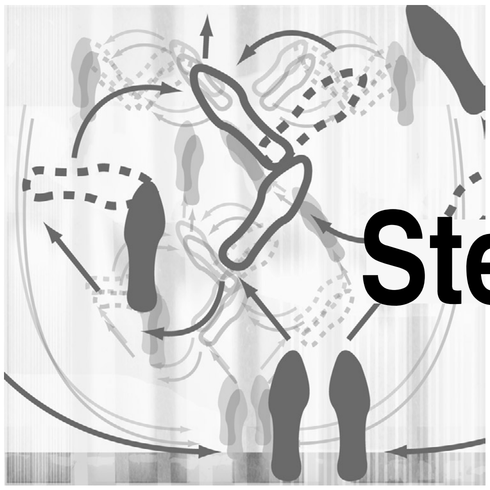


Step 7

A simple program

The book up to now has primarily been a familiarization with the Eclipse workbench. You didn't really create anything usable. However, now you're going to create a program that can actually display data from a database.

Not all at once, though! Instead, we'll proceed slowly. First, you'll create a very simple program, and then you'll debug it. After that you'll add a user interface and finally a database. This step will cover only that first task: creating the initial simple program.

✔ Here is your step checklist:

❑ 7.3(c) Leave Package blank 116

❑ 7.3(d) Leave Enclosing type unchecked 116

❑ 7.3(e) Enter HelloWorld in the Name field 116

❑ 7.3(f) Make sure public is checked, abstract and final are unchecked 116

❑ 7.3(g) Leave Superclass as java.lang.Object, Interfaces blank. 116

❑ 7.3(h) Make sure public static void main(String[] args) is checked 116

❑ 7.3(i) Leave the other two unchecked 116

❑ 7.3(j) Click Finish 116

❑ 7.4(a) Enter the following source code as your main() method 119

❑ 7.4(b) Right-click in the editor pane and select Save 120

❑ 7.4(c) Insert the included CD-ROM into your CD-ROM drive 121

❑ 7.4(d) Right-click on the Hello project and select Import 121

❑ 7.4(e) Select File system in the Import wizard and click Next 122

❑ 7.4(f) Enter “R:\Source\Step 7” in the From directory field, where R is
the drive letter of your CD-ROM drive, and press the Tab key 123

❑ 7.4(g) Left-click on the Step 7 folder 124

❑ 7.4(h) Select HelloWorld.java and click Finish. 125

❑ 7.4(i) Click Yes on the confirmation box that pops up. 125

❑ 7.5(a) Expand HelloWorld.java 127

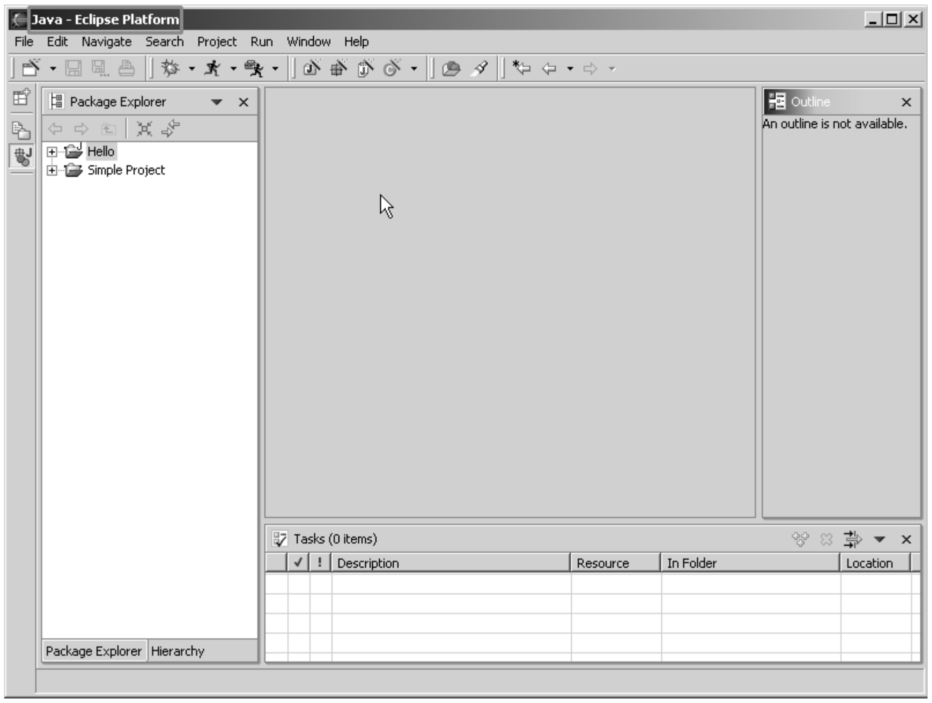
Step 7.1—Get into the Java perspective

GOAL

In this step, you will open the Java perspective using Eclipse's main menu bar.

If you're already in the Java perspective, you can skip this step and continue with Step 7.2.

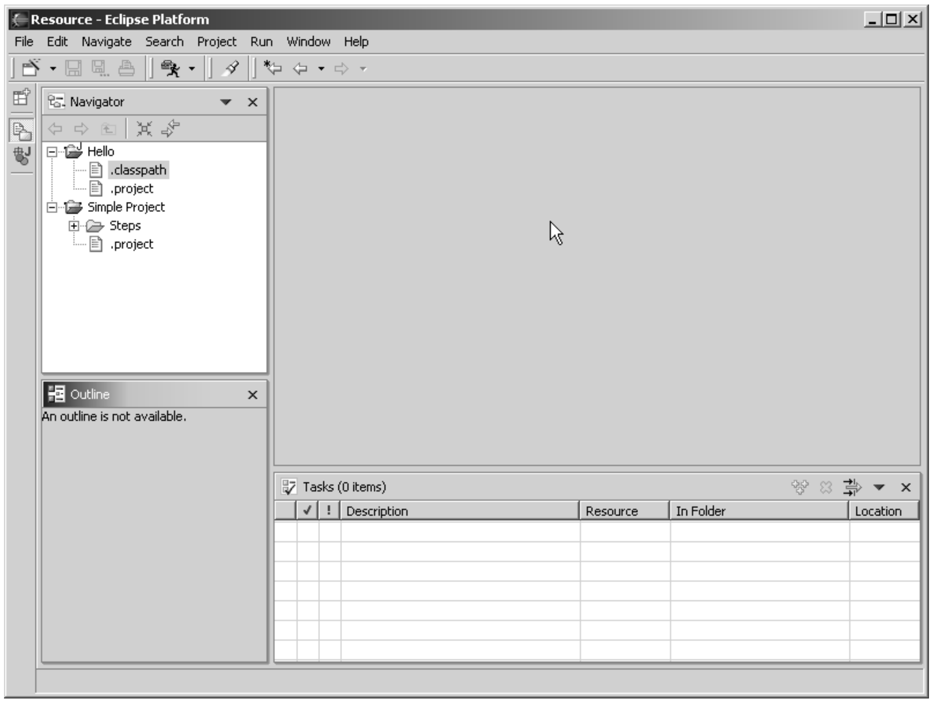
Note: The Title Bar of the Eclipse IDE will indicate which perspective you're currently in. If your Title Bar says "Java - Eclipse Platform" as in Figure 7.1, then you're ready to move on.



