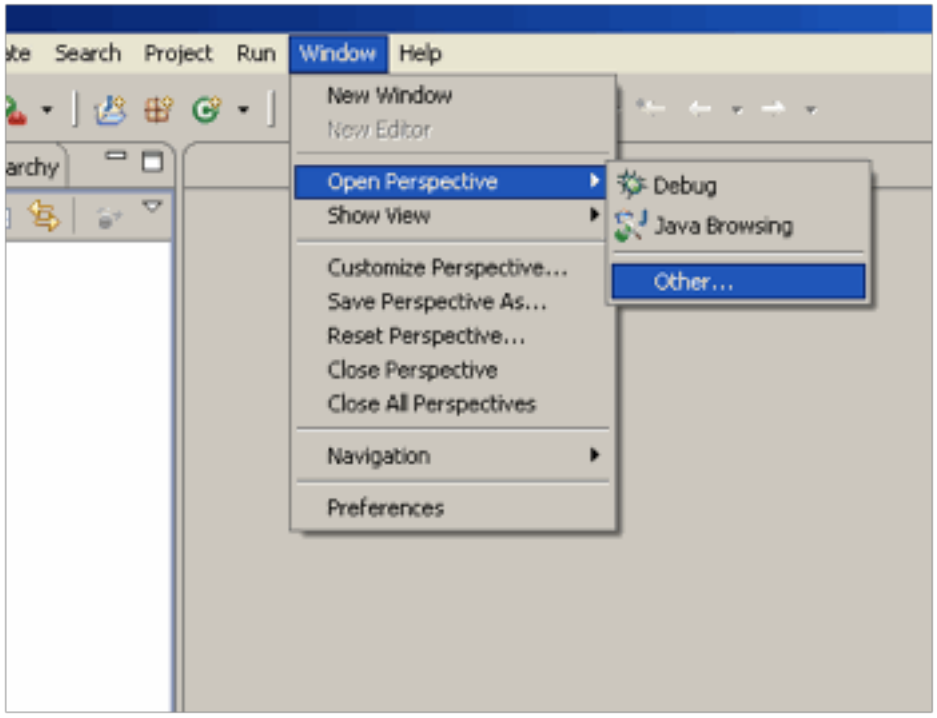
Tutorial: Using the Subversive Plug-in
By Jonathon Lundy and Ron Cytron

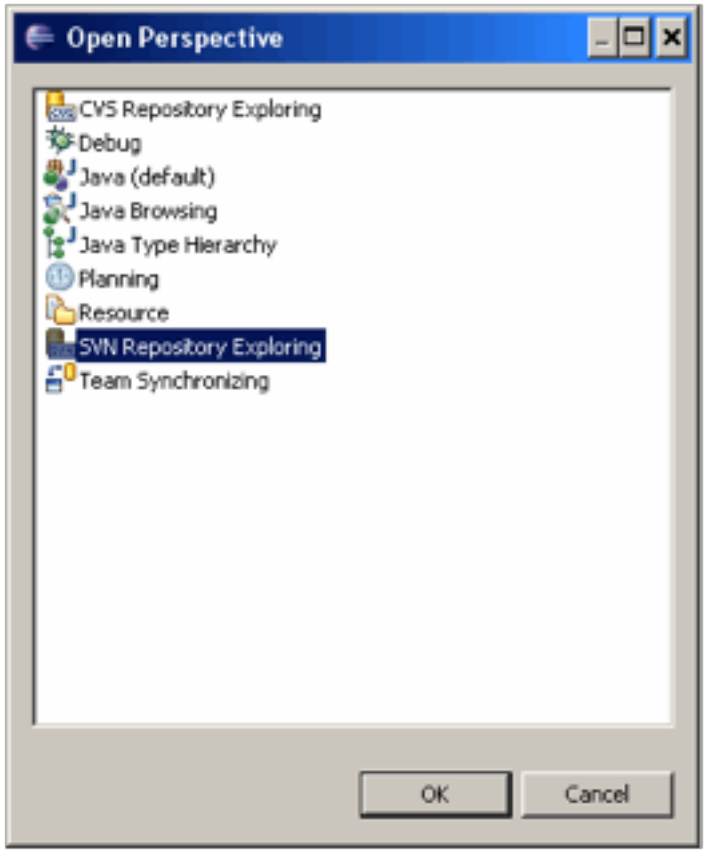
Subversive is a plug-in for the Eclipse IDE which allows Eclipse to use Subversion (SVN), which is a version control system. Use of this system will make it easy for you to easily move between computers, work in groups, and electronically submit your work. However, Eclipse does not support SVN out of the box; a plug-in is required to make it work. This plug-in is already installed on CEC computers. Instructions on how to install the plug-in (if youâ€™re using Eclipse on your own computer) are covered in another tutorial. Here, we address only the use of Subversive. You may wish to read more about Subversion (see link above) to have a better understanding of how it works before reading on.

Opening the SVN Repository Perspective

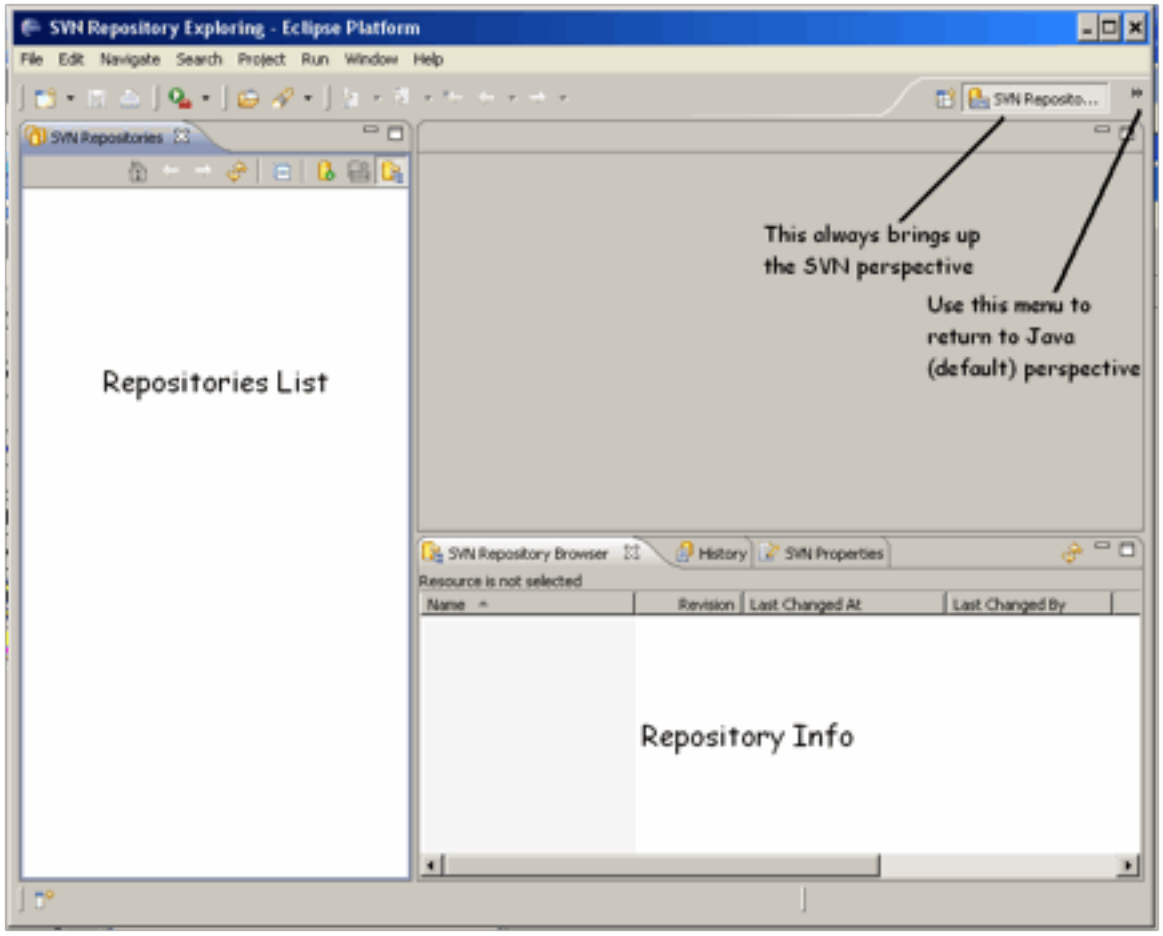
The first time you use Subversive, youâ€™ll need to open the SVN Perspective using the *Window* drop-down menu. Go to *Window >> Open Perspective >> Otherâ€¦*.



