
Visualization Environment for Rich Data Interpretation (VERDI 1.5): Developer Instructions

U.S. EPA Contract No. EP-W-09-023, "Operation of the Center for
Community Air Quality Modeling and Analysis (CMAS)"

Prepared for: William Benjey and Donna Schwede
U.S. EPA, ORD/NERL/AMD/APMB
E243-04
USEPA Mailroom
Research Triangle Park, NC 27711

Prepared by: Liz Adams and Jo Ellen Brandmeyer
Institute for the Environment
The University of North Carolina at Chapel Hill
100 Europa Drive, Suite 490
CB 1105
Chapel Hill, NC 27599-1105

Date: October 26, 2014

Contents

1	Introduction.....	1
2	Install Developer Environment.....	2
2.1	Install Java Development Kit.....	2
2.2	Download and Install Eclipse	3
3	Start Eclipse.....	5
3.1	Install Subclipse in Eclipse	7
4	Import VERDI Source Code.....	10
4.1	Select File→Import.....	10
4.2	Checkout Projects from SVN.....	11
4.3	Create a New Repository Location	12
4.4	Specify Location of VERDI SourceForge Repository.....	12
4.5	Select Folders for Checkout.....	13
4.6	Source Code for Libraries	16
5	Configure Apache Ant to Use tools.jar from the JDK	18
6	Set Eclipse Preferences	21
6.1	Workspace Preferences	21
6.2	Verdi_core Properties	22
6.2.1	Java Build Path	22
6.2.2	Java Compiler	23
7	Build the NetCDF-Java Library with Modifications for VERDI.....	25
7.1	Set up NetCDF-Java Library in Eclipse.....	25
8	Test VERDI Using Scripts within Eclipse	27
8.1	View Scripts within Eclipse.....	27
9	Update Source Code from the Repository	30
9.1	Open the Synchronization Window	30
9.2	Synchronize with Repository Using SVN	32
9.3	Resolve Updates and Conflicts	34

10 Build the VERDI Distribution	35
10.1 Prepare to Build VERDI Distribution.....	35
10.1.1 Microsoft Windows	35
10.1.2 Linux	36
10.1.3 Mac OS X	36
10.1.4 Build_dist.xml.....	36
10.1.5 Build Using Ant	36
10.1.6 Check Console for Error Messages.....	39
10.1.7 Add Java Compiler to Ant	39
10.2 Tag Released Versions.....	42
10.2.1 SVN Tag	42
11 Logging Messages in VERDI	44
11.1 Implementing Log4J2	44
11.2 Log files	45
12 List of Libraries and Source Code Files	46

Figures

Figure 2-1. Java –version command on 64-bit Windows 7	2
Figure 3-1. Startup screen for Eclipse	5
Figure 3-2. Select a VERDI workspace for Eclipse	6
Figure 3-3. Eclipse workbench	6
Figure 3-4. Example Eclipse development environment for Java projects	7
Figure 3-5. Result of search for Subclipse within Eclipse Marketplace	8
Figure 3-6. List of available software when downloading Subclipse	9
Figure 4-1. File/Import in Eclipse	10
Figure 4-2. Import a project into Eclipse from SVN	11
Figure 4-3. Checkout Projects from SVN	11
Figure 4-4. Create a new repository location.....	12
Figure 4-5. Checkout VERDI source code from SVN	13
Figure 4-6. Select projects to check out from SVN	14
Figure 4-7. SVN Checkout includes a meter indicating progress of download.....	15
Figure 4-8. Projects listed in the Package Explorer of the Eclipse workspace.....	15
Figure 4-9. Location of source code for open source libraries	16
Figure 4-10. Cross-reference of source code and class libraries within an Eclipse project	17
Figure 5-1. Eclipse preferences	18
Figure 5-2. Ant preferences within Eclipse.....	19
Figure 5-3. Add tools.jar to Ant classpath	20
Figure 6-1. Eclipse preferences	21
Figure 6-2. Project references for verdi_core.....	22
Figure 6-3. Dependent projects for verdi_core	23
Figure 6-4. Project-specific settings for the Java compiler	24
Figure 8-1. Run configurations.....	27
Figure 8-2. Sample configuration for a VERDI script.....	28
Figure 8-3. Tab to set environment variables for a run configuration	29
Figure 8-4. Specify VERDI_HOME environment variable	29
Figure 9-1. Show View → Other	30
Figure 9-2. Expand team folder, highlight Synchronize, and press the OK button	31
Figure 9-3. Synchronize window added to bottom of workspace.....	31
Figure 9-4. Click on Synchronize symbol to bring up pop-up window.....	32
Figure 9-5. Select SVN in Synchronize window	33
Figure 9-6. Synchronize SVN with all resources selected	33
Figure 9-7. Check for updates and conflicts.....	34
Figure 9-8. Update verdi_core	34
Figure 10-1. Review and edit a build.properties file	35
Figure 10-2. Window → Show View → Ant.....	37
Figure 10-3. Double-click on build.wind.dist to build VERDI distribution with Ant	38
Figure 10-4. Console error message related to not finding Java compiler.....	39
Figure 10-5. Open Window → Preferences	40
Figure 10-6. Expand Ant, select runtime, select global entries	41
Figure 10-7. Add tools.jar to Ant preferences.....	41

1 Introduction

This manual contains instructions on how developers can set up, run, build, and obtain updates from the software repository for the Visualization Environment for Rich Data Interpretation (VERDI) software. Developers are encouraged to develop and contribute code for VERDI. Anyone who experiences a bug should check and, if appropriate, update the existing bug list at <http://bugz.unc.edu>, which is maintained by the Community Modeling and Analysis System (CMAS) Center. If the bug is new please submit a new bugzilla report along with available test datasets that demonstrate the problem. Requests for enhancements are always welcome.

Initial development of VERDI was done by the Argonne National Laboratory for the U.S. Environmental Protection Agency (EPA) and its user community. Argonne National Laboratory's work was supported by the EPA through U.S. Department of Energy contract DE-AC02-06CH11357. Further development has been performed by the University of North Carolina at Chapel Hill's (UNC) Institute for the Environment under U.S. EPA Contract Nos. EP-W-05-045 and EP-W-09-023, by Lockheed Corporation under U.S. EPA contract No. 68-W-04-005, and by Argonne National Laboratory. VERDI is licensed under the GNU Public License (GPL) version 3, and the source code is available through verdi.sourceforge.net. VERDI is supported by the CMAS Center under U.S. EPA Contract No. EP-W-09-023. The CMAS Center is located within the Institute for the Environment at UNC.

VERDI 1.5.0 is distributed for the following computing environments:

- Windows 7, 64-bit
- Windows 7, 32-bit
- Linux, 64-bit
- Linux, 32-bit
- Mac, 64 bit

2 Install Developer Environment

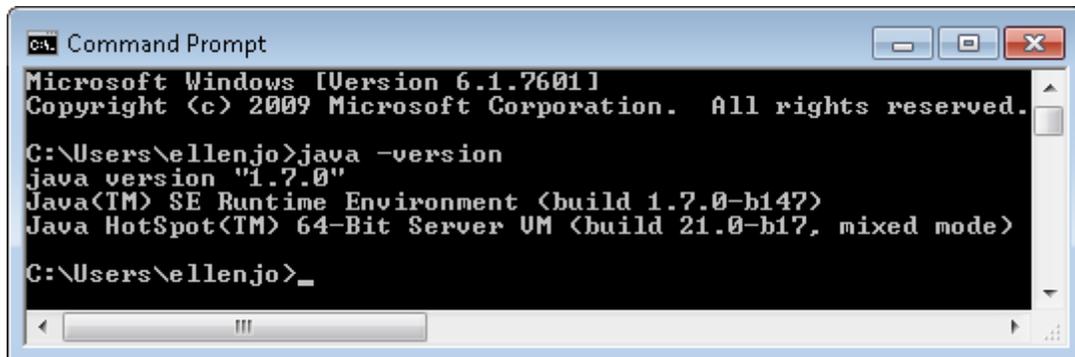
To install VERDI 1.5 on Windows 7, you no longer need administrator privileges. You should exit all programs before installing software. NOTE: VERDI 1.5 was developed under Java 7 and has been minimally tested under Java 8.

The remainder of this chapter discusses how to install the Java Development Kit, the Eclipse Integrated Development Environment (IDE), and Subversion on your development system.

2.1 Install Java Development Kit

Check to see if the Java Development Toolkit (JDK) and the Java Run-time Environment (JRE) 7 are already installed and properly configured on your computer. VERDI v1.5.0 was developed using the JDK version 7u55. If your version of Java is at least that release of Java 7, you can skip to section 2.2.

1. Under Windows 7, open a Command window to get to a Command prompt.
2. Type the command “java -version” as shown below (Figure 2-1). If you see the proper response from that java command, then your java version is already installed and included in your PATH.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ellenjo>java -version
java version "1.7.0"
Java(TM) SE Runtime Environment (build 1.7.0-b147)
Java HotSpot(TM) 64-Bit Server VM (build 21.0-b17, mixed mode)