An outline is not available.

Figure 7.1: The Java Perspective, as indicated by the title bar.

Okay, for whatever reason, you're not in the Java perspective. Maybe you've skipped the first part of the book, or you were exploring, or whatever. You might be in the Resource explorer, as shown in Figure 7.2.



An outline is not available.

Figure 7.2: A possible alternative view of the IDE; in this case, the Resource perspective.

No matter where you are, our job now is to make sure you get to the proper perspective. That's actually quite easy. You can get to the Java perspective by using Eclipse's main menu bar.

❑ **7.1(a) From Eclipse's main menu bar, select Window/Open Perspective/Java.**

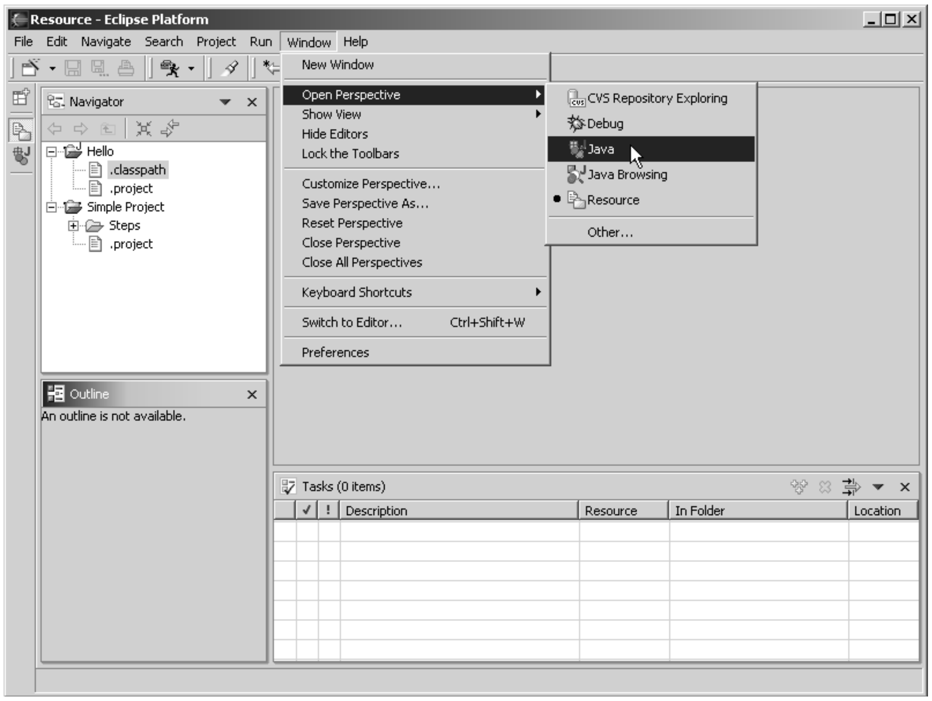


Figure 7.3: Select Window/Open Perspective/Java from the main menu bar.

The Java perspective will appear.

Step 7.2—Create the Java project

GOAL

In this step, you will create a Java project.

If you already see a project named Hello in your Package Explorer, as in Figure 7.4, then you may skip ahead to Step 7.3.

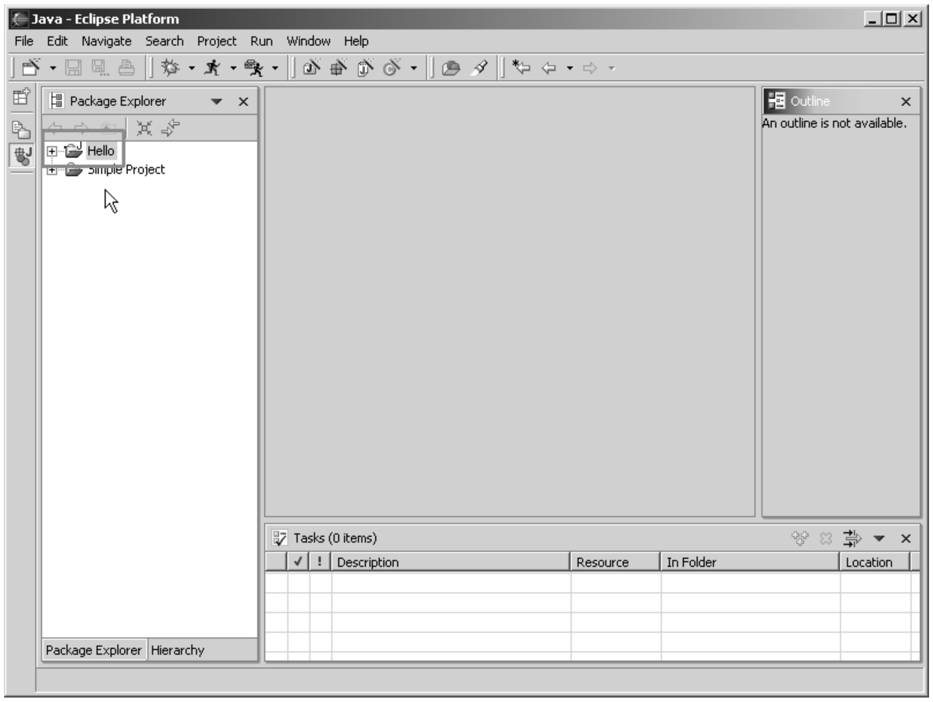


Figure 7.4: The Java perspective showing the Hello project.

Otherwise, you'll need to create a project. To do this, you'll simply do the same actions as outlined in Step 5.2.