Select *SVN Repository Exploring* from the dialog box that appears, and click OK.

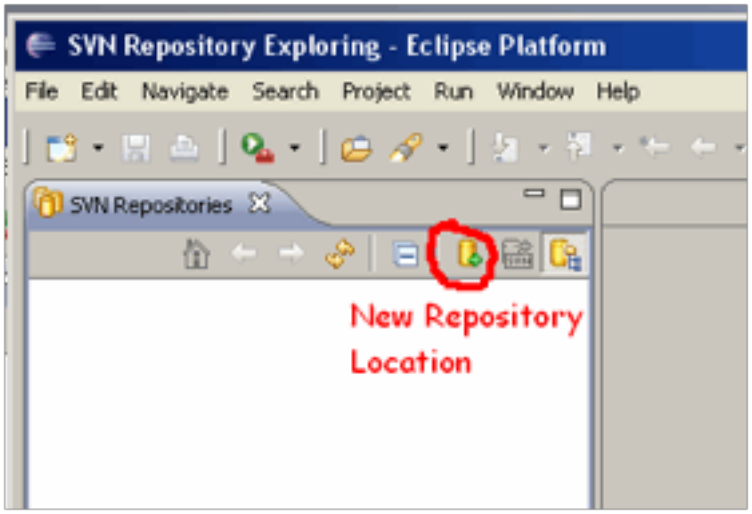


This will open the SVN Repository Exploring perspective, which should look (hopefully) something like the screenshot below. The left pane is where a list of Subversion repositories would appear, if any had been added yet. Likewise, the bottom pane would give us details about the files in the repository, if any had been added. Note that in the upper-right corner of the perspective, an icon with a yellow badge and the label SVN, followed by the words SVN Repository, is depressed. In the future, you can use this button to return to this view once you leave it (i.e., you donâ€™t have to go to the *Window* menu again). To return to the default perspective (the Java perspective), click on the double-arrows next to the SVN Repository icon to pull up a drop-down menu of perspectives, and select the Java one. (Note that it may also appear on the same bar as the SVN Repository icon, and not in the drop-down menu).

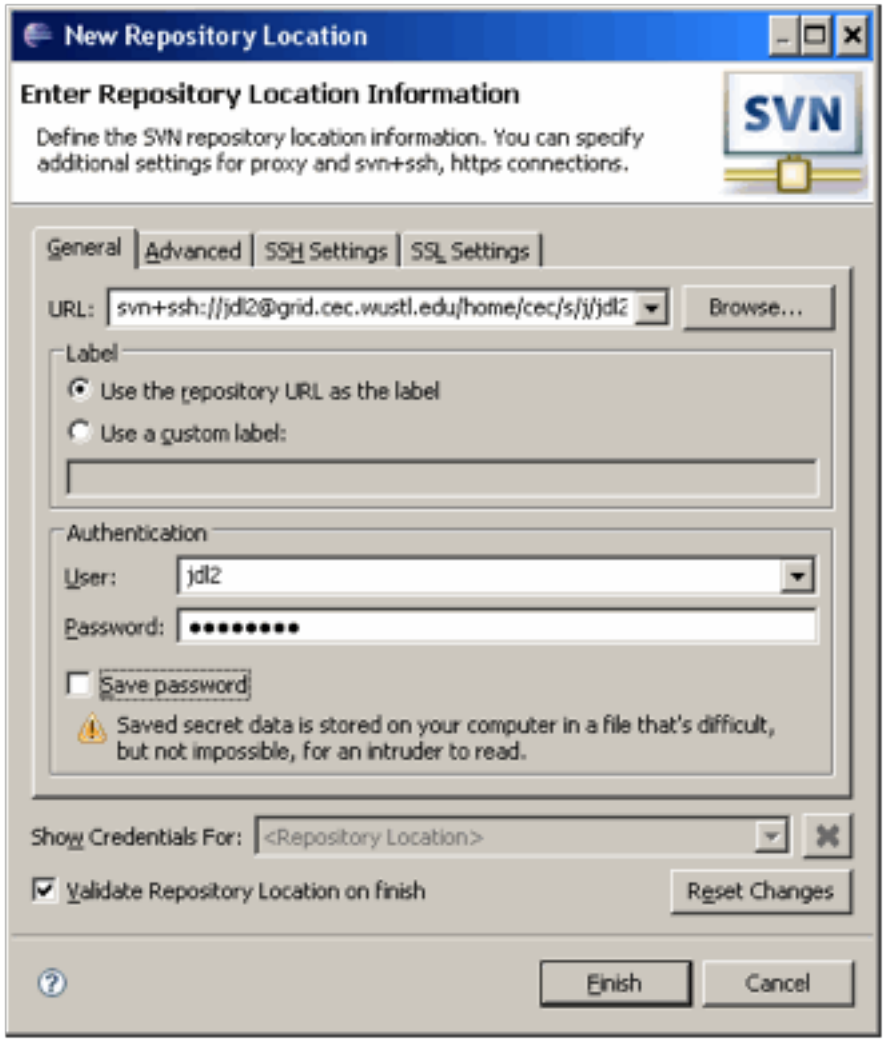


Importing a Project from the Repository into Eclipse

Now that you are in the SVN Repository view, letâ€™s go ahead and add a repository to the list. To do so, press the *New Repository Location* icon at the top of the SVN Repositories pane.



This brings up the New Repository Location dialog box.

