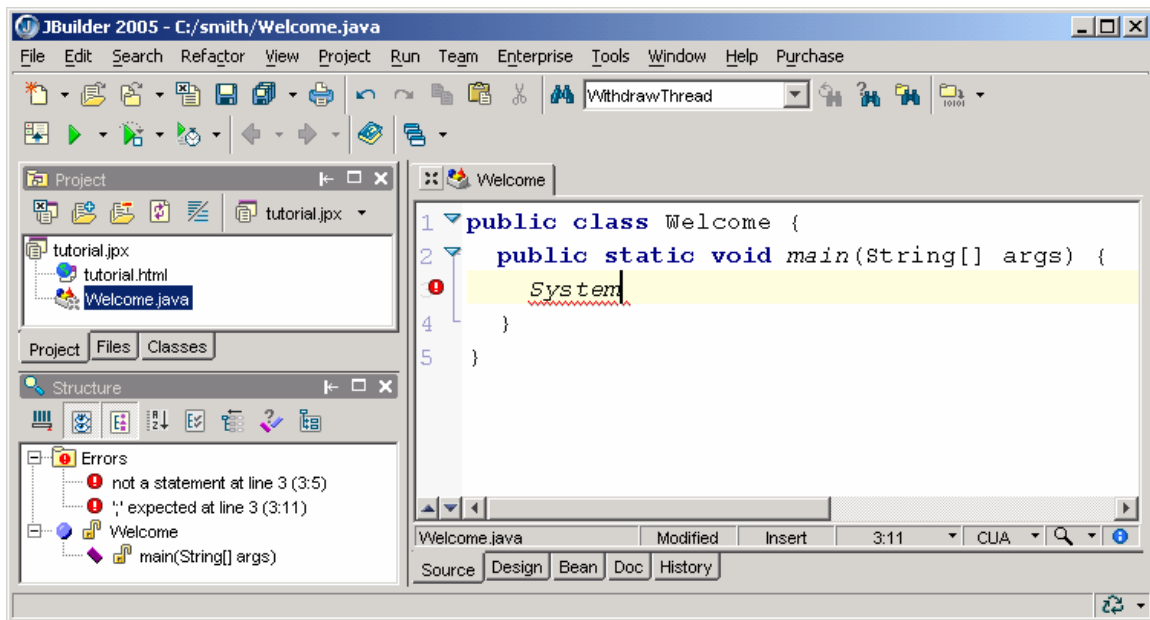
**Figure 3.6**

*Welcome.java* and *Welcome.class* are placed in `c:\smith`.

TIP: You can delete the `.class` file by choosing the *Clean* command from the context menu of the `.java` file in the project pane. You can delete all `.class` files in the project by choosing the *Clean* command from the context menu of the project node in the project pane.

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 3.5. 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 3.7.



**Figure 3.7**

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

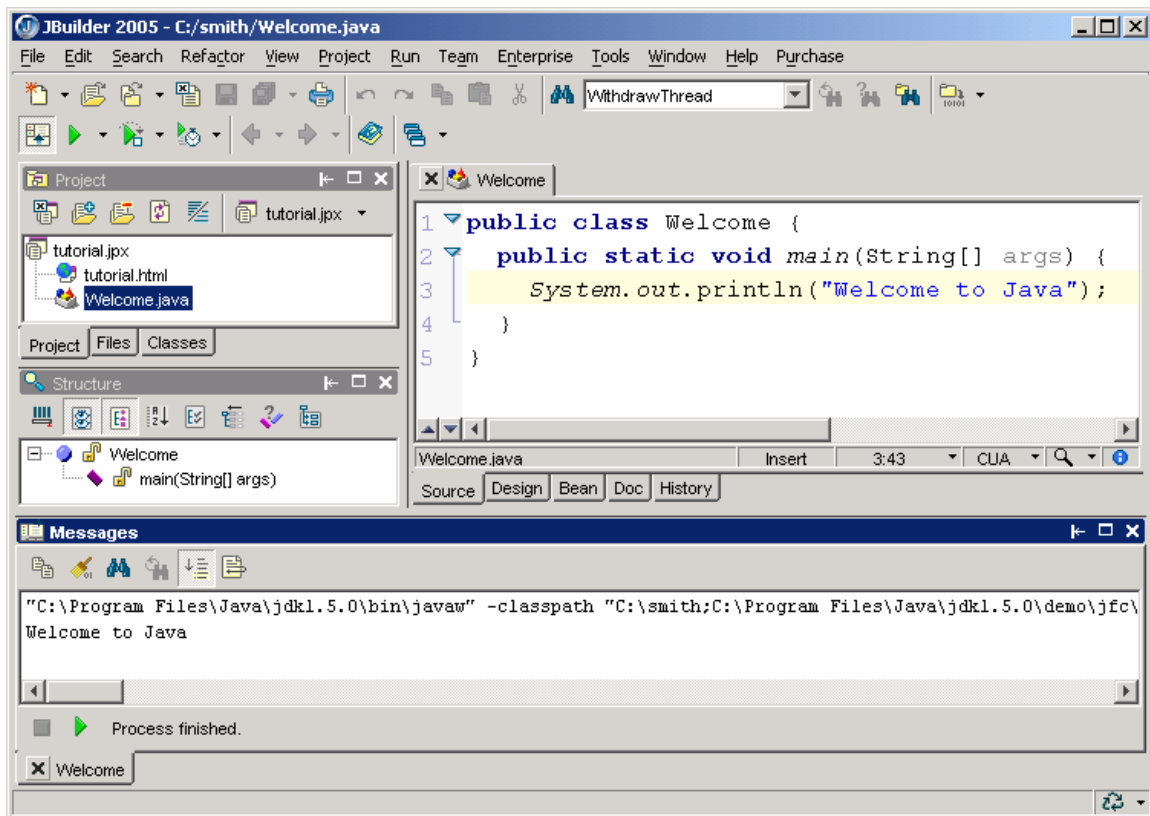
### 3.3 Executing a Java Application

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 3.8. The execution status is displayed below the message pane.



**Figure 3.8**

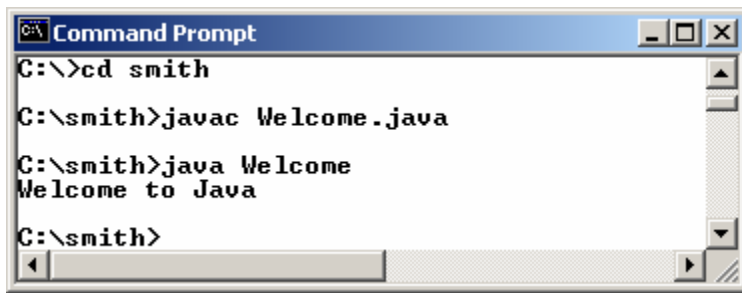
*The execution result is shown in the message pane.*

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

### 3.4 Run Java Applications from the Command Window

So far you have run programs in JBuilder IDE. You also can run a 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:\Program Files\java\jdk1.5.0\bin**  
**set classpath=.;%classpath%**
3. Type **cd c:\smith** to change the directory to **c:\smith**.
4. Type **java Welcome** to run the program. A sample run of the output is shown in Figure 3.9.