❑ **7.2(a) Right-click on the Navigator pane and select New/Project . . .**

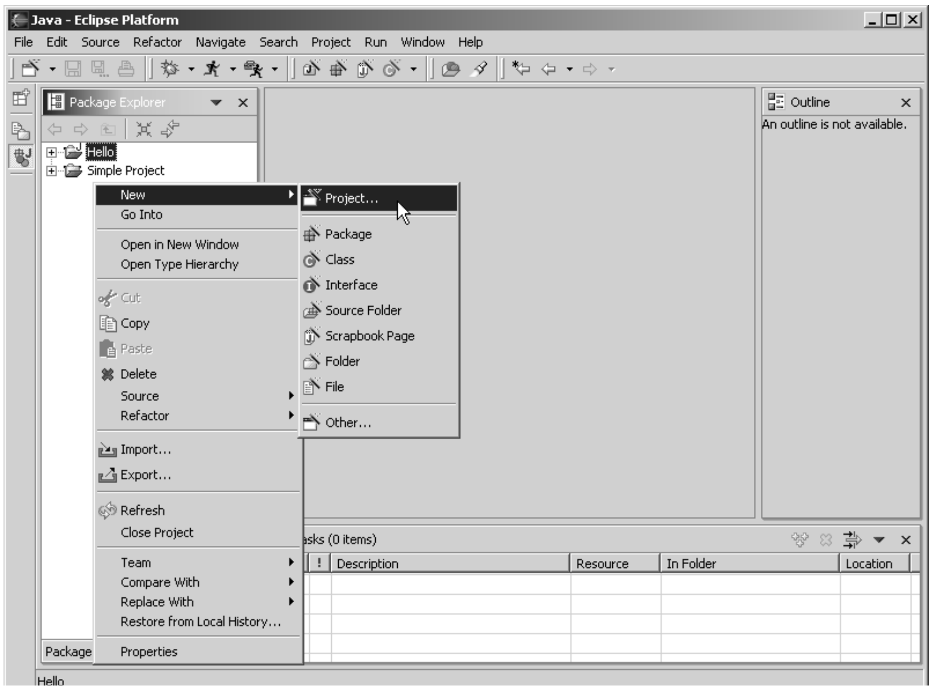


Figure 7.5: Bringing up the New Project wizard by right-clicking on the Navigator pane and selecting New/Project . . .

The New Project wizard will appear. Use it to select the type of project you are creating. For more complex projects, you may be asked for more information, but for a basic Java project, it’s pretty simple.

❑ **7.2(b) Select Java on the left, Java Project on the right, and click Next.**

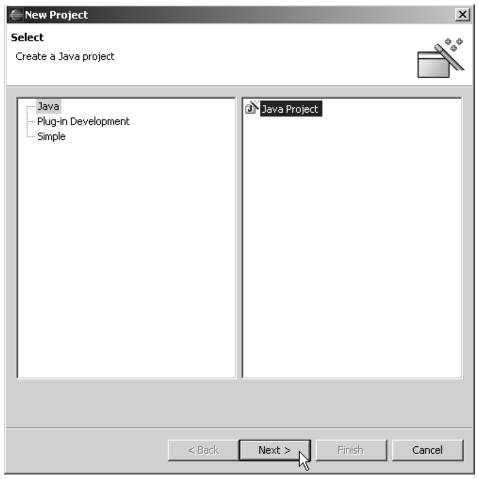
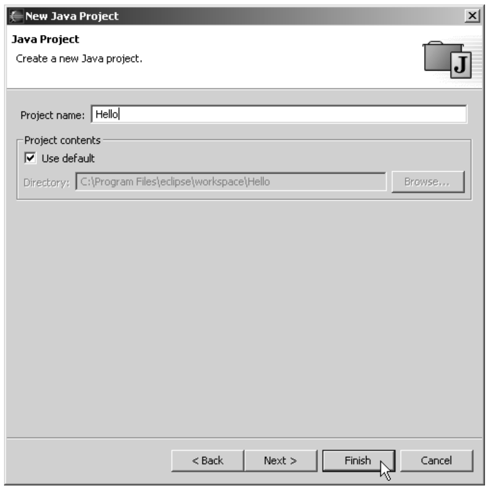


Figure 7.6: Select Java and Java Project on the left and right, and click Next.

Enter the name, click Finish, and you're done.

❑ **7.2(c) Type in "Hello" and click Finish.**



Project name: Hello

☑ Use default

Directory: C:\Program Files\eclipse\workspace\Hello

Figure 7.7: Finish the wizard by entering the name and clicking Finish.

Step 7.3—Creating a new class

GOAL

In this step, you will create a new Java class that you can actually run and debug (although debugging will wait for Step 8).

Creating a class is quite easy, but how you do it depends on what you plan to do with the class. For this step, you'll be creating a class that you can actually run from the command line (or the console, as it's called within Eclipse).

❑ **7.3(a) Right-click on Hello in the Package Explorer and select New/Class.**

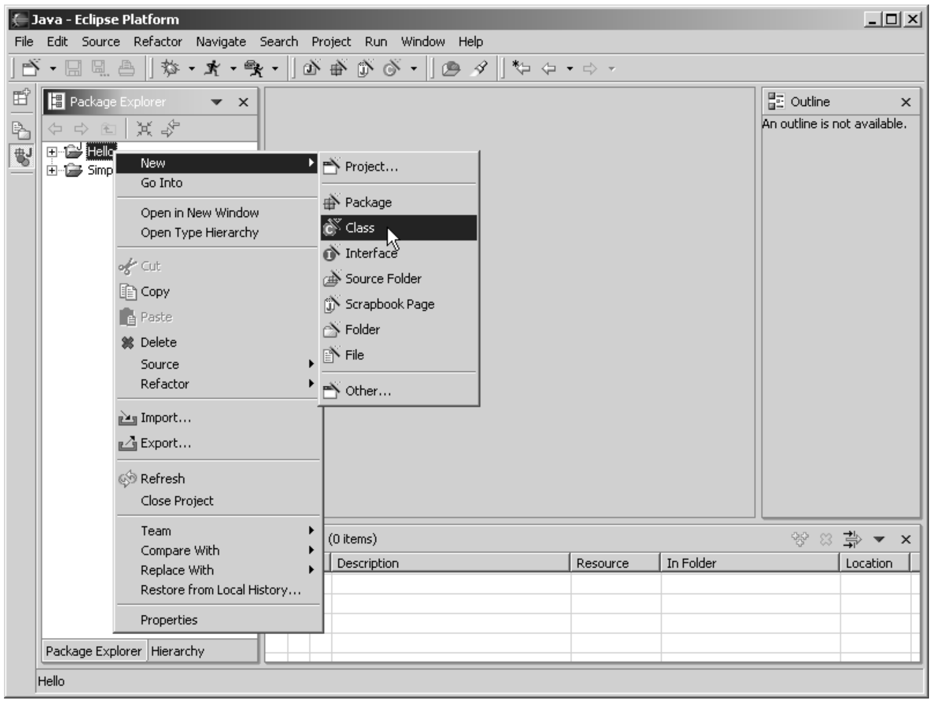
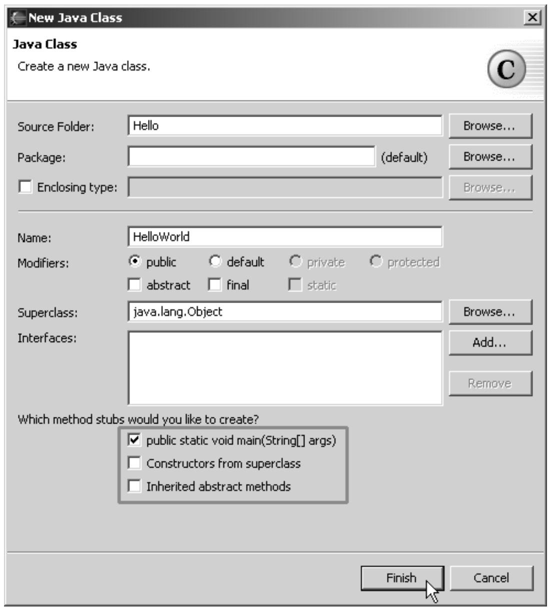