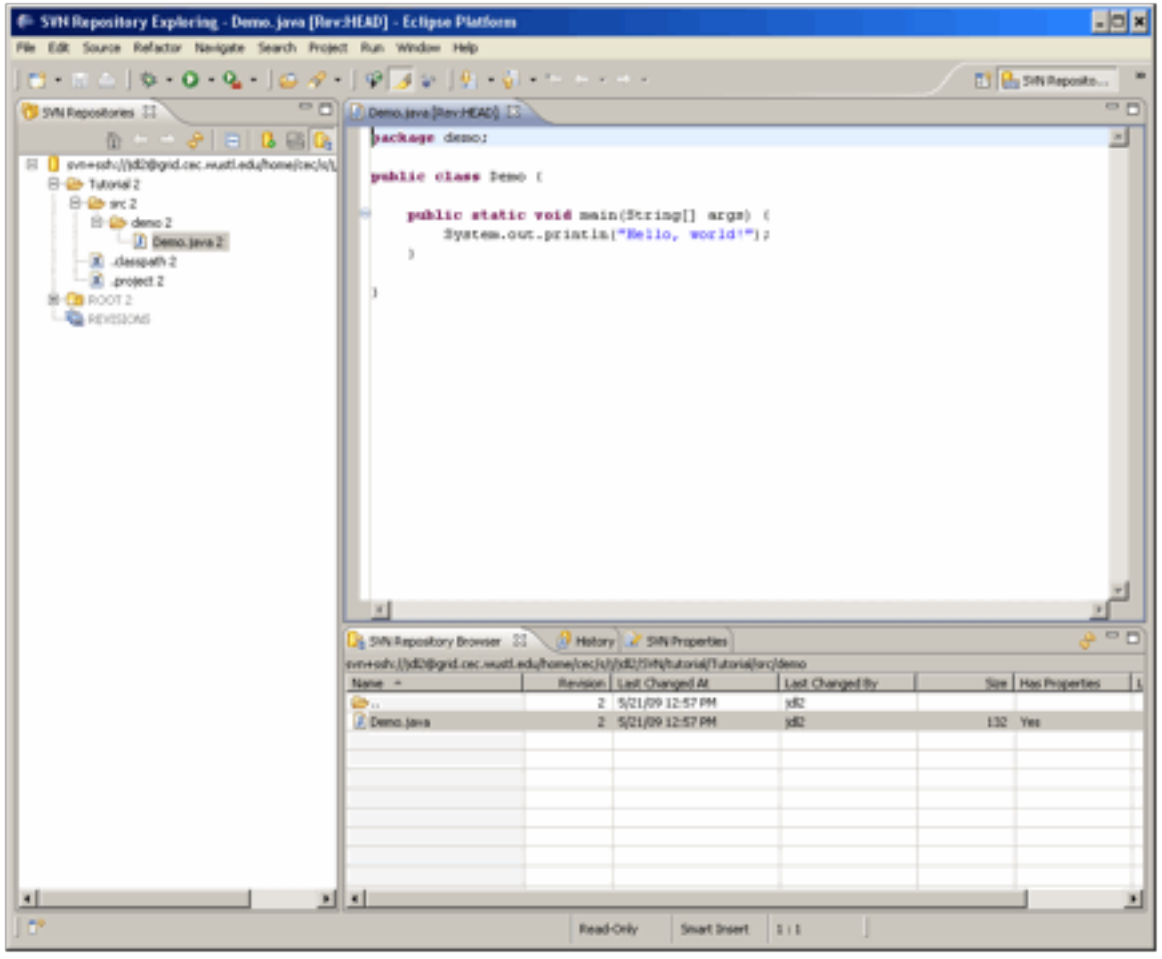


In the *URL* field, enter the **full and complete** location of the SVN repository. Letâ€™s look at the address entered above as an example:

- The *svn+ssh://* indicates the protocol for accessing the SVN repository. Here, we are using Subversion over Secure Shell, which is what weâ€™ll always use for this course.
- *XXXX@grid.cec.wustl.edu* indicates the server. grid is the name of the CECâ€™s SSH server, and *XXXX* is where youâ€™ll type in your CEC username (*e.g.,* jdl2).
- Everything following that is the location of the root of the repository on the server. Itâ€™s easiest to copy and paste the location from a lab assignment or other document.

Under the *Authentication* section, fill in your CEC user name and password. You do not need to click â€œSave passwordâ€. Make sure the *Validate Repository Location on finish* box is checked, and then press *Finish.*

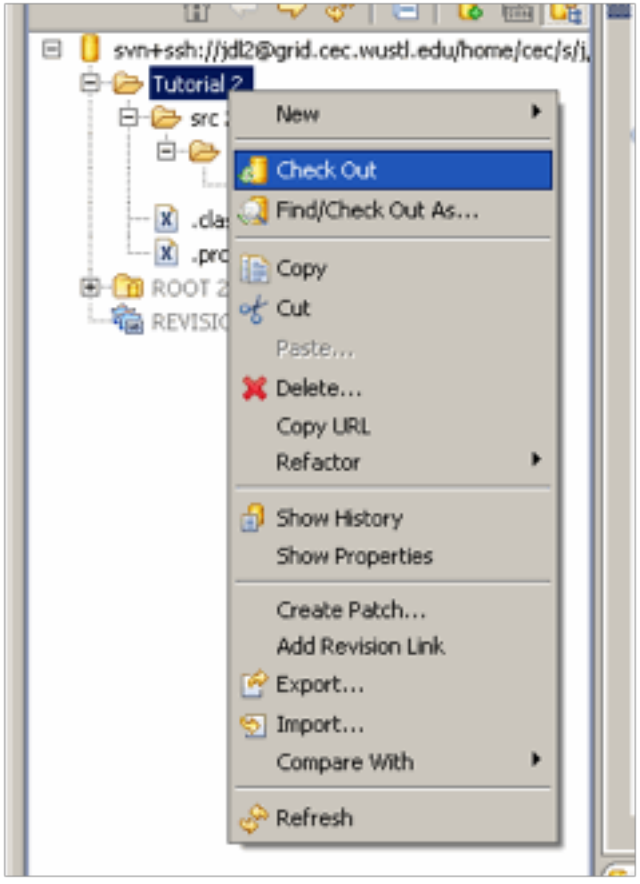
The repository location should then appear in the Repositories pane, if it was added successfully. You can expand the repository by pressing the plus icon to the left of its entry in the list, and then you can select files or folders to look at their properties in the bottom pane or to open them.



Note that if you open Java files in this view, they will be in read-only mode. This is because it is forbidden by Subversion to modify files in the repository directly. They first must be **checked out**, which is a process that

creates a local **working copy** of the files on your hard drive, which you may then modify. However, for your changes to appear on the repository, you must later re-upload the files to the repository, a process called **committing.**