**Figure 3.9**

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

Insert the following two lines

```
set path=%path%;c:\Program Files\java\jdk1.5.0\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 3.9.

## **4 Debugging in JBuilder**

The debugger utility is integrated in JBuilder. You can pinpoint bugs in your program with the help of the JBuilder debugger without leaving the IDE. The JBuilder debugger enables you to set breakpoints and execute programs line by line. As your program executes, you can watch the values stored in variables, observe which methods are being called, and know what events have occurred in the program. Let us use Example 2.4, "Displaying the Current Time," to demonstrate debugging. Create a new class named ShowCurrentTime in tutorial.jpj. The source code for ShowCurrentTime.java can be obtained from Example 2.4.

### *4.1 Setting Breakpoints*

You can execute a program line by line to trace it, but this is time-consuming if you are debugging a large program. Often, you know that some parts of the program work fine. It makes no sense to trace these parts when you only need to

trace the lines of code that are likely to have bugs. In cases of this kind, you can use breakpoints.

A *breakpoint* is a stop sign placed on a line of source code that tells the debugger to pause when this line is encountered. The debugger executes every line until it encounters a breakpoint, so you can trace the part of the program at the breakpoint. Using the breakpoint, you can quickly move over the sections you know work correctly and concentrate on the sections causing problems.

There are several ways to set a breakpoint on a line. One quick way is to click the cutter of the line on which you want to put a breakpoint. You will see the line highlighted, as shown in Figure 4.1. You also can set breakpoints by choosing *Run, Add Breakpoint*. To remove a breakpoint, simply click the cutter of the line.

As you debug your program, you can set as many breakpoints as you want, and can remove breakpoints at any time during debugging. The project retains the breakpoints you have set when you exit the project. The breakpoints are restored when you reopen it.

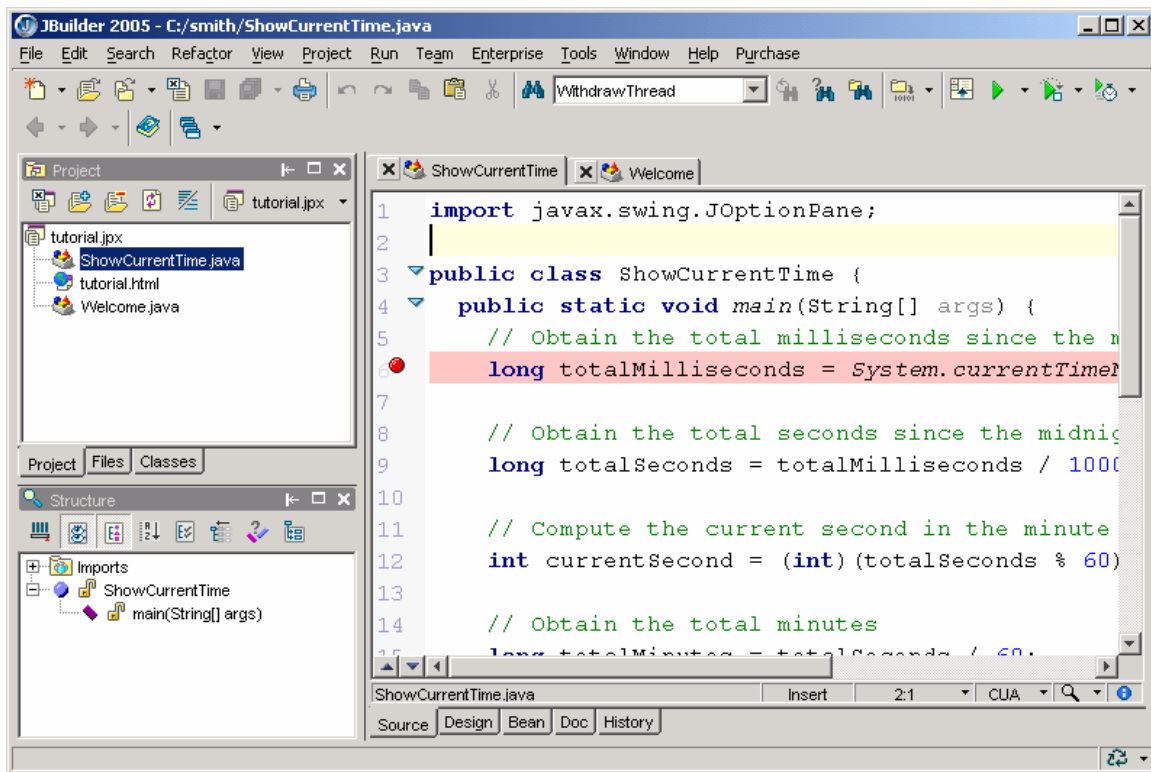
