Eclipse and Subclipse Lab
CS 245
Oct. 2, 2006

Basics
In the following, replace [Name] with your name. For example, [Name]Project becomes TomProject.

1. Login to a Windows workstation and start an X Window session on phoenix using the Cygwin@phoenix link. Login to phoenix.

2. Open a shell and use it to start eclipse.

3. From the Java Perspective, create a new Java project, named [Name]Project.

4. Within the project, create the class Colors within the package myPackage.

5. From the SVN Repository Perspective, open the https://phoenix.goucher.edu/svn/sample repository. There will be a Practice directory within the repository. Open Practice and double-click Colors.java within Practice to open it. Copy the source code, pasting it into the file containing your Colors.java class. Be careful to keep the correct package statement.


7. Using the Team menu, share your project out to the repository, into the directory [Name]Project/Trunk. Commit all your project resources to the repository. This is how you initially share new work with your project team, following the Subversion convention of associating the Trunk directory with the main line of project development.

   Confirm that the project is now in the repository.

8. Using the Team menu, disconnect your project from the repository, deleting the meta-data. (This isn’t something you would ordinarily do, but is necessary for the next step.)

   Completely delete the local copy of your project. (Again, this isn’t something you would ordinarily do.)

9. From the SVN Repository Perspective, right-click the Trunk directory of your project repository and check-out the project. This is how you get a local, working copy of something in the repository that you don’t currently have locally.
Resolving Editing Conflicts

What happens if two team members make a change to the same line of a source file, and then try to commit their separate changes? This is known as a conflict and must be resolved. The team members would need to discuss the conflicting changes, resolve them, and commit the final change. Work in teams of two for this part of the lab.

1. One of you should check-out the other’s project from the repository.

2. Both of you should edit line 119 of Colors.java, one changing blue to red, the other changing blue to green.

3. One of you should now commit your working copy of the project to the repository.

4. The second of you should now try to commit your working copy. The commit will fail. From the Team menu, select Update, getting the current version of resources from the repository. Choose one of the conflicted files and use the Edit conflicts Team command to see your version and the current version side-by-side. You would use the Team command Show in Resource History to determine who committed the last version, so you could discuss changes.

5. Once you’ve resolved the differences and saved the final version, use Mark resolved from the Team menu to mark the conflict resolved and then commit the update.

If you think about it, while you are working so are others, and they are committing changes to the repository. How do you pick-up these changes? This is what the Update command on the Team menu is for. When your own changes are stable, but before you commit, use update to get the latest, most complete set of project files and re-test your changes. At the time of this update, you may be alerted to editing conflicts, which must be resolved. You should also always perform an update just before committing changes to the repository.