How to Set Up MASON in Eclipse

**Prepared by: Steve Lytinen (http://condor.depaul.edu/~slytinen/) and
Steve Railsback (Lang, Railsback & Associates)**

Disclaimer: This document is provided in hopes that it will be useful to agent-based
modelers, but with no guarantee that it is accurate, up to date, or even helpful. Eclipse and
MASON are both complex, evolving projects and there are many ways to go wrong when
setting up and using MASON in Eclipse. We recommend you start by trying to follow our
instructions exactly; if they fail, then experiment or get help. If you find mistakes or
improvements, please contact us through the Swarm Development Group wiki
(www.swarm.org) page where this document is posted, or post your comments to the agent-
based modeling email list (see: www.swarm.org/wiki/Swarm:_Mailing_lists).

I.   Introduction

Eclipse is a widely used and powerful integrated development environment, and makes a
convenient platform for developing MASON models. Eclipse is free and available for Linux and
Windows. Eclipse provides features such as a Java-specific editor, graphical displays of program
and package hierarchies, and a debugger. This how-to describes setting up Eclipse so the MASON
libraries can be viewed and accessed in a new model. It was written for Eclipse 3.1 for Windows.

Notes:
- The approach is to set up MASON as an Eclipse project; then set up new projects for your
  MASON models. Your new projects will access the MASON project.
- If you deviate from any of these instructions, your setup is very likely to fail.
- When you import code (including MASON, or an existing Swarm model) into Eclipse, it
  creates a complete new copy of the code in your Eclipse workspace. This means you will end
  up with two copies of MASON, which also means that you can delete models or MASON from
  Eclipse without necessarily removing them from your computer.

- If you need to start over, be sure to first delete the folders for MASON and any other projects you created in the Eclipse workspace directory, by right-clicking on them in the Package Explorer panel of Eclipse. Tell Eclipse to remove their content too.

II.   Installing Eclipse and MASON in Eclipse

1. Install Eclipse, using downloads from: http://www.eclipse.org/. Install MASON by uncompressing its files somewhere on your computer.

2. Start Eclipse. First, Eclipse will ask you to select a directory for its workspace: the place where it stores the program files you create. Then, click on "Workbench" (an icon in the upper right corner of the Welcome screen) to go to the Eclipse workbench. (If you have previously used Eclipse, you might want to create a new, separate workspace for MASON projects.)

3. Create a Java project to contain MASON.

  a. In the Eclipse menu, select `File -> New -> Project`. This opens a window where you select the `Java Project` wizard and click `Next`.

  b. In the new project window that opens, provide the project name ("MASON"), and click on `Create new project in workspace,` and `Create separate source and output folders`. Then click on `Finish` to close the window.

  c. The new MASON project now appears in Eclipse's Package Explorer window. Expand the MASON project by clicking on the + to its left. Then right-click on the "src" directory under MASON and select `Import -> File system -> Next`. Browse to the MASON directory on your computer, then down to the "mason" subdirectory. (This might be C:\mason\mason.) Click `OK`, then click the + to expand the "mason" folder icon.

Then check the boxes for `ec` and `sim` on the left side.
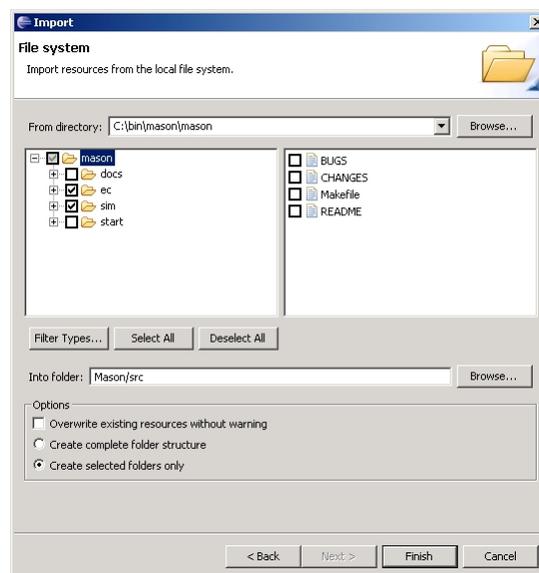
Finally, click `Finish` and wait for Eclipse to load the MASON libraries. Ignore any problems that show up in Eclipse's Problems window.

III.   Creating a Project for a New MASON Model

Use this step if you want to create a new MASON model from the start. (See the following section to import and modify an existing MASON model.)

1. In the Eclipse menu, select `File -> New -> Project`. Then, in the "Select a wizard" window, select `Java Project` and hit `Next`.

2. In the new project window that opens, provide a name for your project, and click on `Create separate source and output folders`. Click `Next` to move to the `Java settings` window.