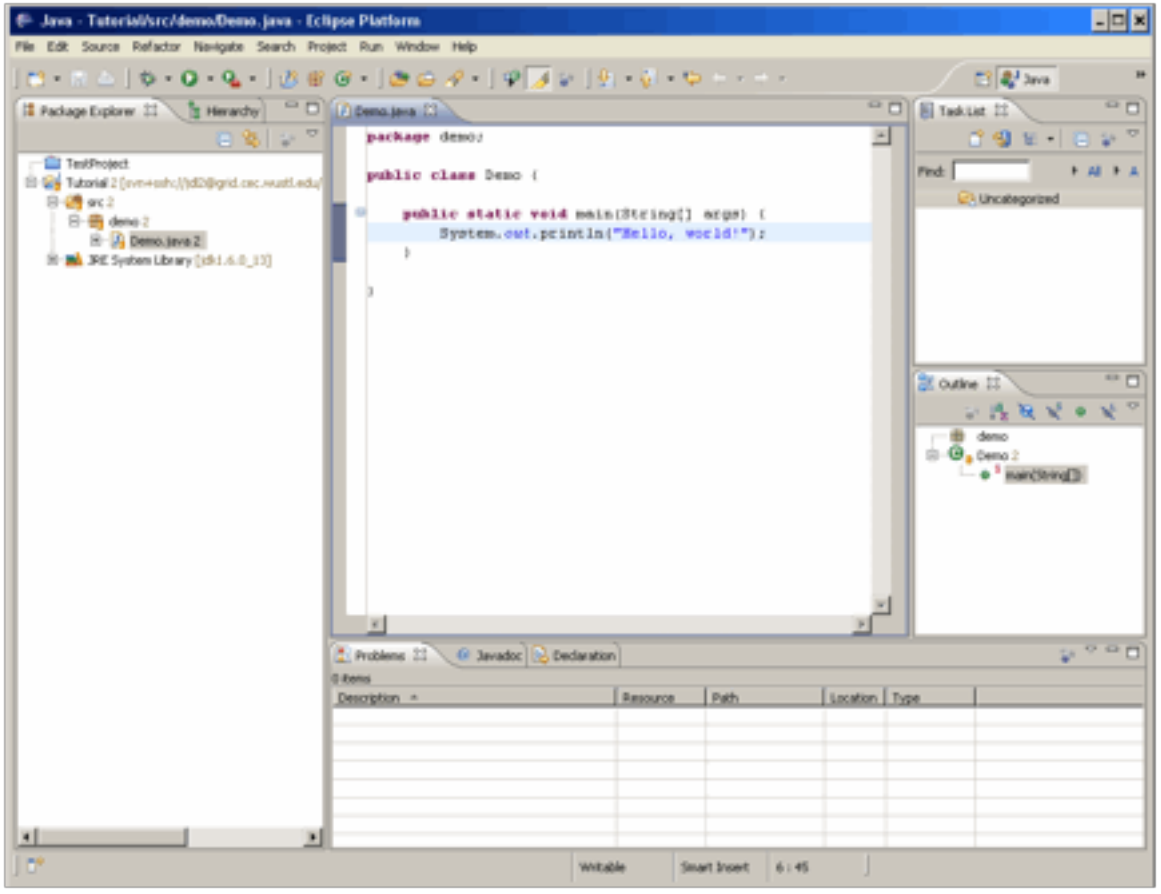
To check out a project from the repository, right click on the folder within the repository containing the Java project (in the screenshot above, the folder is called *Tutorial*â€¦ itâ€™s the folder which contains the *.classpath* and *.project* files, as well as the *src* folder) and then select the *Check Out* option. This will create a local copy of the project inside your Eclipse workspace.



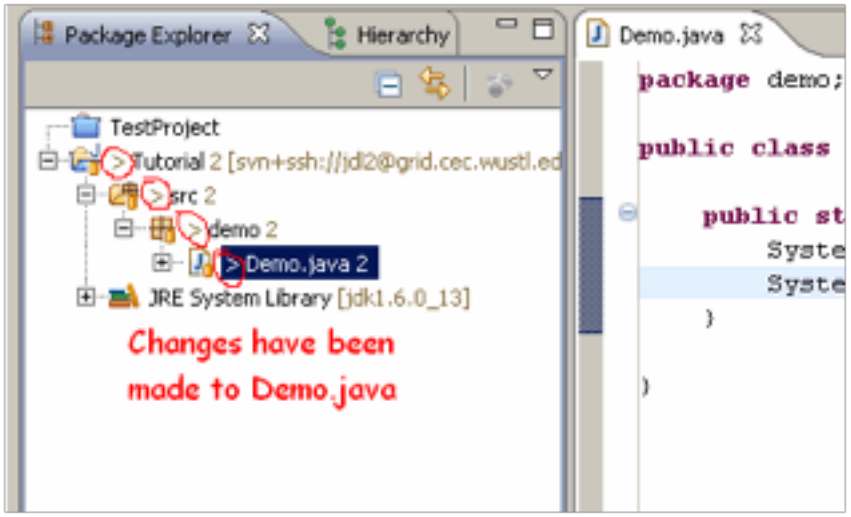
Once the Check Out process is complete, return to the Java perspective. You should see the project that youâ€™ve checked out from the repository in the Package Explorer pane. Note the yellow badge in the

lower-right hand corner of the projectâ€™s icon; this indicates that it is associated with a SVN repository. You may now open, edit, add, and/or delete files from this project as if it were entirely your own, not linked to a repository at all.

Note: the numbers next to the file and folder names indicate the most recent revision in which that file was modified.



Important Note: notice that when you make changes to a file and save them, a pointed bracket appears next to the file name and that of the folders it is contained in. This means that your copy of the file has been modified from what is contained in the repository, and that you should **commit** your changes at some point.