#### 4.2 Starting the Debugger

Perform the following steps to start debugging

ShowCurrentTime:

1. Select ShowCurrentTime.java in the project pane.
2. Click the cutter (the gray column on the left edge of the content pane) of the first non-comment line in the main method to set a breakpoint, as shown in Figure 4.1. A solid circle is displayed at the cutter of the line.
3. Choose ShowCurrentTime.java in the project pane, and right-click the mouse button to display the context menu. Click *Debug Using Defaults* in the context menu to start debugging.

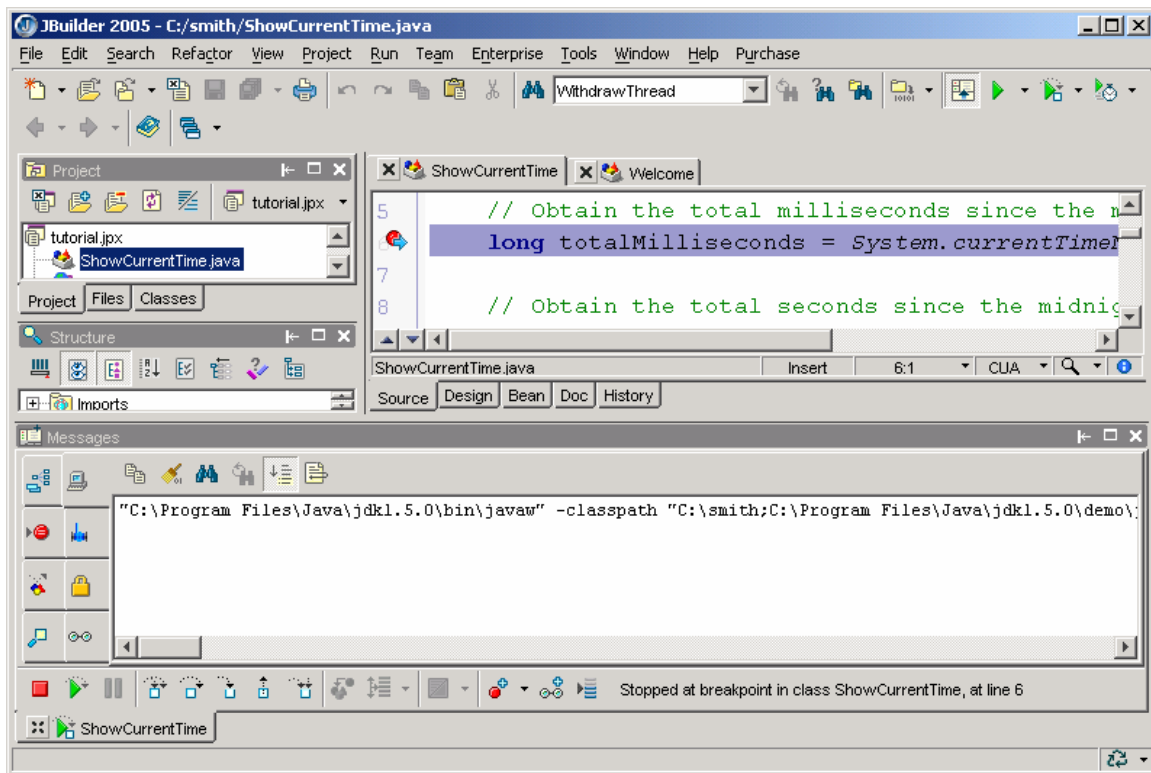
JBuilder NOTE: There are several ways to start the debugger. Steps 2 and 3 are the most convenient.



**Figure 4.1**

*A breakpoint is set in ShowCurrentTime.java.*

If the program compiles without problems, the message pane becomes a debugger pane, as shown in Figure 4.2.



**Figure 4.2**

*The message pane became a debugger pane.*

There are six tabs on the left side of the message pane: console view (🖨️), stack view (📁), watch view (👁️), loaded class view (📦), breakpoint view (🔴), and class tracing view (🔍). The *console view* displays output and errors and enables the user to enter input from the console. The *stack view* displays threads running in the program. The *watch view* displays the contents of the variables in the watch view. To add a variable to the watch view, see Section 4.4.1, "The Add Watch Command." The *loaded class view* displays classes currently loaded by the program. The *class tracing view* displays classes with tracing disabled. This feature is not available in JBuilder Foundation. It is only available in the Developer and Enterprise Editions of JBuilder. By default, tracing for the classes in the Java library is disabled. To enable tracing a Java library class, choose the package that contains the class, and right-click the mouse to display a context menu. Click *Remove* to remove the package from the Class Tracing view.

### 4.3 Controlling Program Execution

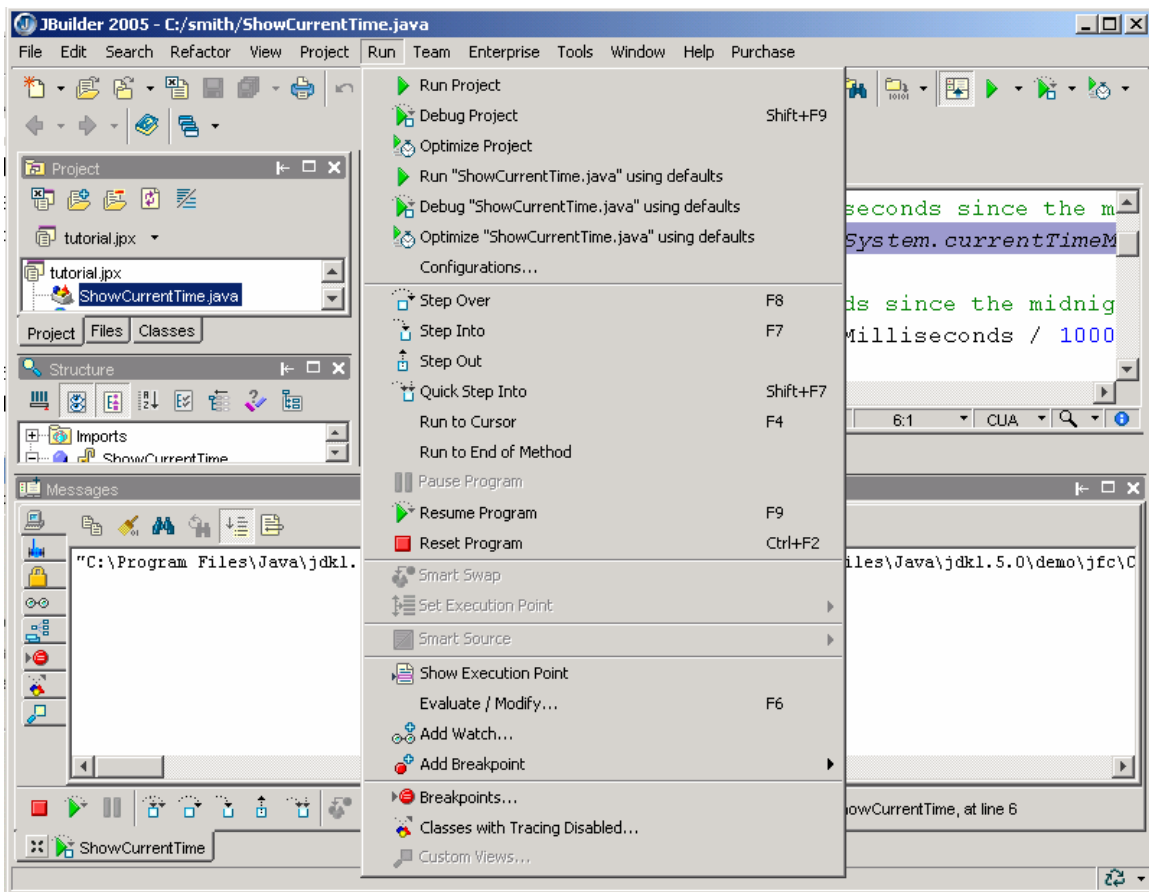
The program pauses at the first breakpoint line encountered. This line, called the *current execution point*, is highlighted and has a green arrow to the left. The execution