3. Click on the `Projects` tab, and hit `Add`. Click on the MASON check box to add the project containing MASON to your new project. Click on `OK`, then `Finish` to close the window. Ignore any problems in the MASON libraries that appear in the Eclipse problems panel.
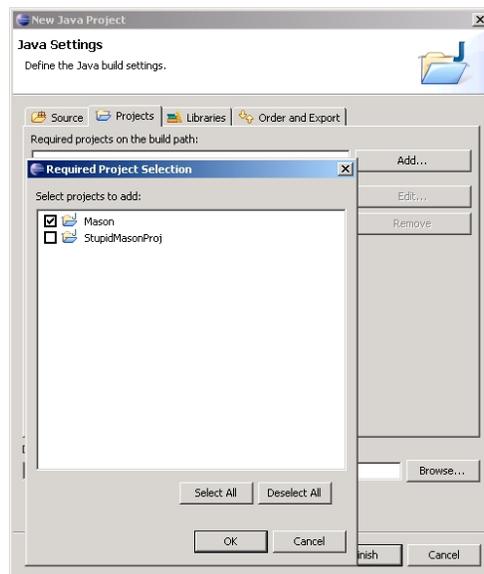
4. Create a new Java package for your model. In the Eclipse Package Explorer window, click on the new project you just created. Then on the Eclipse window, hit `File -> New -> New Java Package` (not "project"). Enter a valid package name (Eclipse will tell you if you try to use an invalid name).

Now, you can start using Eclipse tools to create new classes, etc. By expanding the MASON project in the Package Explorer window, you can see all the MASON classes and methods that you can use.

IV.   Importing A MASON Model into Eclipse

If you want to import an existing MASON model (including one developed in Eclipse), follow these steps.

1. In the Eclipse menu, select `File -> New -> Project`. Then, in the "Select a wizard" window, select `Java Project` and hit `Next`.

2. In the new project window that opens, provide a name for your project, and click on `Create separate source and output folders`. Click `Next` to move to the `Java settings` window.

3. Click on the `Projects` tab, and hit `Add`. Click on the MASON check box to add it to your new project. (See the figure in Section III, above.) Click on `OK`, then `Finish` to close the window. Ignore any problems in the MASON libraries that appear in the Eclipse problems panel.

4. See (6) below if you want to import an existing Eclipse project. If instead you are importing a directory of existing .java files, first create a new package within the new project. In the Eclipse Package Explorer window, expand the new project you just created and make sure it contains a "src" folder. (If there is no "src" folder, you forgot to hit `Create separate source and output folders` in step 2, above.) Then on the Eclipse menu, hit `File -> New -> New Java Package` (not "project"). Enter a valid package name (Eclipse will tell you if you try to use an invalid name, such as one starting with a capital letter). This will create a new package directory under "src".

5. In the Eclipse Package Explorer, right-click on the new package and select `Import -> File system`. In the window that opens, browse to and select the directory of source code to import. After you select the directory, Eclipse will let you select some or all of the files in the directory.

One common problem at this point is that the existing source code you import has no package statements, or wrong package statements: at the top of each .java file must be a statement "`package thePackageName;`" where `thePackageName` must match the package name in the Eclipse Package Explorer panel. (If there are Package statements but with the wrong package name, see "Renaming", below.)

6. If you are importing an Eclipse project, just select the project's entire directory. A checkbox appears for the project; click it to get the whole project. Click "no" for whether to overwrite the existing class path, but "yes" to overwrite other stuff.

V. Running a Model

To attempt compiling and running a MASON model in Eclipse:

1. On the Eclipse main menu, hit `Run -> Run`. This opens a window in which you select a "run configuration"- a set of code you want to execute. First, on the left side of the window select `Java Application`. Then hit `New`, and enter the name of the package and the name of the file containing its Main class. Then hit either `Apply` or `Run`.
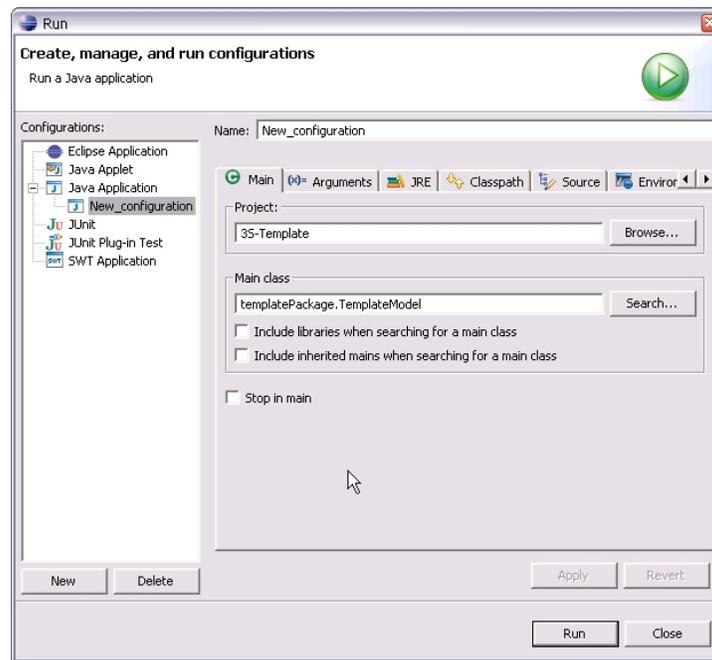
Your application will now attempt to compile and run, with errors appearing in the Problems window. Eclipse may issue a warning that there are errors in Mason and ask if you want to continue; say yes.