Figure 7.8: Creating a new class using New/Class.

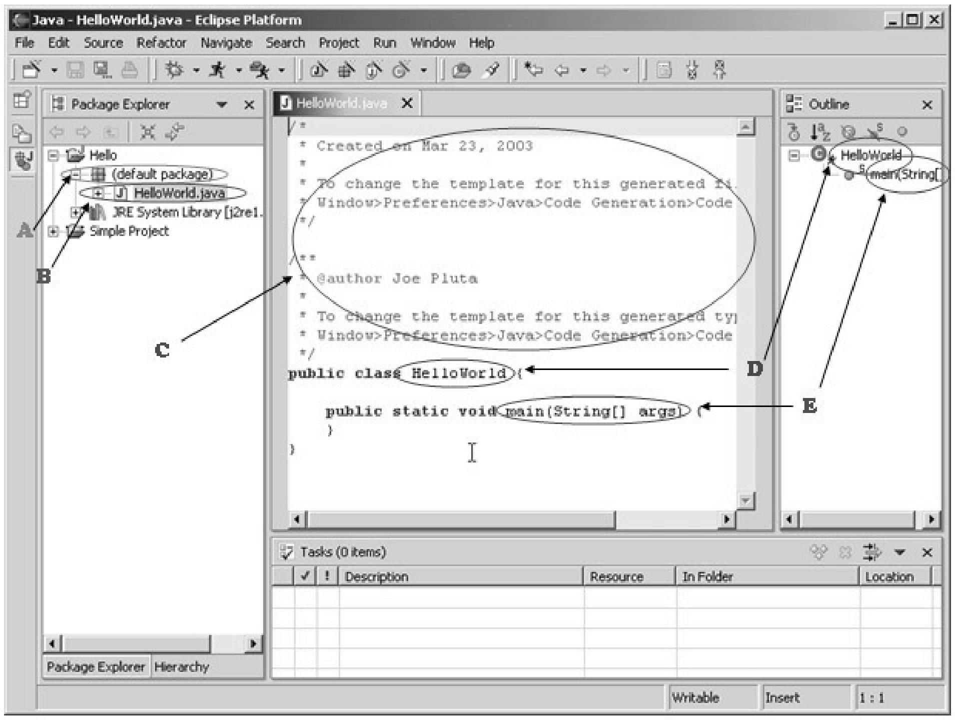
- ❑ **7.3(b) Make sure the Source Folder is Hello.**
- ❑ **7.3(c) Leave Package blank.**
- ❑ **7.3(d) Leave Enclosing type unchecked.**
- ❑ **7.3(e) Enter HelloWorld in the Name field.**
- ❑ **7.3(f) Make sure public is checked, abstract and final are unchecked.**
- ❑ **7.3(g) Leave Superclass as java.lang.Object, Interfaces blank.**
- ❑ **7.3(h) Make sure public static void main(String[] args) is checked.**
- ❑ **7.3(i) Leave the other two unchecked.**
- ❑ **7.3(j) Click Finish.**



Source Folder: Hello
Package: (default)
☐ Enclosing type:
Name: HelloWorld
Modifiers: ● public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
☑ public static void main(String[] args)
☐ Constructors from superclass
☐ Inherited abstract methods

Figure 7.9: Setting the fields properly for a new class.

Following these steps will create a class HelloWorld in the default package (because you left Package blank) in project Hello. It will be a public class, neither abstract nor final. The class has no superclass (except for the implied superclass Object) and implements no interfaces. Finally, the IDE has been instructed to create a standard "main" method, but no other methods. That being the case, we will see the result as in Figure 7.10.



```
/*
 * Created on Mar 23, 2003
 *
 * To change the template for this generated fi
 * Window>Preferences>Java>Code Generation>Code
 */


/**
 * @author Joe Pluta
 *
 * To change the template for this generated ty
 * Window>Preferences>Java>Code Generation>Code
 */
public class HelloWorld {

    public static void main(String[] args) {
    }
}
```

Figure 7.10: The result of the addition of the new class HelloWorld.

There are a number of interesting results from this simple operation.

A. A default package is added, with the icon. The default package is added because you left the package name blank in Step 7.3(c). Had you entered a package name, that package would have been created instead. Had you used an already existing package, the class would simply have been added to that package.

B. The class HelloWorld was added. More correctly, the Java source code was added. That's what the icon in the Package Explorer indicates. The class is also created; we'll look into this a little further in a moment.

C. A class is added with some initial comments. These comments are pretty useless at this point, but you can change them.

Note: To change the default documentation for a new class, go into Window/Preferences. In the Preferences dialog, select Java/Templates and then edit the template named typecomment. You can modify quite a number of templates using this dialog.

