# Getting Started with Eclipse for Java

Maria Litvin Phillips Academy, Andover, Massachusetts

> Gary Litvin Skylight Publishing

#### 1. Introduction

- 2. Downloading and Installing Eclipse
- 3. <u>Importing and Exporting the Preferences</u>
- 4. Configuring Eclipse
- 5. <u>Running "Hello World"</u>
- 6. Bringing Existing Java Files into Eclipse
- 7. Command-Line Arguments and User Input
- 8. Running GUI Applications
- 9. Using Jar Files
- 10. Creating Jar Files
- 11. Content Assist and the Debugger

### 1. Introduction

*Eclipse* is a vast extendable set of tools for software development. Here we are interested in Eclipse's Integrated Development Environment (IDE) component for writing Java programs.

Eclipse is an open source project of Eclipse Foundation; you can find information about Eclipse Project at <u>http://www.eclipse.org/eclipse</u>. Eclipse is available free of charge under the <u>Eclipse Public License</u>.

Eclipse was developed by software professionals for software professionals; it may seem overwhelming to a novice. This document describes the very basics of Eclipse, enough to get started with Java in an educational setting.

Eclipse runs on multiple platforms including Windows, Linux, and Mac OS. There may be minor differences between Eclipse versions for different platforms and operating systems, but the core features work the same way. Here we will use examples and screen shots from *Windows*.

### 2. Downloading and Installing Eclipse

First make sure the Java Development Kit (JDK) is already installed on your computer. See <u>www.skylit.com/javamethods/faqs/GettingStartedJava.pdf</u> for directions.

Go to <u>https://www.eclipse.org/downloads/eclipse-packages/</u> and download the latest version of **Eclipse IDE for Java Developers**, appropriate for your operating system (Windows, Linux, or Mac). Earlier releases of Eclipse are listed in the "More Downloads" frame on the right. It appears that the latest version of Eclipse does not support 32-bit systems; download an older version (Photon or 2018/09, 4.9) for a 32-bit system.

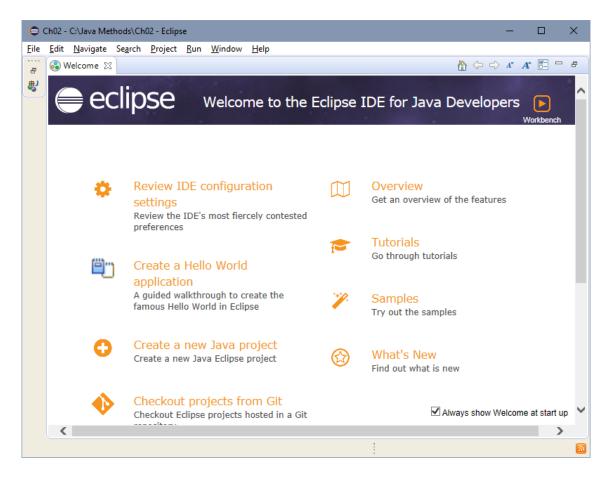
Under Windows, you will download a zip file, for example, eclipse-java-photon-R-win32x86\_64.zip. The zipped file contains the folder eclipse. Copy it to the destination of your choice to unzip or right click on the zip file and choose "Extract All...". We prefer to use C:\Program Files as the destination for the eclipse folder. You might want to rename eclipse into Eclipse (with a capital "E") for consistency with the names of other application folders. But if you prefer, you can install the eclipse folder in C:\.

You will find eclipse.exe in the eclipse folder. This is the Eclipse executable. Create a shortcut to it on the desktop (by dragging eclipse.exe to the desktop while holding down Ctrl+Shift or Alt).

Double click on the shortcut or on eclipse.exe. Eclipse may ask you to choose a "workspace."

A workspace in Eclipse is just a folder on your computer that will hold your programming projects. It is a good idea to create a folder that is initially empty; do not use any of the Eclipse installation folders or any folders that hold original files from your textbook.

Eclipse comes up with a Welcome screen:



Go over the overview and/or tutorials or click on Workbench (the arrow icon) or simply close the "Welcome" tab to start coding.

# 3. Importing and Exporting Preferences

Configuring Eclipse is a daunting and time-consuming task for a novice. Eclipse has thousands of configurable options, basic and advanced, all mixed together. For example, "Insert spaces for tabs" and "Show affordance in hover on how to make it sticky" appear in the same dialog (General/Editors/Text Editors). Factory defaults are chosen for experienced software developers and are not always appropriate for educational use.