point marks the next line of source code to be executed by the debugger.

When the program pauses at the execution point, you can issue debugging commands to control the execution of the program. You also can inspect or modify the values of variables in the program.

When JBuilder is in the debugging mode, the *Run* menu contains the debugging commands (see Figure 4.3). Most of the commands also appear in the toolbar under the message pane. The toolbar contains additional commands that are not in the *Run* menu. Here are the commands for controlling program execution:

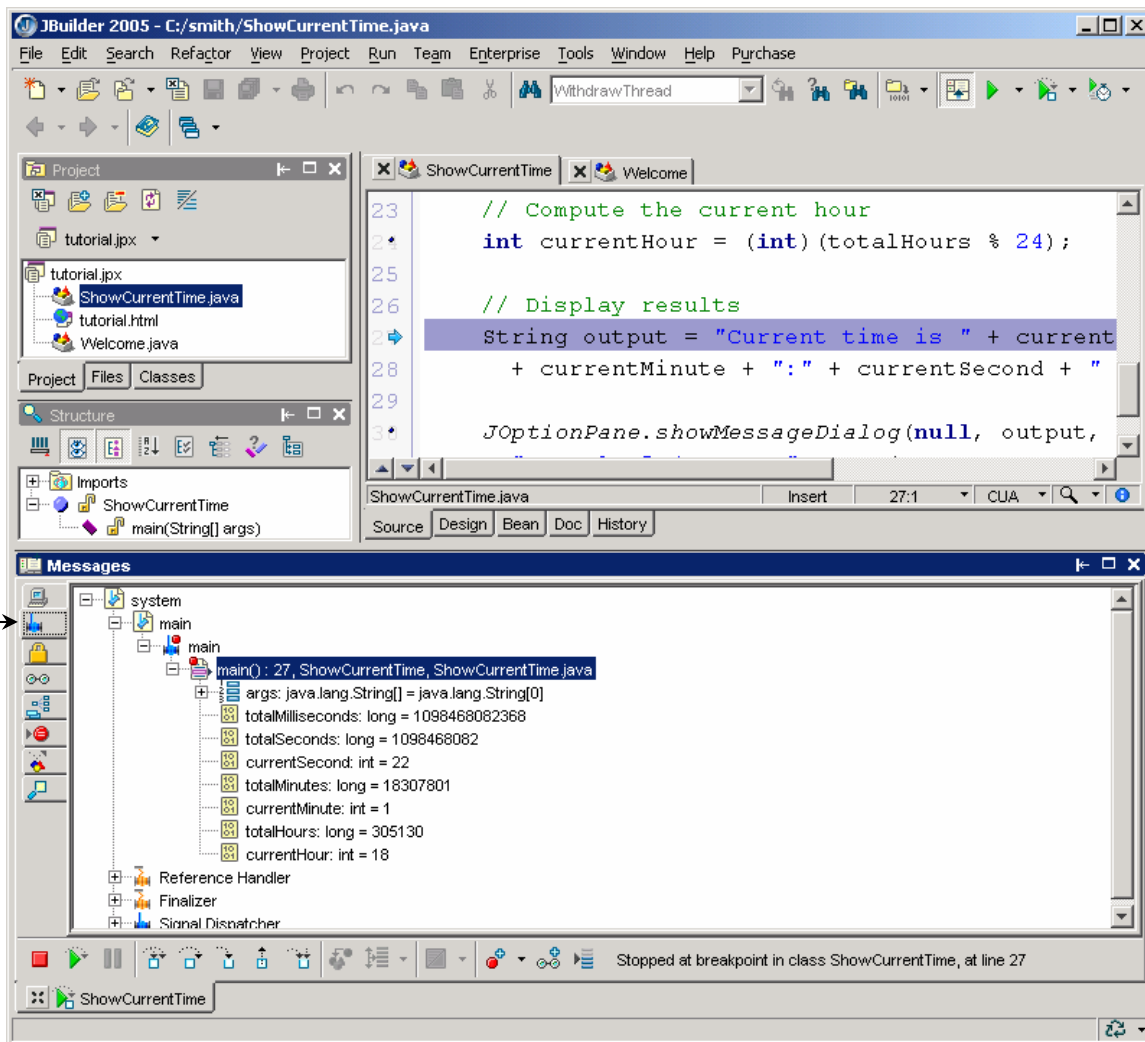
- **Step Over** executes a single statement. If the statement contains a call to a method, the entire method is executed without stepping through it.
- **Step Into** executes a single statement or steps into a method.
- **Step Out** executes all the statements in the current method and returns to its caller.
- **Run to Cursor** runs the program, starting from the current execution point, and pauses and places the execution point on the line of code containing the cursor, or at a breakpoint.
- **Run to End of Method** runs the program until it reaches the end of the current method or a breakpoint.
- **Resume Program** continues the current debugging session or restarts one that has finished or been reset.
- **Reset Program** ends the current program and releases it from memory. Use Reset to restart an application from the beginning, as when you make a change to the code and want to run again from the beginning, or if variables or data structures become corrupted with unwanted values. This command terminates debugging and returns to the normal editing session.
- **Show Execution Point** positions the cursor at the execution point in the content pane.



**Figure 4.3**

*The debugging commands appear under the Run menu.*

TIP: The Thread tab is very useful. As shown in Figure 4.4, the values of the variables are displayed as you execute the program line by line.



**Figure 4.4**

*The overall execution information is displayed in the Thread tab.*

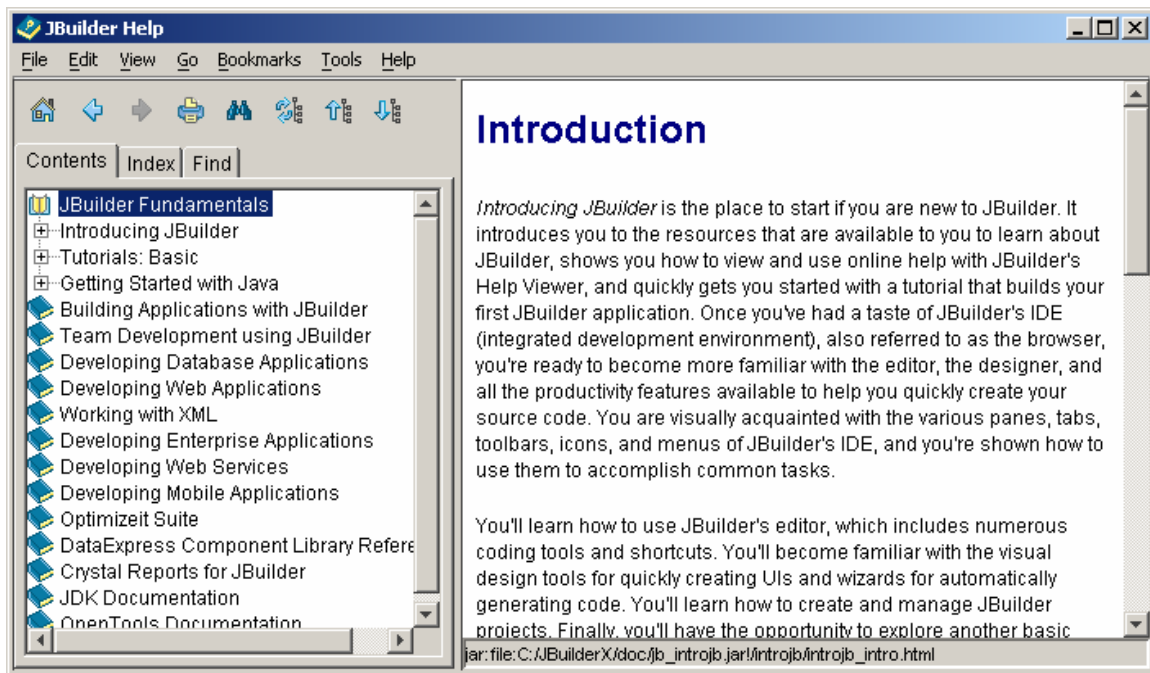
NOTE: The debugger is an indispensable, powerful tool that boosts your programming productivity. It may take you some time to become familiar with it, but your investment will pay off in the long run.