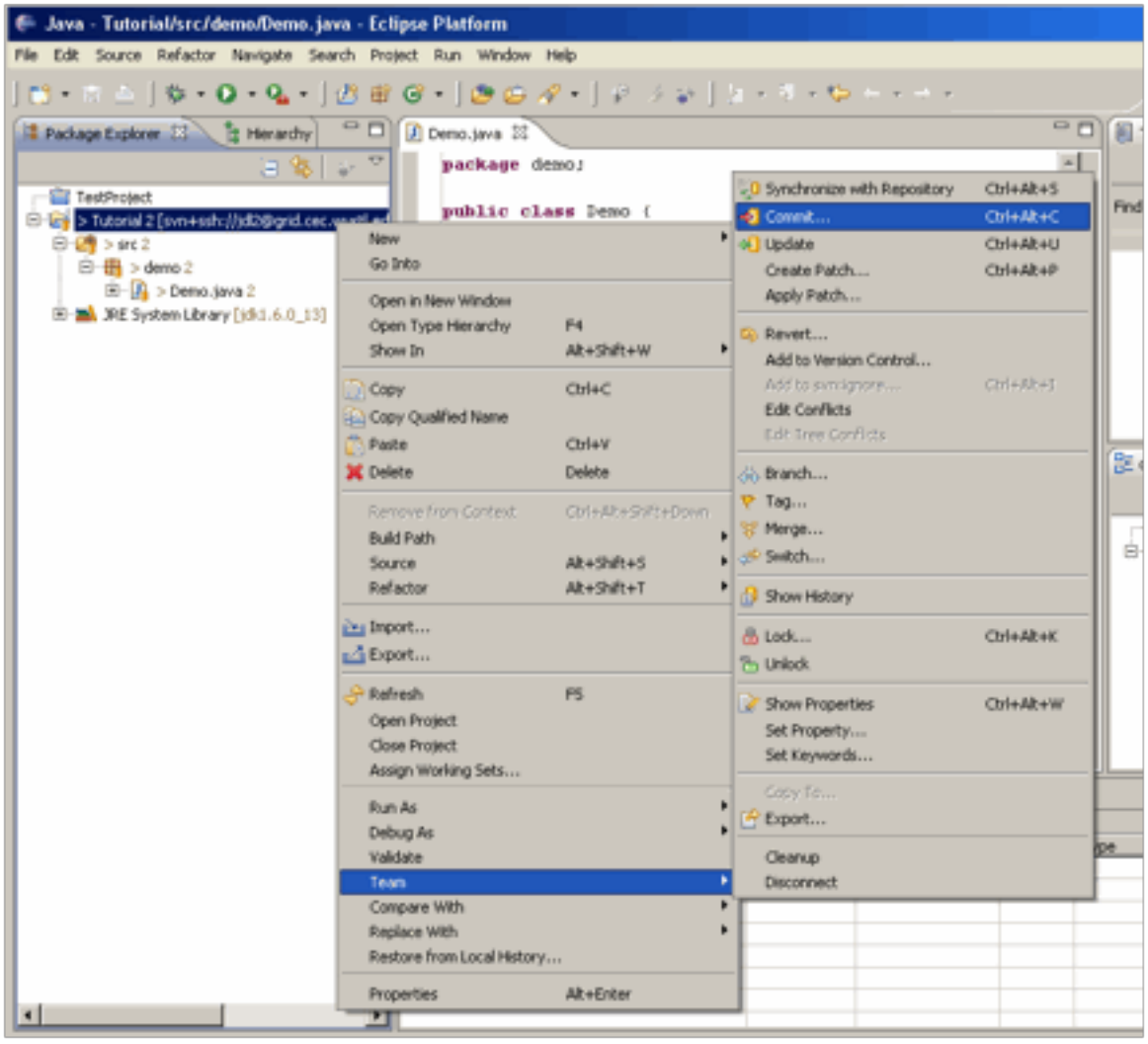


Committing Changes to the Repository

When youâ€™re done working on your Java project for a while, you should commit the changes to the repository to ensure that it contains the most recent version of your code.

Important Note: you should commit whenever youâ€™re done with a work session or after making large changes to ensure that the repository is always up to date. Remember that when we grade, we look at the most recent version of the code contained within the repository, *not* your local copy! Itâ€™s also a good method for backing up your work, as older copies of files in the repository can be retrieved if you should ever need them.

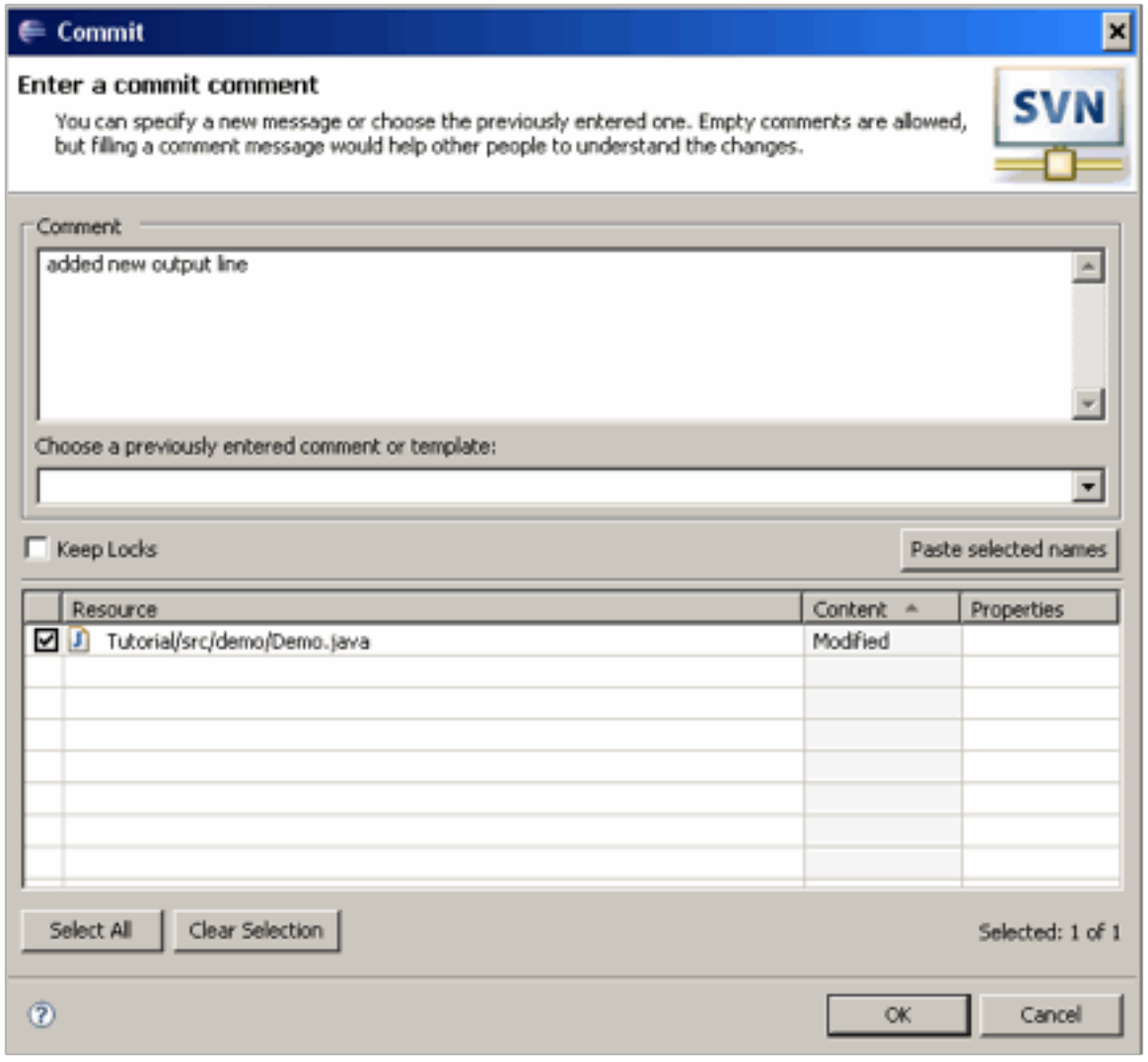
To commit a project, right click on the project in the Package Explorer pane of the Java view, and in the drop-down menu that appears, select *Team >> Commitâ€¦*



Note: you can also right click on a specific folder or file to commit just that part of the project, if youâ€™d prefer. But itâ€™s safer to always commit the entire project.

In the next window that appears, enter a comment describing what changes you have made since the last revision. Youâ€™ll also see which files are to be modified, added, or deleted in the repository. Then press OK to execute the commit.