C:\Users\ellenjo>_
```

Figure 2-1. Java -version command on 64-bit Windows 7

3. Make certain that your Java version is at least that shown above. If you have an earlier Java version (e.g., 1.6 or before) you need to download and install an update.

Download the JDK and the JRE from the current repository for Java. Follow the installation instructions provided by the download site.

1. <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
2. For Linux 64:
 - a. Download the self-extracting <version>-linux-x64.bin file
 - b. Run the command `chmod +x *linux-x64.bin` to give it executable permissions
 - c. Run the self-extracting file `./jdk7*-linux-x64.bin`

- d. Run the alternatives program to tell the system about the existence of your new installation:
 Alternatives –config java
 (this will list how many versions are installed. If there is only one then install the JDK as number 2)
 alternatives –install /usr/bin/java java /opt/jdk1.6.0_20/bin/java 2
 Run the alternatives program again, to choose the new installation
 alternatives –config java, select number 2
- e. Verify that your computer is finding the correct version of Java:
 \$ java –version
- f. Repeat the above steps for javac:
 Alternatives –install /usr/bin/javac javac /opt/jdk1.6.0_20/bin/javac 2
 Alternatives –config java, select number 2
3. For Windows 64: Start Button> Control Panel> System > Advanced System Settings> Environment Variables
 - a. Edit the PATH environment variable to add the fully qualified path to your java executable (e.g., C:\Program Files\Java\jdk1.7.0_55\bin).
 - b. Note that the list is semicolon-delimited.
 - c. Follow the instructions at the beginning of this section and shown in Figure 2-1
4. For Windows 64 with Powershell: Use the following command to set the path, then exit Powershell and restart it for the path to be set.
 - a. setx PATH "\$env:path;\the\directory\to\add" –m
 - b. You should see SUCCESS: Specified value was saved.
 - c. **Restart your session** and the variable will be available. setx can also be used to set arbitrary variables type setx /? at the prompt for documentation.

2.2 Download and Install Eclipse

1. Use your Web browser to go to:
<https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr1>
If the URL has changed, go to <http://www.eclipse.org/> and navigate to IDE and Tools, Desktop IDEs, Java IDE.
2. Look for Download Links compatible with your development system.
3. Download and install the Eclipse IDE for Java Developers.
 - a. Windows: Download the zip file for the appropriate version of Eclipse.
 - b. Select a convenient directory where you have permission to install software; that is, you do not have to install Eclipse under one of the “Program Files” directories or on the root file system on one of your drives.
 - c. Unzip the downloaded file into your chosen directory.
 - d. Linux/Mac: Install to local directory and extract using tar –xzvf
4. If you are running Eclipse on Microsoft Windows, you can put the Eclipse icon where it will be convenient for you. Using the Windows Explorer, navigate to where you installed Eclipse. Go to the eclipse subdirectory and right-click on the eclipse.exe file. Windows

displays a context menu associated with eclipse.exe. Select one or more of three main options.

- a. Pin to Taskbar: Place the icon in the taskbar at the bottom of your main window. Once there, you can click and drag the icon along the taskbar to place the icon at a logical place for you.
- b. Pin to Start Menu: Place the icon in the Start menu. The icon will be placed as the last item in your list of pinned items (i.e., just above the horizontal line that divides your pinned items from your frequently used items).
- c. Place on Desktop: Hover over “Send to”, move your mouse into the menu that opens to the right and select “Desktop (create shortcut)”. You can then move your shortcut to a different location on your desktop or rename it.

3 Start Eclipse

Using Windows: Select the Eclipse icon on your taskbar, start menu, or desktop, or go to the directory where your installed Eclipse (e.g., C:\Program Files\eclipse directory and double-click on eclipse.exe.

Using a Mac: Go to the applications directory, to the Eclipse folder, and click on the Eclipse icon.

Using a Linux machine: Go to the directory where Eclipse was installed and run the eclipse executable.

Figure 3-1 shows the startup window for Eclipse. Note the progress bar at the bottom that indicates how Eclipse is being configured. Eclipse requires more startup time as you add tools into the Eclipse environment.



Figure 3-1. Startup screen for Eclipse

Next, select the workspace for Eclipse (Figure 3-2). If you have not yet used VERDI 1.5 on your computer, you can select the Browse button to select where you want to install VERDI's source code. Eclipse will create the directory for you. Eclipse's startup screen is once again displayed while more tools are loaded and the VERDI project is opened.

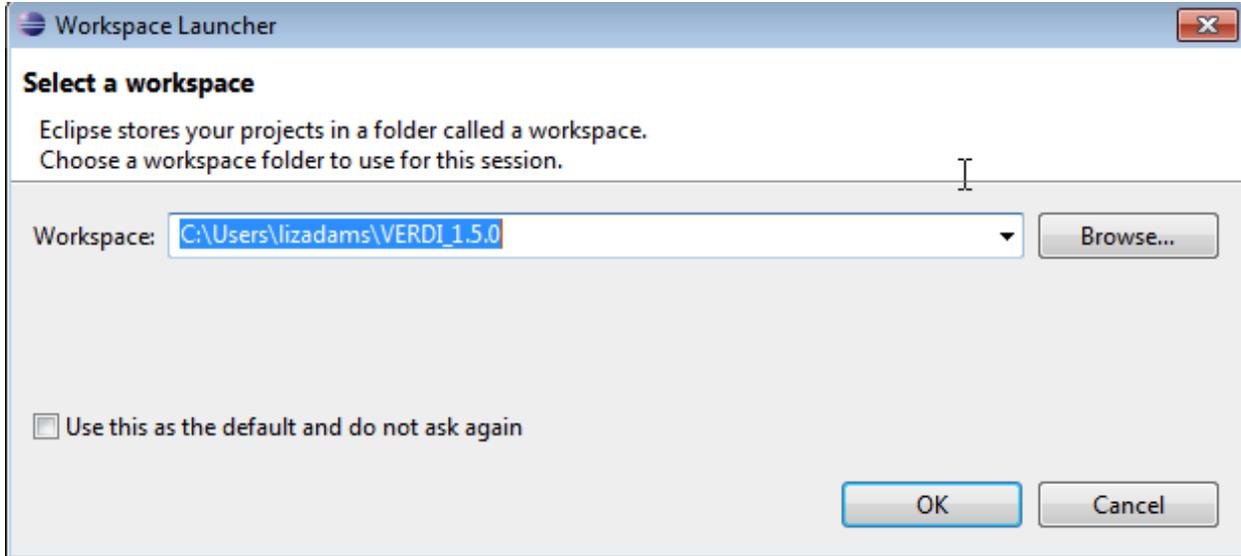


Figure 3-2. Select a VERDI workspace for Eclipse

To enter the developer workspace, click on the arrow at the right-hand side of the Welcome screen; if you hover over the arrow it will say “Workbench – Go to the workbench” (Figure 3-3). The Eclipse workbench contains several windows that allow you to view source code, edit, and build within a single developer environment (see Figure 3-4).

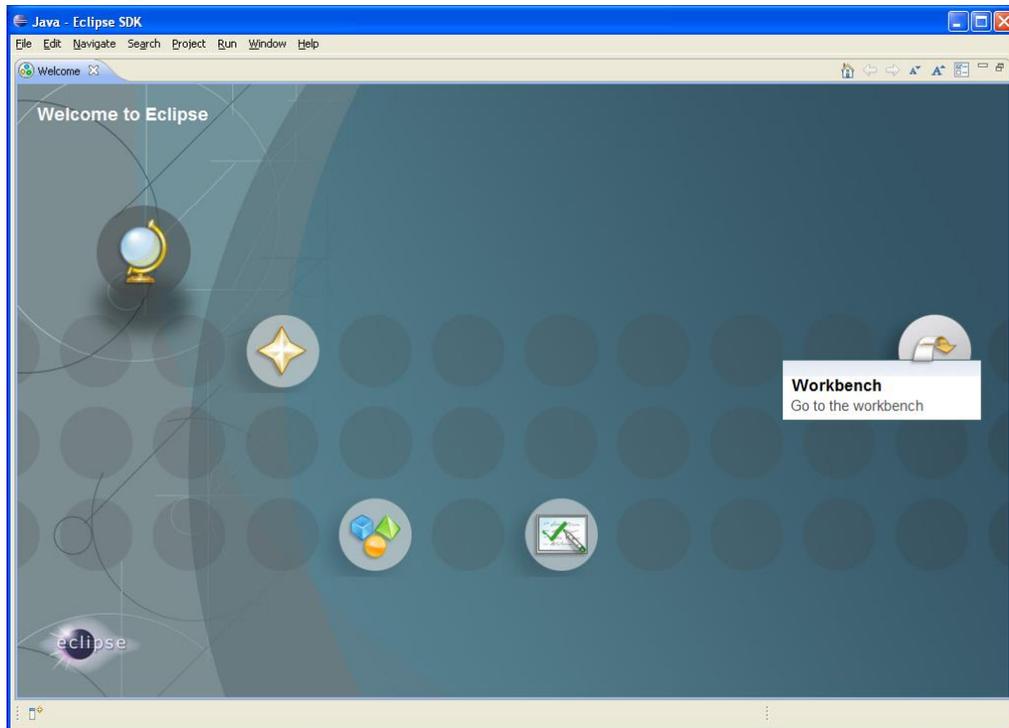


Figure 3-3. Eclipse workbench

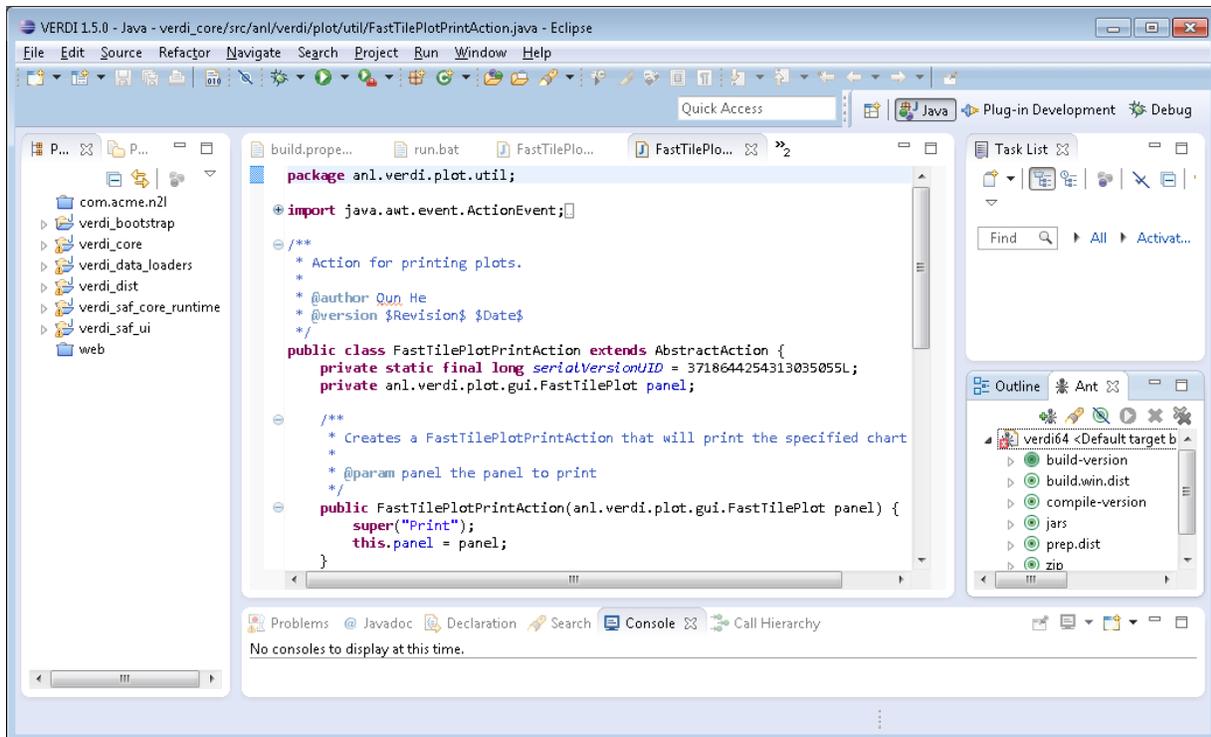


Figure 3-4. Example Eclipse development environment for Java projects

The Eclipse IDE window as shown in Figure 3-4 has a title bar at the top showing the name of the file currently being edited, menus and icons below the title bar, the Package Explorer down the left-hand side, multiple tabbed panes in the central file editor with Java keyword highlighting, messages along the bottom, and the Ant build environment in the bottom right-hand corner. These and other windows may be added, closed, moved, and resized as-needed for the work being performed.