Luckily, Eclipse provides a way to export the preferences from the current workspace into a file (an .epf file) and import the preferences from a file into a workspace.

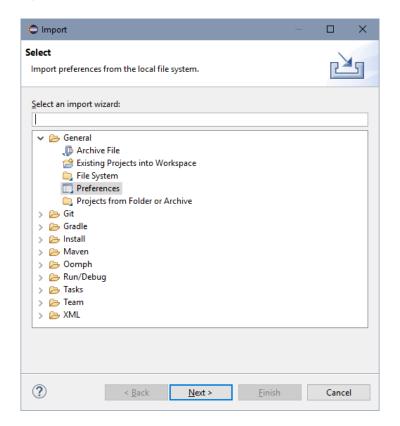
Our preferences are available in the LitvinsPreferences.epf file. You can download LitvinsPreferences.zip, which contains LitvinsPreferences.epf, from www.skylit.com/javamethods/faqs/LitvinsPreferences.zip.

### Our Eclipse preferences are described in Section 4.

You might want to just import these preferences into your workspace and leave Section 4, "Configuring Eclipse," until later, when you are ready to experiment with your own settings.

To import LitvinsPreferences.epf into your workspace, follow these steps:

- 1. Download LitvinsPreferences.zip and extract from it LitvinsPreferences.epf into a folder of your choice.
- 2. Choose the Import... command on the File menu.
- 3. Expand "General", select "Preferences" and click Next:



4. Browse to LitvinsPreferences.epf, select it, and click Finish.

\* \* \*

If you want to configure your own preferences and save them, perhaps for backup or for using them in another workspace, export them into a file. Follow these steps:

- 1. Choose the Export... command on the File menu.
- 2. Expand "General", select "Preferences", and click Next.
- Type the pathname of the file (or browse to the folder where you want to store the .epf file and add the file name). No need to include the .epf extension it will be added automatically. Click Finish.

Preferences created with an earlier release of Eclipse might not work properly with a newer version.

Eclipse configuration settings apply only to the current workspace and revert to defaults when you switch to a new workspace. You need to configure or import preferences into every workspace that you create. \*

It is possible to always use the same workspace, but as the number of projects in it grows, it may become hard to manage. You might prefer to use a separate workspace for each chapter in the textbook.

\* \* \*

The preferences settings includes an entry for "Installed JREs", that is, the Java Run-Time Environments installed on your computer.

Eclipse might not work properly if the JRE selected in the preferences does not match the JRE installed on your computer.

To examine and change the "Installed JREs" setting, choose Preferences on the Window menu, and go to Java⇒Installed JREs:

Preferences			-	
type filter text	Installed JREs			(> ▼ ⊂> ▼ ▼
<ul> <li>Ant</li> <li>Code Recommenders</li> <li>Gradle</li> <li>Help</li> <li>Install/Update</li> </ul>	newly created Java pro	-		
v Java	Name	Location	Туре	<u>A</u> dd
> Appearance	🗹 🛋 jre-10.0.1	C:\Program Files\Java\jre-10	Standard	Edit
> Build Path Code Coverage				Dupli <u>c</u> ate
> Code Style				Remove
> Compiler				<u>R</u> emote
> Debug				<u>S</u> earch
Editor     Installed JREs				
JUnit				
Properties Files Edito				
> Maven				
> Mylyn				
> Oomph				
> Run/Debug				
> Team				
Validation				
> XML				
< >				Apply
?		[	Apply and Close	Cancel

Remove the JRE listed there if it doesn't match the JRE on your computer, and add the one you have. Check the box for the desired JRE. If in doubt, click on Execution Environments (on the left panel, under "Installed JREs") to see how your selected JRE matches the version of Java installed on your computer.

\* \* \*

<sup>\* &</sup>lt;u>https://www.eclipse.org/forums/index.php/t/1088367/</u> offers a way to install preferences for all workspaces at once using *Oomph* installation editor.

While at this screen, you might want to attach Java docs to the selected JRE. Click on your JRE, click Edit, click on it again, and click Javadoc Location. If you had installed the docs locally, browse to the doc/api folder on your computer (for example,

file:/C:/Program Files/Java/jdk-10.0.1/docs/api/); otherwise enter the URL for the Java documentation page online (for example, <u>https://docs.oracle.com/javase/10/docs/api/</u>). Now, when you position the cursor over a library class name in the editor and press Shift+F2 for contest-sensitive help, Eclipse will open a window with the javadoc description of the library class.