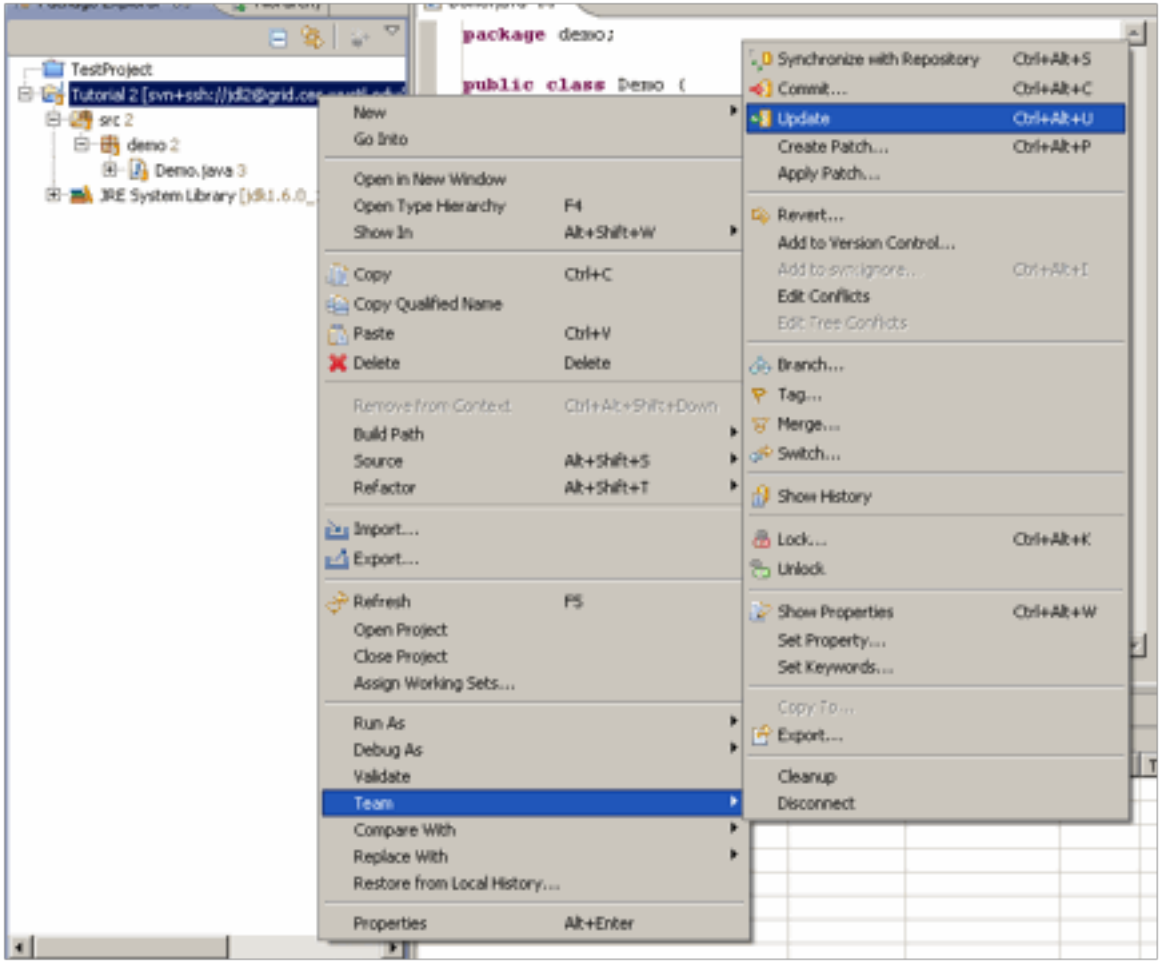
Note: adding comments are strongly recommended as a reference for yourself and others.



To ensure that the commit was finished successfully, check in the Package Explorer that the version numbers of modified files have increased appropriately and that the pointed brackets indicating differences between your version and the repository version of the file have been removed.

Updating Your Local Working Copy from the Repository

When youâ€™re working in groups or in multiple locations, youâ€™ll want to ensure that your working copy is the most recent repository version before starting to work on your project. This is done by updating your local working copy. To do this, find your local copy of the project in the Package Explorer pane of the Java perspective and then right click. On the resulting drop-down menu, go to *Team >> Update*.



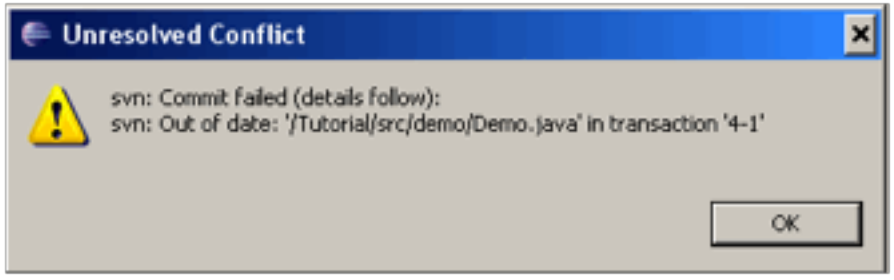
Your local working copy will automatically be updated to the newest repository version, should a newer version exist.

Note: itâ€™s a good idea to run an update whenever you start a new working session, to ensure that you have the most recent version of the project before modifying it.

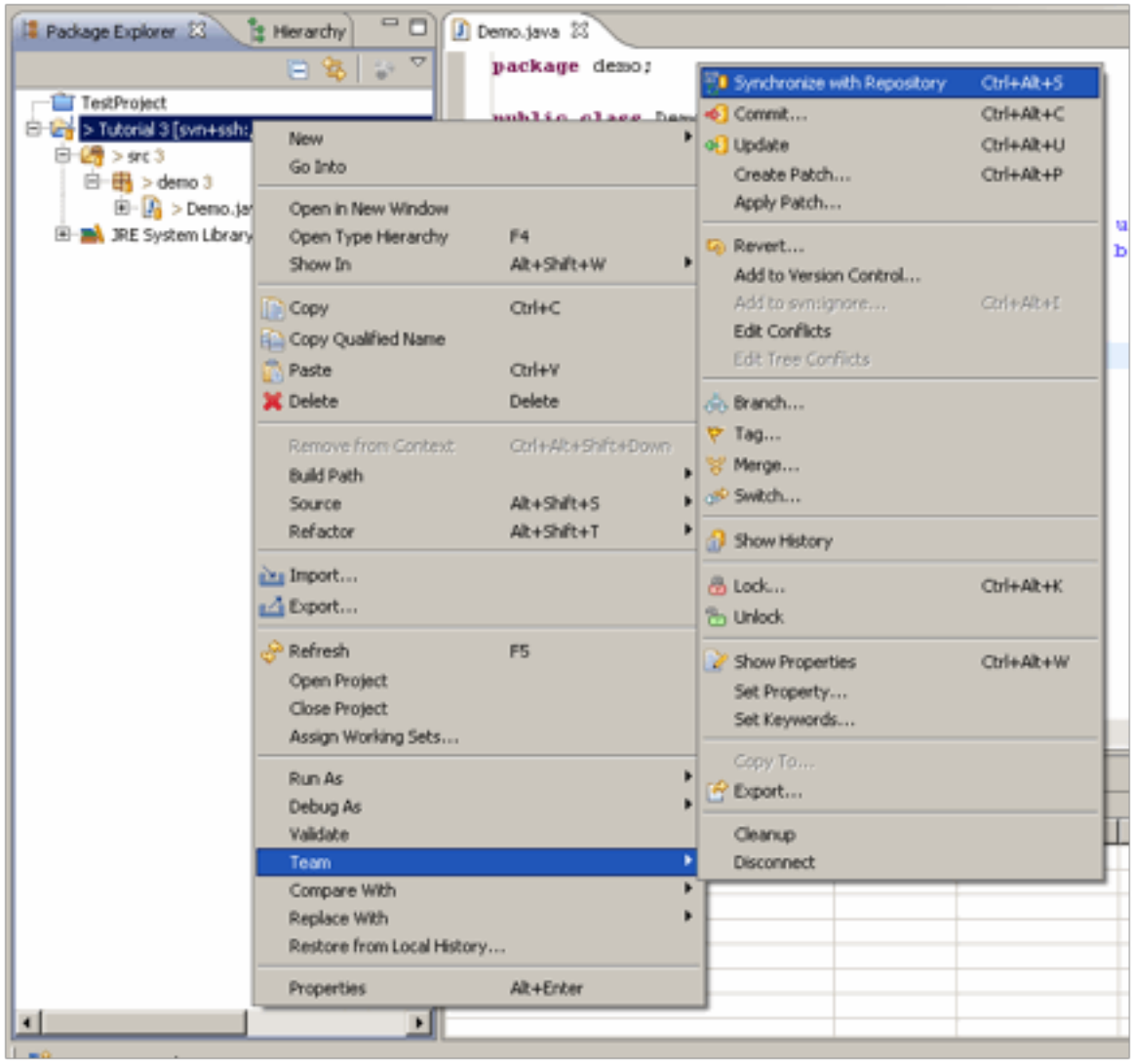
Resolving Conflicts

It may happen, especially if youâ€™re working in a team, that two people make changes to the same file and then try to commit them, resulting in a **conflict**. (This could also happen if youâ€™re working in multiple locations and donâ€™t always remember to update at the beginning of your session and commit at the end!)