## 5 Getting Help in JBuilder

JBuilder provides a large number of documents online, giving you a great deal of information on a variety of topics pertaining to the use of JBuilder and Java.

### 5.1 Accessing from the Help Menu

To access online help, choose *Help, Help Topics* to display JBuilder Help, as shown in Figure 5.1.



**Figure 5.1**

*All help documents are displayed in JBuilder Help.*

JBuilder Help behaves like a Web browser and contains the main menus, navigation pane, and content pane. From the main menus, you can open a URL from the File menu, add bookmarks from the Bookmarks menu, and get help on using JBuilder Help from the Help menu.

The navigation pane contains eight action buttons on top of the three tabs. The buttons are *Home*, *Previous*, *Next*, *Print*, *Find in Page*, *Synchronize Table of Contents*, *Previous Topics*, and *Next Topic*. The Home, Previous, and Next buttons let you go to the first, previous, and next topics in the history list. The Print button prints the document in the content pane. The Find in Page button enables you to search the current topic. The Synchronize Table of Contents button synchronizes the topic with the contents in the content pane. The Previous Topic and Next Topic buttons let you go to the previous and next topics.

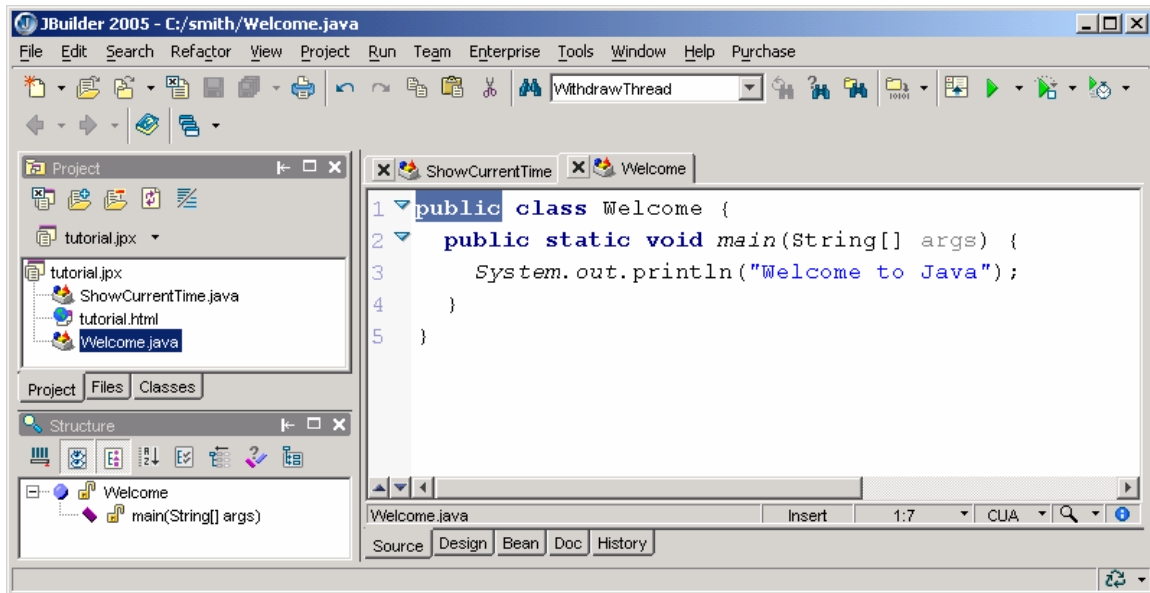
The three tabs are *Contents*, *Index*, and *Find*. The Contents tab displays available documents. The table of contents of the document is displayed in a tree-like list in the navigation pane. To view a given topic, select the node in the tree associated with the topic. JBuilder Help displays the document for the topic in the content pane.

The Index tab shows the index entries for the current document. The Find page shows the combined index entries for

all the available documents in JBuilder. To display the index, simply type the first few letters in the entry. As you start typing, the index scrolls, doing an incremental search on the index entries to find the closest match. Select and double-click the index in the entry to display the document for the entry in the content pane.

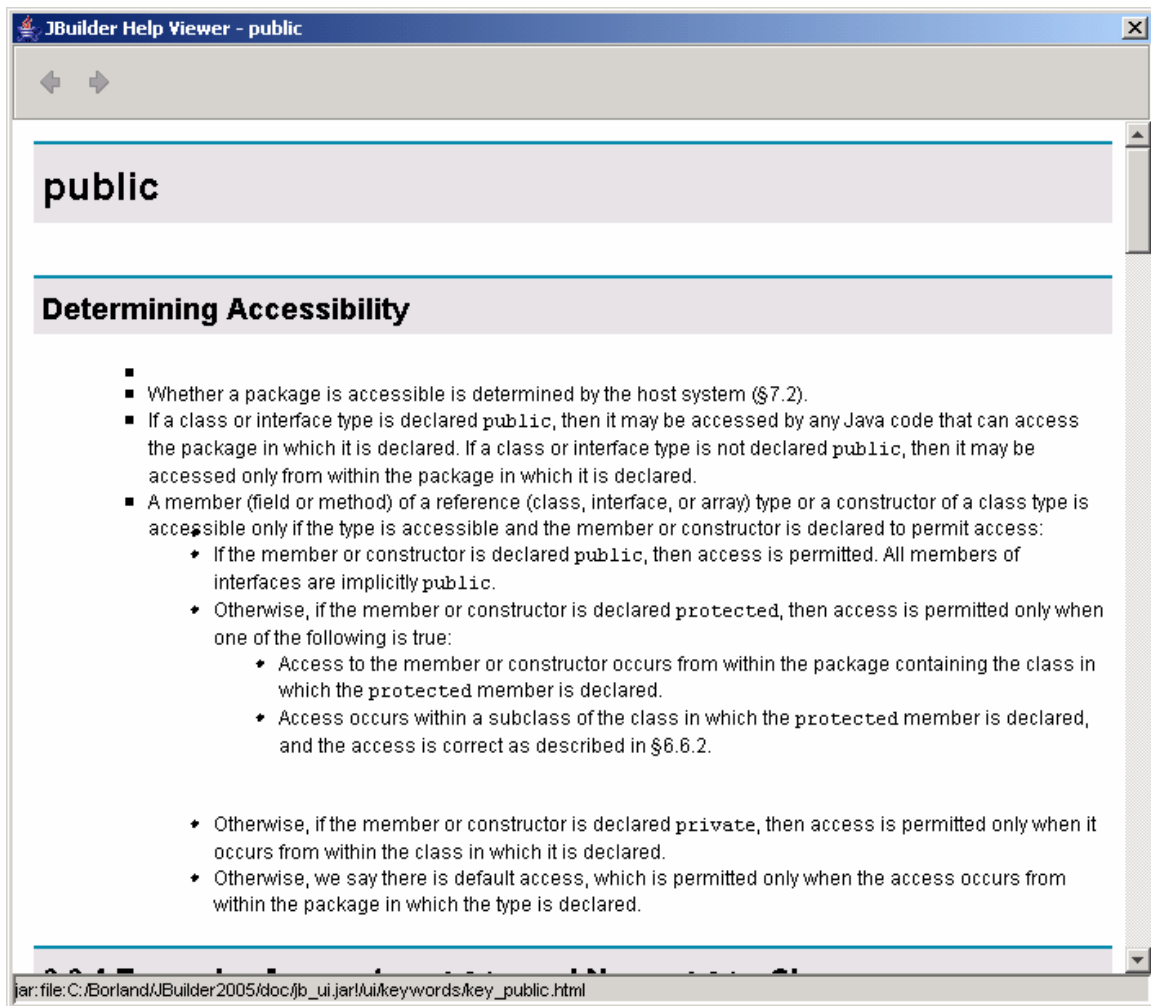
