CSE 331 Software Design and Implementation

Handout T1: *Tools Overview*

Quick Links to Other Tools Handouts:

- Editing, Compiling, Running, and Testing Java Programs
- Version Control (SVN) Reference
- Eclipse Reference for CSE 331
- Working at home
- Daikon invariant detector

Contents:

UW CSE Instructional Computing

Students may use both the Windows and Linux machines in the UW software labs. However, in certain cases, you will need a Linux prompt to perform an action.

Windows users should SSH into the the Unix host `attu` to perform command line actions. For some actions, Linux users will also be required to SSH into `attu` in order to ensure consistent results.

Using Unix

In order to perform some CSE 331 tasks, you will need a basic knowledge of Unix/Linux commands. If you are unfamiliar with working with the Unix command-line interface, go through the Unix tutorials provided by the student ACM chapter.

Initial Setup

You must setup your CSE 331 account via the student-setup script early in the quarter.

You must be registered for CSE 331 and have a UW CSE account in order to perform the setup. If you are on the overload list or have some other special circumstance, contact the course staff immediately.

To run the student-setup script, SSH into attu and enter the following commands at the `attu` prompt:

```
/cse/courses/cse331/13sp/bin/student-setup
```

If you see the message "CSE 331 setup complete!", you are all set!

If you see an error message, contact the course staff for assistance (see the staff list for email information, or ask for help on the CSE 331 forum.)

Logging into attu with SSH

`attu` is the name of an Instructional Workstation (IWS) machine. You will occasionally need to log into this machine. See the Working at Home document for instructions on logging in to attu, whether from home or from another CSE computer.

Java

Oracle's Java Development Kit (JDK) is available on the UW CSE instructional machines, and may be used for coding CSE 331 assignment solutions. This quarter, CSE 331 will use version 1.7 (a.k.a. 7) of the JDK.

Editing, Compiling, Running, and Testing Java Programs

Please see the document **Tools Handout: Editing, Compiling, Running, and Testing Java Programs** for detailed instructions.

Useful Java Websites

- Oracle's main Java site: http://java.com/
- Oracle's Java Documentation: http://docs.oracle.com/javase/
- Oracle's 1.7 JDK, including documentation of all APIs: http://docs.oracle.com/javase/7/docs/api/
- Oracle's Java tutorial: http://docs.oracle.com/javase/tutorial/
- Java Language Specification: http://docs.oracle.com/javase/specs/
- Cafe au Lait: Java FAQ, News, and Resources

Using javadoc to generate specs

Oracle's Java Development Kit includes javadoc, a tool that produces specifications from source code annotated with special comments. The comments may include "tags", which are introduced by an at-sign (@).

We have extended the javadoc program to recognize additional CSE 331 tags, as well as all the tags accepted by the Oracle Standard Doclet. These additional tags declare specification fields for classes and requires, modifies, and effects clauses for methods. Note that these tags must appear **after** all non-tag comments for classes and methods.

- Some extended tags belong in the overview for a class; they are used to formally define what a given Abstract Data Type represents.

@specfield *name* : *T* // *text*	Indicates that *name* is a abstract specification field of type *T* for the class, adding *text* as a comment if present
@derivedfield *name* : *T* // *text*	Same as specfield, except that this also adds the property "derived" to the output information

  Derived fields can be viewed as functions on preexisting state; thus if a class had a specfield `@specfield n : integer` we could define a derived field:

  ```
  @derivedfield pos : boolean // pos = true iff n > 0
  ```

  Derived fields are not allowed to hold any information that could not be already calculated from the already existing state in the object. Thus, you use specfields to introduce new state variables and derived fields to introduce functions on those state variables.

  Derived fields are not strictly needed in specifications, but they may reduce complexity and redundancy.

- Other extended tags belong in the specification for a method; they define the method's preconditions, postconditions, and side effects.

@requires *X*	Declares *X* to be a precondition for the method
@modifies *Y*	Declares that nothing besides *Y* will be modified by the method (as long as *X* holds when it is invoked)
@effects *Z*	Declares that *Z* will hold at exit from the method (as long as *X* holds when it is invoked)

The preferred way to generate API documentation with the CSE 331-extended javadoc is to use the ant `doc` target for your assignment. See running automated tasks with ant in the "editing, compiling, running and testing" handout. The instructions for eclipse, briefly:

a. Right-click on `build.xml` in Package Explorer
b. Select **Run Ant...**
c. Toggle the checkboxes so that **doc** is the only box selected
d. Click **Run**
e. After you see a **BUILD SUCCESSFUL** message in the console at the bottom of the screen, right-click on `doc` in Package Explorer
f. Select **Refresh**
g. Open the `doc` folder and double-click `index.html`.

