Map a network drive to  `\ntfiler\sr2007`.
(You may map it to any drive letter, but for the purpose of this explanation, I will assume it is mapped to drive s: )
1) To map a drive:
2) Open My computer.
3) Click Tools: Map network drive at the top
4) Select drive S:
5) In the folder box, type: `//ntfiler/sr2007`
6) (If you are doing this from home, you would need to also click ‘connect using a different user name’, then enter in your username as CPSC/myusername and then your password.)
7) Click Finish

Open Eclipse:
1) Select your workspace on the drive you just created.
2) E.g., select: `s:/myusername/workspace` (or `s:/myusername/eclipse/workspace` or whatever you called it)
3) If you see a mostly blue screen (not white), select the ‘my workspace’ icon in the upper right corner
4) You should now see a mostly white screen of three-four sections.
5) I prefer the ‘Package Explorer’ view. If you don’t see it at the right, then, at the top, click Window → Show View → Other, then select Java → Package Explorer
6) Change the view to Hierarchical. At the top right corner of the Package Explorer, you will see a down arrow. Click it and select Package Presentation → Hierarchical
7) Make sure that your java version is set to 1.5 (or 6). At the top, select Window → Preferences → Java → Compiler → Compiler Compliance Level → 5.0 (or 6.0 – although this is not tested yet, so try this on your own time – you will need to check that 6.0 is installed on your machine)

Checkout tamudrg Project:
1) Select File → New → Project
2) Select CVS → Projects from CVS
3) Select Create a new repository location (unless you already have repository in list below :`extssh:username@csdll.cs.tamu.edu:/cvsroot`)
4) Enter Host: csdll.cs.tamu.edu
5) Repository path: `/cvsroot`
6) User: username
7) Password: password
8) Connection type: extssh
9) Use default port
10) Select Save password (if you wish)
11) Press enter
12) Select use specified module name
13) Type tamudrg
14) Select check out as project in the workspace
15) Enter tamudrg in as project name
16) Select checkout subfolders
17) Select use default workspace location (should be the same as the workspace location you typed again on drive s)
18) Select Next
19) HEAD should be selected under matching tags.
20) Press Finish.

Checkout class working space:
1) Following the instructions from above, check out the SR2007 project from the repository.
2) Create a directory with your name on it in SR2007 (similarly to the hammond or kdahmen directory) This directory is your project space – only check in code from here unless you have spoken to me.
3) Also checkout SRLData

Synchronize Repository:
5) This will offer to bring up a new window. Select okay.
6) You will need to enter in the class user name and password. (On the board.)
7) This will take a minute or two.
8) In the CVS window, at the top you will see many arrows. Select the yellow arrow pointing to the LEFT (not the RIGHT).
9) A window should pop up: “Confirm update! Are you sure you want to update X resources?” Select Yes. (If it asks you to enter in a text string or you see a text box, cancel. You selected the wrong arrow.)
10) DO NOT CHECK IN CHANGES FROM tamudrg without speaking to me first. (You will want to receive updates, however.)
11) However, please do check in your changes in your directory – use the RIGHT arrow to check in changes.
12) This will take about a minute.
13) Switch back to the package explorer. At the top right of your screen there should be a box stating ‘Team Synchronize’, select the button directly to the left of that with a screen on it. Select “Resource”

Start GUILD (generator of user interfaces from LADDER descriptions)
1) In the Package Explorer, find tamudrg → code/src → edu → mit → sketch → language → debugger → GUILD.java. Right click, and select Run As → Java Application
2) You should see a small box pop-up box asking you to “Please select your drg directory”. Traverse to your workspace (e.g., s:/myusername/workspace), then when you see the tamudrg directory, select it by single-clicking on it (not double). Make sure it did not go inside that directory, but that the tamudrg directory is still viewable. Select the ‘drg directory’ button below.
3) You will see another small pop-up box, stating: “Use this directory every time?”
Select “No”. (Once you have run the application a few times and are confident
that you have set the previous variable correctly, you may then select, “Yes,” and
the application will no longer ask you for either piece of information. If you then
want to change it, you must select Project ➔ Clean at the top of Eclipse to
recompile everything, which will reset all of these settings.)
4) You should now see a directory that says: “Open a LADDER domain list.” If you
set your previous tamudrg directory correctly, you will also see many .ldl files.
5) If you do not see ldl files, then search for