D. The class itself is defined as HelloWorld. You'll also notice that the class is shown in the Outline view.

E. Last but not least, the main() method is created. It, too, has an entry in the Outline view.

Step 7.4—Write the code for the new class

GOAL

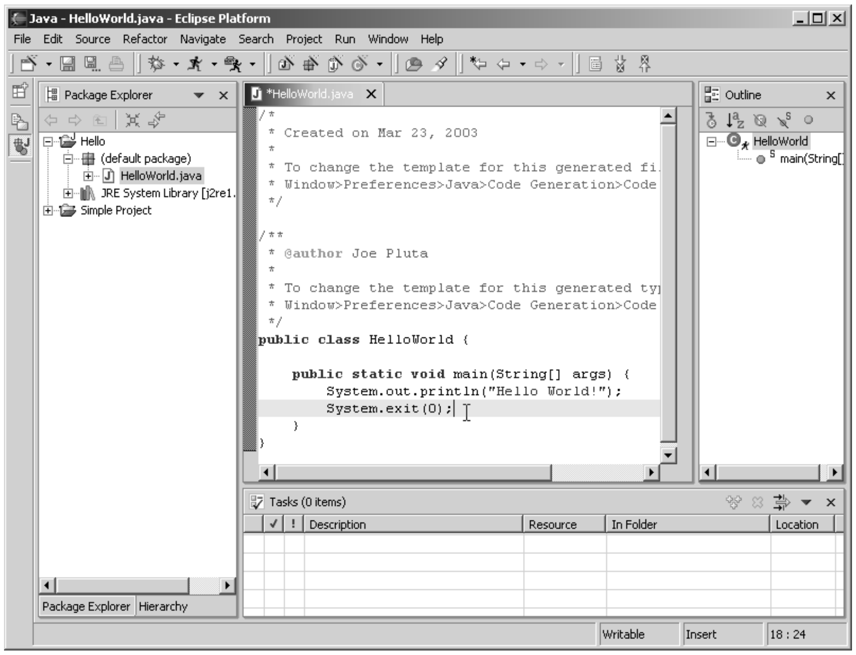
The last editing activity is to enter the code for this class. Up to this point, you've simply been defining a few basic attributes of your class, but now it' s time to write the code.

You may either enter the code directly into the editor or use the Import feature to import the source from the supplied CD-ROM. If you wish to enter the source yourself and get a feel for the source editor, use Option 1 below. Otherwise, skip ahead to Option 2.

Option 1—Source Entry

❑ **7.4(a) Enter the following source code as your main() method.**

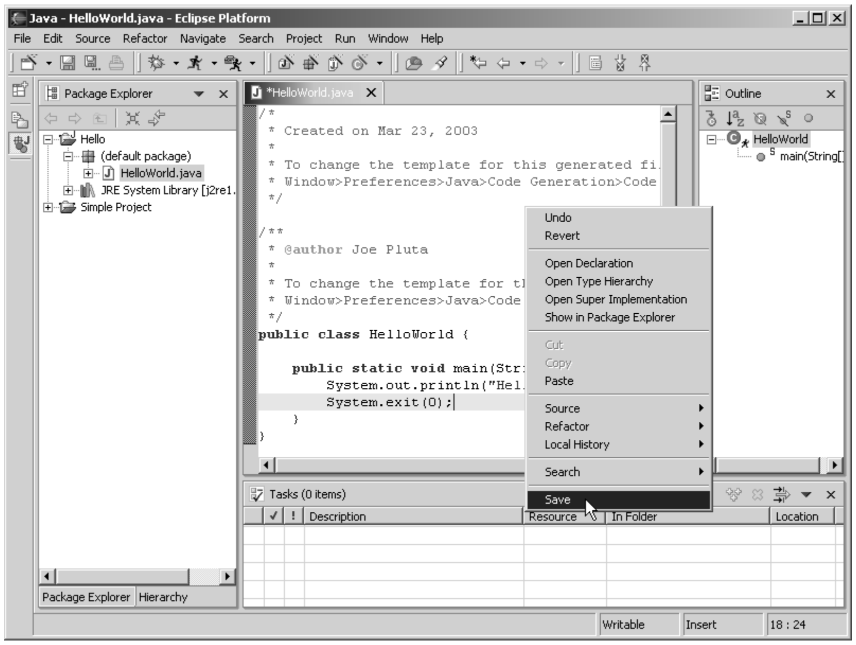
```
System.out.println("Hello World!");
System.exit(0);
```



```
/*
 * Created on Mar 23, 2003
 *
 * To change the template for this generated fi
 * Window>Preferences>Java>Code Generation>Code
 */

/**
 * @author Joe Pluta
 *
 * To change the template for this generated ty
 * Window>Preferences>Java>Code Generation>Code
 */
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.exit(0);
    }
}
```

Figure 7.11: Source code entered into the editor pane.

❑ **7.4(b) Right-click in the editor pane and select Save.**



```
/*
 * Created on Mar 23, 2003
 *
 * To change the template for this generated fi
 * Window>Preferences>Java>Code Generation>Code
 */

/**
 * @author Joe Pluta
 *
 * To change the template for tl
 * Window>Preferences>Java>Code
 */
public class HelloWorld {

    public static void main(Str:
        System.out.println("Hel.
        System.exit(0);
    }
}
```

Figure 7.12: Right-click in the editor pane and select Save to save the code.

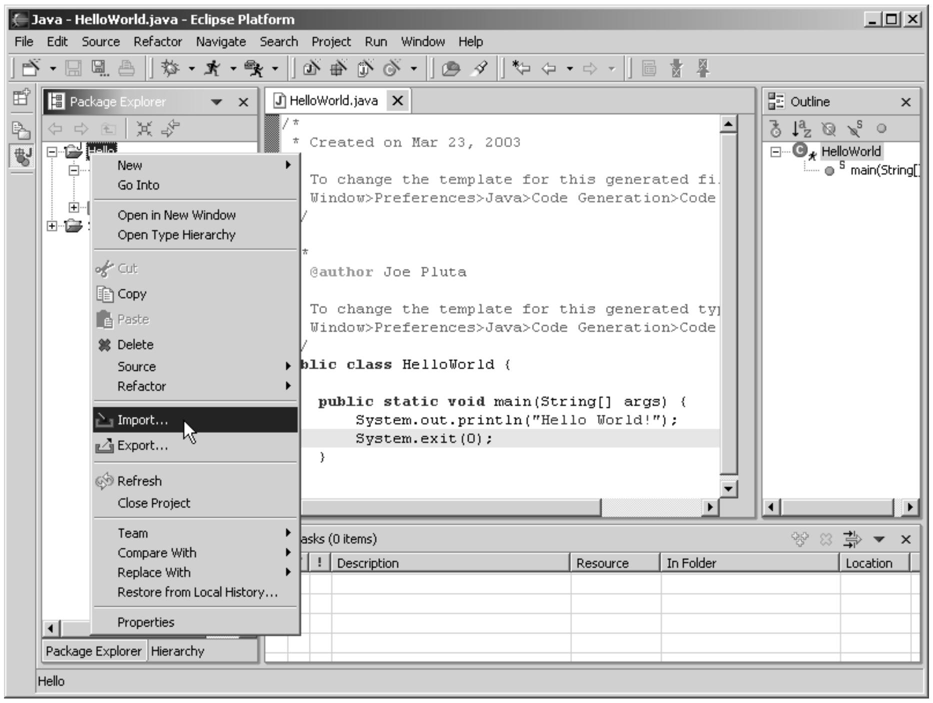
Another option is just to press Ctrl-S. At this point, you can skip ahead to Step 7.5. However, you might want to execute Option 2 anyway. This will import the code from the CD-ROM and overwrite the code you just entered. It will look exactly the same as what you typed in except for the author name, and you'll learn how to use the Import feature, which I'm sure you'll be using in the future.