3.1 Install Subclipse in Eclipse

- a. In Eclipse select Help> Eclipse Marketplace
- b. In the Search box type Subclipse, click Go
- c. Subclipse (the version compatible with eclipse) will be displayed (Figure 3-5), click on Install
- d. Select the packages to install as shown in Figure 3-6.
- e. Review and accept the software license.
- f. Agree to allow unsigned software to be loaded onto your computer.
- g. You will then be asked to allow Eclipse to restart to complete adding Subclipse into Eclipse. When Eclipse starts again, you will be asked if you want to allow the Subclipse team to receive anonymous usage statistics for your Eclipse instance. The default is to report usage. If you do not want to report usage of Subclipse uncheck the box and then press the OK button.

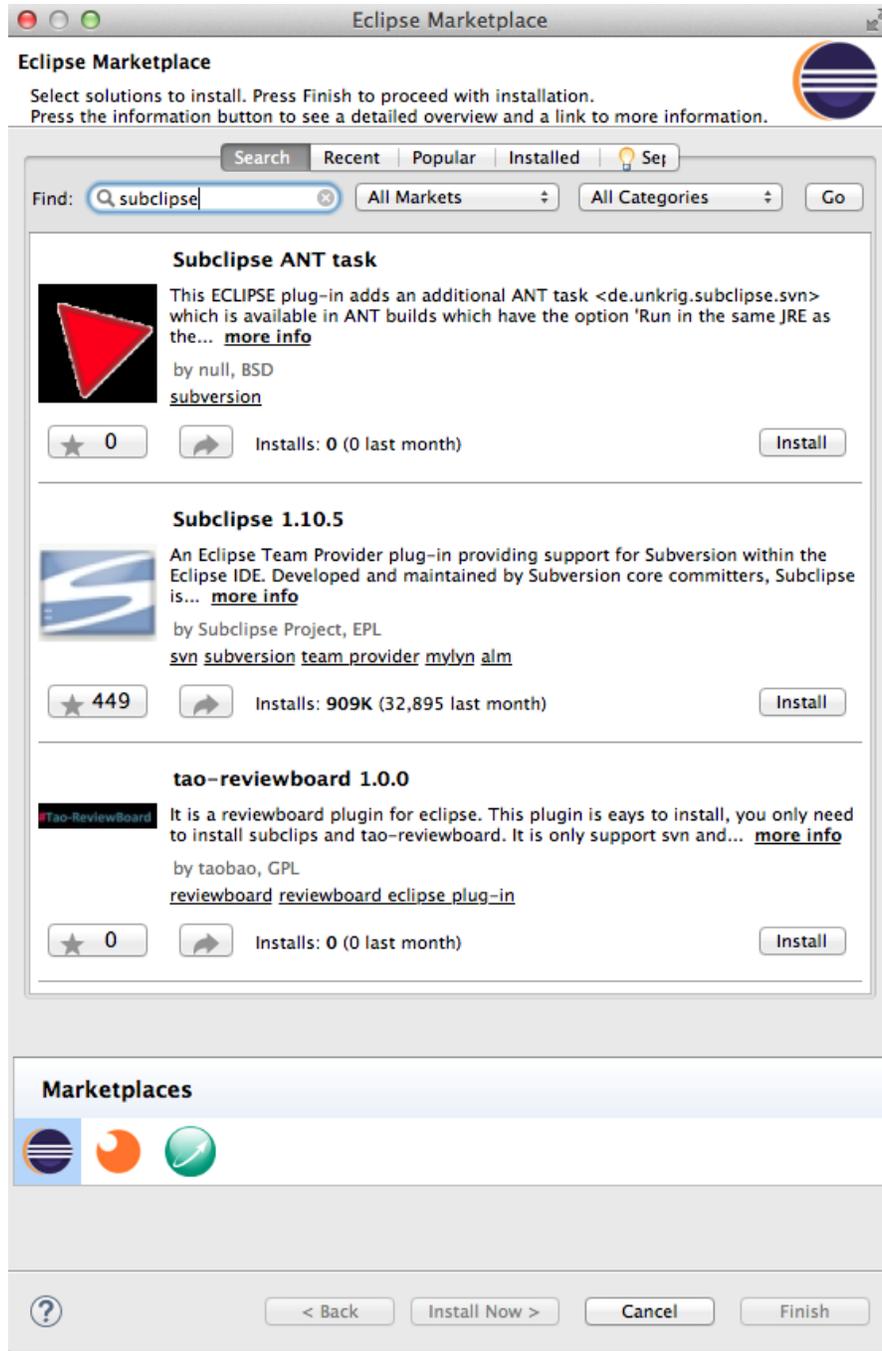


Figure 3-5. Result of search for Subclipse within Eclipse Marketplace

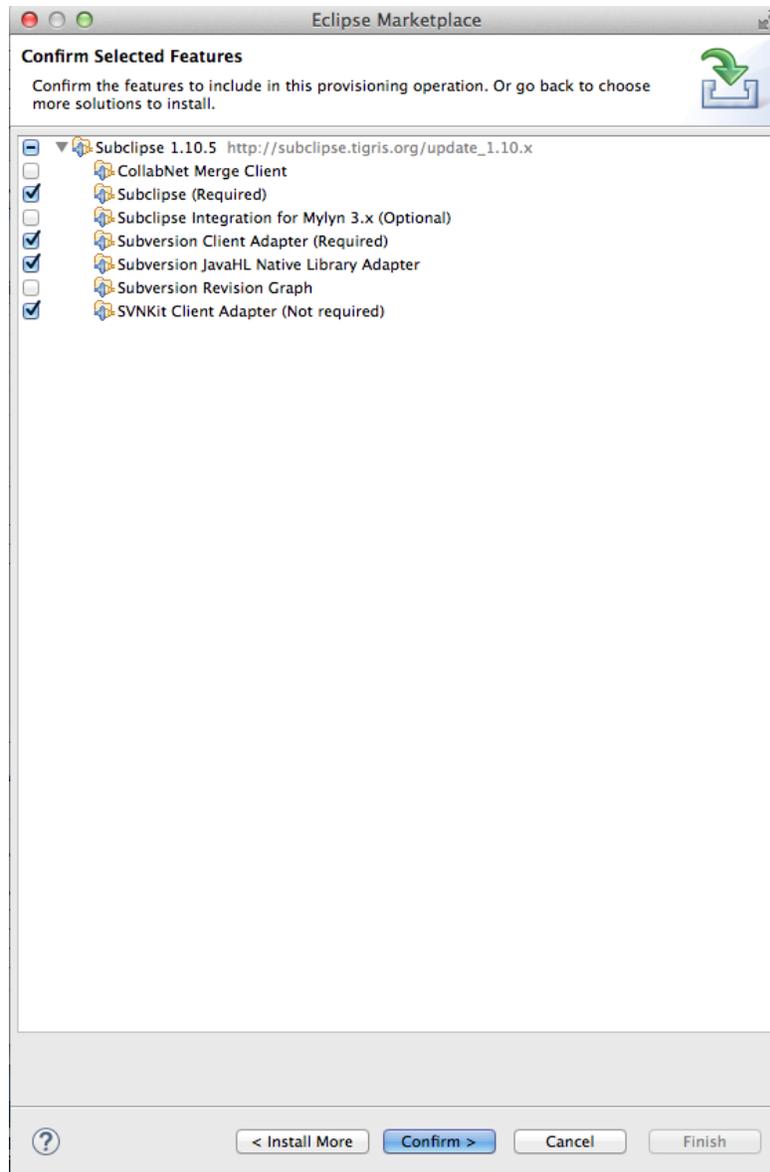


Figure 3-6. List of available software when downloading Subclipse

4 Import VERDI Source Code

4.1 Select File→Import

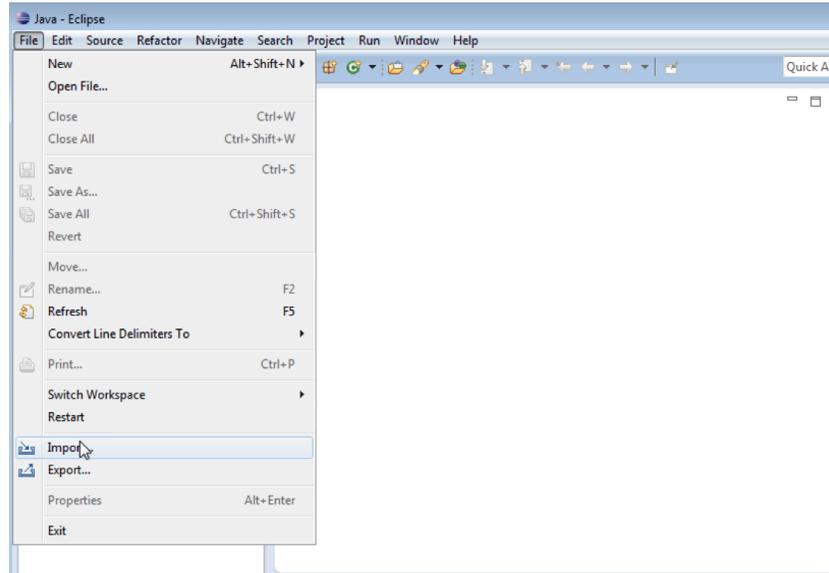


Figure 4-1. File/Import in Eclipse

To import the VERDI source code, use the mouse to select **File→Import** (Figure 4-1). This will generate a pop-up window titled Import (Figure 4-2).

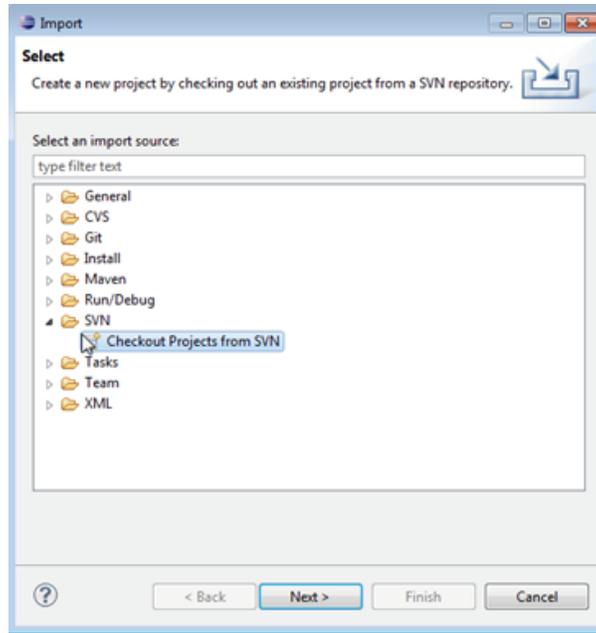


Figure 4-2. Import a project into Eclipse from SVN

4.2 Checkout Projects from SVN

Click on the **SVN** Folder to open it, and then select **Checkout Projects from SVN** by clicking on it (see Figure 4-3). Then click next.

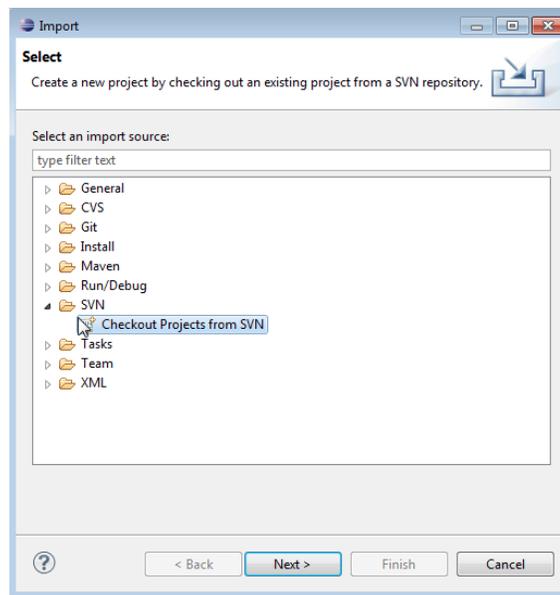


Figure 4-3. Checkout Projects from SVN

4.3 Create a New Repository Location

Use the mouse to highlight the button next to **Create a new repository location** (Figure 4-4), then click **next**.

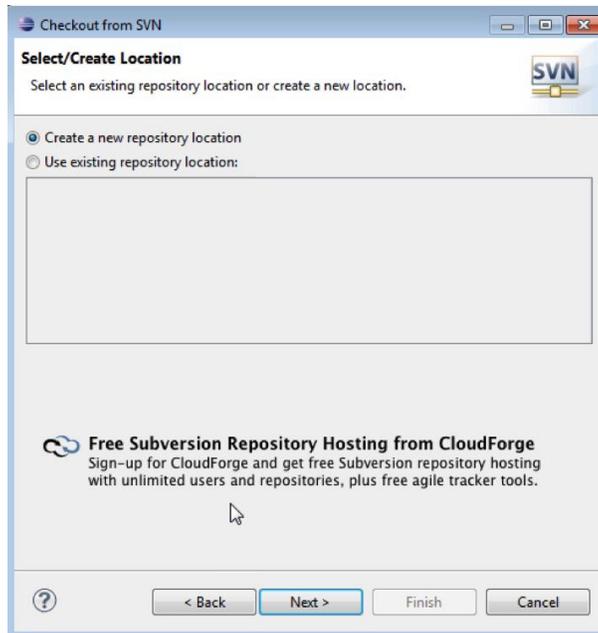


Figure 4-4. Create a new repository location

4.4 Specify Location of VERDI SourceForge Repository

Copy and paste the URL: https://svn.code.sf.net/p/verdi/code/branches/verdi_1.5 into the location field (Figure 4-5), then click next. You will see an “operation in progress” window while Eclipse establishes its connection to SourceForge.

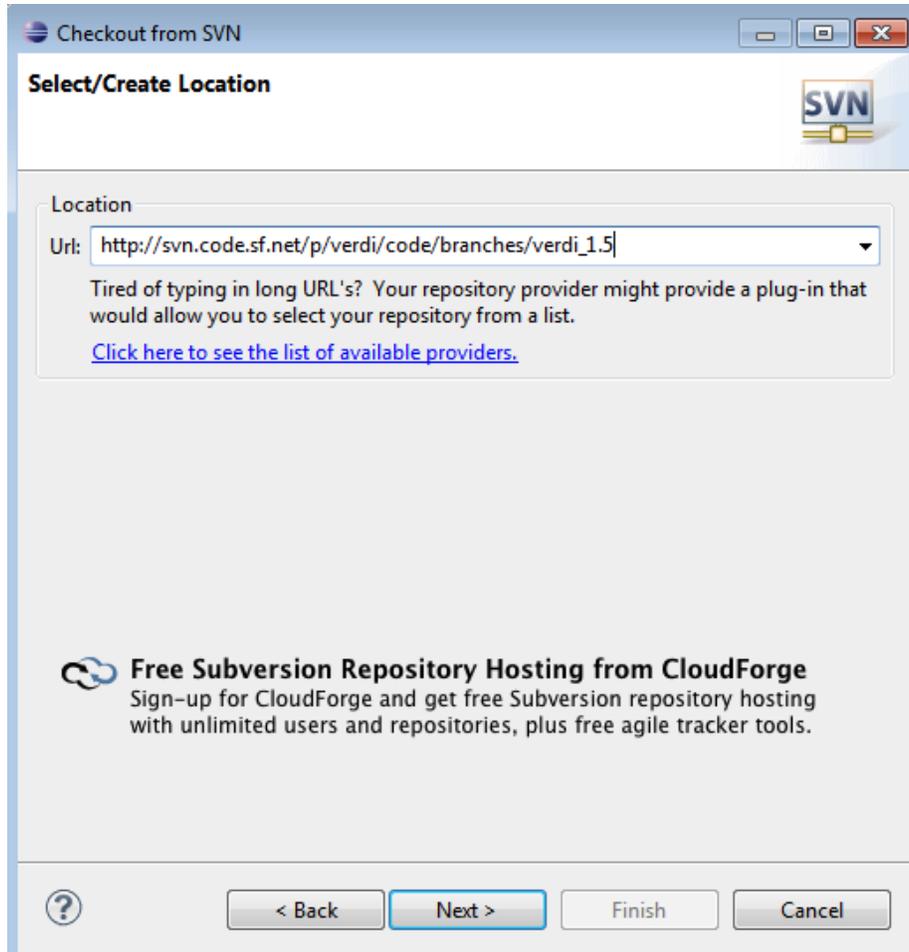


Figure 4-5. Checkout VERDI source code from SVN

4.5 Select Folders for Checkout

To download the latest version of the code (VERDI_1.5), click on `verdi_bootstrap` and then use the mouse and the shift key to highlight the following group of subfolders: **`verdi_bootstrap`**, **`verdi_core`**, **`verdi_data_loaders`**, **`verdi_dist`**, **`verdi_saf_core_runtime`**, **`verdi_saf_ui`**. Click the Finish button (Figure 4-6).

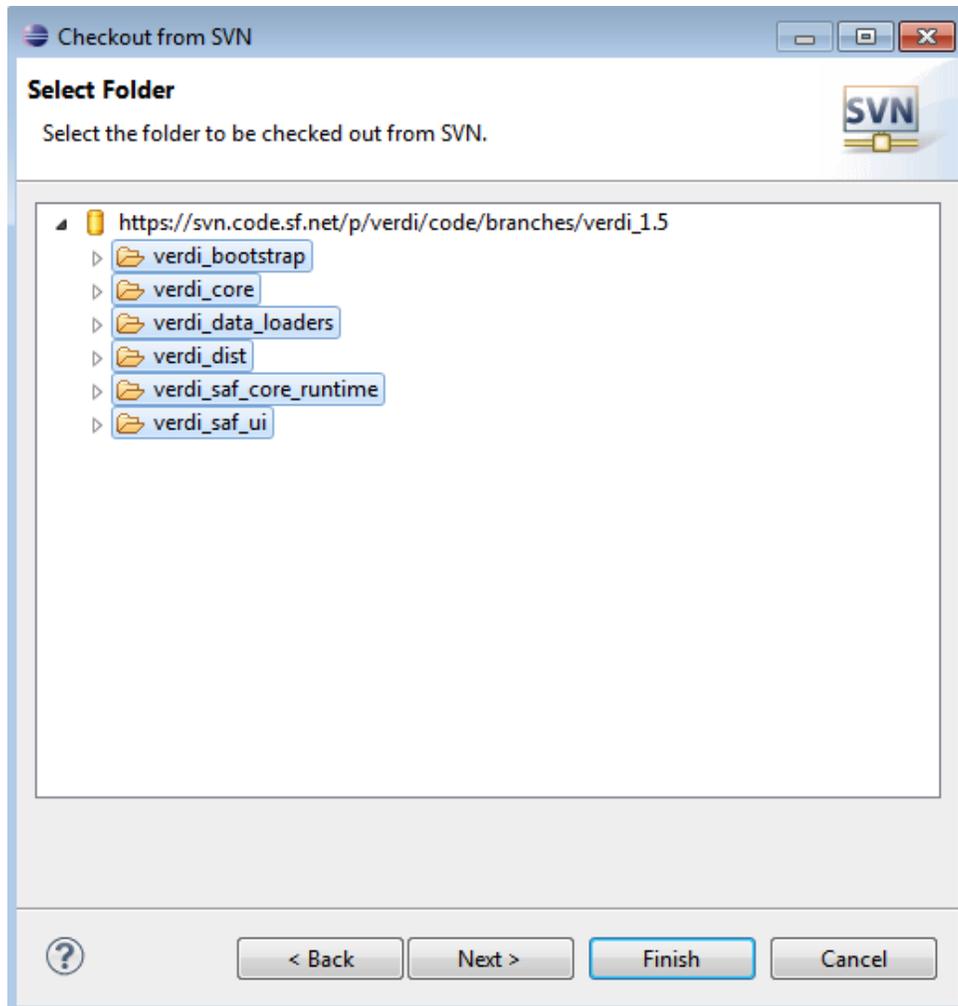


Figure 4-6. Select projects to check out from SVN

Eclipse then checks out the latest version of VERDI from the repository. The SVN checkout routine provides the option to run in the background, and also provides a meter indicating progress as the projects are being checked out (Figure 4-7). Eclipse displays a message in the “Problems” tab at the bottom of the workspace if there is an error. For example, Eclipse will display an error if you use a directory that already exists and Eclipse has permission problems copying files to the workspace directory. The workspace and the directory where the VERDI software is installed should not share the same location. Figure 4-8 shows that the code has been successfully imported into the workspace. A red X by one of the folders, on the other hand, indicates a problem.

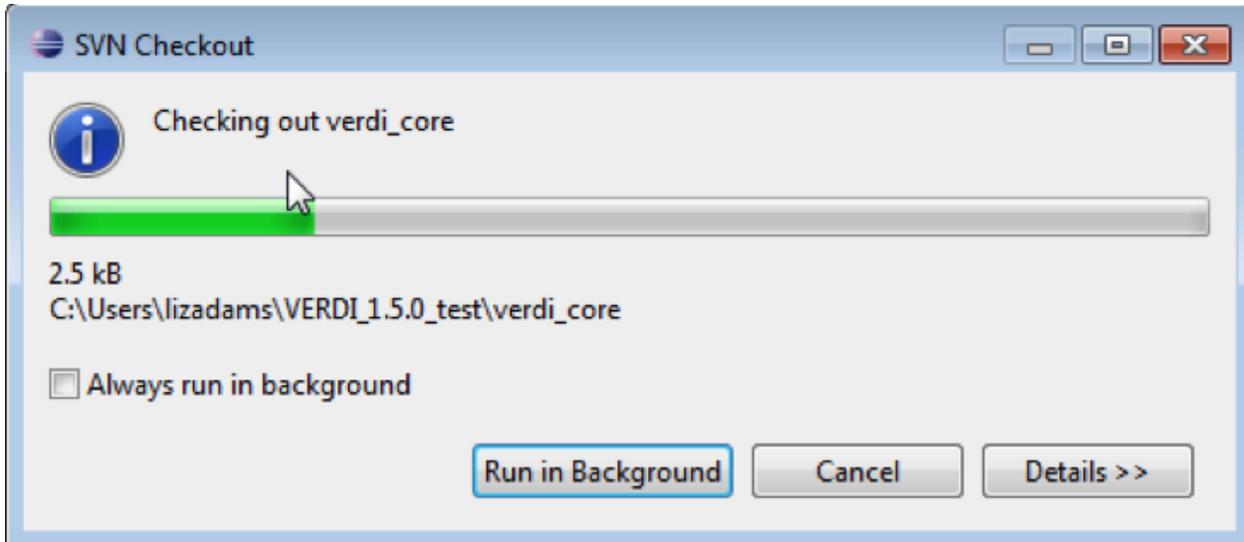


Figure 4-7. SVN Checkout includes a meter indicating progress of download

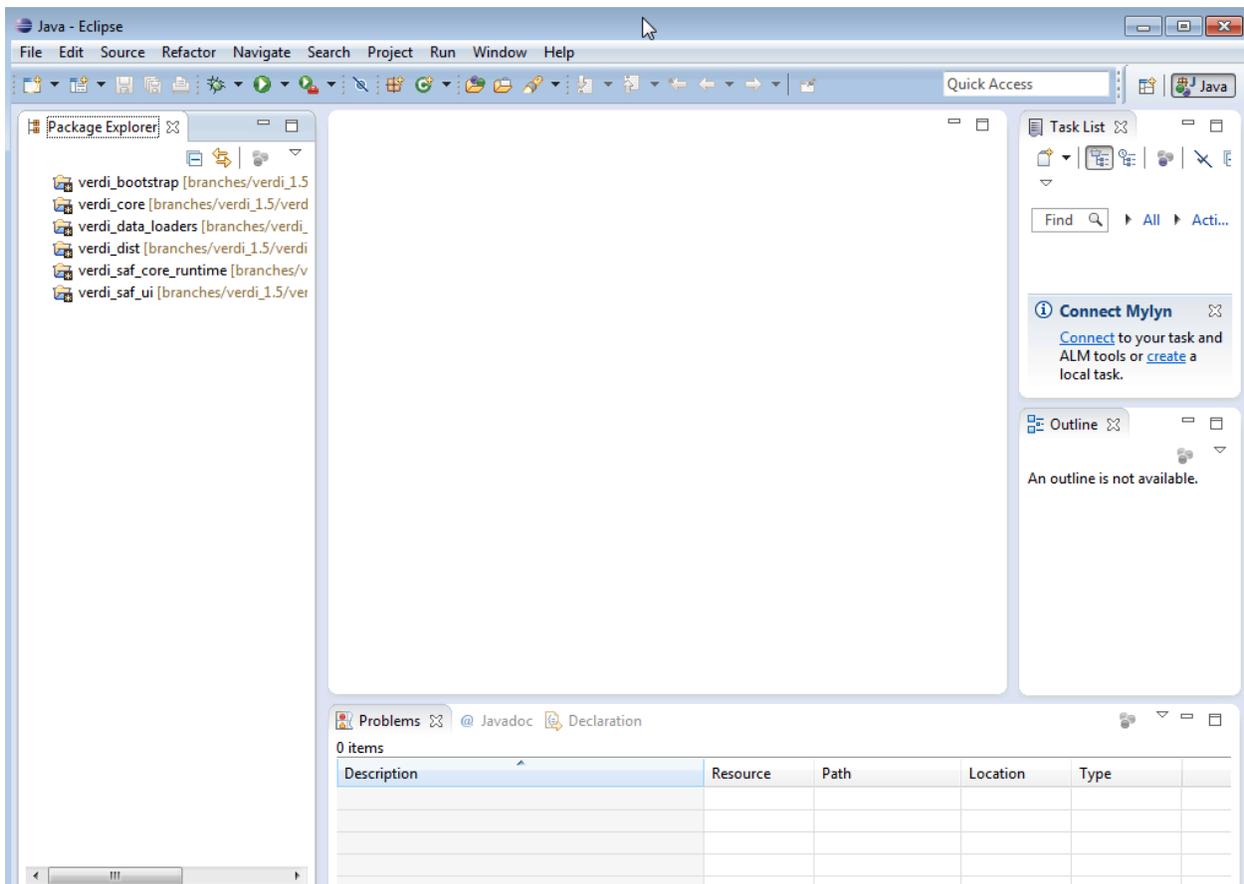


Figure 4-8. Projects listed in the Package Explorer of the Eclipse workspace

4.6 Source Code for Libraries

Some of the libraries that VERDI uses are open source and have their source code readily available. The source code to many of these libraries are distributed with VERDI 1.5.0 and are linked within the Eclipse project. Now if your debugging session needs to go into a library for which the source code is distributed, Eclipse should be able to display the source code for you.

All of this source code is included in the `verdi_core/lib_src` directory of your VERDI source code installation. As shown in Figure 4-9, the library source files are provided as jar or zip files.

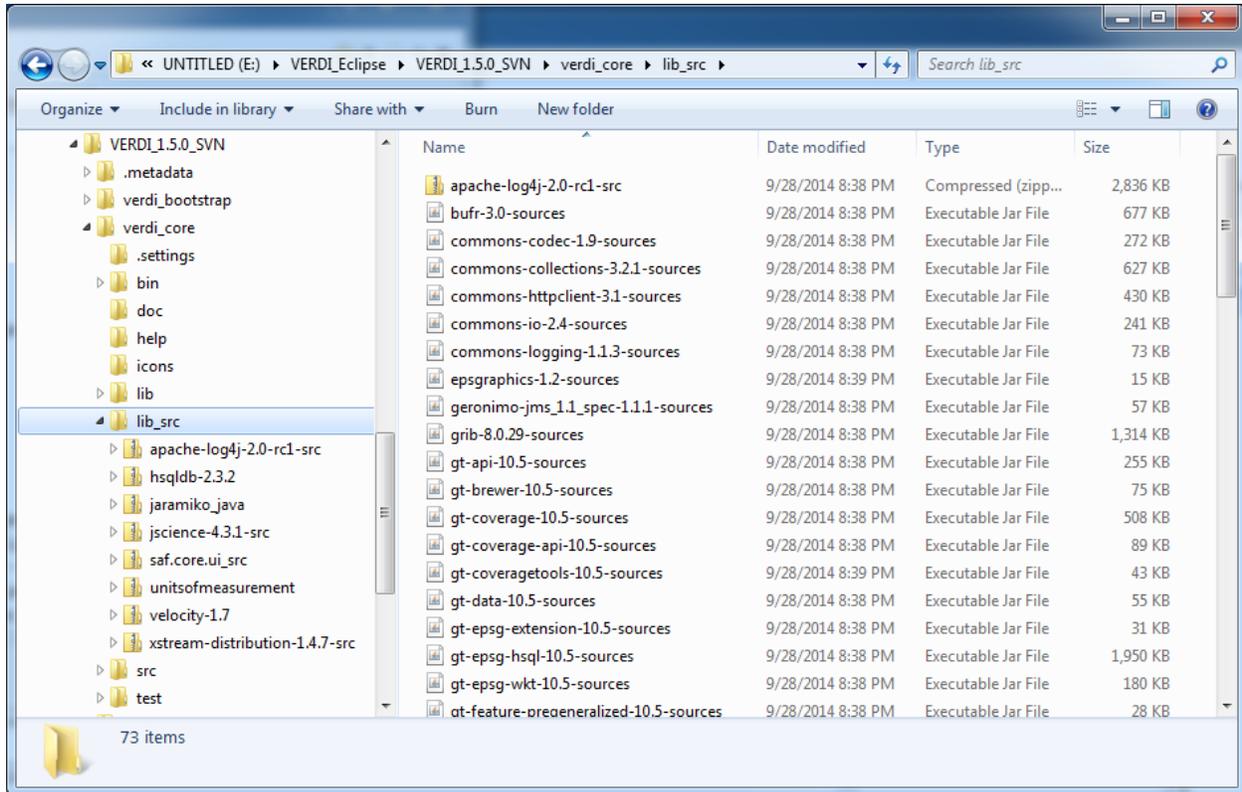


Figure 4-9. Location of source code for open source libraries

Each library is cross-referenced within Eclipse from its executable jar file to its source file. To see the libraries used by one of the projects, `verdi_data_loaders` for example, right-click on `verdi_data_loaders` in the Eclipse Package Explorer. This brings up the Properties box. Select Java Build Path on the left-hand side and the four tabs then open – Source, Projects, Libraries, and Order and Export (Figure 4-10).

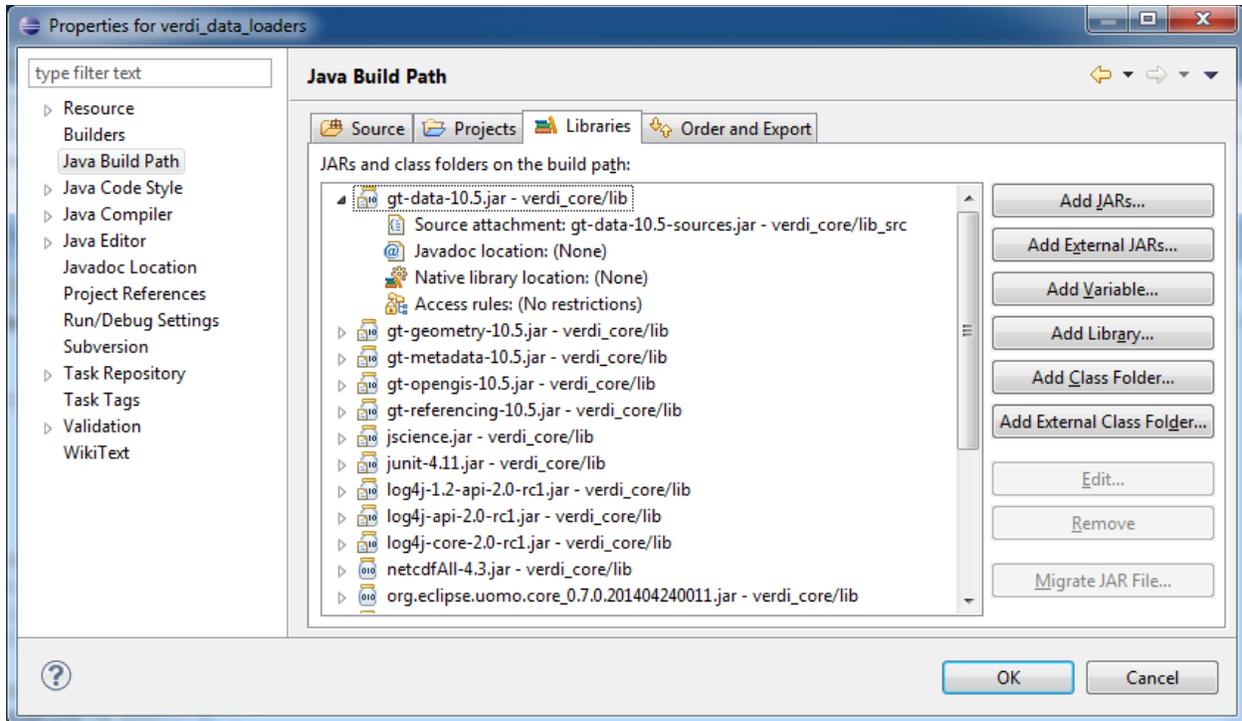


Figure 4-10. Cross-reference of source code and class libraries within an Eclipse project

As shown in Figure 4-10, the library `gt-data-10.5.jar` within the directory `verdi_core/lib` is cross-referenced to the `gt-data-10.5-sources.jar` within `verdi_core/lib_src`. Note that both the class and source libraries are located in `verdi_core` directory structure, although the properties for the `verdi_data_loaders` project are shown. The libraries that are used for multiple Eclipse projects within VERDI are stored under `verdi_core`, which is the largest project. All library source code that is distributed with VERDI is located within `verdi_core/lib_src`.

5 Configure Apache Ant to Use tools.jar from the JDK

Apache Ant is a software tool for automating the software build process. It is provided with Eclipse.

You need to edit the General Ant Preferences to add the tools.jar from the JDK. To do this, select Window> Preferences as shown in Figure 5-1.

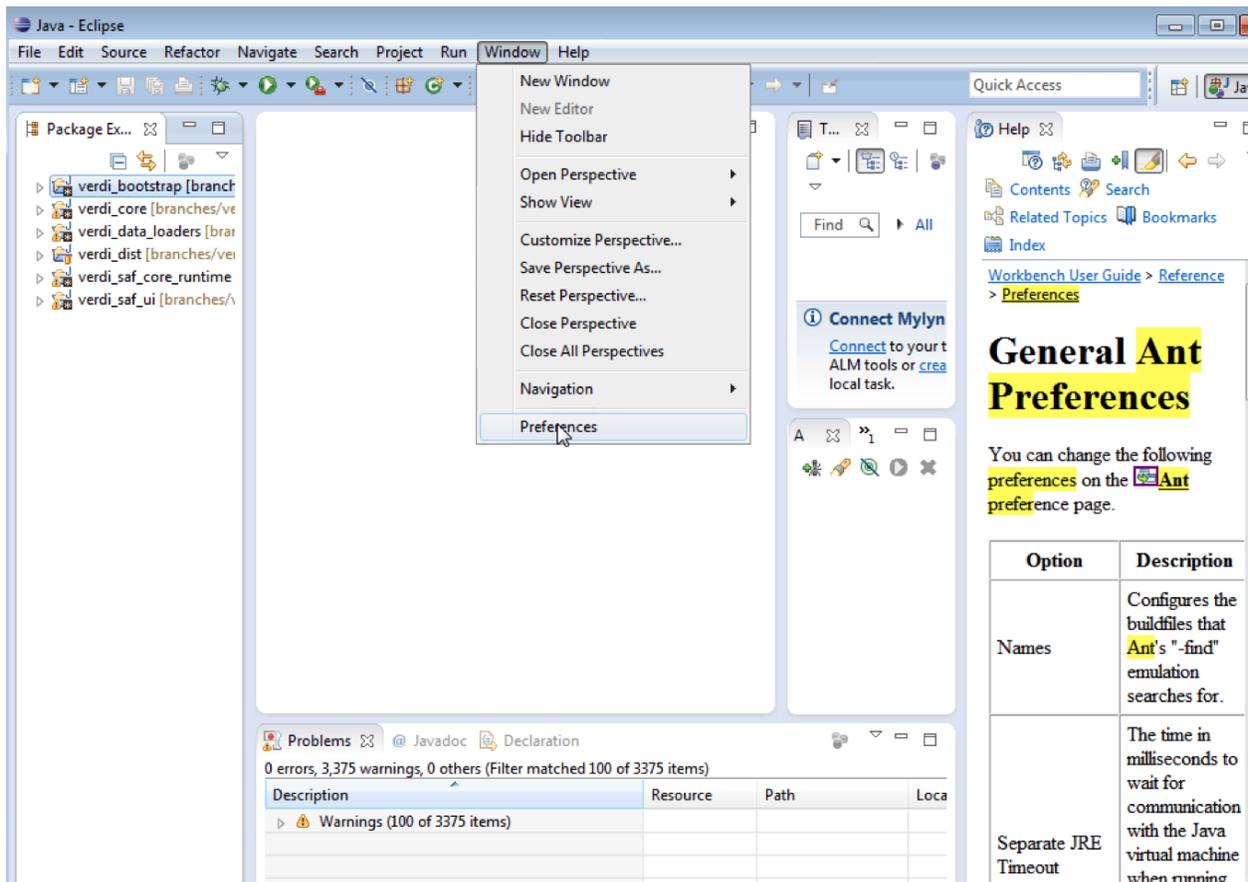


Figure 5-1. Eclipse preferences

Next, select Ant> Runtime> Global Entries as shown in Figure 5-2.