#### \* \* \*

You may want to change some of the Eclipse settings frequently. For example, to change the editor font size, go to Window/Preferences, navigate to General⇒Appearance⇒Colors and Fonts, expand the "Java" line, choose "Java Editor Text Font", click Edit, and choose the font size. To set the console font, go to General⇒Appearance⇒Colors and Fonts "Debug" line.

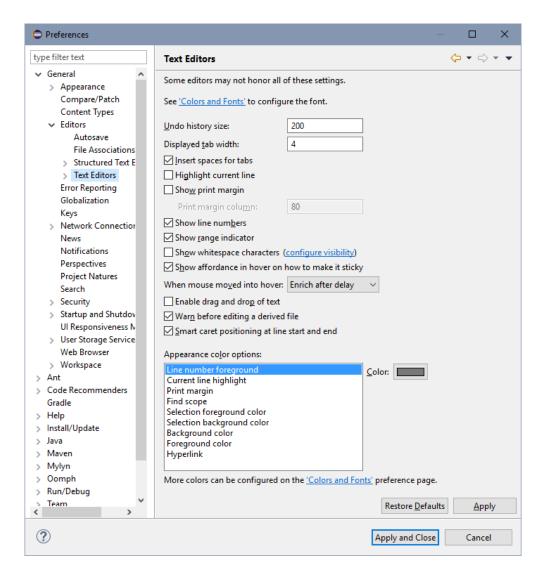
### 4. Configuring Eclipse

In this section we give a few suggestions for setting preferences and show where different types of options are located. The options we recommend below simply reflect... well, our preferences.

### In Eclipse, the **Preferences** command is located under the **Window** menu.

Click on it.

Under General⇒Editors⇒Text Editors check "Insert spaces for tabs" and "Show line numbers" if you want them, and uncheck "Highlight current line" and "Enable drag and drop":



Under General⇒Startup and Shutdown uncheck "Confirm exit when closing last window" and all of the "Plug-ins activated on startup." Also increase the number of recent workspaces under General⇒Startup and Shutdown⇒Workspaces.

Under General⇒Workspace check "Save automatically before build" and "Show workspace path in window title".

The next step is setting Java-specific options:

Preferences	— 🗆 X
type filter text	Java 🗘 🗸 🗸 🗸
> General > Ant	General settings for Java development:
> Code Recommenders Gradle	Action on double click in the Package Explorer
> Help > Install/Update	Expand the selected element
<ul> <li>Java</li> <li>Appearance</li> </ul>	When opening a Type Hierarchy
> Build Path Code Coverage	<ul> <li>Open a new Type Hierarchy Perspective</li> <li>Show the Type Hierarchy View in the current perspective</li> </ul>
<ul> <li>Code Style</li> <li>Compiler</li> <li>Debug</li> <li>Editor</li> <li>Installed JREs</li> </ul>	Refactoring Java code Save all modified resources automatically prior to refactoring Rename in editor <u>wi</u> thout dialog
Execution Environm JUnit Properties Files Editor	Search
> Maven > Mylyn > Oomph > Run/Debug	Enable <u>n</u> ew Java index Re <u>b</u> uild Index
> Team Validation > XML	Java dialogs Clear all 'do not show again' settings and show all hidden dialogs <u>Clear</u>
	again
< >	Restore <u>D</u> efaults <u>Apply</u>
?	Apply and Close Cancel

Under Java⇒Code Style uncheck the "Add '@Override'" box.

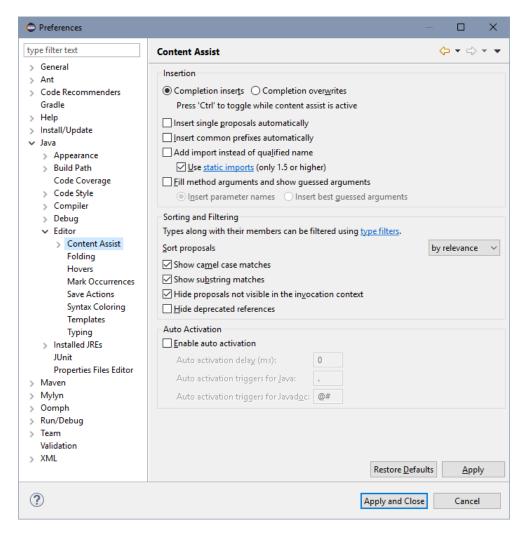
Under Java⇔Code Style⇒Formatter click New and enter a name of your choice for a new profile:

🖨 New Profile		×
<u>P</u> rofile name: Litvin		
Initialize settings with the follo	wing profile:	
Eclipse [built-in]		~
Open the edit dialog now		
?	OK	Cancel

Click OK. Under the "Indentation" tab choose the "Spaces only" tabs policy and set both the indentation size and the tab size to 2. Under the "Braces" tab change all brace positions except the last one, "Array initializer," to "Next line." Under the "White Space" tab **Arrays**⇒**Array initializers** uncheck "after opening brace" and "before closing brace" boxes. Under the "Control Statements" tab check all "Insert new line" boxes.

Define another profile under Java⇒Code Style⇒Clean Up. Under the "Code Organizing" tab check "Remove trailing whitespace"; under the "Missing Code" tab uncheck "Add missing annotations." Under the "Unnecessary Code" tab uncheck "Remove unused imports."

Under Java⇒Editor⇒Content Assist uncheck all the boxes in the Insertion section and uncheck "Enable auto activation":



Under **Java⇔Editor⇒Folding** uncheck all the "Initially fold" boxes or disable folding altogether by unchecking the "Enable folding" box.

Under Java⇒Editor⇒Mark Occurrences uncheck the "Mark occurrences" box.

If you do not like italics in your code editor, go to Java⇒Editor⇒Syntax Coloring, choose the Java⇒Static fields element, and uncheck the "Italic" box. The same for Java⇒Static final fields.

Under Java⇔Compiler⇔Errors/Warnings expand "Potential programming problems" and change "Serializable class without serialVersionUID" from "Warning" to "Ignore."

When you are finished setting the preferences, click OK. If Eclipse asks you whether it is OK to reload the workspace, click Yes.

### 5. Running "Hello World"

Eclipse does not tell you which workspace is currently selected, unless you have selected the "Show workspace path in window title" option in the preferences (under **General Workspace**). Go to the File Switch Workspace menu item and click Other... to see the path.

Never use Eclipse's default workspace. Create your own, for example, C: \mywork or C: \JavaMethods\Ch02, and import our (or your own) preferences into it.

On the File menu choose New Java Project (or click on the pull-down arrow next to the "New" button on the toolbar and choose Java Project). A dialog box pops up. Enter the project name, for example, "Hello"; leave the "Use default location" box checked:

🖨 New Java Project		-	- 🗆	×
Create a Java Project Create a Java project in the workspace or in an e				
<u>P</u> roject name: Hello				
Use default location				
Location: C:\Java Methods\Ch02\Hello			Brows	e
JRE				
• Use an execution environment JRE: Ja	ivaSE-10			$\sim$
○ U <u>s</u> e a project specific JRE: jr	e-10.0.1			$\sim$
O Use default JRE (currently 'jre-10.0.1')		1	Configure J	IREs
Project layout <u>U</u> se project folder as root for sources and <u>C</u> reate separate folders for sources and cla		Co	nfigure def	ault
Working sets Add project to working sets Working sets: The default compiler compliance level fo project will use a project specific compile			Ne <u>w</u> S <u>e</u> lect 1.6. The ne	
? < Back New	dt >	<u>F</u> inish	Car	ncel

In Eclipse Oxygen or an earlier version, click Finish.

\* \* \*

Java 9 introduced *modules* — another level in the hierarchy of classes and packages. A module can combine several packages. Modules are not useful in an educational setting; students can rely on the dummy <u>unnamed module</u> provided by default.

Eclipse *Photon* has added features for supporting modules, which are not useful to students, especially for projects created with older versions of Java. When creating a project in Photon, and after you typed in the name of your project, click Next and uncheck the Create module-info.java file box, or click Finish and choose Don't Create on the popup dialog:

🗢 New modul	e-info.java			×
Create module	<b>2-info.java</b> module name. By convention, module names usually start with a lowe	rcase lett	er <b>G</b>	)
<u>M</u> odule name:	Ch02			
	<u>C</u> reate	D	on't Crea	te

\* \* \*

If you want to see the contents of the bin folder (where the .class files are placed), from the Window menu, choose Show View Navigator.