Option 2—Importing Source from CD

Importing is quite easy. The only part that can get a little confusing is the relationship between folders and packages. When you are importing from a file system, you must be certain to import from the correct point in the file tree. To make it easy for this book, I have located the source for each step in a folder named for that step, as you are about to see.

❑ **7.4(c) Insert the included CD-ROM into your CD-ROM drive.**

❑ **7.4(d) Right-click on the Hello project and select Import . . .**



```
/*
 * Created on Mar 23, 2003

 To change the template for this generated fi
 Window>Preferences>Java>Code Generation>Code
/

*
 @author Joe Pluta

 To change the template for this generated ty
 Window>Preferences>Java>Code Generation>Code
/
blic class HelloWorld {

  public static void main(String[] args) {
      System.out.println("Hello World!");
      System.exit(0);
  }
```

Figure 7.13: Use the popup menu in the Package Explorer to import a file into the Hello project.

The Import wizard gives you several options for your import source. Since your source is in a folder on the CD-ROM, you'll use the File system option.

❑ **7.4(e) Select File system in the Import wizard and click Next.**

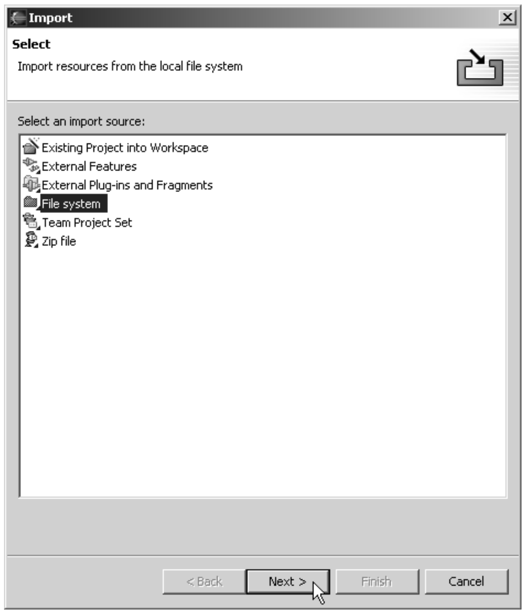
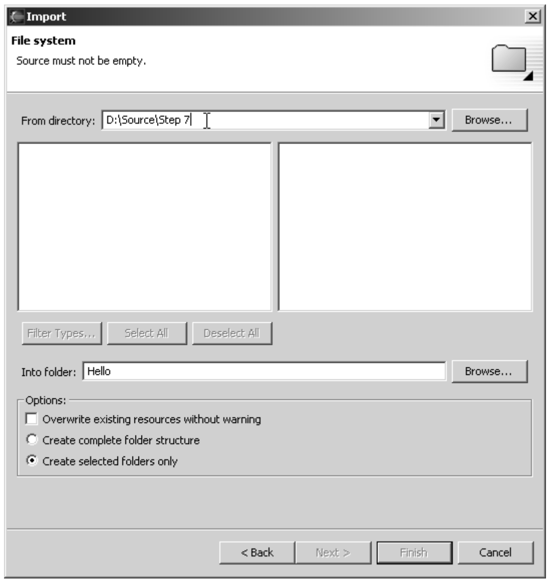


Figure 7.14: Select File system as the source for the import.

Now you need to specify the location of the files. All source for this step is located on the CD-ROM in a folder called Source, in a subfolder called Step 7, so specify

R:\Source\Step 7, where R is the letter of the CD-ROM drive where you loaded the disk. In Figure 7.15, the disk is loaded in the D: drive.

❑ **7.4(f) Enter “R:\Source\Step 7” in the From directory field, where R is the drive letter of your CD-ROM drive, and press the Tab key.**



Import

File system

Source must not be empty.

From directory: D:\Source\Step 7

Into folder: Hello

Options:

- [ ] Overwrite existing resources without warning
- ( ) Create complete folder structure
- (●) Create selected folders only

Figure 7.15: Enter R:\Source\Step 7, where R is the drive where you loaded the included CD-ROM.

The Tab key will cause the left pane to display an icon for the Step 7 folder. Select the Step 7 folder by clicking on it.

❑ **7.4(g) Left-click on the Step 7 folder.**

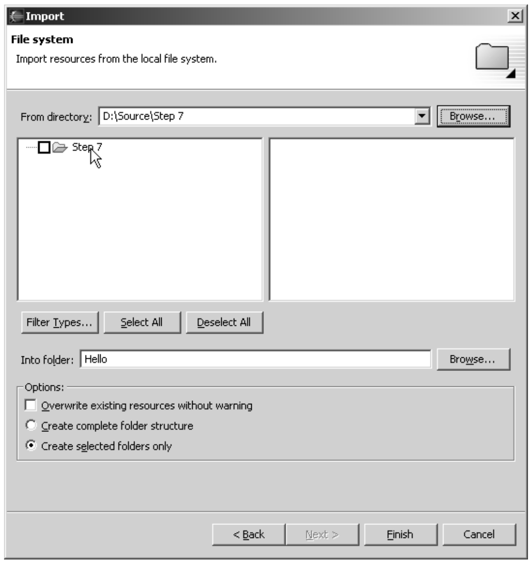

Import
File system
Import resources from the local file system.
From directory: D:\Source\Step 7
Browse...
Step 7
Filter Types...
Select All
Deselect All
Into folder: Hello
Browse...
Options:
Overwrite existing resources without warning
Create complete folder structure
Create selected folders only
< Back
Next >
Finish
Cancel

Figure 7.16: Select the Step 7 folder by left-clicking on it.