After running the ant `doc` target, you should check the output. You may find that you need to add line breaks (`<br>`) or paragraph breaks (`<p>`) to your javadoc comments for readability. Also, if you omit certain tags, subsequent text may fail to appear in the output. Finally, since much of the text of javadoc comments is inserted in a HTML document, you must be careful with text that can be interpreted as HTML markup, such as the less-than (<) and greater-than (>) characters. For instance, if you write:

```
@effects Adds <x> and <y>
```

then <x> and <y> will be interpreted as HTML tags in the output (and won't be displayed by a browser). It's usually better to just write

```
@effects Adds x and y
```

Report any weird behavior or complaints about `javadoc` to cse331-staff@cs.washington.edu.

Eclipse

Eclipse is an integrated development environment (IDE) that you can use to develop your Java code. It is already installed at UW CSE. If you are working on your own computer, then you should install the latest version of "Eclipse IDE for Java Developers", then install Subclipse.

For tips about using Eclipse, please consult the following documentation:

- **Eclipse Reference for CSE 331**
- **Editing, Compiling, Running, and Testing Java Programs**

Daikon invariant detector

See the separate Daikon handout.

SVN

Please consult the **Tools Handout: Version Control Reference** for details on using SVN to manage your source code.

Forums, email, etc.

CSE 331 Forums

The CSE 331 forums are a great place to post technical questions about the assignments, Java, and the course in general. The TAs, professor, and other students monitor the forums, so responses can arrive relatively quickly and be shared with all members of the class.

When you are posting to the forums, please provide enough detail so that others can reproduce your problem. Strive to post the most succinct message that gets your point across. Extraneous details such as pages of obscure compiler warnings can distract from your question and make it less likely for others to look at it carefully on first glance.

Keep in mind that TA office hours may be more appropriate for getting help with coding bugs and environment problems.

Class Mailing List

The course staff will use the class mailing list to send important course announcements. Your @u.washington.edu address is subscribed. Do not attempt to send questions to the mailing list, instead either email the course staff or use the forum.

Setting Project Directory Permissions

Your project directories were created by support@cs with full read and execute permissions. As a result, anyone can check out of your SVN repository. To prevent other students from viewing your work, modify the directory by connecting to attu and executing the command:

```
  chmod -R o-rwx /projects/instr/13sp/cse331/$USER
```

Do not modify the cse331 group permissions. The cse331 group is for course staff and full permissions are required to deliver assignments to your repository.

Errors When Setting Project Directory Permissions

If the course staff has recently imported files into your repository, you will receive errors of the following form:

```
  chmod: changing permissions of `/projects/instr/13sp/cse331/YourUserName/REPOS/db/...':
Operation not permitted
```

These errors are expected and occur because SVN creates some files owned by the course staff whenever we commit to your repository. You do not need to rerun the command.

Drawing diagrams

You may occasionally wish to draw diagrams, such as object models and module dependency diagrams. One fast and effective way to do this is to draw in longhand, then scan your results in order to submit them electronically. Alternately, you may wish to use a drawing program. Here are some suggestions. Many others exist; you are welcome to use any one that you feel comfortable with.

Visio

Microsoft Visio is a diagramming and drawing application for Microsoft Windows.

Visio templates and stencils for CSE 331 are available as zip files. See the README file in the zip archive for instructions; essentially, the files should be placed in the `Solutions` subdirectory of your Visio installation.

OmniGraffle

OmniGraffle is a diagramming and drawing application for Mac OS X.

Here are some OmniGraffle templates and stencils you might find helpful when drawing diagrams for CSE 331: OmniGraffle stencils

Dia

Dia is a cross-platform (Unix, Mac, Windows) diagramming and drawing application.

You can download Dia from its homepage at http://www.gnome.org/projects/dia/.

Dia can export its diagrams as Encapsulated PostScript files, which you can then convert to PDF. Or, you can print the diagrams from Dia itself.

If you have problems **using** Dia, see the below links:

- Dia homepage
- Dia tutorial

xwrits

xwrits is a Unix/X-window program that reminds you to take typing breaks, which are important for your health. There are lots of ways to customize and use xwrits; here is one way which is strict about preventing you from typing:

```
xwrits typetime=5 breaktime=0.50 +beep +clock +idle +mouse maxhands=5 +multiply +noiconify
+lock &
```

You can also put the xwrits command in your `.startup.X` file.

Working at home

If you choose to do some or all of your work on your own computer, instead of in the Allen Center software labs, then see the document on working at home.

For problems or questions regarding this page, contact: cse331-staff@cs.washington.edu.