🖨 Ch02 - C:\Java Methods\Ch02	2 - Eclipse					-		$\times$
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>I</u>								
📑 🗕 🔚 🕼 🗢 🔺 🔌 🔅	- 🜔 🕶 🎴 🕶 (	9a - 😰 O	🔹 🤔 🖨	🔗 🔹 🖢	💌 🖓 💌 🏷	$\leftarrow \star \Rightarrow \star$		
						Quick Acces	s 🛛 🖻	影
😫 P 😫 🖳 🗖								
(+ -> @   <b>= 5</b>   <b>&gt;</b>								
🗸 🔁 Hello								
> 🗁 .settings								
🗁 bin								
🗁 src 🕅 .classpath								
x .project								
	🖹 Problems 🔀	@ Javadoc	Declarati	on		⊉ 5		
	0 items					-+1   °		
	Description	^			Resource	Path		Locatic
	<							>
Hello								<u></u>

If you expand the Hello folder in Navigator, you will see the src subfolder:

That's where Java source files go. If you are starting from scratch, select a project (in this case "Hello"), then click on the New Java Class button on the toolbar. In the dialog box that pops up, enter the name for your class (for example, HelloWorld) and check the feature(s) you want automatically generated for your class (for example, public static void main):

🗢 New Java Class		-	
Java Class	efault package is discouraged.		C
Source fol <u>d</u> er:	Hello/src		Br <u>o</u> wse
Pac <u>k</u> age:		(default)	Bro <u>w</u> se
Enclosing type:			Bro <u>w</u> se
Na <u>m</u> e: Modifiers:	HelloWorld       ● public     ○ package     ○ private       □ abstract     □ final     □ static	) pro <u>t</u> ected	
Superclass:	java.lang.Object		Brows <u>e</u>
Interfaces:			<u>A</u> dd <u>R</u> emove
	s would you like to create? public static <u>void main(String[] args)</u> Constr <u>u</u> ctors from superclass In <u>h</u> erited abstract methods comments? (Configure templates and defaul	t value <u>here</u> )	
?	Generate comments	<u>F</u> inish	Cancel

Click Finish. Type in the code for your class in the editor:

```
/**
 * Displays a "Hello World!" message on the screen
 */
public class HelloWorld
{
   public static void main(String[] args)
   {
     System.out.println("Hello, World!");
   }
}
```

From the Run menu choose Run, or click the "run" button Or, or press Ctrl+F11. This will build or rebuild your project, if necessary, and run it.

# It is easy and convenient in Eclipse to have several programs in the same project and choose which one of them to run.

If you press Ctrl+F11 when a Java class is selected or open it in the editor, Eclipse will run the selected class.

You can double click on any Java file to open it in the editor window.

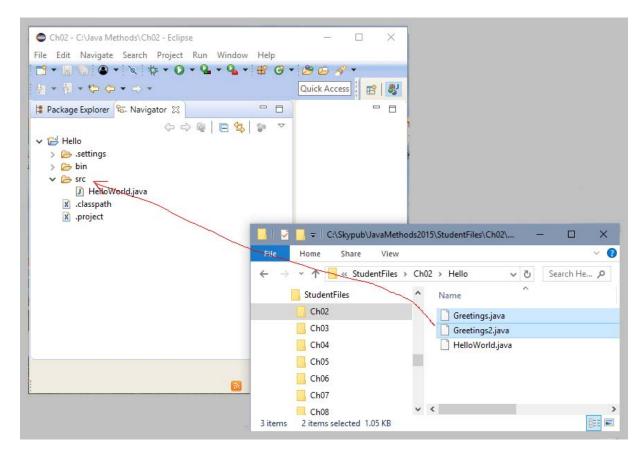
There are many keyboard shortcut keys in Eclipse. You can see the complete list in **Window⇒Preferences⇒General⇒Keys**. In particular,

**Ctrl+Shift+F** formats the file open in the editor according to the active profile, defined in Preferences, fixing indentation and removing the white space.

Ctrl+/ can be used to comment a highlighted block of code, thus temporarily disabling the code. Pressing Ctrl+/ again uncomments the block.

### 6. Bringing Existing Java Files into Eclipse

Suppose the files Greetings.java and Greetings2.java already exist on your computer (for example, downloaded with student files for your textbook). You want to bring them into an Eclipse project. If the project does not yet exist, create it first. Then open your operating system's file manager (for example, Windows Explorer) <u>outside Eclipse</u>. Select the desired .java files and drag and drop (or copy and paste) them into the src folder in the project.