This will cause the contents of the Step 7 folder to appear in the right-hand pane as shown in Figure 7.17.

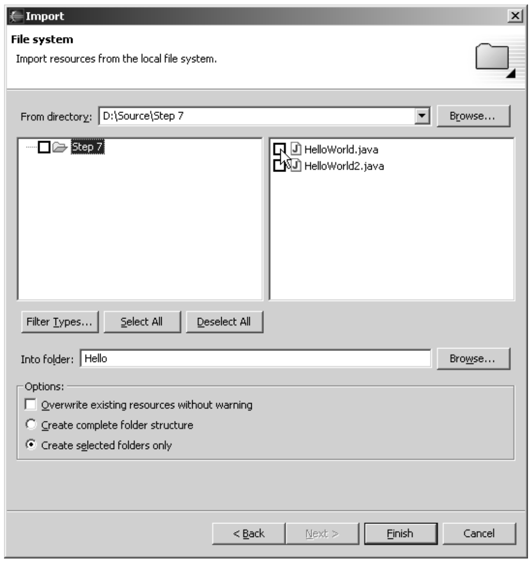
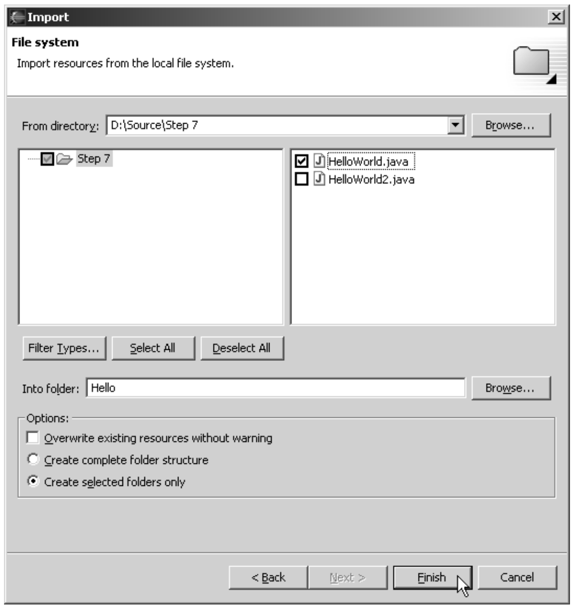

Import
File system
Import resources from the local file system.
From directory: D:\Source\Step 7
Browse...
Step 7
HelloWorld.java
HelloWorld2.java
Filter Types...
Select All
Deselect All
Into folder: Hello
Browse...
Options:
Overwrite existing resources without warning
Create complete folder structure
Create selected folders only
< Back
Next >
Finish
Cancel

Figure 7.17: The contents of the Step 7 folder will appear in the right hand pane.

Select HelloWorld.java by clicking on its checkbox.

❑ 7.4(h) Select HelloWorld.java and click Finish.



Import

File system

Import resources from the local file system.

From directory: D:\Source\Step 7

☑ Step 7

☑ HelloWorld.java
☐ HelloWorld2.java

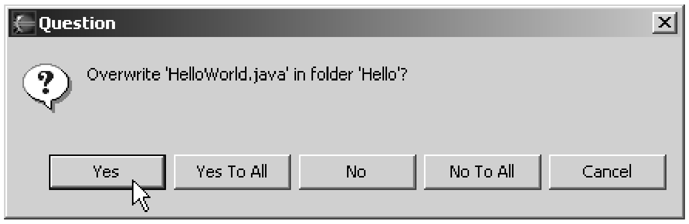
Into folder: Hello

Options:
☐ Overwrite existing resources without warning
○ Create complete folder structure
● Create selected folders only

Figure 7.18: Select only HelloWorld.java, then click Finish.

If you also performed Option 1, you will get a confirmation dialog like the one in Figure 7.19.

❑ 7.4(i) Click Yes on the confirmation box that pops up.

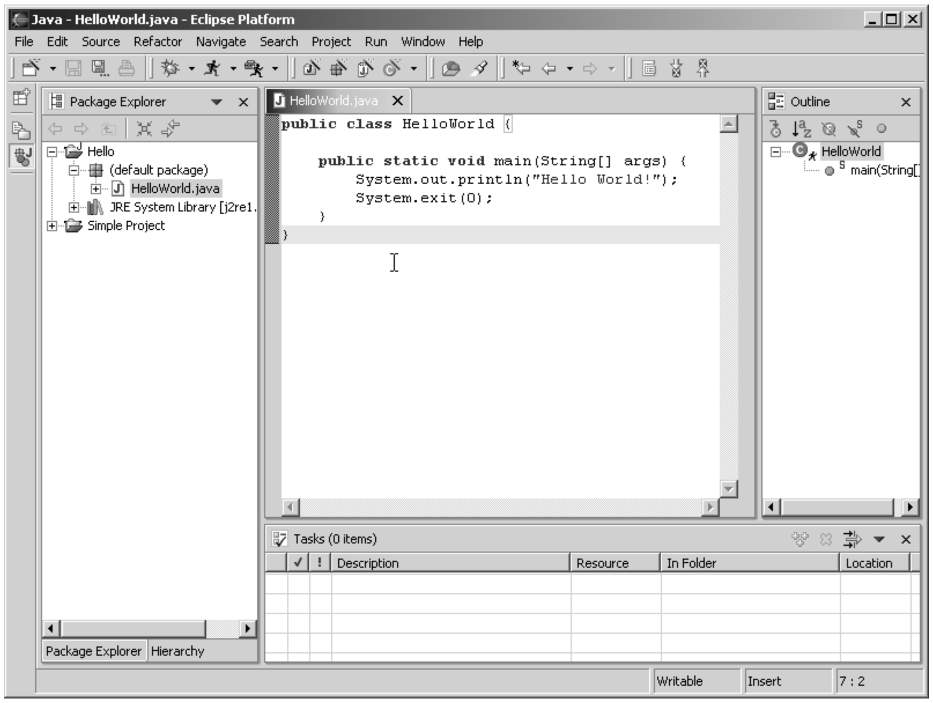


Question

Overwrite 'HelloWorld.java' in folder 'Hello'?

Figure 7.19: This prompt is used to be sure you really want to overwrite your source.

When you are done, your display should look just like the one shown in Figure 7.20. Since the source on the CD doesn't have all the fancy auto-generated comments, it'll be a little more austere than what the tool generated, as you can see.



```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.exit(0);
    }
}
```

Figure 7.20: The workbench display after importing source code from the included CD-ROM.