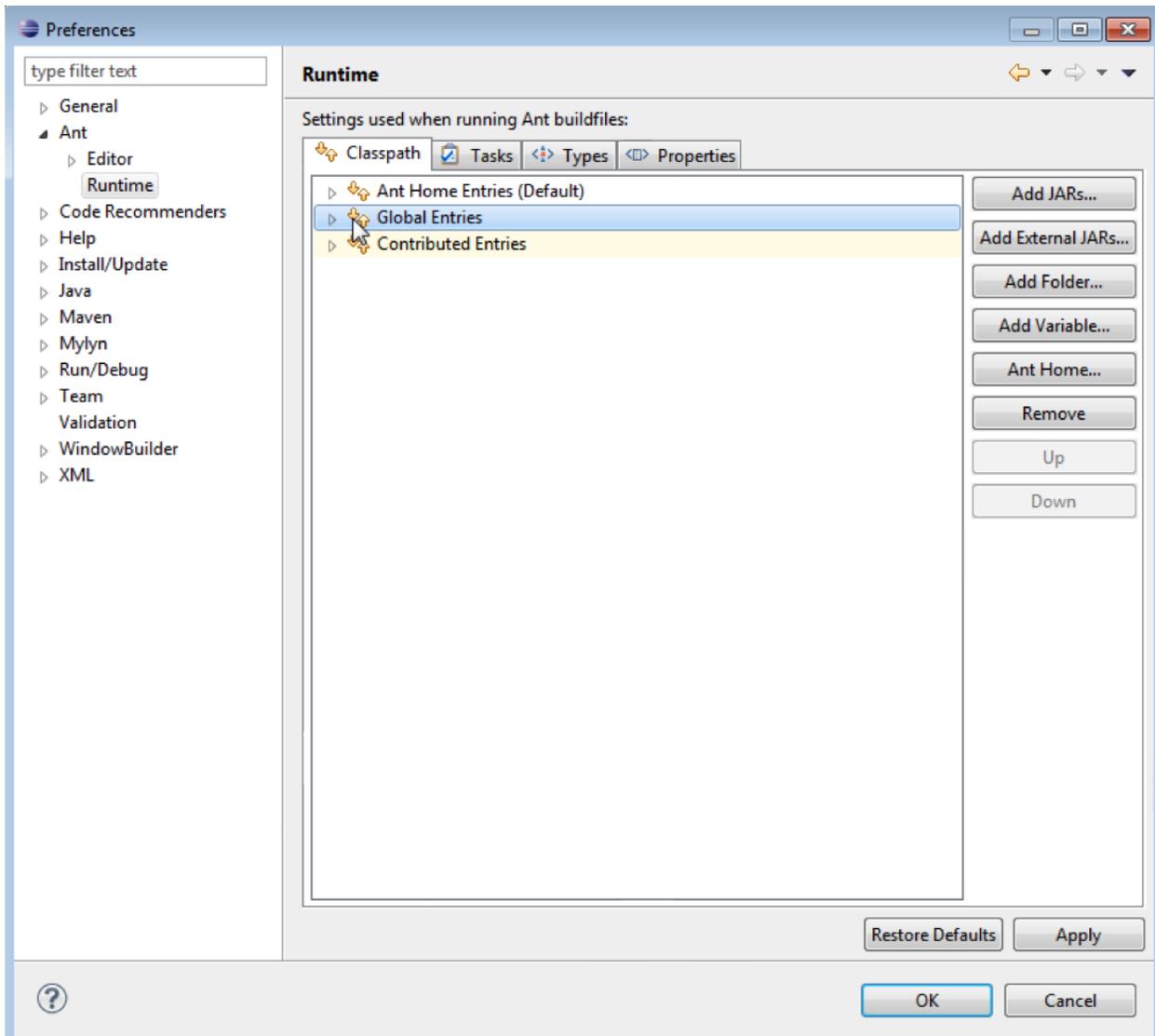


Figure 5-2. Ant preferences within Eclipse

Then select **Add External JARS** and navigate to the location where the JDK is installed on your computer. Next, browse to the lib, select tools.jar and press the Open button (Figure 5-3). Finally, press the Apply button followed by the OK button.

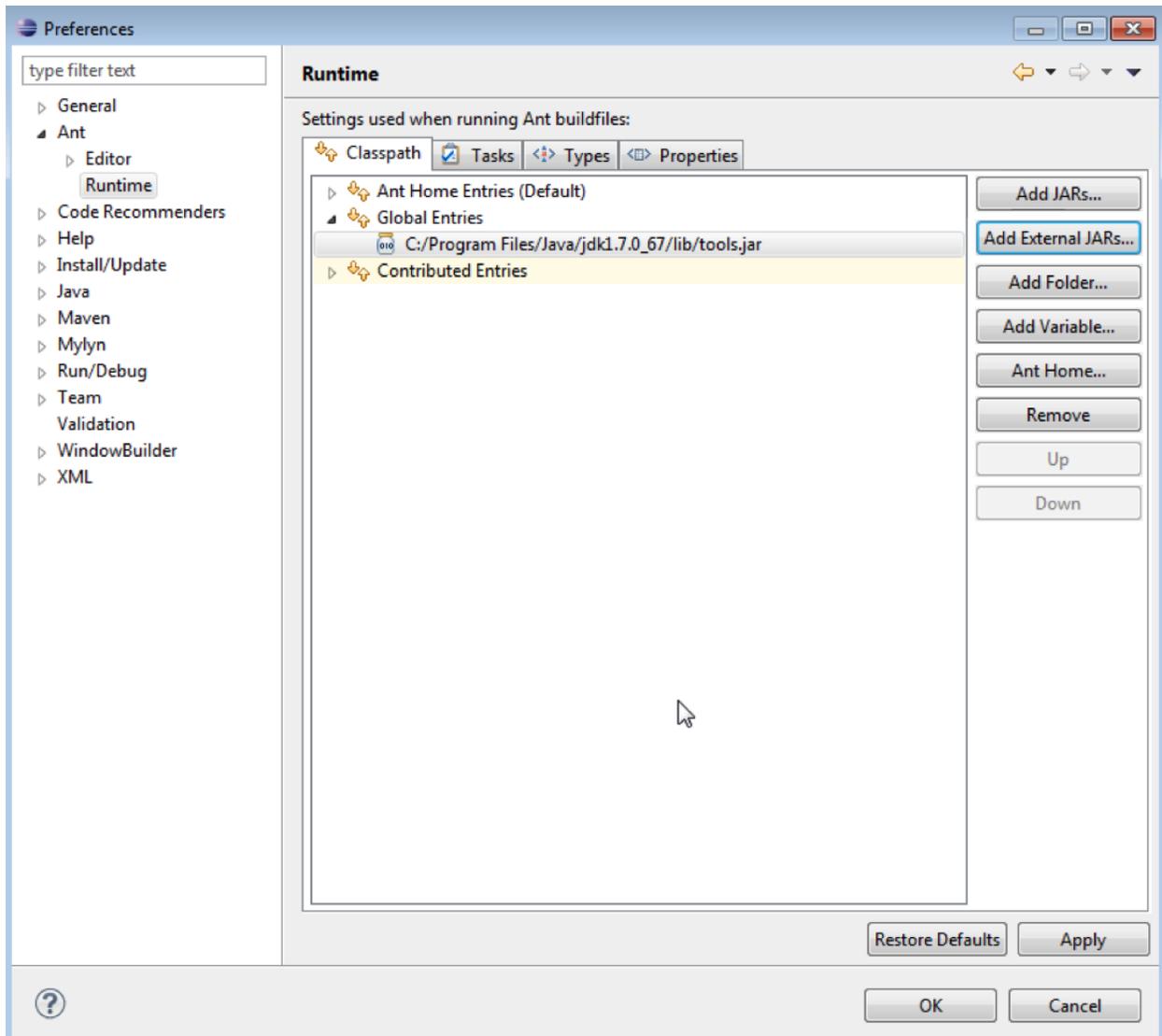


Figure 5-3. Add tools.jar to Ant classpath

6 Set Eclipse Preferences

This chapter provides recommended Eclipse settings for building VERDI.

6.1 Workspace Preferences

Eclipse can be set up to build the projects automatically after a developer makes local changes to the Java source code. To automatically build after source code changes are made, enable this preference using the Eclipse menus (Figure 6-1):

Window > Preferences > General > Click on Workspace >

NOTE: Any options that you set here are for all of the projects in this workspace.

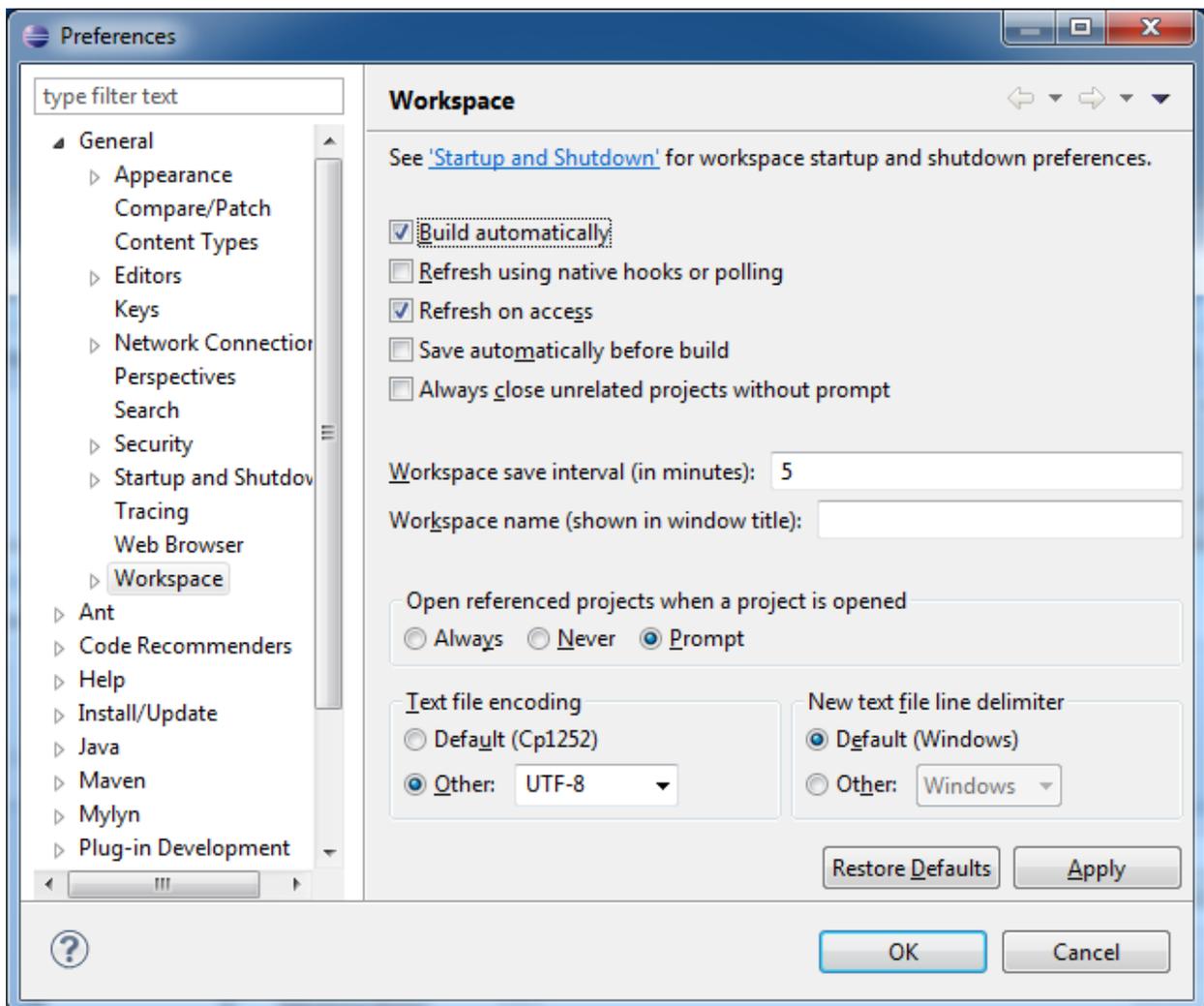


Figure 6-1. Eclipse preferences

If you want Eclipse to automatically rebuild your projects as you change a file, check the Build Automatically checkbox (Figure 6-1). Note that this option can slow down your development if you are making several changes because your projects will rebuild after each change.

There is also a setting to automatically recognize files that are added to the workspace. To automatically synchronize the workspace with the underlying file system, check the Refresh on Access option (Figure 6-1).

If your code is to be used on multiple platforms, go to “Text File Encoding” near the bottom of the window. Click the radio button to the left of “Other” and select “UTF-8”. Click the Apply button and then the OK button.

6.2 Verdi_core Properties

In the Package Explorer view, right click on verdi_core and select Properties at the bottom of the pop-up menu. A pop-up window titled Properties will appear for verdi_core (Figure 6-2).

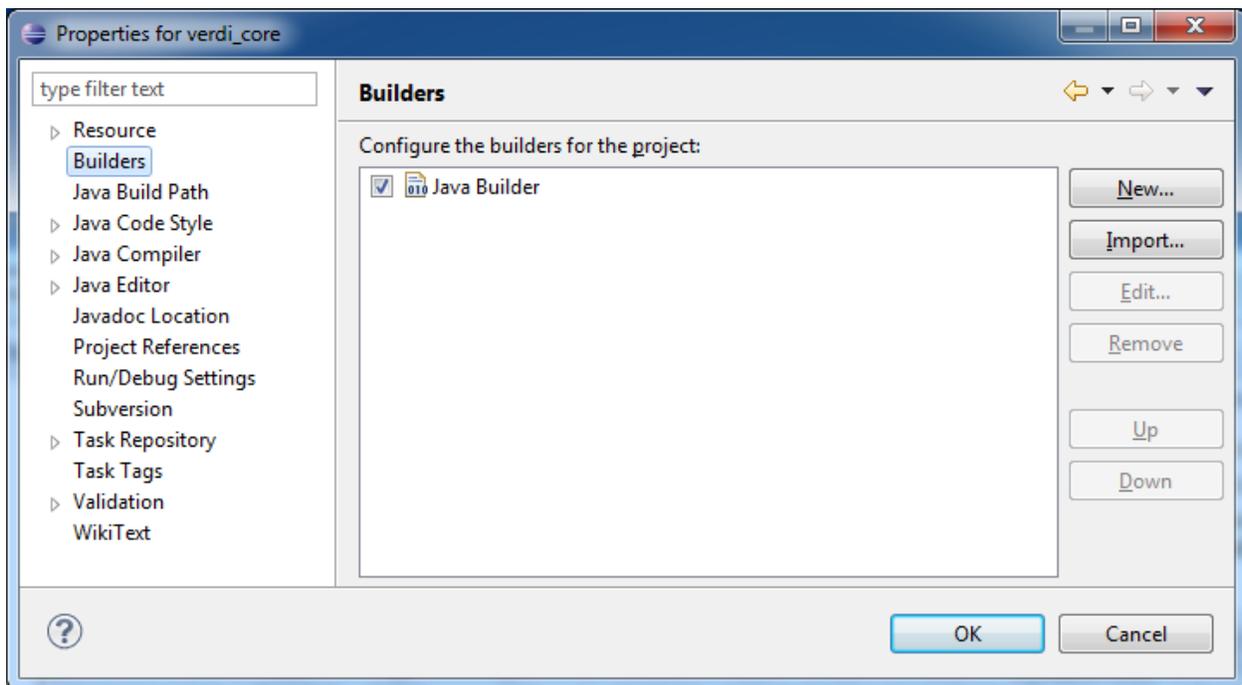


Figure 6-2. Project references for verdi_core

6.2.1 Java Build Path

Select Java Build Path. The right side of the window then shows four tabs, containing information on the **Source** folders, the required **Projects**, the **Libraries** (Java ARchives (JARS) and class folders on the build path), and the **Order and Export** (entries that are selected for export to dependent projects) (Figure 6-3).

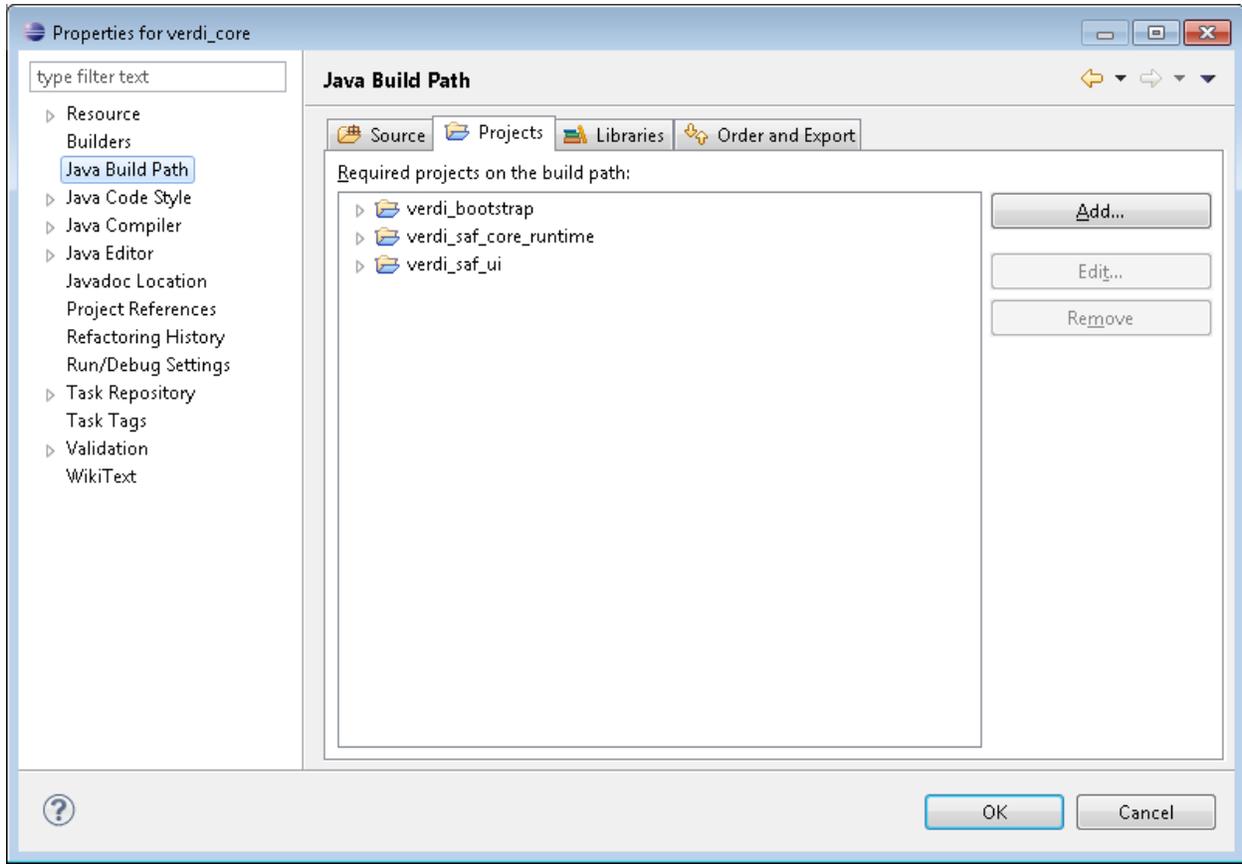


Figure 6-3. Dependent projects for verdi_core

Figure 6-3 shows that the `verdi_core` project depends upon three other projects – `verdi_bootstrap`, `verdi_saf_core_runtime`, and `verdi_saf_ui`. Therefore, these projects must be built and available to the Java compiler when `verdi_core` is built. Also, note that the latter two projects are part of the Repast Symphony library. You should not need to change these dependencies.

6.2.2 Java Compiler

From the `verdi_core` Properties window, select Java Compiler (Figure 6-4). The panel on the right side shows the version of the JDK that is currently being used by VERDI. Note that the compliance settings are all set to Java 1.7. Also, the check mark at the top enables project-specific settings for the Java Compiler.

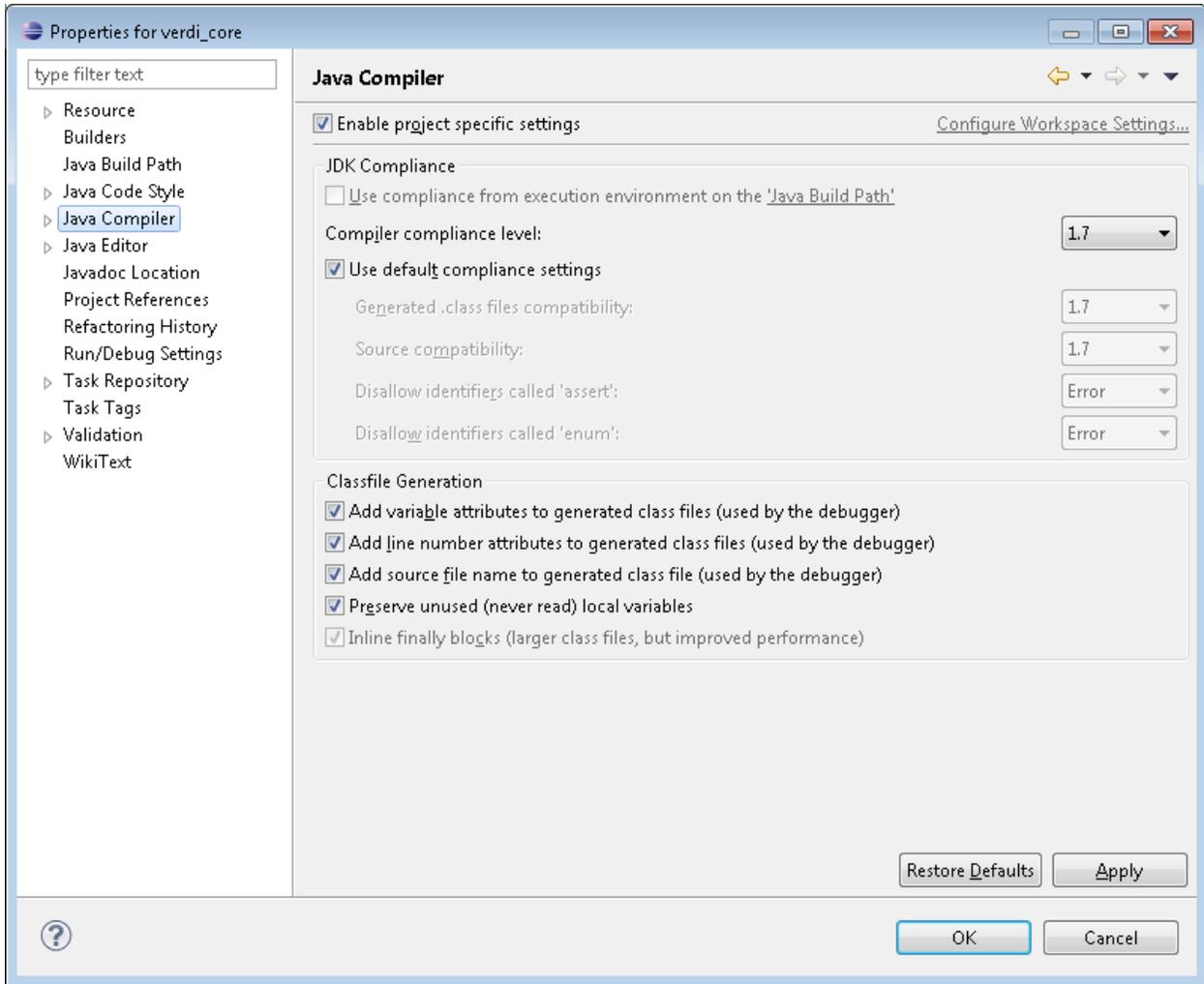


Figure 6-4. Project-specific settings for the Java compiler

The Classfile Generation section, which is beneath the compliance settings, has several options that are used by the debugger. If you plan on running VERDI in the Eclipse debugger, these settings should be checked.

After verifying all of your settings, click the Apply button and then the OK button.

7 Build the NetCDF-Java Library with Modifications for VERDI

You may skip this chapter unless you need to make changes to the NetCDF-Java library. Note that VERDI uses a modified version of the netcdfAll library.

7.1 Set up NetCDF-Java Library in Eclipse

Follow these steps to download version 4.3.22 of the NetCDF-Java Library from GitHub. Git and maven are required.

1. Check if mvn is installed on your machine using
 - a. `mvn --version`
 - b. The output should be something close to: Apache Maven 3.2.3
2. If mvn is not found, install Maven on your machine by following the Installation Instructions found on the Maven Download site: <http://maven.apache.org/download.cgi>
3. Determine if git is installed on your machine using
 - a. `git --version`
 - b. The output should be something close to: git version 1.8.5.2
4. If git is not found, install Git on your machine using the following instructions: <http://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
5. To obtain the thredds source code from the Unidata/thredds GitHub site: <https://github.com/Unidata/thredds>
 - a. Download the 4.3.22 version using the command:
`git clone -b target-4.3.22 git://github.com/Unidata/thredds.git`
6. Within Eclipse, select File>Import>Maven>Existing Maven Projects
7. Browse to the directory where git downloaded the 4.3.22 version. Use the top level thredds directory as the Root Directory to import.
8. Eclipse will import and use maven to build the class files for thredds.
9. There will be some errors in the opendap and ui projects. Change package import statement on files that did not compile correctly from `opendap.util.gui` to `opendap.tools.gui`. Under `ucar.nc2.dods` remove `#import ucar.nc2.DODSNode;`. For `CoordSysTable.java` use `import ucar.ma2.DataType` to solve the issue of the unresolved `DataType`. The following link has tips on solving these type of errors: <https://meteo.unican.es/trac/wiki/TutorialMaven>
10. There are three files that have been modified for VERDI: `WRFConvention.java`, `M3IOConvention.java` and `Stereographic.java`
11. The versions specific to VERDI are found under:
`verdi_data_loaders/src/ucar/nc2/dataset/conv/M3IOConvention.java`
`verdi_data_loaders/src/ucar/nc2/dataset/conv/WRFConvention.java`
`verdi_data_loaders/src/ucar/unidata/geoloc/projection/Stereographic.java`

These versions need to be copied from the above directories to replace these files under thredds:

```
cp Stereographic.java thredds/cdm/src/main/java/ucar/unidata/geoloc/projection
```

```
cp M3IOConvention.java thredds/cdm/src/main/java/ucar/nc2/dataset/conv/
```

```
cp WRFConvention.java thredds/cdm/src/main/java/ucar/nc2/dataset/conv/
```

12. Refresh the cdm project. Note the WRFConvention.java uses the logger log4j, so you need to add the log4j to the build path.
13. After the files have been compiled in eclipse, then do a mvn install from the command line (outside of eclipse) to build the jar file needed by VERDI.
14. This will create a netcdfAll-*.jar under thredds/ui/target
15. Copy the netcdfAll-*.jar to /verdi_core/lib/netcdfAll-4.3.jar
16. Refresh the verdi_core project and rebuild VERDI

8 Test VERDI Using Scripts within Eclipse

Scripts are available for testing VERDI within Eclipse for several plot types.

8.1 View Scripts within Eclipse

From the Eclipse Main Menu, select either Run>Run Configurations or Run> Debug Configurations if you want to run the script within the debugger. Then, select Java Application to view the scripts (Figure 8-1).

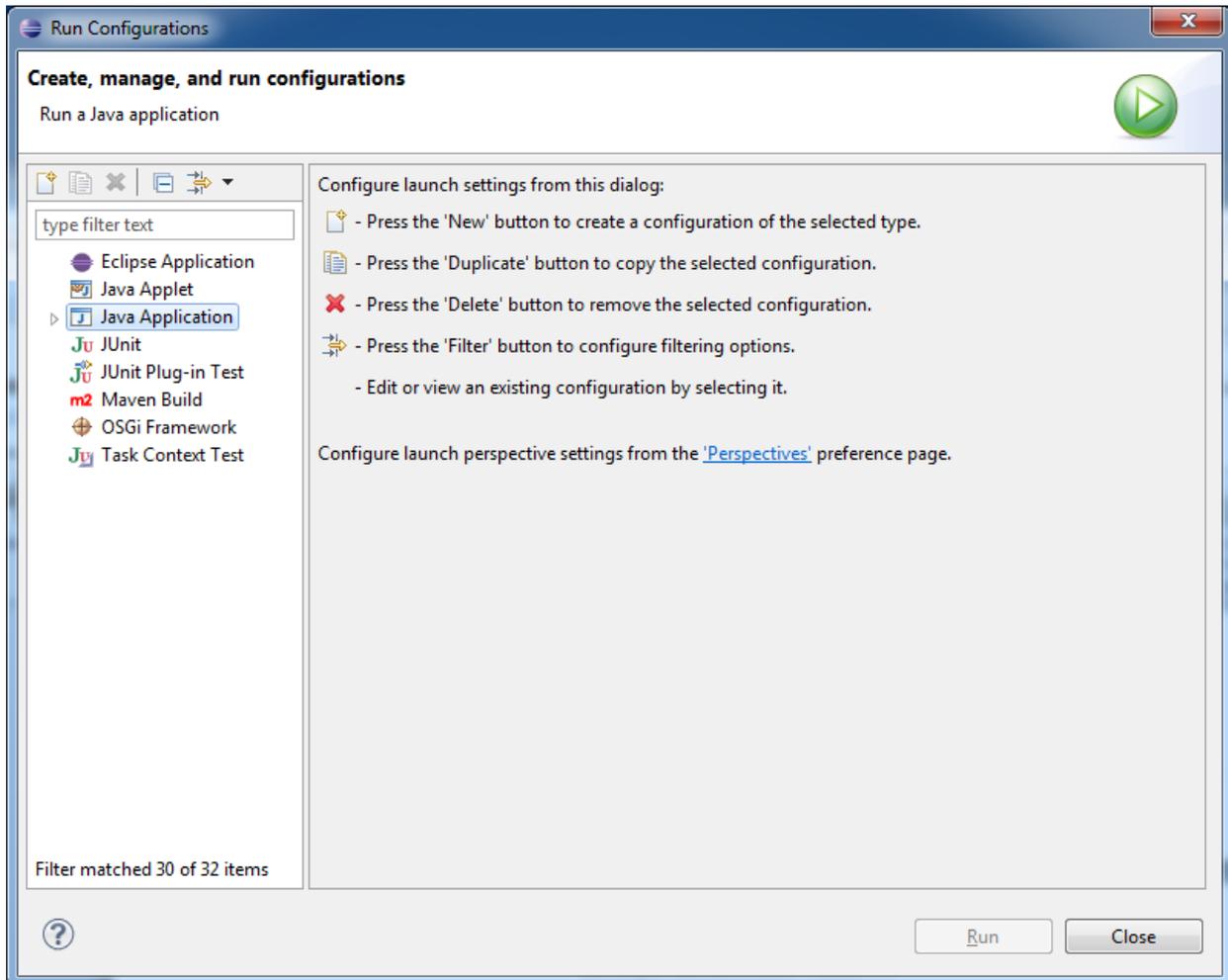


Figure 8-1. Run configurations

The script names include `verdi_script`, `verdi_script_batch`, `verdi_script_camx`, `verdi_script_twoBars`, and `verdi_script_vertical_crosssection`. Select `verdi_script`, and then click on the arguments tab to view the command-line arguments that are passed to VERDI in the script (Figure 8-2). When you run the script, VERDI automatically loads the data and creates plots using the script commands specified in the arguments tab. Setting up and running scripts shorten the time required to debug plot issues because plots can be reproduced more quickly. (Note: The pathnames are specified relative to the `distfiles/data` directory in the arguments tab. This allows developers to run the test scripts on different platforms [Windows, Linux, or Mac] without having to edit the pathname to load the data correctly.)

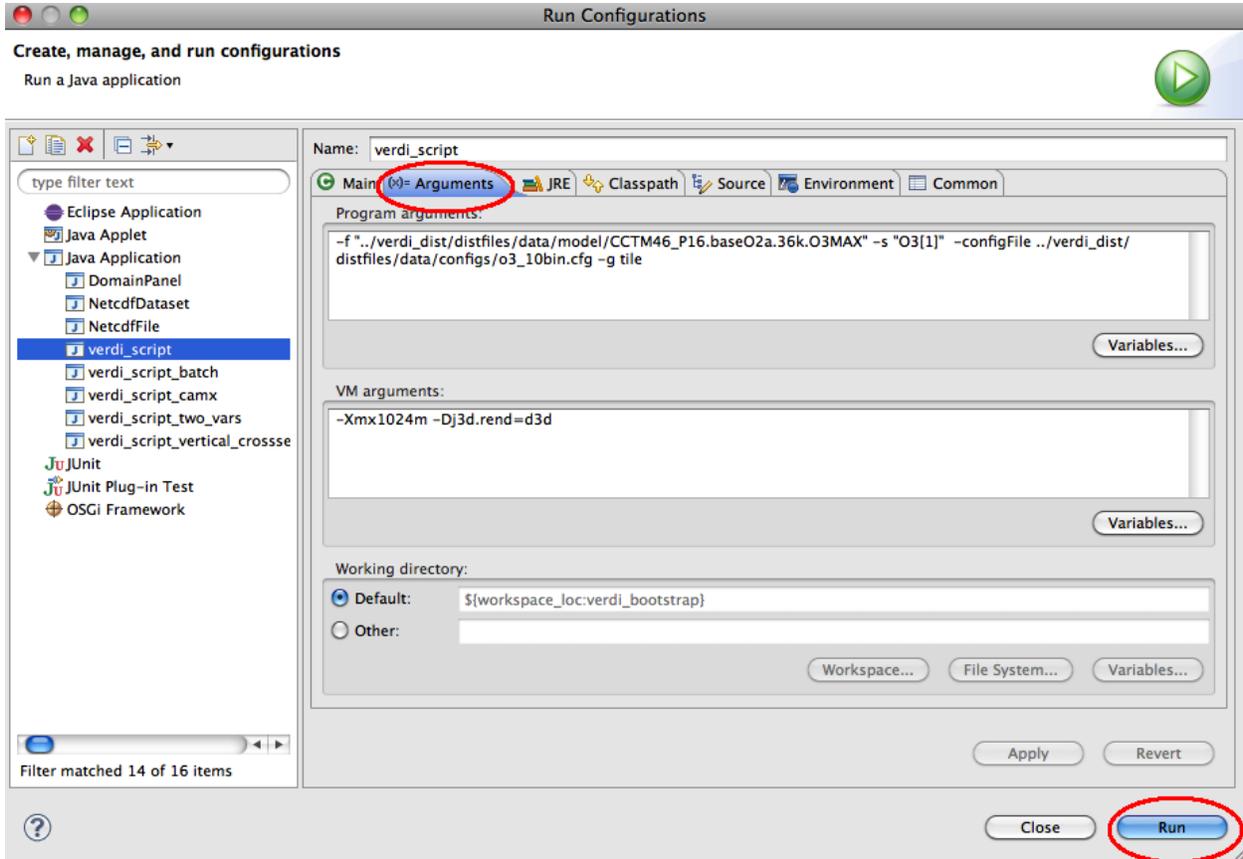


Figure 8-2. Sample configuration for a VERDI script

Before you run the script, you need to add the `VERDI_HOME` environment variable to point to a location with the distribution files (the files that are available after you build VERDI within eclipse). Click on the **Environment** Tab shown in Figure 8-3.

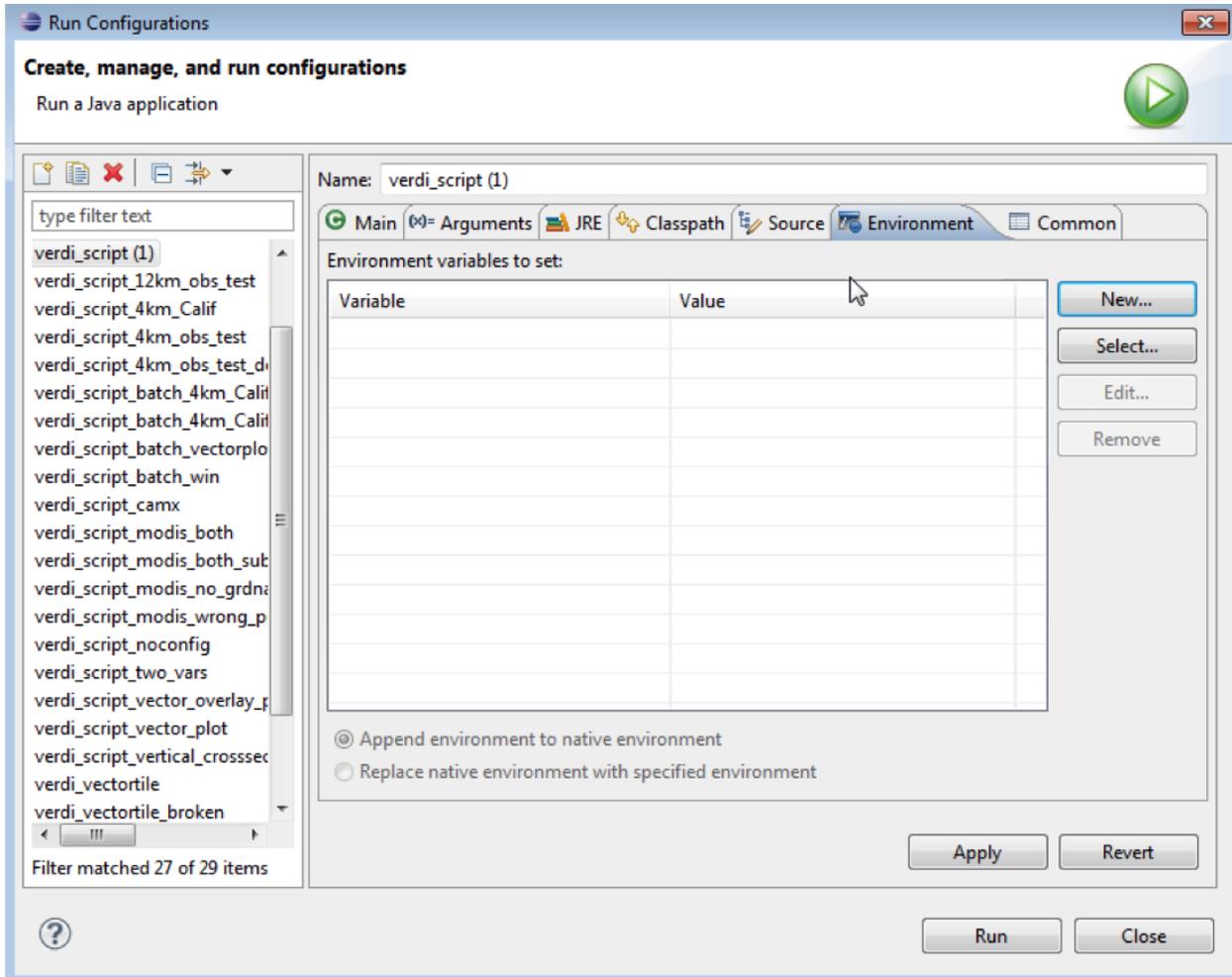


Figure 8-3. Tab to set environment variables for a run configuration

Add the environment variable VERDI_HOME and have it point to your eclipse workspace VERDI (see Figure 8-4). At this point you are able to debug VERDI within Eclipse using the verdi_script.launch file.

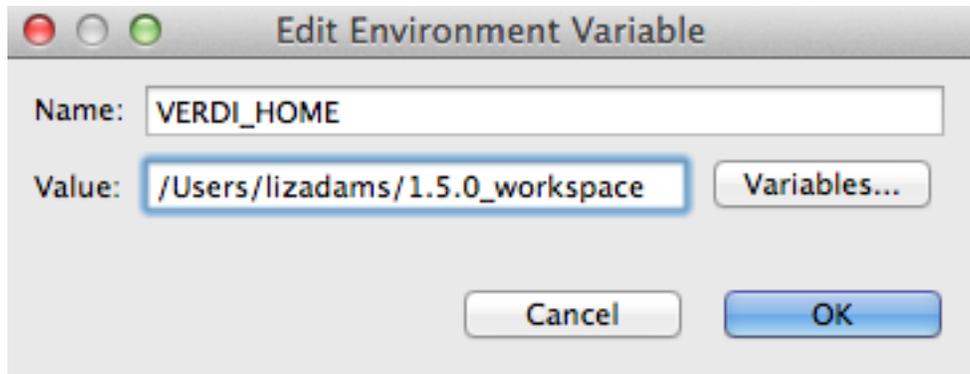


Figure 8-4. Specify VERDI_HOME environment variable

9 Update Source Code from the Repository

If you have previously obtained the VERDI source code from the repository, the synchronization window allows you to obtain any bug fixes or new features that developers have committed.

9.1 Open the Synchronization Window

Select Window→Show View→Other to open the **Show View** pop-up window (Figure 9-1). Expand the Team Folder (Figure 9-2) by clicking on the plus symbol, then highlight the word Synchronize and click OK. The synchronize window will be created at the bottom of the workspace (Figure 9-3).

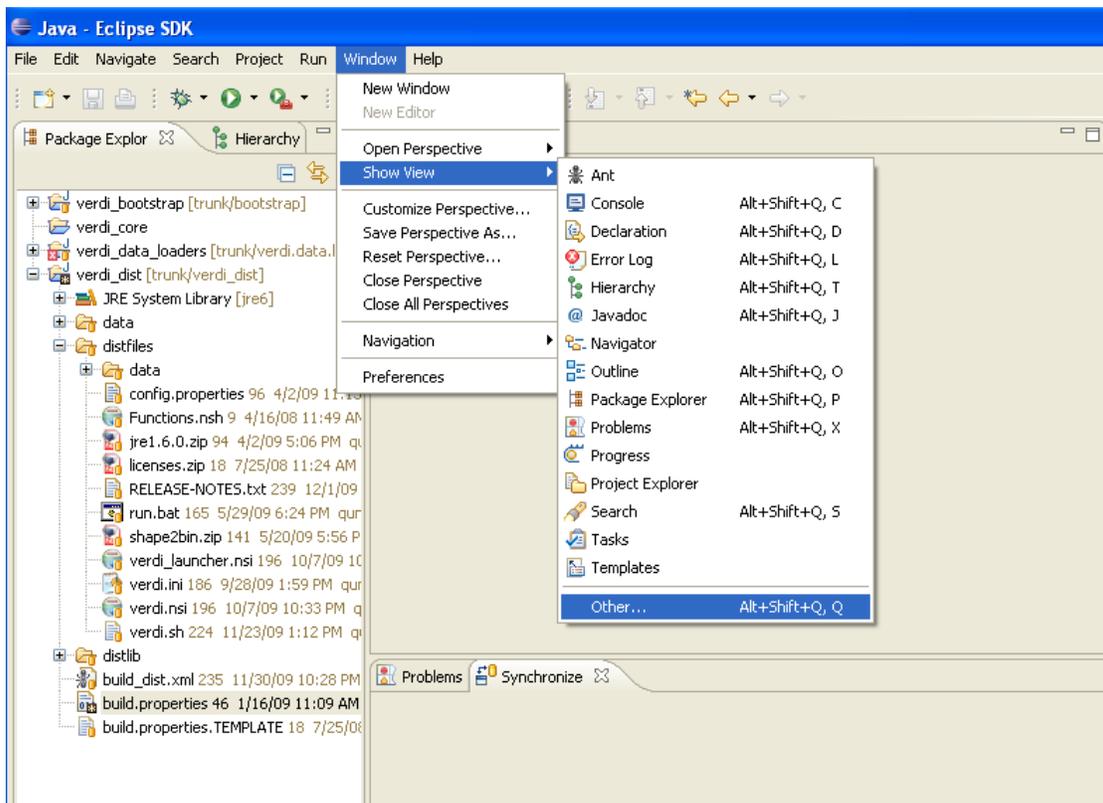


Figure 9-1. Show View → Other

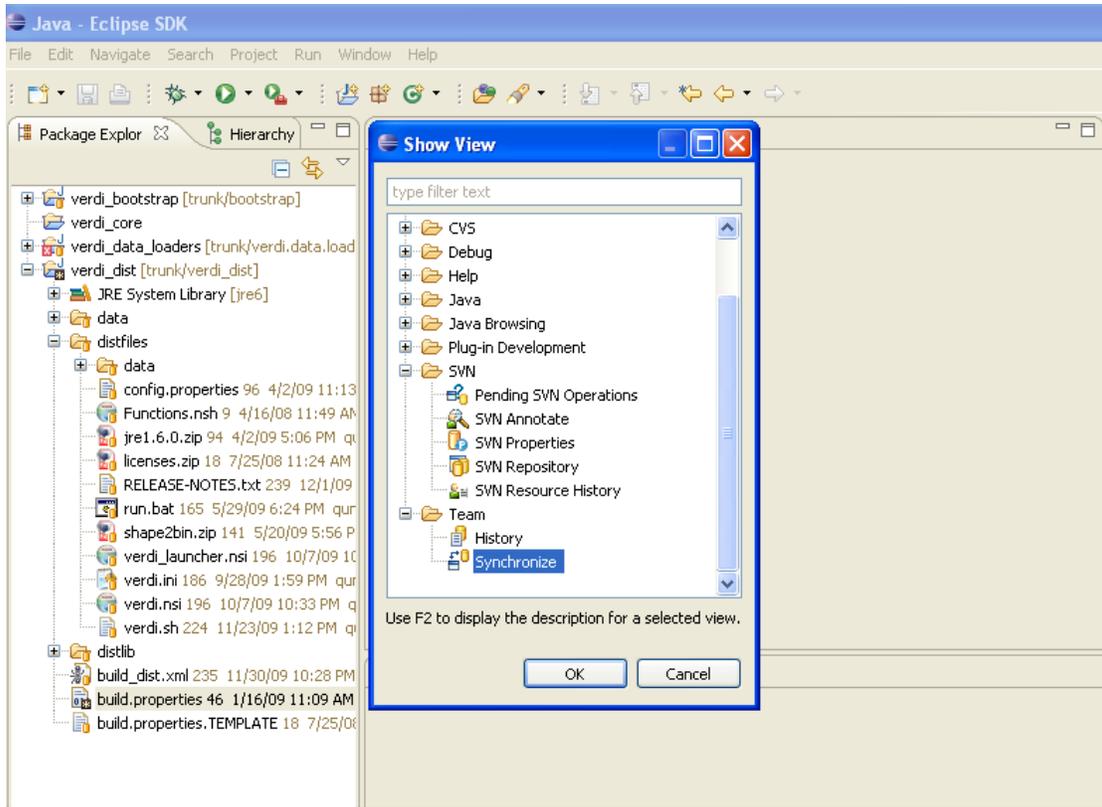


Figure 9-2. Expand team folder, highlight Synchronize, and press the OK button