A pop-up dialog will ask you whether you want to copy or "link" the files:

➡ File Operation ×
Select how files should be imported into the project:
✓ Create link locations <u>r</u> elative to: PROJECT_LOC ∨
Configure Drag and Drop Settings
OK Cancel

Choose "Copy files" to copy the Java files into the project folder. If you chose "Link" you will work with the original files in their original location and you may accidentally ruin or delete them.

See the next section for instructions for running Greetings and Greetings2.

If your program reads data files (.txt, .wav, .gif, etc.), place them into the project folder, such as Hello, at the same level as src and bin.

### 7. Command-Line Arguments and User Input

The Greetings program (*Java Methods* Section 2.4) expects command-line arguments; Greetings2 accepts input from the user.

If your program expects run-time arguments from the command line, you need to define a run-time configuration. From the Run menu, choose Run Configurations. Enter the configuration name, for example, "Greetings". Under the "Main" tab choose the main class for your program (such as "Greetings"). Under the "(x)=Arguments" tab enter the program's "command line" arguments in the top text area. For example:

<b>reate, manage, and run c</b> Run a Java application	nfigurations	
<ul> <li>Image: Second system of the sy</li></ul>	Name:       Greetings            • Main (w)= Arguments        IRE % Dependencies % Source & Environment [         Program arguments:             Program arguments:           Annabel Lee             VM arguments:           VM arguments:             VM arguments:           VM arguments:             VM arguments:           VM arguments:             VM arguments:           VM arguments:             VM orking directory:           Verking directory:             Other:           S(workspace_loc:Hello)             Other:           Workspace	Variables

Then click Run. You have to set the run-time arguments once, as long as you keep them the same. When the Greetings class is selected, the "Run" button and the Ctrl+F11 key will run the correct configuration.

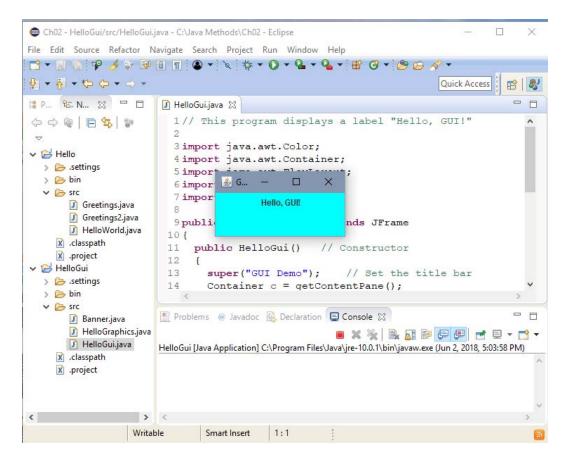
\* \* \*

The Greetings2 program prompts the user for input.

Unfortunately, Eclipse doesn't position the cursor correctly on the input line in the console window. If the cursor is in an editor window and you start typing, you will mess up the source code. Make sure to click on the console window before responding to a prompt.

### 8. Running GUI Applications

Create another project in the same workspace (from File menu choose New Java Project, or click on the pull-down arrow next to the New button on the toolbar and choose Java Project). For example, create a new project named "HelloGui" and copy (drag and drop) all the files from the *Java Methods* Chapter 2 HelloGui folder to the project's src folder:



Click on any class to select it, then press Ctrl+F11 to run it.

# 9. Using Jar Files

A JAR (Java archive) file can hold several compiled Java classes and serve as a library for a Java project. The file name for a JAR file has the extension .jar. Standard Java library supplied with the JDK is a JAR file, and it is automatically added to Java projects. A programmer or a third party can supply their own JAR files. For example, the authors of the *Java Methods* textbook supply EasyClasses.jar, which holds the class files for EasyReader, EasyWriter, EasySound, and EasyDate classes (see www.skylit.com/javamethods/JM-Appendix-D.pdf).