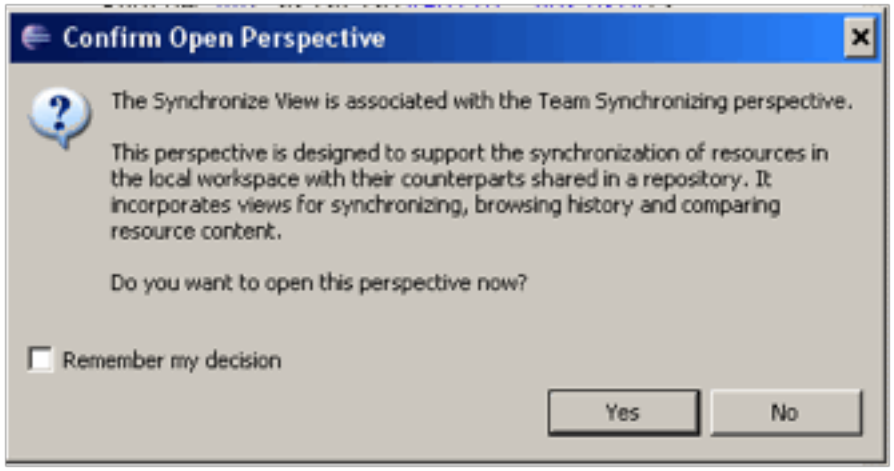
When a conflict occurs, youâ€™ll receive an error message like this one when you try to commit to the repository:



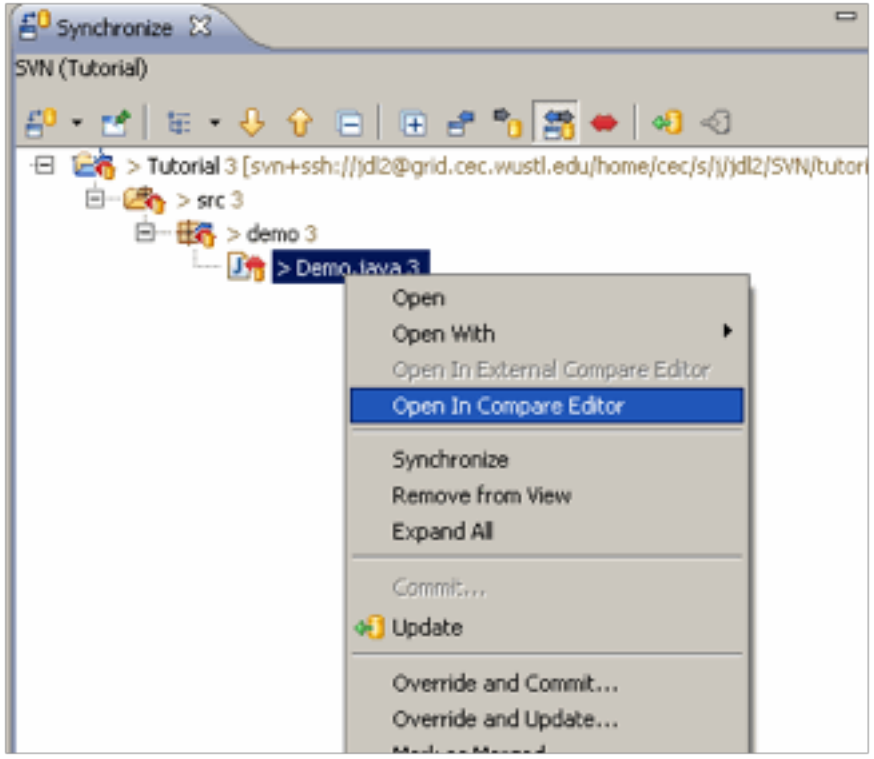
In order to resolve this conflict, youâ€™ll want to take a look at both versions of the code side by side. In order to do this, you need to synchronize your working version of the project with the version on the repository. To do this, right click on the project folder in the Java view, then in the drop-down menu go to *Team >> Synchronize with Repository.*



Eclipse should offer to open the Team Synchronizing perspective, which will allow you to compare the two versions of code side by side. Click yes when this box appears. If Eclipse does not give you this option, and it does not open the Team Synchronizing perspective automatically, you can access it by going to *Window* >> *Open Perspective* >> *Otherâ€¦* and then choosing Team Synchronizing.

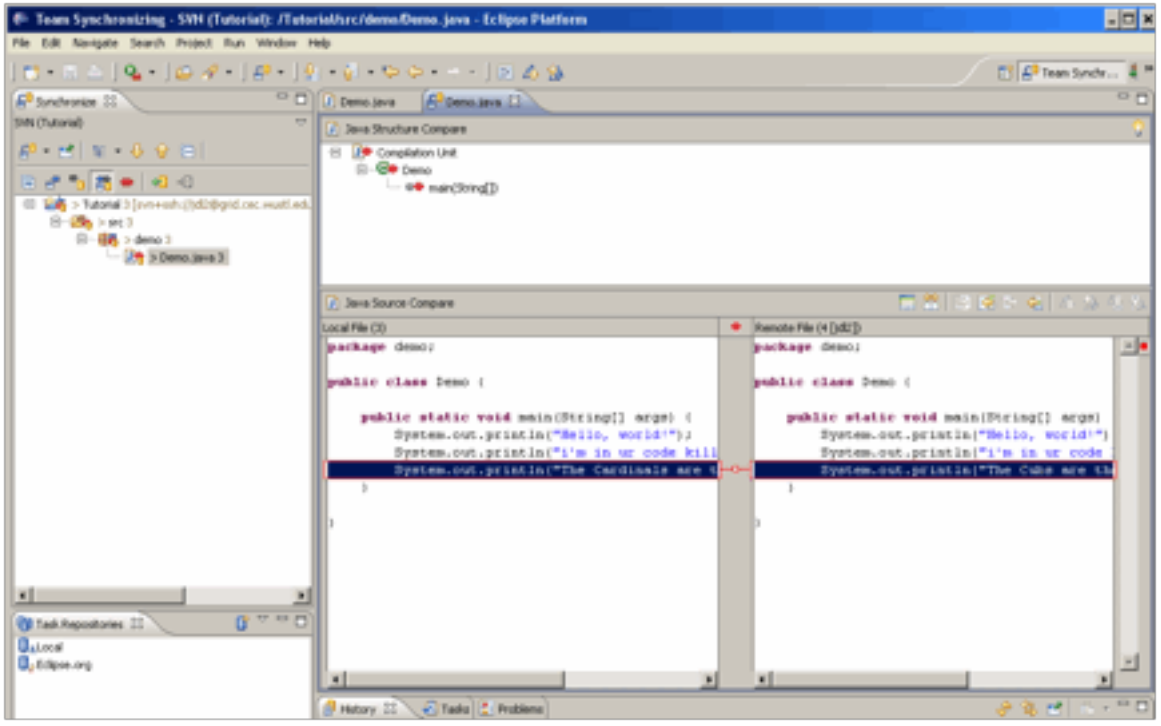


Notice that in the project tree on the right, some of the files and folders have red flags in their lower-right corners. This indicates that a conflict exists in that file (or if itâ€™s a folder, in something contained in the folder). Right-click on a file with the conflict flag and select *Open in Compare Editor*.