Step 7.5—Reviewing the source and the generated class

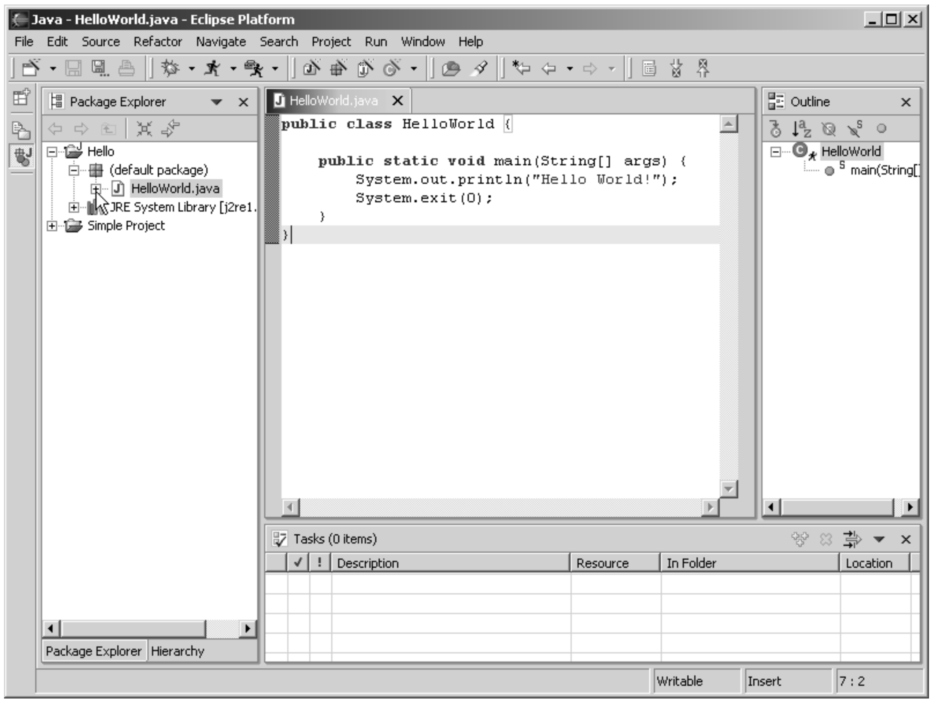
GOAL

In this step, you will use the Package Explorer to verify that Eclipse automatically compiles code when you save it, and to inspect the generated class.

You may have noticed that when you added HelloWorld.java that it was expandable—you could see the little plus sign to the left, which indicated there were objects underneath it. That may seem a little unusual, since this is just a source file. In the Resource perspective you wouldn't see the plus sign. So why does the Java perspective see Java source files differently?

Well, it's because the Java perspective considers the Java source file to be the parent of the Java class file. Let's take a look. Expand the source by left-clicking on the plus sign as shown in Figure 7.21, which will in turn give you the display in Figure 7.22.

❑ **7.5(a) Expand HelloWorld.java.**


Java - HelloWorld.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Hello
(default package)
HelloWorld.java
JRE System Library [j2re1.
Simple Project
Package Explorer Hierarchy
HelloWorld.java
Outline
HelloWorld
main(String[]
Tasks (0 items)
Description Resource In Folder Location
Writable Insert 7 : 2

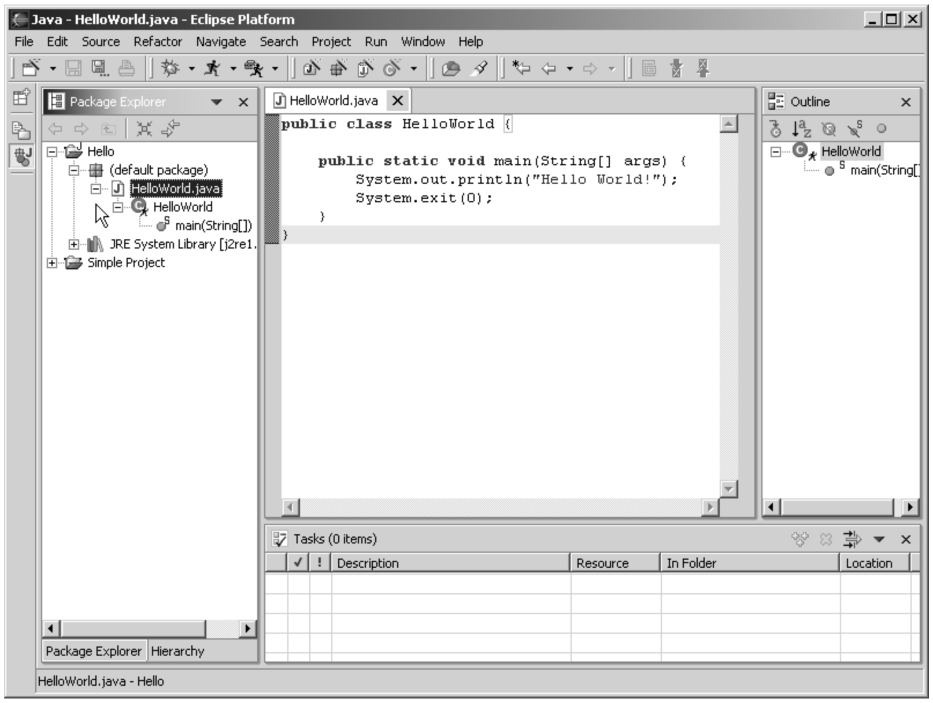
```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.exit(0);
    }
}
```

Figure 7.21: Left-click on HelloWorld.java to expand it.

You'll see that underneath HelloWorld.java source code, with the J icon, is the HelloWorld class, with a class icon. The white C inside a green circle denotes a class, and the little running guy underneath indicates that the class has a main() method and is therefore executable (i.e., runnable—cute, eh?).

You'll notice that the HelloWorld class, which is a child of the HelloWorld.java source code in the Package Explorer view, itself has a child. Each method in the class is considered a child of the class. So beneath the HelloWorld class, you see the main() method, with a green ball icon. The green ball indicates a public method, and the red S declares it to be static (Eclipse is big on icons declaring a lot of information).

Note also that these same symbols are used in the Outline view in the upper right-hand pane.



```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.exit(0);
    }
}
```

Figure 7.22: The expanded HelloWorld shows the source, the class, and the method.

Okay, you've entered the class. Now you get to run it. Step 8, here we come!