As an example, let's set up a project for the *BalloonDraw* program (Java Methods, Section 4.2), which uses the balloondraw.jar library. Create a new project and add BalloonDraw.java to its src folder:

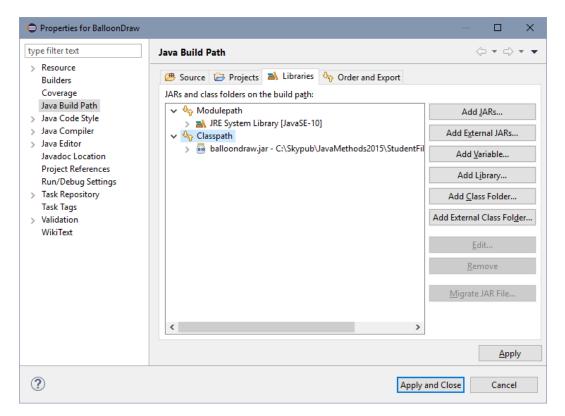
Ch04 - Balloons/src/BalloonDraw.java	C:\Java Methods\Ch04 - Eclipse				- 0	×
<u>File Edit Source Refactor Navigate</u>	Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp					
📑 • 🗟 🐘 • 🕥 • 💁 • 🚱	• 🖶 🞯 • 🤔 🗁 🖋 • 🕸 🖉 🔳 🔳	۵ - ۵	¥ ▼ 🕅 ▼ 🏷 (> ¬	▼ ⊲> ▼ Quick	Access	影
		create p other sha Gary Litvi nds JFrame nstructor DrawingP ew Contro ane(); t.CENTER); out.SOUTH)	<pre>ctures with so pes) .n</pre>	everal	Access	-
					3	
	Problems 🕄 @ Javadoc 🔯 Declaration			-*	,   🏭 🗸 🗆	• 🗆
	4 errors, 0 warnings, 0 others		<b>D</b>	1	-	
	Description	Resource	Path	Location	Туре	
	> 🔕 Errors (4 items)					
<b>N</b>	Writable Smart Insert 1:1			<	2 🕅 🔊	2 🕲

The project shows a list of errors because several required classes from balloondraw.jar are missing. To add a JAR library to a project, follow these steps:

 Select the project (that is highlight its name), then choose Properties on the Project menu (or right-click on the project name and choose Properties; or select the project and press Alt+Enter). 2. Select Java Build Path and click on the "Libraries" tab and select Classpath:

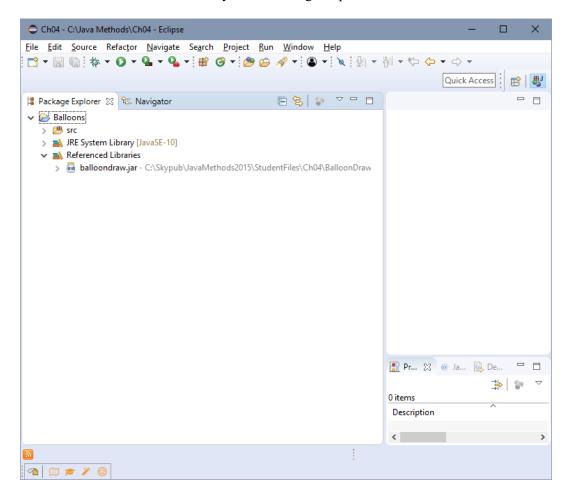
Properties for BalloonDraw		- 🗆 X
type filter text	Java Build Path	<
<ul> <li>&gt; Resource Builders Coverage</li> <li>Java Build Path</li> <li>&gt; Java Code Style</li> <li>&gt; Java Compiler</li> <li>&gt; Java Editor</li> <li>Javadoc Location</li> <li>Project Natures</li> <li>Project Natures</li> <li>Project References</li> <li>Run/Debug Settings</li> <li>&gt; Task Repository</li> <li>Task Repository</li> <li>Task Tags</li> <li>&gt; Validation</li> <li>WikiText</li> </ul>	Source       Projects       Libraries       Source and Export         JARs and class folders on the build path:            Source       Order and Export          Source       Classpath           Source       Source           Source       Source        <	Add JARs Add External JARs Add Variable Add Library Add Class Folder Add External Class Folder Edit Edit Migrate JAR File
?	Apply	and Close Cancel

3. Click Add External JARs..., navigate to the folder that holds the JAR file, choose the file, and click Open.



Click Apply and Close. Now the errors disappear and you can run the program.

You can see a "Referenced Libraries" entry in the Package Explorer:



# If you add to your project a class with the same name as a class in the JAR, the explicitly added class will be used, rather than the class in the JAR.

For example, we can add Balloon.java to the project and edit it. Balloon.class in the project will take precedence over Balloon.class in balloondraw.jar, that is, your edited version will be used when you run the program.