Play Tic Tac Toe
1) Select tictactoe.ldl
2) You should see a blank screen on the right. Try to play tic tac toe.
3) Draw X’s and O’s and a Board. Try to make the system tell you that you have
won a game.
4) When finished, close the application and restart.

Experiment with Drawing in other Applications
I recommend trying Finite State Machines, Military Course of Action Diagrams,
Mechanical Engineering (you may need to install Working Model. Contact me for
instructions), Electrical Engineering (you may need to install Spice), UML class
diagrams, or Flow charts. (Some of them may be out of date. If you encounter and error
please email me.)

Create Your Own Tic Tac Toe Game
1) Click File ➔ Domain List ➔ Open/New
2) Instead of clicking on an ldl file, type in yournameTTT (note this does not yet
work in Macs, you need to hand create a new text file, we will fix the bug)

Creating a new shape (Cross in our example).
1) In the Domain List Panel, select ‘Add’.
2) You should be in the domain descriptions directory. Click sketch ➔ shapes to
open. Then create a new directory there entitled yournameTTT. Inside that
directory, create a new shape by typing ‘Cross’.
3) Open the Shape Definition Panel.
4) Under components: Enter the components that make up your shape one at a time
and name each of them. (e.g., a Cross is made of two lines that could be named
pos and neg: Line pos, Line neg)
5) Under Display Specification, set the color: (color green)
6) In the Drawing Space Panel draw your shape (two lines), it should turn the
appropriate color (green). It should also accept any shape made up of those
primitives (two lines).
7) Go back to the Shape Definition Panel, add a constraint one at a time. Test each
constraint by drawing an example of the shape and making sure that it turns the
correct color, and then drawing another shape with the same primitives but with
the constraint false, and it should not change color.
8) Experiment with using context shapes already on the screen to help recognize shapes.
9) Experiment with adding different editing behaviors
10) Experiment with adding aliases (these allow more shape parts to be accessible when defining shapes hierarchically)

Create Abstract Shapes
1) You can define a new shape with nothing in the definition.
2) Other shapes can extend this abstract shape.
3) You can then use this shape as a more general component.
4) For instance, both Circle and Cross can be defined as a ‘Piece’ in tic tac toe. You could then define a tie condition to exist when there are 9 ‘Piece’s on a Board.

Adding a Backend
1) If you are good at Java, you can create a backend application that runs whenever the ‘Run (Connect) button is pressed.
2) In the eclipse package explorer, go to the tamudrg\code\src\edu\mit\sketch\language\applink directory.
3) Right click the applink directory, and select New \ Other, then select Class.
4) A window entitled: “New Java Class” should pop up.
5) In Name, type the exact name of your ldl class, but with out the .ldl. E.g., for your MynameTTT.ldl class, type MynameTTT. Make sure it starts with a capital letter
6) In the superclass textfield, type AppLink, then click browse and select the AppLink class.
7) Inside of the connects method, add the code:
   ```java
   System.out.println("Yippee! There are " +
   getViewableShapes().size() + " shapes.");
   for(DrawnShape s : getViewableShapes()){
       System.out.println(s);
   }
   ```
8) You will also have to import edu.mit.sketch.language.shapes.DrawnShape;
9) Click Run on your application, you should see a printout in the Eclipse Console.
10) Examine the AppLink.java class to see what methods are available to you.
11) Examine TicTacToe.java, FSM.java, MechEng.java, and Circuits.java, for examples.
12) Code something.