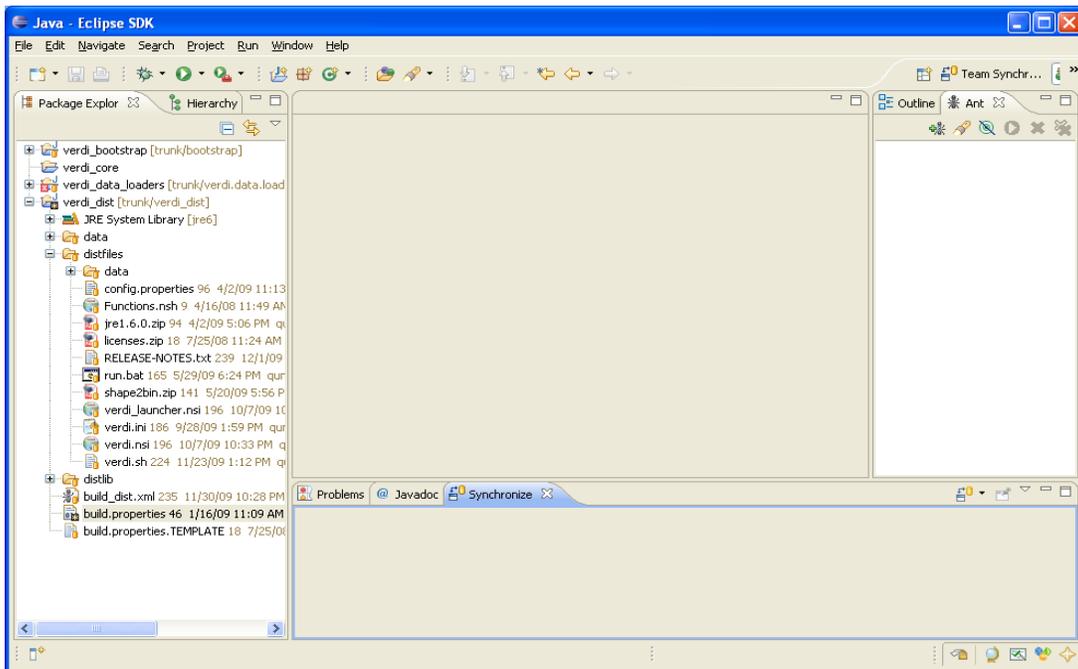


Figure 9-3. Synchronize window added to bottom of workspace

9.2 Synchronize with Repository Using SVN

The Synchronize window has a synchronize symbol in its upper right corner (Figure 9-4). Click on the synchronize symbol to open the Synchronize pop-up window. Click on SVN to select the subversion software package manager, then click the Next button (Figure 9-5). A Synchronize SVN pop-up window then appears listing the packages that are available as resources for synchronization (Figure 9-6). Click the Select All button to select all the packages, then click the Finish button. A pop-up window labeled Confirm Open Perspective will ask if you would like to change from the Java Perspective to the Team Synchronization Perspective. If you opt to change perspectives, there is a right arrow button in the upper right-hand corner of the workspace to switch back to the Java Perspective when you finish reviewing the code in the Team Synchronization Perspective.

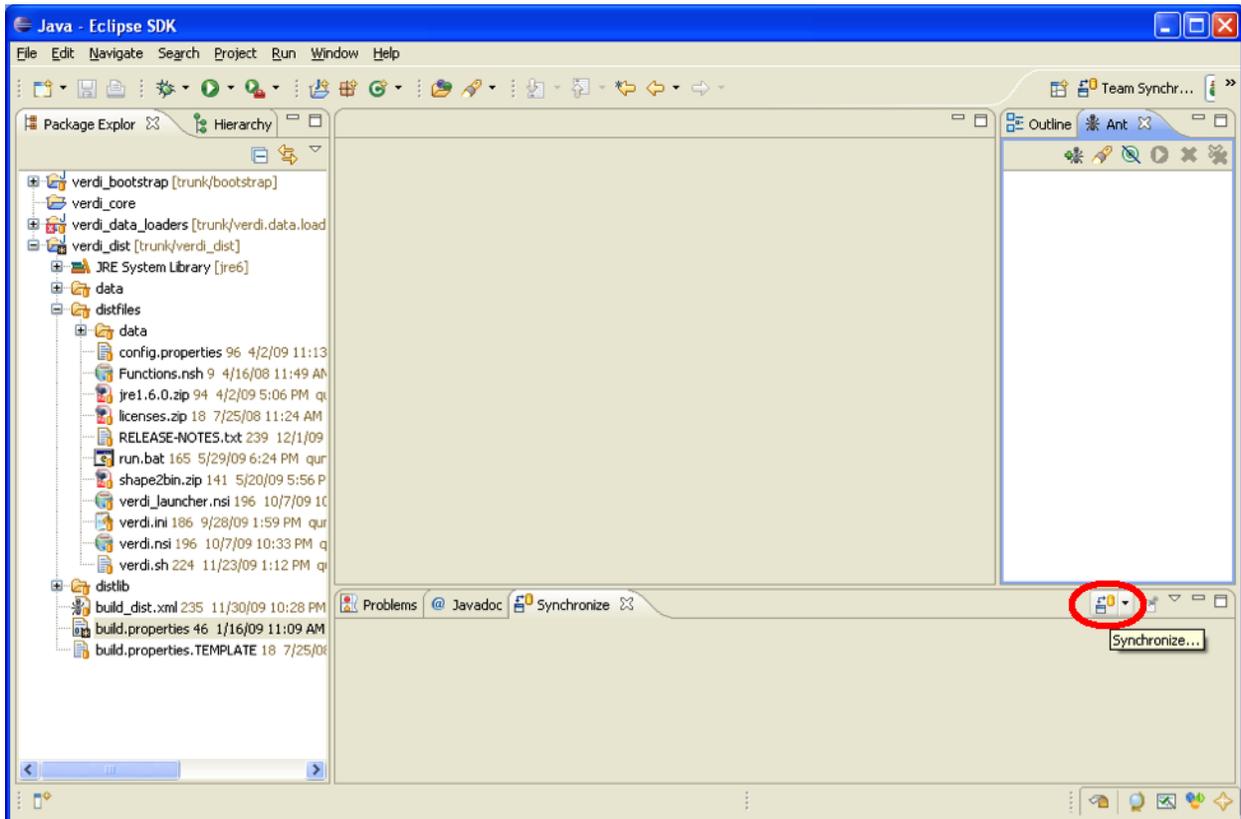


Figure 9-4. Click on Synchronize symbol to bring up pop-up window

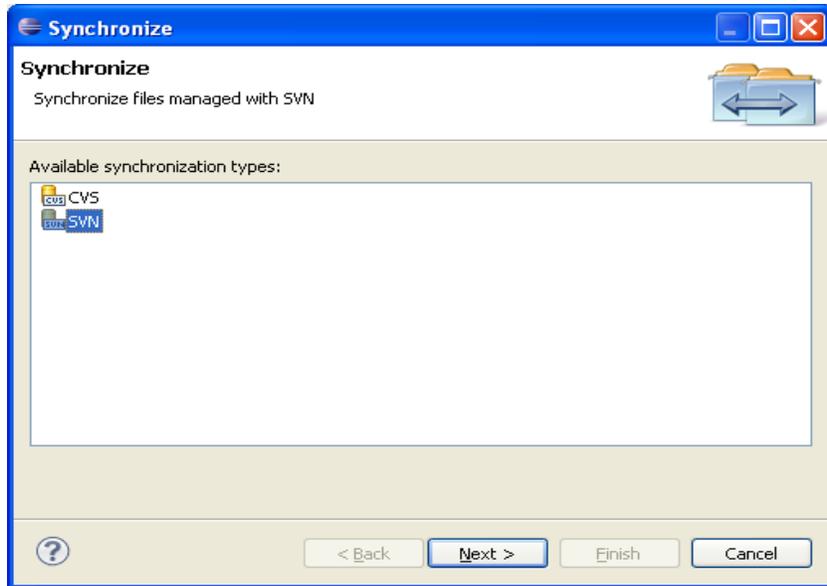


Figure 9-5. Select SVN in Synchronize window

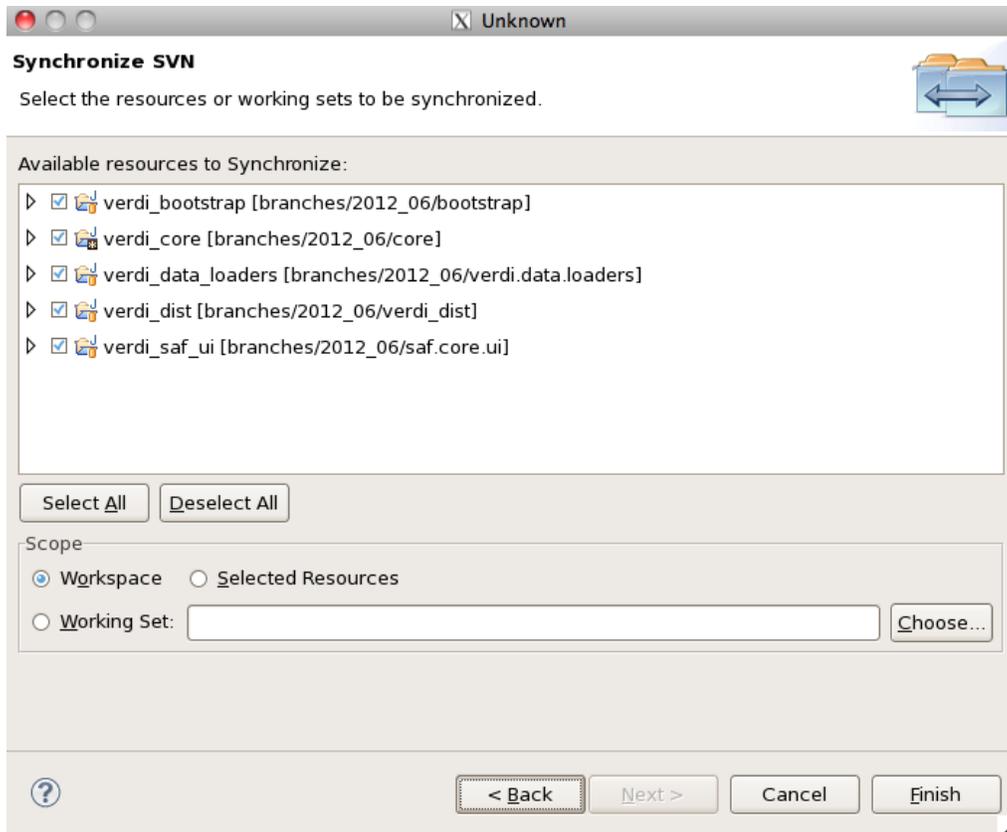


Figure 9-6. Synchronize SVN with all resources selected

9.3 Resolve Updates and Conflicts

After synchronization, Eclipse will show three colored arrows at the bottom of the SVN Workspace window. Each arrow has a number next to it that indicates the number of a type of change for the entire workspace (Figure 9-7).

- Blue incoming arrow: The number of incoming changes
- Green outgoing arrow: The number of outgoing changes
- Red double conflict arrow: The number of conflicting changes

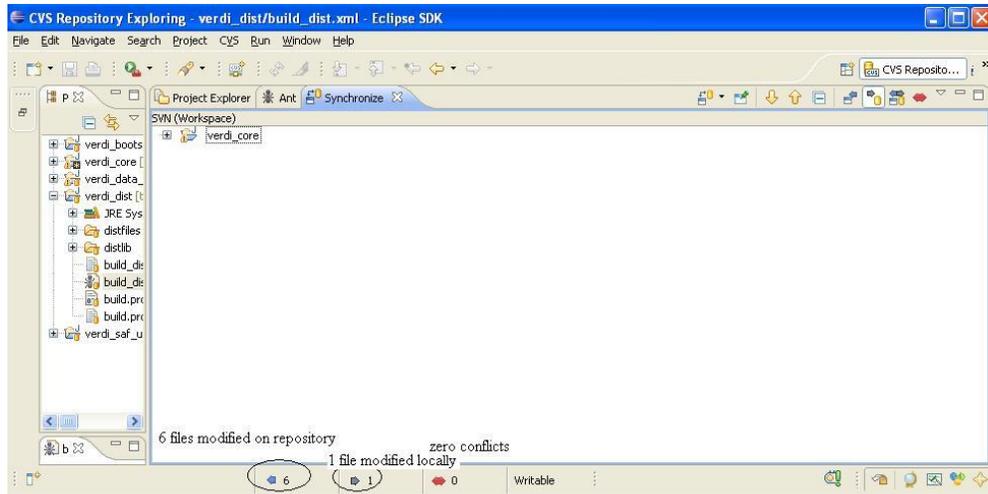


Figure 9-7. Check for updates and conflicts

If you click on the blue arrow, and then right click on verdi_core, a pop-up menu will allow you to select Update. This action downloads the code updates to your local workspace (Figure 9-8).

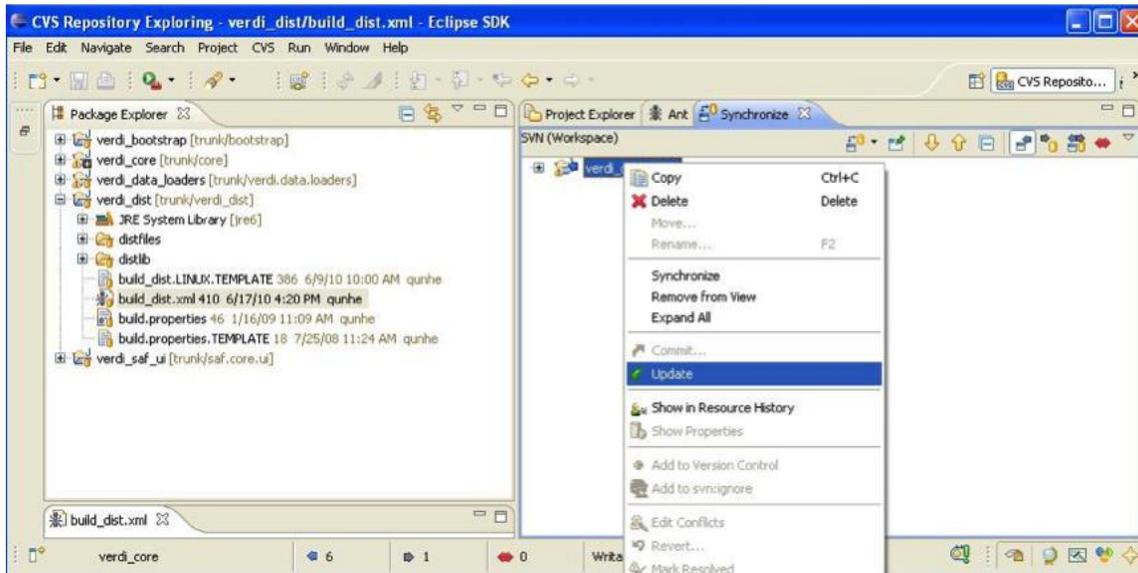


Figure 9-8. Update verdi_core

10 Build the VERDI Distribution

10.1 Prepare to Build VERDI Distribution

Once VERDI has been checked out of the repository, the folders will be displayed in the Project Explorer Window on the Workbench. The next step is to edit the appropriate build.properties file.

10.1.1 Microsoft Windows

If you are building VERDI for the Windows platform, open and edit either the 32-bit build.properties.win32 or the 64-bit build.properties.win64 file that matches your JDK; double-click on the appropriate file to open it in the text editor (Figure 10-1). Edit the build.properties file to specify the JDK used to compile VERDI and the directory where Eclipse will build the VERDI distribution. Save your new file both under its initial name and as the new build.properties file. An example JDK location is:

C:\\Program Files\\Java\\jdk1.7.0_55

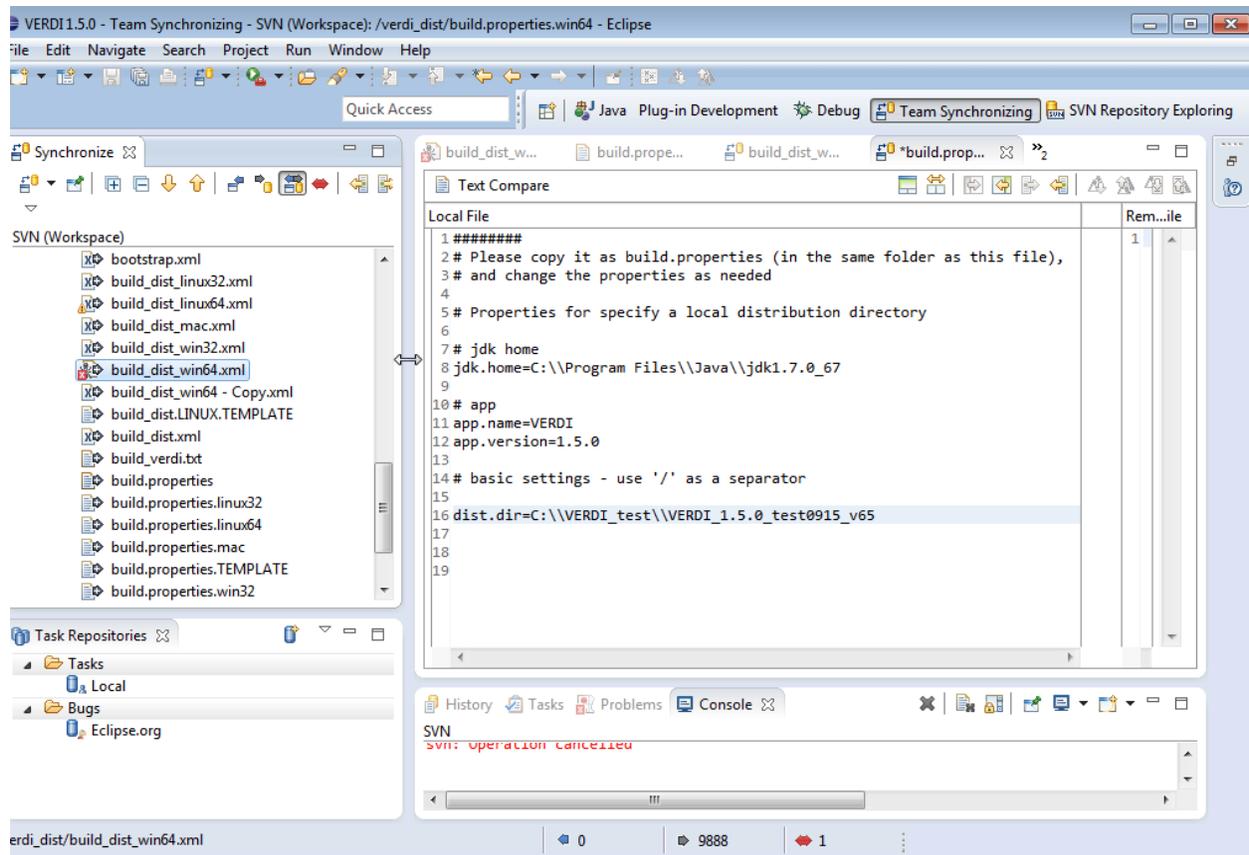


Figure 10-1. Review and edit a build.properties file

10.1.2 Linux

If you are building VERDI for a Linux platform, open and edit either the 32-bit `build.properties.linux32` or the 64-bit `build.properties.linux64` file that matches your JDK. Edit the file to specify the JDK used to compile VERDI and the directory where Eclipse will build the VERDI distribution. Save your new file both under its initial name and as the new `build.properties` file. An example JDK location is:

```
/home/lizadams/jdk1.7.0_55
```

10.1.3 Mac OS X

If you are building VERDI for a Mac, open and edit the `build.properties.mac` file. Specify the JDK used to compile VERDI and the directory where Eclipse will build the VERDI distribution. Save your new file both under its initial name and as the new `build.properties` file. An example JDK location is:

```
/System/Library/Java/JavaVirtualMachines/1.7.0.jdk/Contents/
```

10.1.4 Build_dist xml

There are five `build_dist.xml` files available for Ant building, each for a specific version of JDK: `build_dist_win32.xml`, `build_dist_win64.xml`, `build_dist_linux32.xml`, `build_dist_linux64.xml`, and `build_dist_mac.xml`. These XML files provide the instructions for how to build the respective Windows, Linux, and Mac OS X distributions of VERDI. After editing a file, save it both under its initial name and as the new `build_dist.xml` file.

The `build_dist.xml` file obtains the local directory settings from the `build.properties` file. Although the changes to specify these directories could be made in the `build_dist.xml` file, the `build.properties` file has been created to clearly identify what settings are dependent on the local computer configurations. Also, this structure should reduce errors that might be incurred by a user editing the `build_dist.xml` file.

10.1.5 Build Using Ant

If you have not already set your workspace preferences, you need to set them prior to building VERDI using Ant (see 6.1).

10.1.5.1 Microsoft Windows Distribution

The Windows distribution can be built using the scripts (`build_dist_win32.xml` or `build_dist_win64.xml`) within `verdi_dist` on a Windows 7 computer. Select the Eclipse menu options `Window`→`Show View`→`Ant` to create a subwindow for Ant (Figure 10-2). Drag the corresponding `build_dist_{machineOS}.xml` into the Ant window. Click on the small arrow next to **verdi** to open and display the contents.

1. Double click on `build-version` to update the build version number
2. Double click on `compile-version` to update the compile version number
3. Double click on `build.win.dist` to build the VERDI distribution for a Windows machine (Figure 10-3).

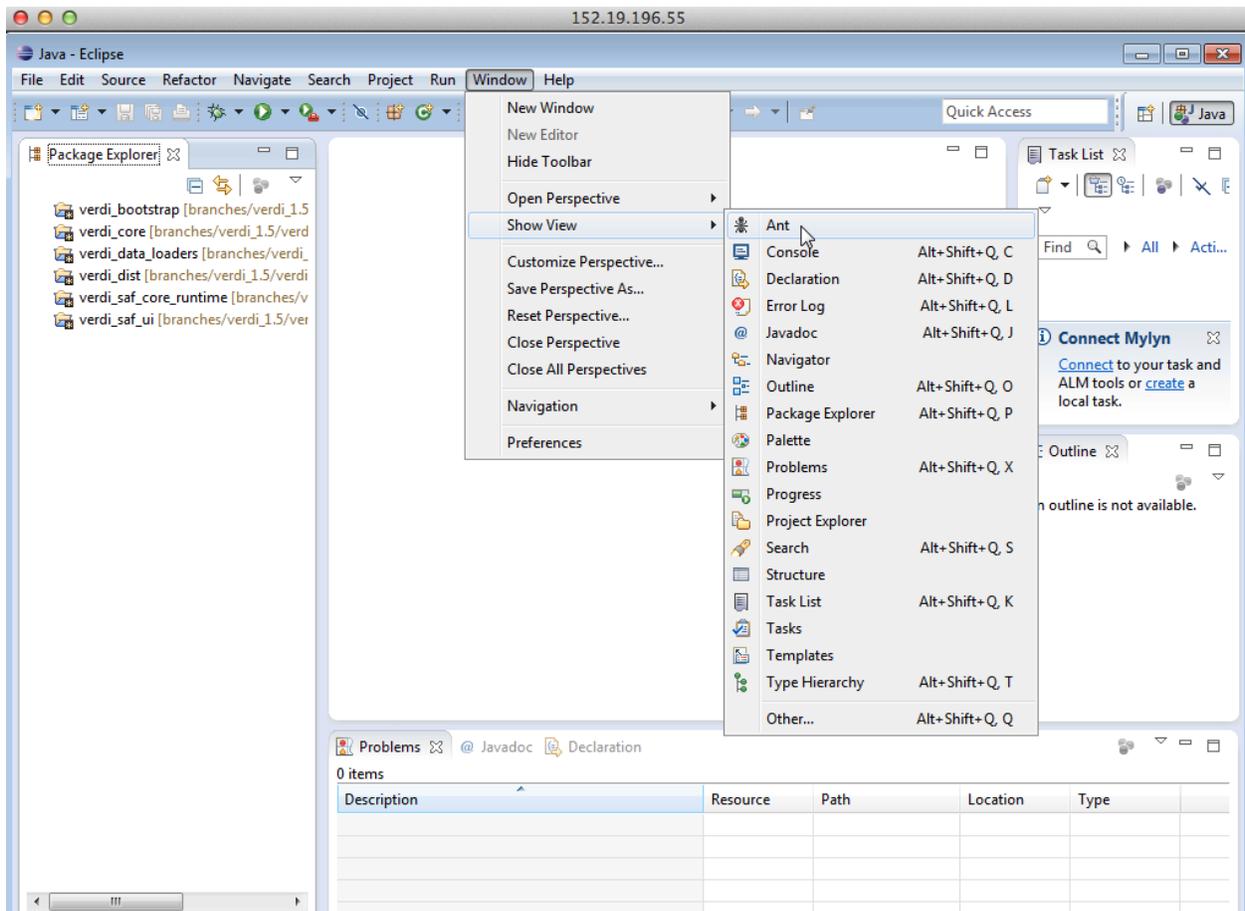


Figure 10-2. Window → Show View → Ant

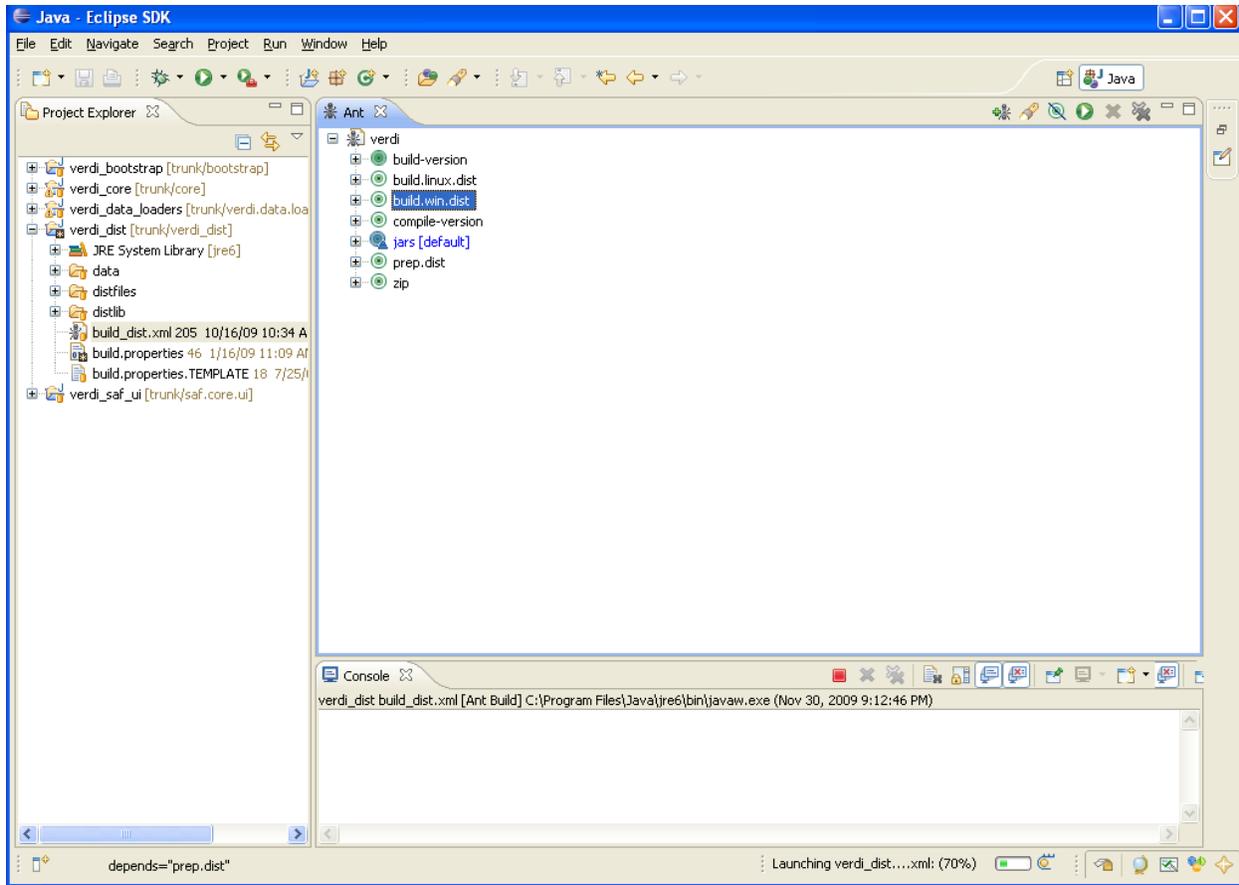


Figure 10-3. Double-click on build.win.dist to build VERDI distribution with Ant

10.1.5.2 Linux Distribution

The Linux distribution can be built by using Ant to run build.linux.dist on a Linux machine. The verdi_dist folder contains the build_dist_linx32.xml or build_dist_linx64.xml script. Select the Eclipse menu options Window→Show View→Ant to create a subwindow for Ant (Figure 10-2). Drag the corresponding build_dist xml into the Ant window. Click on the plus button next to verdi to open and display the contents.

1. Double click on build-version to update the build version number
2. Double click on compile-version to update the compile version number
3. Double click on build.linux.dist to build the VERDI distribution for a Linux machine.

10.1.5.3 Mac Distribution

The Mac distribution can be built by using Ant to run build.mac.dist on a Mac OS X machine. The verdi_dist folder contains the build_dist_mac.xml script. Select the Eclipse menu options Window→Show View→Ant to create a subwindow for Ant (Figure 10-2). Drag the corresponding build_dist xml into the Ant window. Click on the plus button next to verdi to open and display the contents.

1. Double click on build-version to update the build version number
2. Double click on compile-version to update the compile version number
3. Double click on build.mac.dist to build the VERDI distribution for a Mac OS X machine.

10.1.6 Check Console for Error Messages

Error messages will appear in the console window underneath the Ant window. If you obtain the error shown in Figure 10-4, add the Java compiler to your path on the Windows or Linux machine (please review Chapter 5 and Section 6.2.2 to resolve and fix the problem).

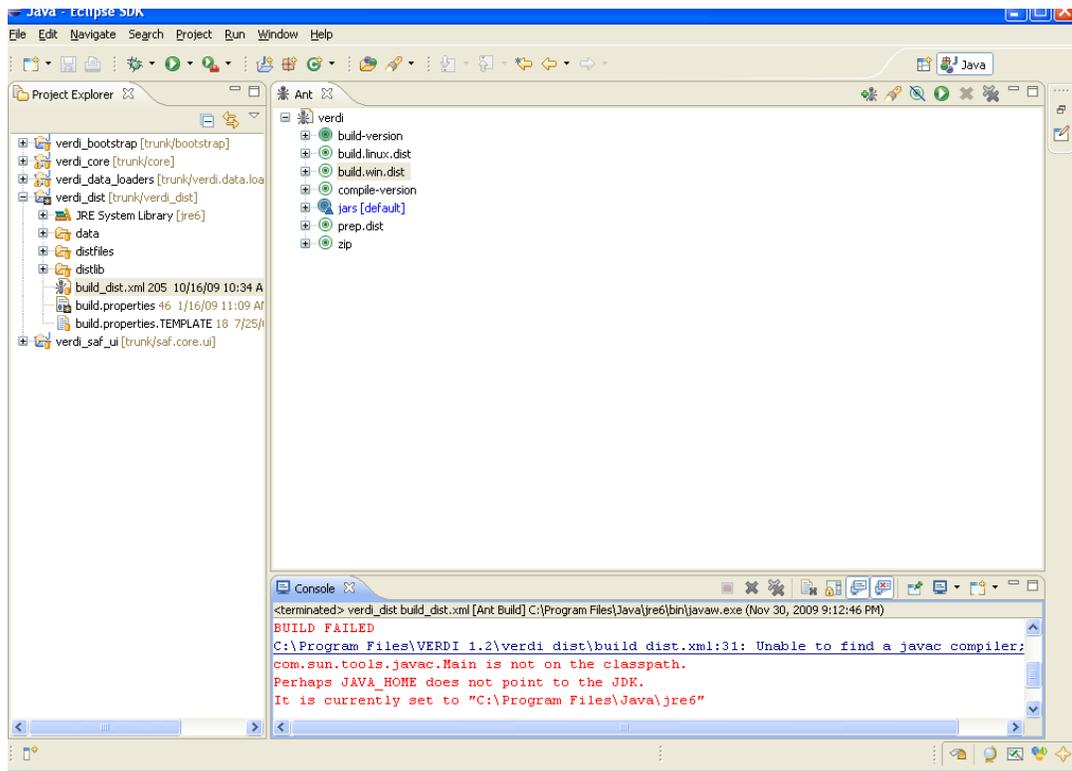


Figure 10-4. Console error message related to not finding Java compiler

10.1.7 Add Java Compiler to Ant

(You will need this section of the manual only if you obtain the error shown in Figure 10-4.)

To allow the Ant compiler to find the Java compiler, you will also need to change the Ant Preferences to add tools.jar as an external jar as follows:

1. In the Eclipse main menu, select Window→Preferences (Figure 10-5)
2. In the Preference Window, select Ant→Runtime (Figure 10-6)
3. Click on the Classpath tab, then select Global Entries
4. Click on Add External JARs

5. Locate the tools.jar under the lib folder on the JDK local installation directory, then click OK, and click OK again (Figure 10-7)

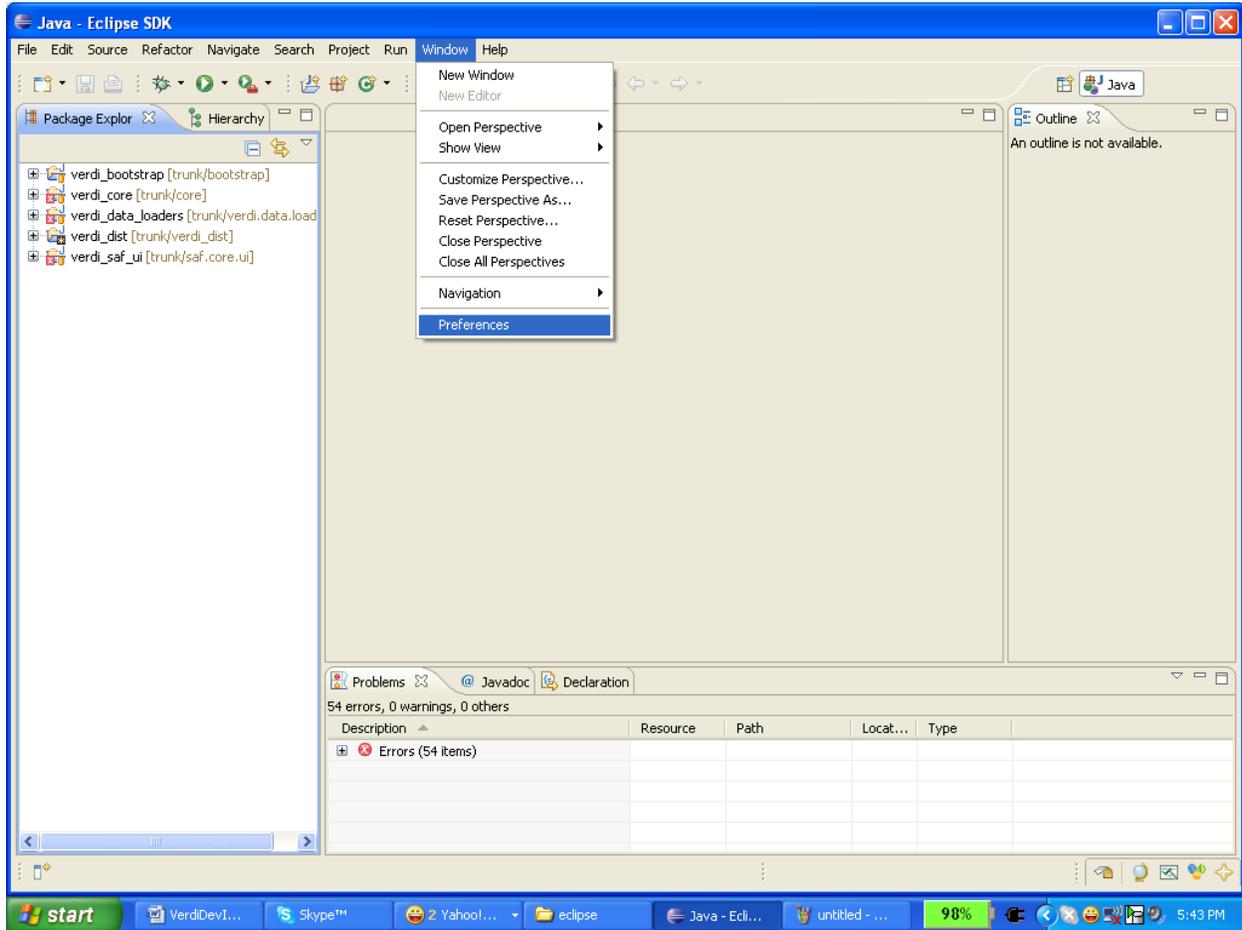


Figure 10-5. Open Window → Preferences

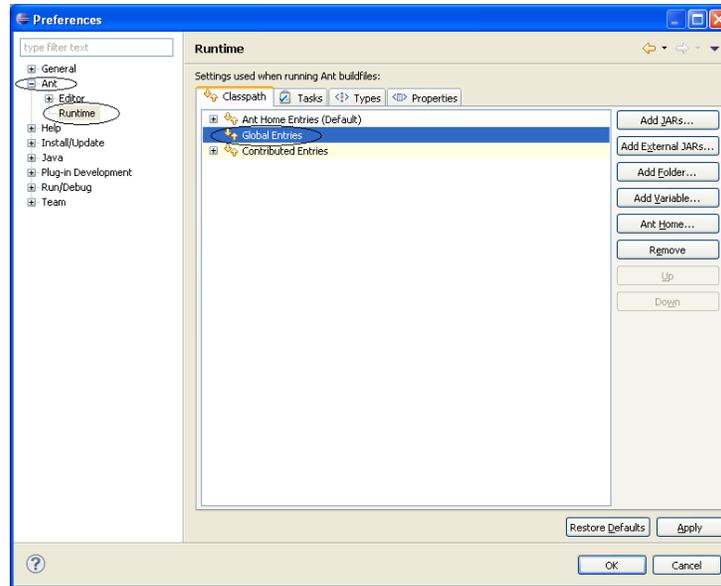


Figure 10-6. Expand Ant, select runtime, select global entries

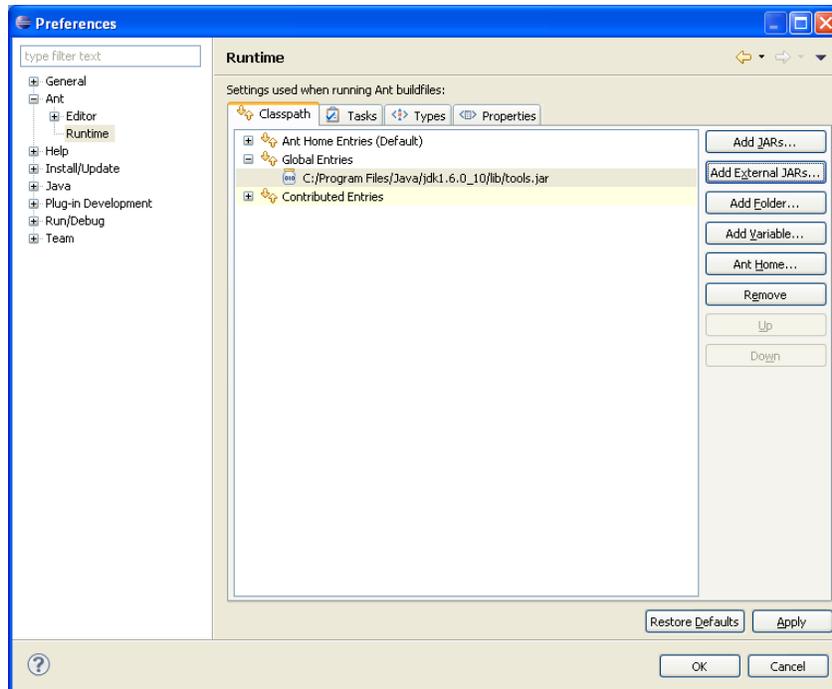


Figure 10-7. Add tools.jar to Ant preferences

10.2 Tag Released Versions

10.2.1 SVN Tag

When a version has been released, make a copy of the projects at the revision number prior to the release and save this to a tagged directory with the version number. Files under the tagged directory should not be changed (ie. nothing should be checked into the tagged directories once a copy of the code prior to the release has been added). The following example will show how a tagged directory containing VERDI 1.4.1 was created from the branches/2012_06 directory.

First see what tags have already been created.