\* \* \*

If you have the source code for the library and/or javadoc documentation, you can "attach" them to the JAR file to take full advantage of Eclipse's interactive tips and to have convenient access to the source code and the docs. Under the "Libraries" tab expand the newly created library item —

Properties for BalloonDraw		- <b>D</b> X
type filter text	Java Build Path	<> ▼ → ▼
type filter text > Resource Builders Coverage Java Build Path > Java Code Style > Java Compiler > Java Editor Javadoc Location Project References Run/Debug Settings > Task Repository Task Tags > Validation WikiText	<ul> <li>Bource Projects Libraries Order and Export</li> <li>JARs and class folders on the build path:</li> <li>Modulepath</li> <li>Otasspath</li> <li>Classpath</li> <li>Classpath</li> <li>Source attachment: (None)</li> <li>Javadoc location: (None)</li> <li>External annotations: (None)</li> <li>Is not modular</li> <li>Native library location: (None)</li> </ul>	Add JARs Add External JARs Add Variable Add Ljbrary Add Class Folder
	Access rules: (No restrictions)	Add External Class Fol <u>d</u> er <u>E</u> dit <u>R</u> emove <u>M</u> igrate JAR File <u>Apply</u>
?	Apply	and Close Cancel

— click on "Source attachment," and click Edit... You will see a source attachment pop-up:

Source Atta	achment Configuration	_		×				
Select the loca	tion (folder, JAR or zip) containing the source for 'balloondraw.jar': location							
<u>P</u> ath:			<u>B</u> rowse.					
○ E <u>x</u> ternal location								
Pa <u>t</u> h:			External <u>F</u> i	le				
E <u>n</u> coding:	Default (Cp1252) 🗸	E	kternal F <u>o</u> l	der				
?	ОК		Cance	el l				

Choose "External location", navigate to the folder that holds the source code (unzip it first, if necessary), and click OK. Repeat the same for "Javadoc location" if you have API documentation for the classes in the JAR. Now if you hover over a library element in the editor, Eclipse displays a tip from that element's javadoc. (Like almost everything else, "hovers" are configurable in Eclipse.)

Pressing F2 redisplays the javadoc tip. Pressing Shift+F2 opens the full javadoc window in the editor. This works for both the standard Java library and your own JAR files.

\* \* \*

### 10. Creating JAR Files

A JAR file can hold a complete Java program (a class that has the main method and all the supporting classes) and it can be configured to be runnable: a double click on the file will run the program. Try, for example, balloondraw.jar in the Java Methods student files, Ch04/BalloonDraw.

### The same JAR file can be runnable and can also serve as a library.

Eclipse allows you to quickly create a library or an executable JAR file for a project. From the File/Export... menu, expand the Java line and choose "JAR file" or "Runnable JAR file":

Export	-		×
Select Export resources into a JAR file on the local file system.		Ľ	3
<u>S</u> elect an export wizard:			
type filter text			
<ul> <li>✓ ➢ General</li> <li>Ant Buildfiles</li> <li>P Archive File</li> <li>File System</li> <li>Preferences</li> <li>&gt; ➢ Install</li> <li>✓ ➢ Java</li> <li>Ĩ JAR file</li> <li>※ Javadoc</li> <li>Ĩ Run/Debug</li> <li>➢ Tasks</li> <li>➢ Team</li> <li>※ XML</li> </ul>			
(?) < <u>Back</u> <u>Next</u> > <u>Finish</u>		Cano	el

Specify the destination and, for a runnable file, the name of the run configuration. See Eclipse help for details: **Help Contents**  $\Rightarrow$  **Java development user guide**  $\Rightarrow$  **Tasks**  $\Rightarrow$  **Creating JAR files**. Follow the instructions for "Creating a new JAR file" or "Creating a new runnable JAR file."

23

### 11. Content Assist and the Debugger

Eclipse offers many features that speed up typing and correcting mistakes. The Content Assist feature provides context-sensitive code completion. Press Ctrl+Space to see the code completion suggestions (called *proposals*). For example, if you type syso and press Ctrl+Space, Eclipse will suggest System.out.println();. We think Content Assist may confuse a novice; a student has to acquire some familiarity with Java to employ this feature effectively.

As far as the debugger is concerned, see Top Ten Reasons Not to Use a Java Debugger in School.