## 5.2 Obtaining Help on Java Keywords and Classes

To obtain help on a Java keyword, highlight it in the content pane and press F1. For example, if you highlight public in the content pane, as shown in Figure 5.2, and press F1, you will see the help on public displayed in Figure 5.3.



**Figure 5.2**

*Highlight public and press F1 to display the documentation on the keyword public.*



**Figure 5.3**

*The documentation on the keyword public is displayed in JBuilder Help.*

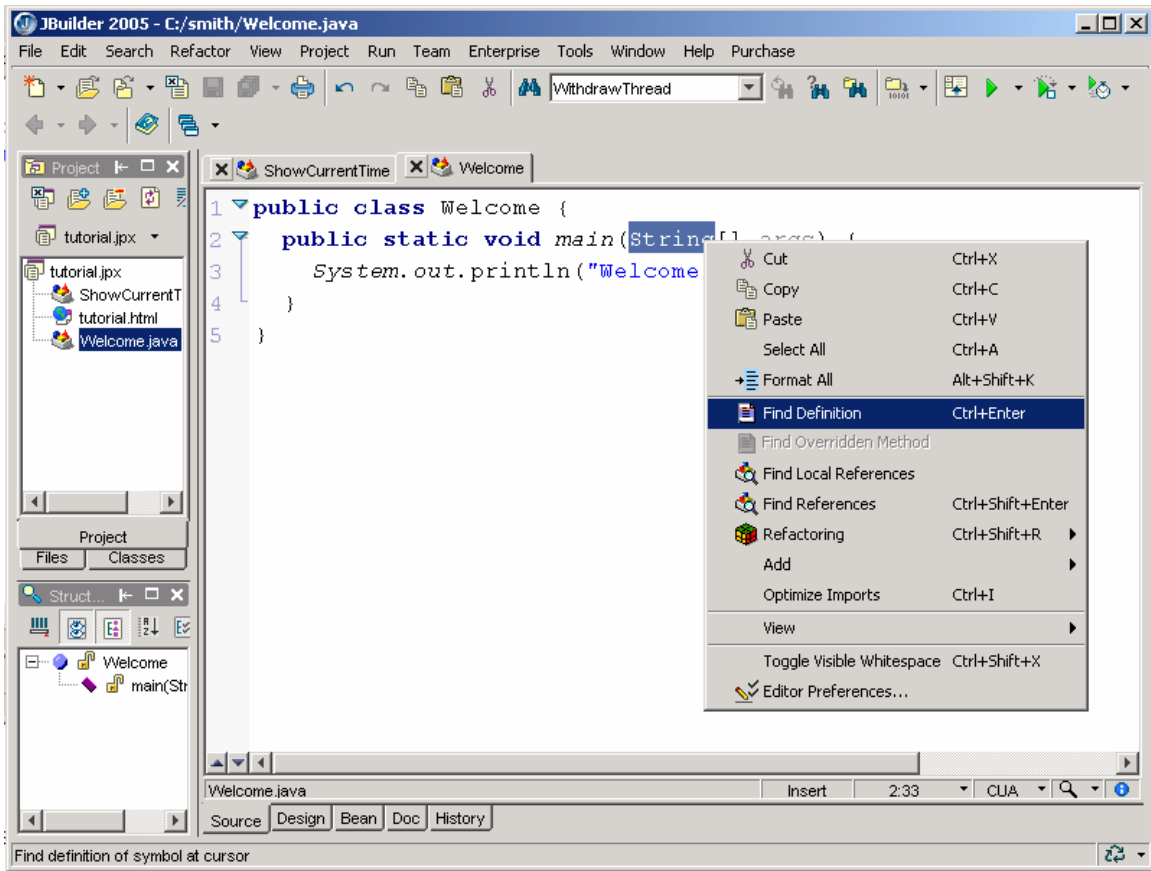
To display the documentation on a Java class, highlight the class name (e.g., String) in the content pane and press F1. The documentation for the class is displayed in JBuilder Help, as shown in Figure 5.4.



**Figure 5.4**

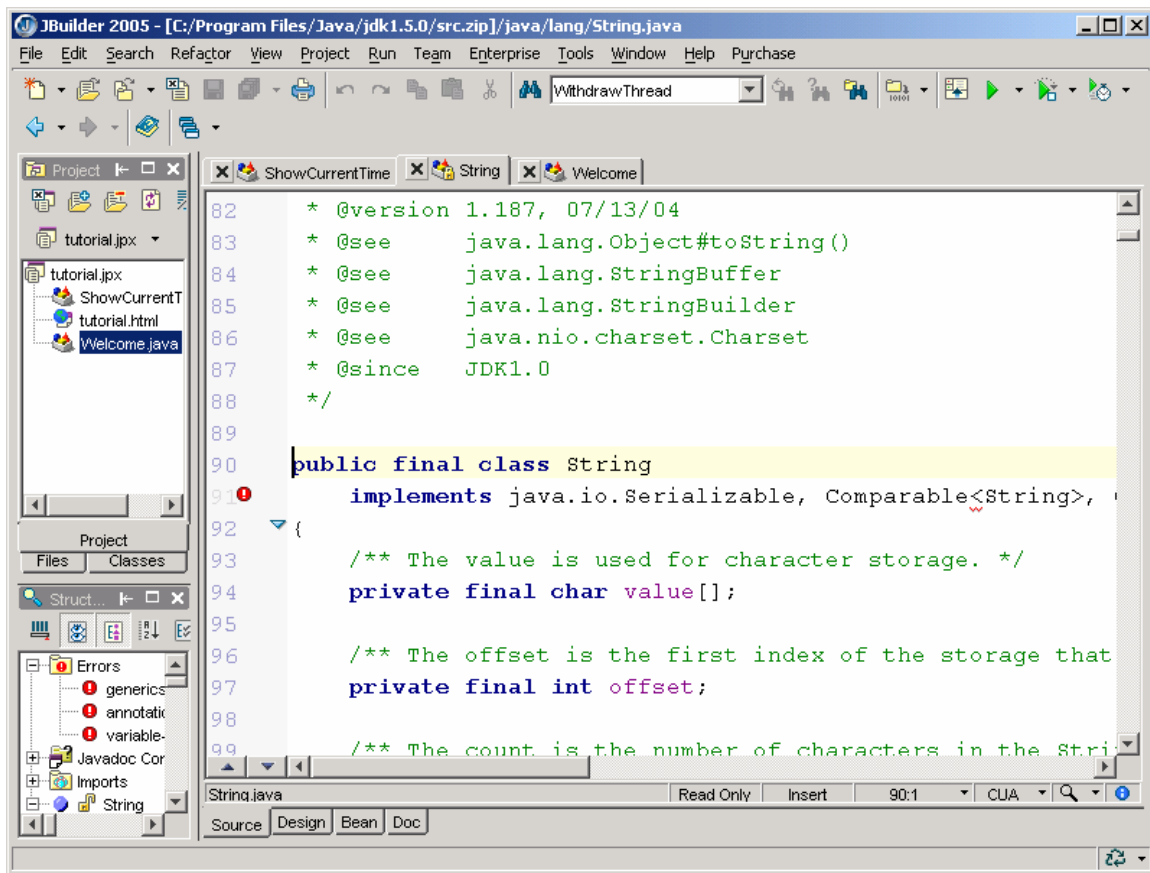
*The documentation on the `String` class is displayed in JBuilder Help.*

You can also display the source code of the class by choosing *Find Definition* in the context menu of the class in the content pane, as shown in Figure 5.5. The class source code is displayed in the content pane, as shown in Figure 5.6.



**Figure 5.5**

*You can display the source code of a class in the Java API.*



**Figure 5.6**

*The source code of the class is displayed in the content pane.*

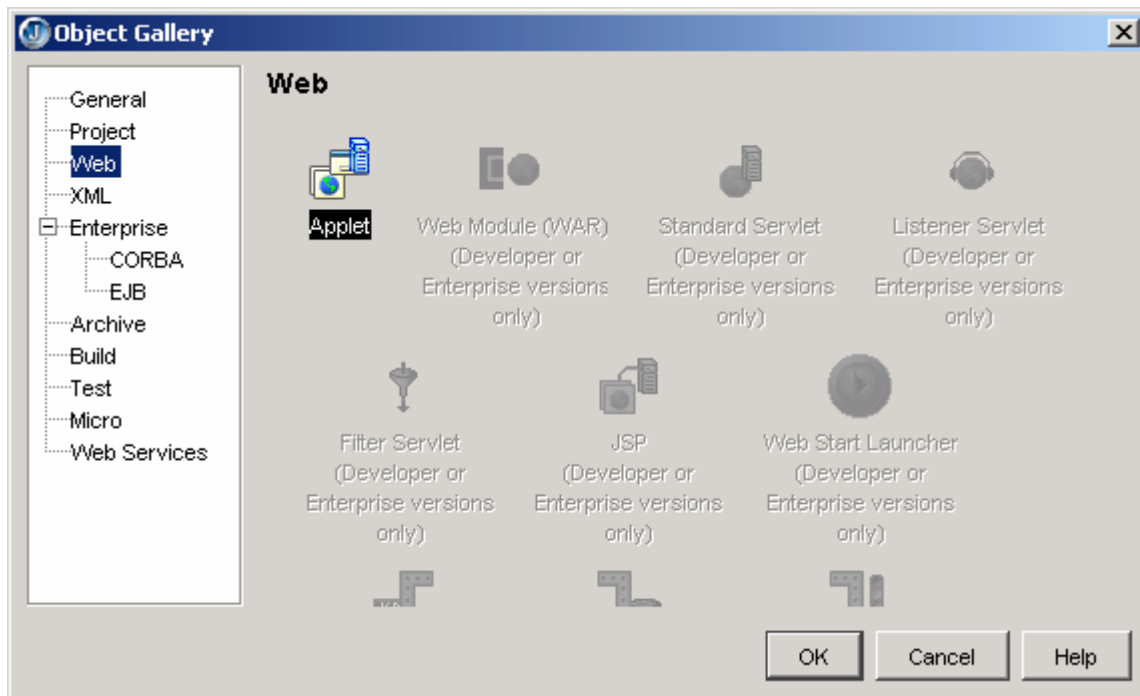
## 6 Creating and Testing Java Applets

You have learned how to create, compile, and execute a Java program. Applets are special type of Java program. JBuilder provides the Applet wizard that can be used to create Java applets.

### 6.1 Creating a Java Applet

The following steps create template files for a new applet:

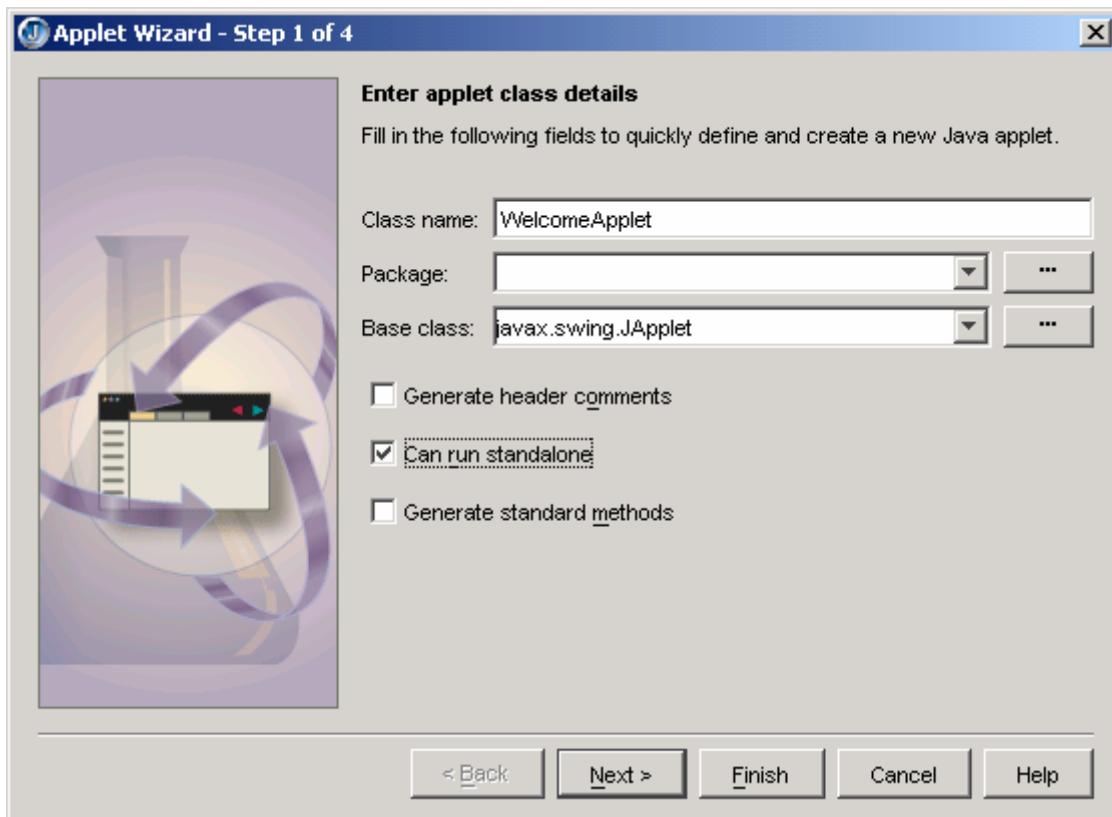
1. Choose *File, New* to display the Object Gallery, as shown in Figure 6.1. In the Web page of the Object Gallery, click the Applet icon to open the Applet wizard.



**Figure 6.1**

*The Object Gallery contains many useful wizards for creating various types of Java programs.*

2. JBuilder starts Applet Wizard Step 1 of 4 to configure your applet (see Figure 6.2). Enter WelcomeApplet in the Class name field, leave the Package field empty, choose javax.swing.JApplet in the Base class field, and check the *Generate header comments* and *Can run standalone* options as shown in Figure 6.2. Click *Finish* to generate the template files for the applet.



