6.170 Laboratory in Software Engineering
Eclipse Reference for 6.170

Contents:

CVS in Eclipse

Setting up CVS in Your Environment

Setting up CVS in Eclipse: These instructions describe how to setup CVS on Athena ONLY. If you are working from home, take a look at the Working At Home document.

1. Launch Eclipse using `eclipse` if you are on Athena.
2. Select **Window** >> **Open Perspective** >> **Other...** >> **CVS Repository Exploring**
3. Right-click in the "CVS Repositories" window, and select **New** >> **Repository Location...**
4. Fill in the fields as follows, and as shown below:
   `Host:` athena**.dialup.mit.edu**
   `Repository path:` **/mit/<*your username*>/6.170/cvsroot/**
   `User:` *your username*
   `Password:` *leave this field blank* if you are on Athena.
   `Connection type:` **ext** if you are on Athena.
5. Click **Finish**

Checkout the Problem Set from CVS

These instructions describe how to checkout the problem sets on Athena ONLY.
If you are working from home, take a look at the Working At Home document.

How to Checkout in Eclipse
1. Launch Eclipse using `eclipse` if you are on Athena. If you are working on another platform, then launch the Eclipse executable that you downloaded. 2. Select **Window** >> **Open Perspective** >> **Other...** >> **CVS Repository Exploring** 3. Expand the CVS Repository directory structure, in the CVS Repositories Perspective Window. Expand **HEAD**, so that "psN" is visible. 4. Right-click "psN," and select **Check Out As**

`psN` is the name of the **module** that you are checking out from CVS. Checking out will create the directory `~/6.170/psN/` which will have the contents of your problem set.

How Do I Add a File to CVS?

How to Add a File to the Repository in Eclipse
1. Open the **Java Perspective** by selecting **Window >> Open Perspective >> Java**. 2. Right-click on the file that you wish to add to the repository. 3. Select **Team >> Add to Version Control** Eclipse should automatically detect which type of file (text/ASCII or binary) that you are adding.

Committing Changes to CVS

CVS Committing Using Eclipse
1. Right-click on the file you would like to commit while in the **Java Perspective**. 2. Select **Team >> Commit...** 3. Enter a descriptive log message into the comment box and press **OK**.

Updating Your Local Copy of the CVS Repository

Running CVS Update from Eclipse
1. Right-click on the file or directory you would like to update while in the **Java Perspective**. 2. Select **Team >> Update**

Compiling Code and Running Programs

Traditionally, programmers switch back and forth between editing code and compiling it. However, Eclipse users perform these two tasks concurrently, as Eclipse takes advantage of **partial recompilation** and compiles code every time that you save it.

Compiling Code

Compiling Java Code in Eclipse
Compile your file by saving it. If your file is saved and Eclipse says that it does not compile but you believe that it should, make sure that all of the files on which your file depends are saved and compiled. If that does not work, try refreshing your project or using **Project >> Rebuild Project** to force Eclipse to recognize the latest version of everything.

Using Ant in Eclipse

Using Ant in Eclipse
1. While in the **Java Perspective**, right-click on the **build.xml** file and choose **Run Ant...** 2. In the dialog that pops up, check off the Ant tasks that you would like to run and press the **Run** button.

Running your program in Eclipse

To run a program, right click on the java source file containing the main method and chose **Run As...** >> **Java Application**.

Running Unit Tests

JUnit is integrated with Eclipse, so you can run the test suite from within the IDE.

- First, select the test you want to run from the package explorer, the left pane. Since you want to test `Card`, you should select `CardTest` from the ps1 classes.
- From the Eclipse menu at the top of the screen, select **Run >> Run As >> JUnit Test**
- The JUnit GUI should pop up in place of the package explorer, momentarily, and run all the tests. You can double-click on failed tests to jump to the code for that test. When you're done inspecting the JUnit results, close the JUnit pane to go back to the package explorer.

Later on, when you have finished all the problems, you might want to run `AllTests` to run all the tests together.

If you are not working on Athena, you might have to explicitly add the `junit.jar` library using **Project >> Properties >> Java Build Path >> Libraries >> Add External JARs** and then telling Eclipse where your copy of `junit.jar` is.