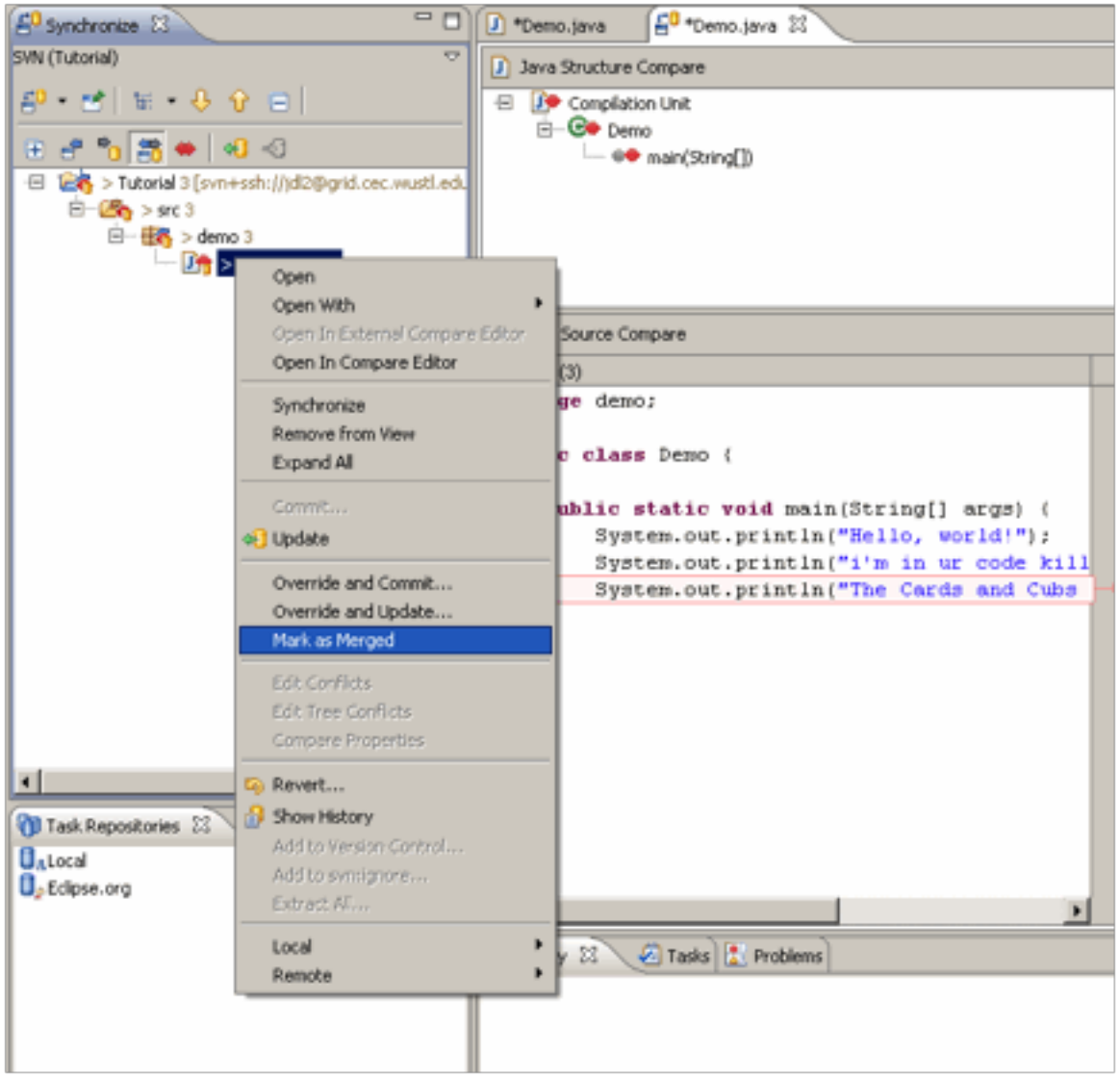


This will open the two versions of the file side by side, with the conflicting line(s) highlighted and surrounded by a red box.

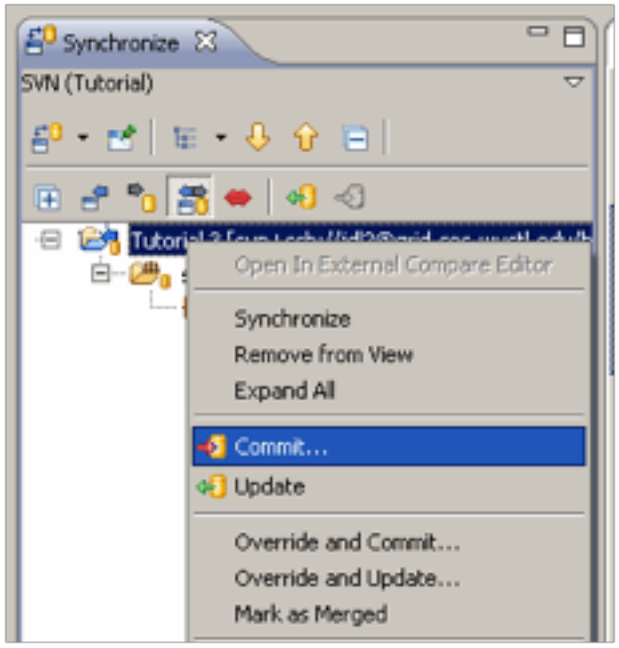
Note: using the toolbar above the side-by-side view, you can also bring up an ancestor pane showing the two versionsâ€™ common ancestor. You may find this useful.



Once youâ€™ve decided how you will correct the conflict, make whatever changes you wish to make to the **left-hand pane** of the compare view (the **local version**). Then right-click on the file youâ€™re working on in the project tree view on the left and select *Mark as Merged* from the drop-down menu.



Repeat this process for any other conflicts that may exist (i.e., as long as there are red flags next to files in the project, or you can check the number of conflicts in the project as indicated by the number next to the double-headed red arrow at the bottom of the Eclipse window). When all conflicts have been resolved, right-click on the project (topmost) folder in the left pane and select *Commitâ€¦* from the drop-down menu that appears. If all conflicts have been resolved, this should work as a normal Commit, and you will be prompted for a commit comment.



When this has been successfully completed, you can close the Team Synchronization perspective and return to the default Java perspective using the shortcuts at the upper-right side of the Eclipse window, or by using the *Window >> Open* Perspective menu.

Note: be sure to update your local working copy before you begin working again to ensure you have the latest version available.

Last Edited: May 21, 2009 by Jonathon Lundy