**Figure 6.2**

*Applet wizard Step 1 of 3 prompts you to enter the package name, the applet class name, and other optional information.*

NOTE: You may choose the *Next* button to display Step 2 of 4 of the Applet wizard to enter the parameters for the applet.

The Applet wizard generates two files: `WelcomeApplet.html` and `WelcomeApplet.java`. The source code for `WelcomeApplet.html` is shown in the content pane of the `AppBrowser` in Figure 6.3. The source code for `welcomeApplet.java` is shown in Figure 6.4.

NOTE: The HTML file `WelcomeApplet.html` is stored in the project source path directory (i.e., `c:\smith`) and the Java source file `WelcomeApplet.java` is stored in the project source path directory/`packageName` (i.e., `c:\smith`). To view the code in `WelcomeApplet.html`, you need to choose the *Source* tab at the bottom of the content pane.

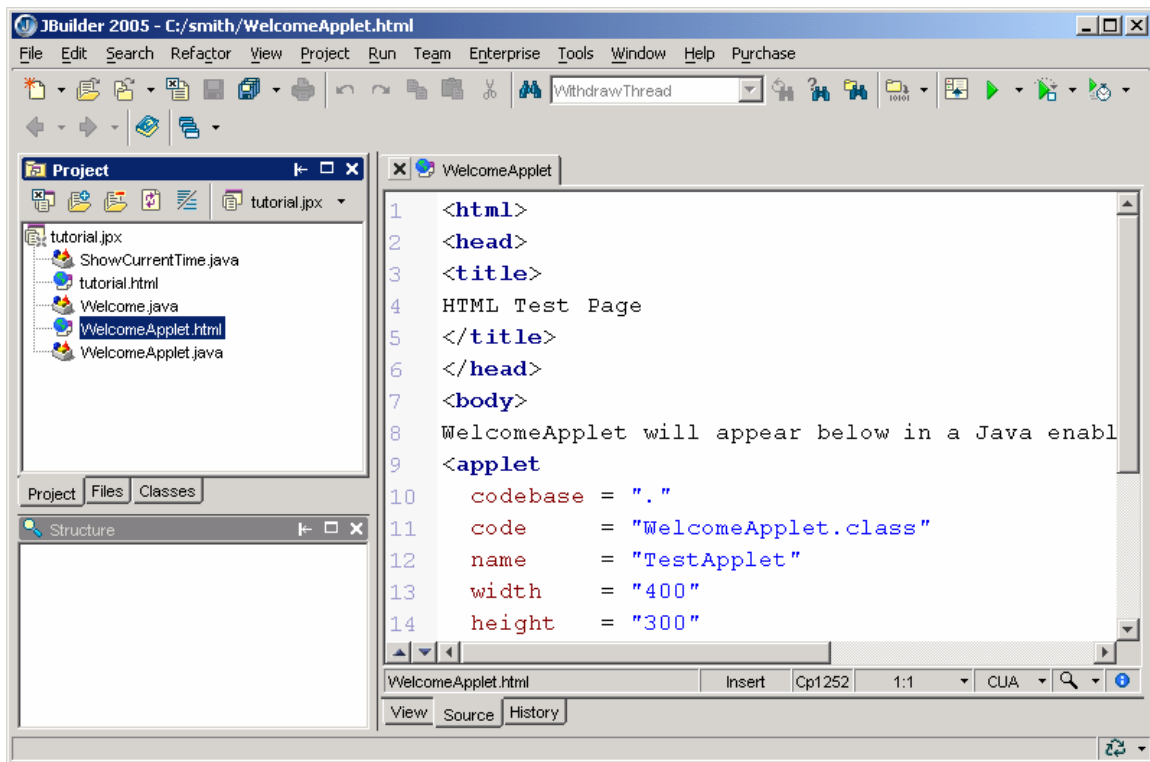
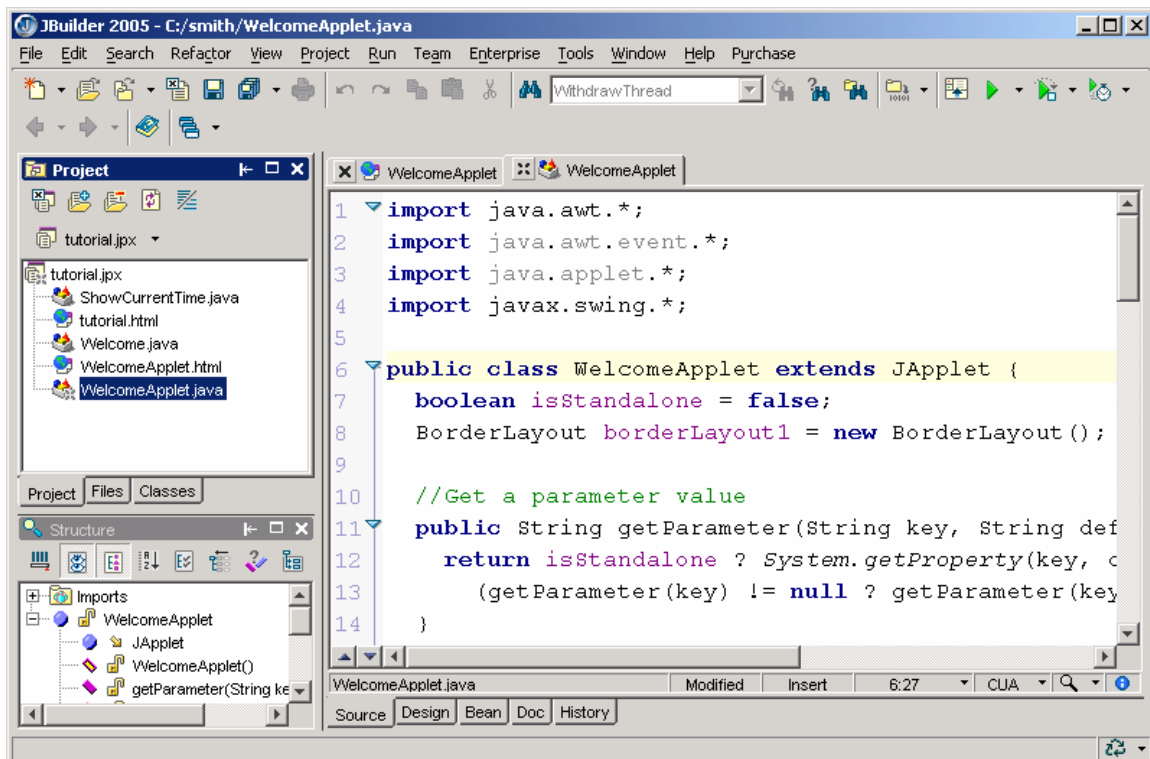


Figure 6.3

*The source code of WelcomeApplet.html is shown in the content pane.*



**Figure 6.4**

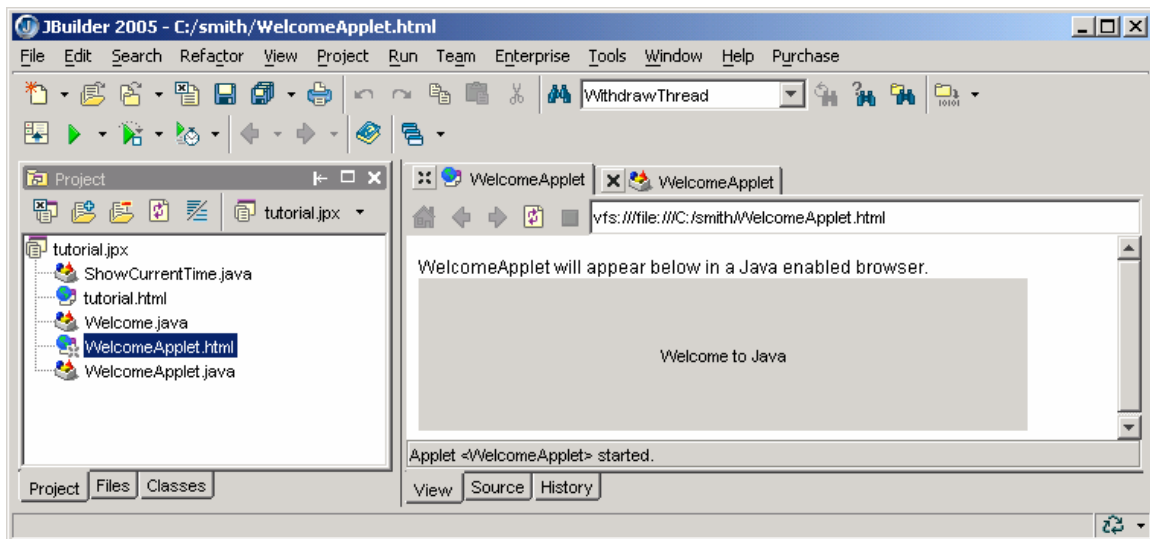
*The source code of WelcomeApplet.java is shown in the content pane.*

## 6.2 Modifying Applets

Replace WelcomeApplet.java with the code on Page XXX in the book to display the text "Welcome to Java" in the applet.

## 6.3 Viewing Applets in the Content Pane

You can view the applet in the content pane using the JBuilder's applet viewer by selecting the applet's HTML file (e.g., WelcomeApplet.html) in the project pane and choosing the View tab in the content pane, as shown in Figure 6.5.



**Figure 6.5**

*The applet is displayed in the content pane inside JBuilder IDE.*

## 6.4 Viewing Applets Using the Applet Viewer Utility

You can also view a Java applet using the Sun applet viewer. Choose the applet's HTML file (e.g., WelcomeApplet.html) in the project pane. Right-click it to display its context menu. Click *Run using defaults* in the context menu. The applet is displayed in the applet viewer as shown in Figure 6.6.



**Figure 6.6**

*The WelcomeApplet program is running from the applet viewer.*

You can also invoke the applet viewer from the DOS prompt using the following command:

```
appletviewer WelcomeApplet.html
```

### 6.5 Viewing Applets from a Web Browser

Applets are eventually displayed in a Web browser. Using JBuilder's applet viewer and Sun's applet viewer, you do not need to start a Web browser. Both applet viewers function as browsers. They are convenient for testing applets before deploying them on a Web site. JBuilder's applet viewer has limited functions. For example, you cannot use menus from JBuilder's applet viewer. Sun's applet viewer fully simulates a Web browser. To display an applet from a Web browser, open the applet's HTML file (e.g., WelcomeApplet.html). Its output is shown in Figure 6.7.

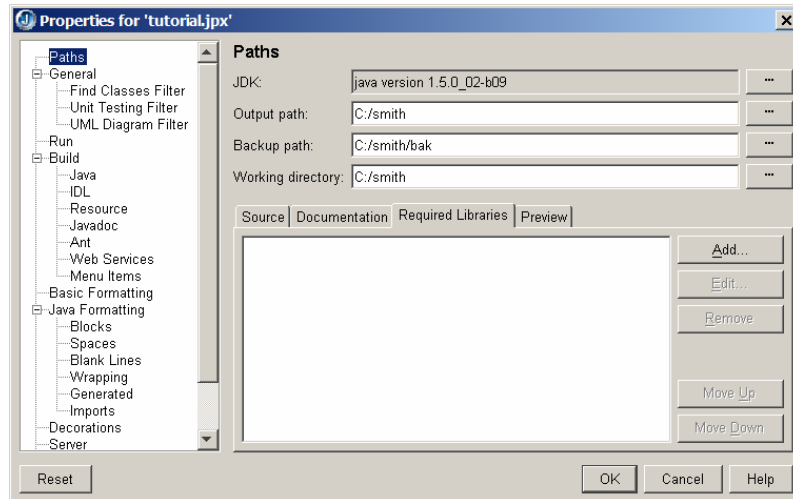


**Figure 6.7**

*The WelcomeApplet program is displayed in Internet Explorer.*

## 7 Adding Classes to the Library

If your project uses a Java class file not in the Java API, the class must be in the project output path or must be added into the Required Library page in the Project Properties dialog, as shown in Figure 7.1.

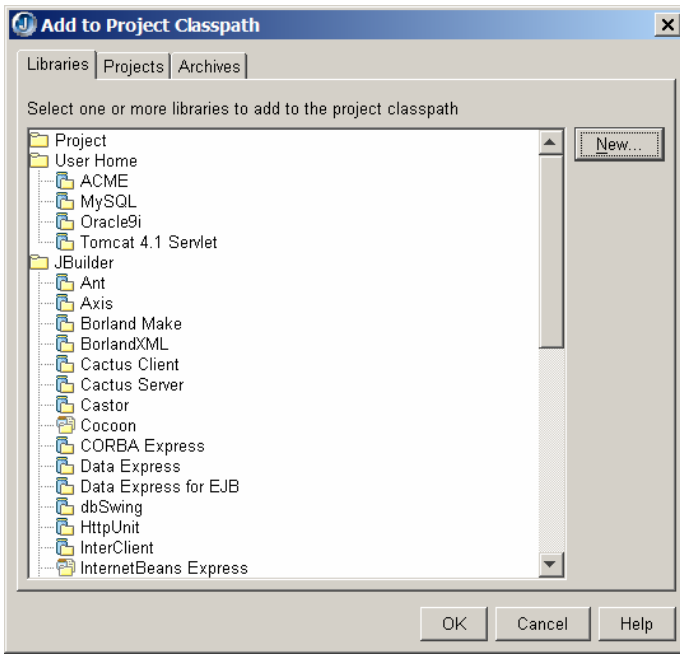


**Figure 7.1**

*You can add .class files, .jar files, and .zip files in the Required Libraries page.*

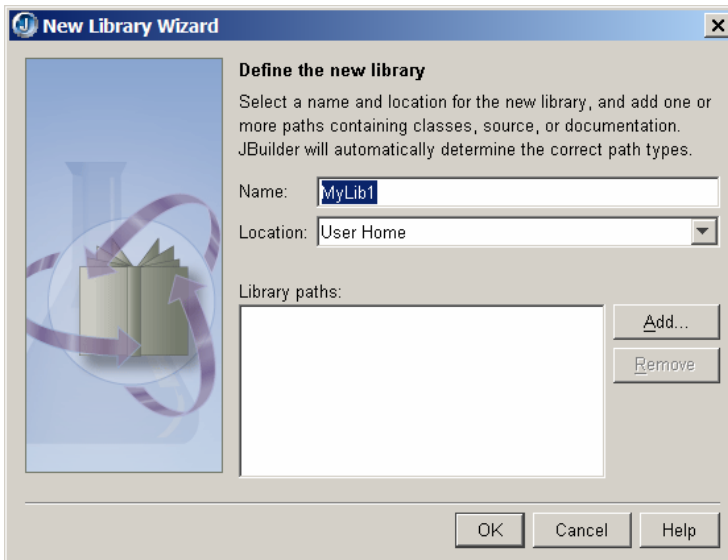
For example, suppose you have a JAR file named `classes12.jar` located in `c:\book\classes12.jar`. Your project needs to use the classes in this JAR file. You can add it into the library as follows:

1. In the Path node of the Project Properties dialog, choose Required Libraries, as shown in Figure 7.1.
2. Click the Add button to display the Add to Project Classpath dialog, as shown in Figure 7.2.
3. Click the New button to display the New Library wizard, as shown in Figure 7.3. Type `MyLib1` in the Name field as the name for the new library.
4. Click the Add button in Figure 7.3 to display a dialog to select one or more directories, as shown in Figure 7.4. Select `classes12.jar` under `c:\book` and click OK. You will see the file added in the New Library wizard, as shown in Figure 7.5.
5. Click OK to dismiss the New Library wizard. You will see `MyLib1` added in the Add to Project Classpath dialog, as shown in Figure 7.6.
6. Click OK to add `MyLib1` to the Required Libraries page, as shown in Figure 7.7.
7. Click OK to close the Project Properties dialog.



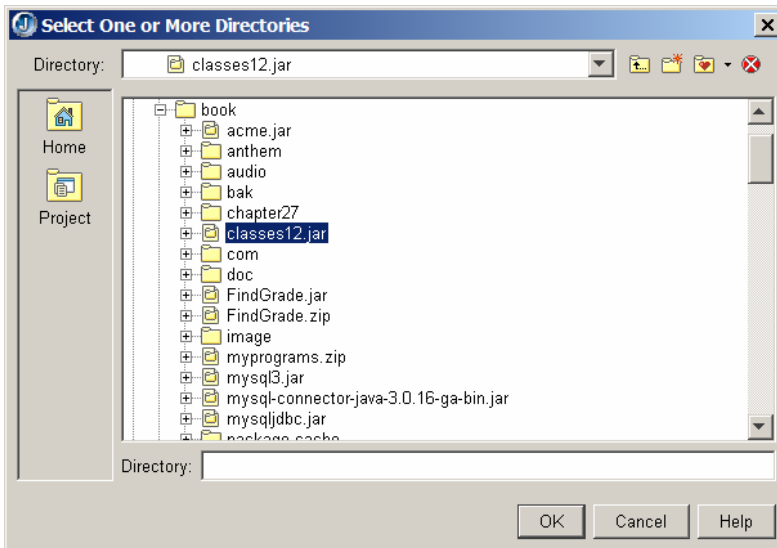
**Figure 7.2**

*You can create add or create a new library from the Library tab.*



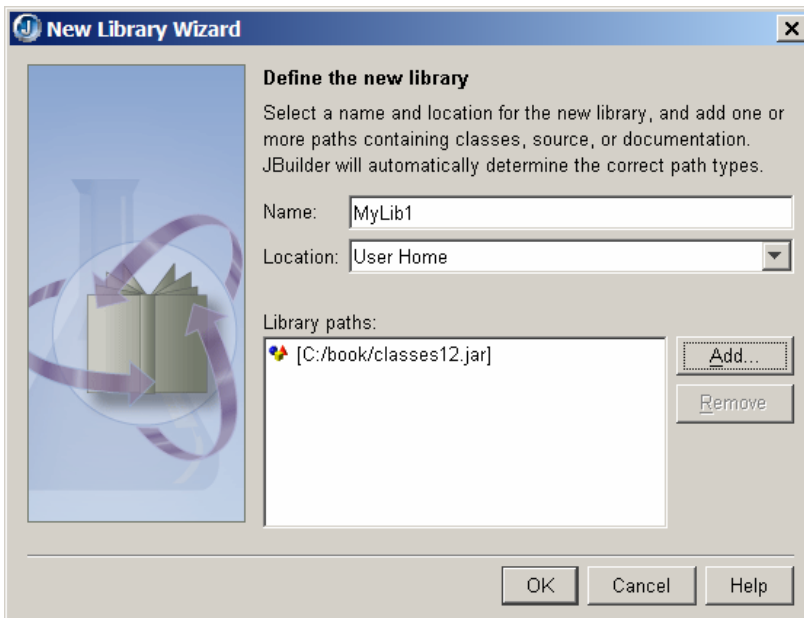
**Figure 7.3**

*You can create a new library in the New Library wizard.*



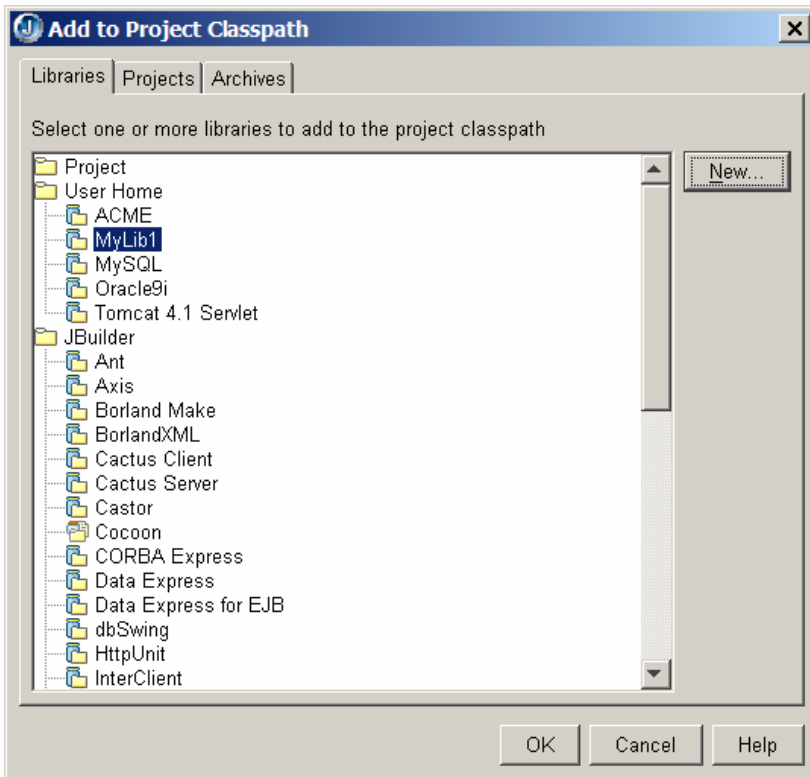
**Figure 7.4**

*You can select one or more directories or JAR files to create a library.*



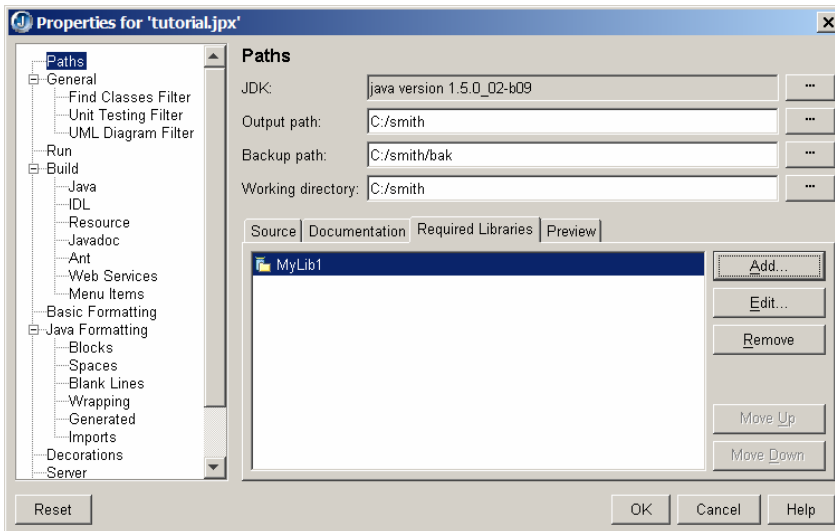
**Figure 7.5**

*A new library is created for the JAR file.*



**Figure 7.6**

*A new library is added in the User Home.*



**Figure 7.7**

*A new library is added in the Required Libraries page.*