Working on Java in Eclipse

Editing Code in Eclipse

Here are some tips for things that can make your life easier in Eclipse:

Autocomplete	`Ctrl-Space` asks Eclipse to help you complete some code you've started. Eclipse can complete lots of things:
	• variables, method names, class names
	  ◦ `ArrayL` Ctrl-Space --> `ArrayList`
	  ◦ `random.next` Ctrl-Space --> `random.nextInt`
	• constructor and method parameters
	  ◦ `new ArrayList(` Ctrl-Space --> popup menu of `ArrayList`'s constructors
	  ◦ `random.nextInt(` Ctrl-Space --> tooltip showing `nextInt`'s parameters
	• methods to override
	  ◦ `class Foo extends Bar {` `Ctrl-Space` --> menu of `Bar`'s methods that can be overridden

Organize import statements	`Ctrl-Shift-O` (that's O as in Organize) automatically updates the `import` statements at the top of your class, adding classes that need to be imported from other packages and removing classes that you're no longer using. If a class name is ambiguous -- e.g., `List` might be either `java.util.List` and `java.awt.List` -- then Eclipse pops up a dialog asking you which one you want.
Look up Java API documentation	`Shift-F2` when your cursor is on a class or method name. (You have to configure this feature with the location of the API documentation; see Problem Set 0 for more details.)
Comment/uncomment a block of code	`Ctrl-/` comments the highlighted region. `Ctrl-\` uncomments the highlighted region.
Mark TODO items for yourself	Start a comment with TODO to leave yourself a note about a piece of code that you need to fix. Eclipse will automatically put the comment in the Tasks pane, the pane where it shows your

	compile errors. (If you don't see the Tasks pane, use **Window** >> **Show View** >> **Tasks**.) You can jump to TODO items or compile errors in your code quickly by double-clicking on them in the Tasks pane.
See which files you haven't committed to CVS yet	Configure Eclipse so that the file icons in the Package Explorer show you which files you've added or changed but haven't committed to CVS: • Go to **Window** >> **Preferences** >> **Keys** >> **Label Decorations**. • Check the CVS box. • Still in the Preferences dialog box, go to **Team** >> **CVS** >> **Label Decorations** >> **Icons**. • Check all the check boxes: outgoing, remote, added, and new resource.
Generate get() and set() methods	Make sure the fields for which you would like to create get() and set() methods are declared in the class, then use Source > Generate Getter and Setter.
Run classes or unit tests that you've run recently	The little running man icon on the Eclipse toolbar runs the last class or unit test you just ran. Pull down its menu for your recent history of runs.
Renaming or moving packages, classes, methods, and variables	Right-click on any package, class, method or variable in the Package Explorer and select Refactor > Rename to rename it or Refactor > Move to put it in another package or class.
Emacs key bindings	If you prefer to use Emacs key bindings while editing code, do: **Window** >> **Preferences**>> **General** >> **Keys** and set **Active Configuration** to **Emacs**

Creating a New File

Creating a New Java File

Regardless of whether you are creating a new Java class or a new Java interface, the file should have the same name (and capitalization) of the class or interface and end with a `.java` extension.

How to Create a Java Source File in Eclipse
1. Select **File** >> **New** >> **Class** or **Interface**, depending on which type of file you

are creating.
2. Fill out the fields of the dialog that comes up that are pertinent the new file that you are creating, especially **Package**, **Name**, and **Source Folder** whose value should be `psN/src`.
3. When the fields are filled out correctly, the **Finish** button will become enabled. Click it.

Creating a New Non-Java File

Relevant textfiles, screen shots, and diagrams should also be created in your `psN` directory.

How to Create a New Non-Java File in Eclipse

1. Select **File** >> **New** >> **File**
2. In the dialog that appears, select the folder that should contain the file from the dialog, enter the name of the new file, and press the **Finish** button.

Note: If you have added a file to your psN directory through another program and you do not see it in your filetree in Eclipse, try right-clicking on the folder where you expect to see the file and choose **Refresh**.

Known Eclipse Bugs

If you are using the milestone (3.1M4) build instead of a stable build, there are some known bugs with the Eclipse compiler:

- Some lint-related warning messages are known to be suppressed by the Eclipse compiler.
- The Eclipse compiler will complain that the return type for `next()` for a class that implements `Iterator` is invalid, even it is not.
- The Eclipse compiler sometimes doesn't build a large project correctly resulting in `AbstractMethodException`s, when such exceptions shouldn't occur. Cleaning the project doesn't help either. The workaround that we discovered was to revert to commandline ant. Basically, the following seems to fix this problem:
-
- `ant clean`
- `ant build`

Since these are compiler-bugs there are no known solutions, except perhaps to install a newer version of Eclipse when one become available. However, the default `javac` compiler that ships with JDK1.5 should not have these bugs and a workaround is to compile using Ant.