```
svn list https://svn.code.sf.net/p/verdi/code/tags
```

You should see:

```
04282011/  
05282010/  
12042009/  
v1.3/  
v1.4.1/
```

Use the following command to determine what the current revision number is for the code in the branches/verdi_1.5 directory.

```
svn info https://svn.code.sf.net/p/verdi/code/branches/verdi_1.5
```

View the comments that were made to verify that the revision you would like to tag was created at the time of the release date. Note, VERDI_1.5 was released in October 2014.

The following instructions were followed to tag the VERD 1.4.1 release

The message log confirms that this revision contained the release notes for version 1.4.1 and it was committed on May 2, 2013. (Note, if you run this same command on revision 714, you would see a message log dated June, so this revision was made after the release.)

```
svn log -r713 https://svn.code.sf.net/p/verdi/code/branches/2012_06
```

```
-----  
r713 | lizadams | 2013-05-02 14:48:00 -0400 (Thu, 02 May 2013) | 1 line  
Release notes for version 1.4.1  
-----
```

Save the code at the last revision just prior to release. First set the editor using the command:

```
export SVN_EDITOR=vim
```

Then use `svn copy` to copy the branches/2012_06 at revision 713 to the tags/v1.4.1 directory:

```
cep00190:svn-verdi lizadams$ svn copy -r713  
https://svn.code.sf.net/p/verdi/code/branches/2012_06  
https://svn.code.sf.net/p/verdi/code/tags/v1.4.1
```

Use the `svn list` command to view the files under the `tags/v1.4.1` directory:

```
svn list https://svn.code.sf.net/p/verdi/code/tags/v1.4.1  
bootstrap/  
core/  
saf.core.ui/  
verdi.data.loaders/  
verdi_dist/
```

Use the `svn log` command on revision `?` to see the comment made at the time that the tagged version 1.4.1 was created:

```
svn log -r721 https://svn.code.sf.net/p/verdi/code/tags/v1.4.1  
-----  
r721 | lizadams | 2014-05-12 18:04:56 -0400 (Mon, 12 May 2014) | 3 lines  
Tag VERDI_1.4.1 release version at revision 713  
-----
```

11 Logging Messages in VERDI

VERDI 1.5 uses the Apache Log4J version 2 (Log4J2) logging services. For complete information see <http://logging.apache.org/log4j/2.x/>. All of the messages produced by VERDI itself are now sent through the Log4J2. Some of the libraries used by VERDI also use Log4, allowing messages generated within these libraries to print in with the VERDI-generated messages in a single log file for debugging purposes.

11.1 Implementing Log4J2

Log4J2 also uses a hierarchical messaging system, allowing the user to specify the level of detail desired in the log file. The complete manual is on-line at <http://logging.apache.org/log4j/2.x/manual/index.html>. VERDI uses the XML version of configuration (<http://logging.apache.org/log4j/2.x/manual/configuration.html>). A configuration file `log4j2.xml` is included in VERDI's directory `bootstrap/lib`. You can customize and enhance this configuration file to fit your needs.

Each Java class file needs to include 2 import statements:

```
import org.apache.logging.log4j.LogManager;  
import org.apache.logging.log4j.Logger;
```

The class then has to include 1 statement at the top of the class definition. For the class named `BatchTaskFactory` the statement is:

```
static final Logger Logger = LogManager.getLogger(BatchTaskFactory.class.getName());
```

Then, instead of using `System.out` to send messages, use `Logger.messages`. For example, at the top of the member function `showGISLayersDialog()` we have a message to tell us where we are in the execution:

```
Logger.debug("in FastTilePlot.showGISLayersDialog()");
```

This tells Java to send the message `in FastTilePlot.showGISLayersDialog()` to the logger at the DEBUG level.

Another type of message prints the value of a variable. In this example we have just calculated the value of an integer variable and we want to see it in a DEBUG logger:

```
Logger.debug("statisticsSelection = " + statisticsSelection);
```

For permanent messages, use a higher message level. Typically messages that are always important, such as those caught in a catch block of a try/catch, are written as an ERROR. If VERDI is configured to print out ERROR, FATAL, or ALL messages, then the ERROR messages will be printed but the DEBUG messages will not.

The hierarchical levels in VERDI are:

TRACE, DEBUG, INFO, WARN, ERROR, FATAL, ALL, OFF

OFF messages are never printed, and the OFF level in the XML file will print no messages. An ALL message is printed for all levels except OFF, a FATAL message is printed for levels ALL and FATAL, etc.

The manual also provides information on implementing different logging levels in different parts of your program. That way, you can have more messages printed (e.g., DEBUG level) in the code on which you are currently focused.

11.2 Log files

The log4j2.xml file distributed with VERDI appends logs into an existing log file. If you have messages printed at the ERROR level, this file will stay relatively small. However, if you use the DEBUG or ALL levels you will need to maintain the log file. Depending upon your needs, you can delete the file periodically, rename the file when you want to keep a set of messages, or use some file maintenance utility (e.g., tail) to reduce the size of the log file.

Also, the XML file included in the distribution is set up to write to the file named verdi.log in the verdi subdirectory of the user's home directory. You can change the directory or the name of the file to suit your needs.

12 List of Libraries and Source Code Files

Compiled Library	Source Code
anlBasic.jar	
ant-contrib-1.0b2-bin.zip	
AppleJavaExtensions.jar	
bufr-3.0.jar	bufr-3.0-sources.jar
clibwrapper_jjio-1.1.jar	
commons-codec-1.9.jar	commons-codec-1.9-sources.jar
commons-collections-3.2.1.jar	commons-collections-3.2.1-sources.jar
commons-httpclient-3.1.jar	commons-httpclient-3.1-sources.jar
commons-io-2.4.jar	commons-io-2.4-sources.jar
commons-logging-1.1.3.jar	commons-logging-1.1.3-sources.jar
dockingFramesCommon.jar	
dockingFramesCore.jar	
epsgraphics-1.2.jar	epsgraphics-1.2-sources.jar
forms-1.0.5.jar	
gt-coverage-api-10.5.jar	gt-coverage-api-10.5-sources.jar
grib-8.0.29.jar	grib-8.0.29-sources.jar
GRIP.jar	
gt-api-10.5.jar	gt-api-10.5-sources.jar
gt-brewer-10.5.jar	gt-brewer-10.5-sources.jar
gt-coverage-10.5.jar	gt-coverage-10.5-sources.jar
gt-coveragetools-10.5.jar	gt-coveragetools-10.5-sources.jar
gt-data-10.5.jar	gt-data-10.5-sources.jar
gt-epsg-extension-10.5.jar	gt-epsg-extension-10.5-sources.jar
gt-epsg-hsql-10.5.jar	gt-epsg-hsql-10.5-sources.jar
gt-epsg-wkt-10.5.jar	gt-epsg-wkt-10.5-sources.jar
gt-feature-aggregate-10.5.jar	
gt-feature-pregeneralized-10.5.jar	gt-feature-pregeneralized-10.5-sources.jar
gt-geometry-10.5.jar	gt-geometry-10.5-sources.jar
gt-grid-10.5.jar	gt-grid-10.5-sources.jar
gt-image-10.5.jar	gt-image-10.5-sources.jar
gt-jts-wrapper-10.5.jar	gt-jts-wrapper-10.5-sources.jar
gt-main-10.5.jar	gt-main-10.5-sources.jar
gt-metadata-10.5.jar	gt-metadata-10.5-sources.jar
gt-opengis-10.5.jar	gt-opengis-10.5-sources.jar
gt-referencing-10.5.jar	gt-referencing-10.5-sources.jar
gt-render-10.5.jar	gt-render-10.5-sources.jar
gt-shapefile-10.5.jar	gt-shapefile-10.5-sources.jar

Compiled Library	Source Code
gt-shapefile-old-10.5.jar	gt-shapefile-old-10.5-sources.jar
gt-shapefile-renderer-9.5.jar	gt-shapefile-renderer-9.5-sources.jar
hsqldb.jar	hsqldb-2.3.2.zip
icu4j-50_1_1.jar	icu4j-50_1_1-src.jar
j3dcore.jar	
j3dcore-ogl.dll	
j3dutils.jar	
jai_codec.jar	
jai_imageio.jar	
jai_imageio_windows-i586.jar	
jai_core.jar	
jaramiko-151.jar	jaramiko_java.zip
javadbf-0.4.0.jar	javadbf-0.1.0-sources.jar
jcommon-1.0.16.jar	jcommon-1.0.16-sources.jar
jdom-2.0.5.jar	jdom-2.0.5-sources.jar
jeo.jar	
jfreechart-1.0.17.jar	jfreechart-1.0.17-sources.jar
jh-2.0_02.jar	
jide-oss-3.5.14.jar	jide-oss-3.5.14-sources.jar
jmf.jar	
jpf.jar	jpf-1.5-sources.jar
jpf-boot.jar	jpf-boot-1.5.sources
jpf-tools.jar	
jscience.jar	jscience-4.3.1-src.zip
jsr-275-1.0-beta-2.jar	
jt-all-1.3.1.jar	
jts-1.13.jar	
junit-4.11.jar	junit-4.11-sources.jar
l2fprod-common-all.jar	
log4j-1.2-api-2.0-rc1.jar	log4j-1.2-api-2.0-rc1-sources.jar
log4j-api-2.0-rc1.jar	log4j-api-2.0-sources.jar
log4j-core-2.0-rc1.jar	log4j-core-2.0-rc1-sources.jar
log4j-jcl-2.0-rc1.jar	log4j-jcl-2.0-rc1-sources.jar
log4j-taglib-2.0-rc1.jar	log4j-taglib-2.0-rc1-sources.jar
milStd2525_png.jar	
netcdfAll-4.3.all	
omcorba.jar	
omj3d.jar	
omsvg.jar	
org.apache.commons.lang_2.6.0.v201205030909.jar	
org.eclipse.osgi-3.9.1.v20130814-1242.jar	
org.eclipse.uomo.core_0.7.0.20140420011.jar	
org.eclipse.uomo.ucum_0.7.0.20140420011.jar	

Compiled Library	Source Code
org.eclipse.uomo.ui_0.7.0.20140420011.jar	
org.eclipse.uomo.units_0.7.0.20140420011.jar	
org.eclipse.uomo.util_0.7.0.20140420011.jar	
org.eclipse.uomo.xml_0.7.0.20140420011.jar	
osx.jar	
piccolo.jar	
piccolo2d-core-3.0.jar	piccolo2d-core-3.0-sources.jar
piccolo2d-extras-3.0.jar	piccolo2d-extras-3.0-sources.jar
piccolo2d-swt-3.0.jar	piccolo2d-swt-3.0-sources.jar
piccolox.jar	
prefsAll.jar	
reference.jar	
repast_simphony_gis_2_1_0.jar	
sgt_v30.jar	sgt_src_v30.jar
swingx-all-1.6.5.jar	swingx-all-1.6.5-sources.jar
TableLayout.jar	
unit-api-0.6.1.jar	unitsofmeasurement.zip
vecmath.jar	
velocity-1.7.jar	velocity-1.7.zip
visad.jar	visad_src.jar
wizard-0.1.12.jar	
xpp3_min-1.1.4c.jar	xpp3_min-1.1.4c-sources.jar
xstream-1.4.7.jar	xstream-distribution-1.4.7-src.zip