2. You can re-run a program, once configured, by hitting `Run -> Run Last Launched` on the Eclipse menu.

VI. Using the Eclipse Debugger

Eclipse has a built-in debugger that lets you step through your code and inspect variables as each step is executed. The debugger is useful for understanding how model results arise as well as for finding and fixing mistakes.

1. Set a breakpoint: a place in your code where the debugger will stop and let you inspect the model's state. In the Eclipse code editor, right-click in the left margin at the line where you want a breakpoint and select `Toggle Breakpoint`. A red message will appear at the bottom of the Eclipse window if it is not a valid place to put a breakpoint. (If right-clicking does not work, click to put the cursor on the line where you want a breakpoint; then use the menu to hit `Run -> Toggle Line Breakpoint`.)

2. On the menu, select `Run -> Debug`. A window appears to let you select which run configuration to debug; the default is the last thing you tried to run. Hit `Debug`. Your MASON model will start up; click the MASON start button to begin execution. The code will run until it hits a breakpoint.

3. A pop-up window will ask you to confirm that it is OK to switch to the "debug perspective" (a different arrangement of Eclipse windows for debugging). It is.

4. A "Debug" window will appear. It has buttons on its toolbar that let you "step into" (execute one line at a time, going into the method calls as they are encountered), "step over" (skip over method calls), and "Step Return" (which runs until the next Return statement is reached - use this when the debugger delves into a bunch of very strange, confusing Java thread stuff that you do not want to know about). See the Eclipse documentation for more details on stepping the debugger.

5. In the Variables window, you can expand objects to see values of their instance variables.

6. To terminate the debugger, click on the red square button in the Eclipse Debug window. Then, you need to right-click on anything in the Debug window and hit `Remove all terminated` to get rid of old debug threads.

VII.   Eclipse Features to Help You Write and Debug Java Code

There are several reasons to use an Integrated Development Environment (IDE) such as Eclipse when writing code. Here, we discuss some of the features provided by Eclipse that make writing Java programs easier.

  1. Declaring a new class
       a. Click on the "New Java class" icon . You will then see a window that looks like this:
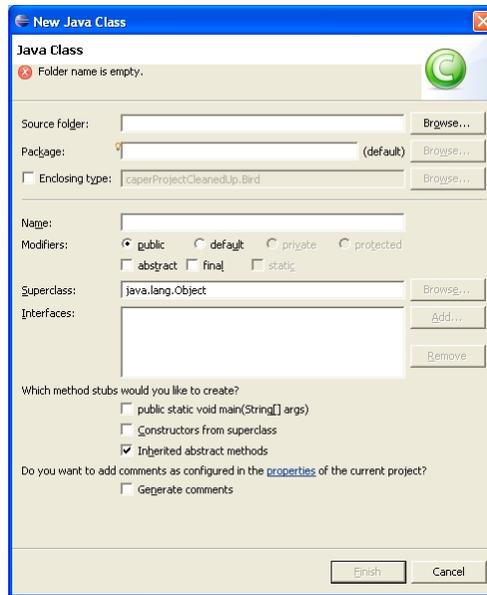
Probably, the "Source folder" and "Package" textboxes will automatically be filled in.

b. Type a class name (in the "Name" textbox). If you wish to extend a class other than Object, click the "Browse" button (it will become clickable once you type a class name). Then you will see a new Window. As you start typing in the name of the desired superclass, you will see a list of built-in (MASON or Java) classes that match what you have typed. Select the class you want, and click OK.

c. If you want your class to implement an interface, click the "Add" button next to the Interface text area. Filling in an interface name is similar to selecting a superclass.

d. Click the appropriate checkboxes (for example, if you want your class to have a main method); then click the "Finish" button.

2. Adding import statements

   If you write code that uses a class in another package (including the MASON packages), it
   is necessary to have an import statement at the top of the source file. Eclipse will
   automatically generate the appropriate import statement if you:
   a. Highlight the class name. You can do this by double-clicking with the cursor over
      any part of the name.
   b. In the Source menu, select "Add import".
   c. Select the appropriate package/class combination to import.

3. Generating "setter" and "getter" methods

   There are many situations in which methods are written to set or return the value of an
   instance variable. These methods are called "setter" and "getter" methods. For example, if
   you wish to make a variable appear in a MASON "inspector", the variable needs getter and
   setter methods.

   a. Highlight the variable name, as described above.
   b. In the Source menu, select "Generate getters and setters".
   c. Click/unclick the checkboxes that are displayed, depending on whether you want
      both a getter and a setter, or just one or the other.

4. Renaming

   If you ever decide that you want to rename something in a Java program, it is easy to do in
   Eclipse. Simply highlight the item (variable, class, package, etc.) that you wish to rename.
   Then, in the Refactor menu, choose "Rename". Eclipse will propagate the name change
   throughout the entire project.

5. Correcting syntax errors

   If you miswrite a Java statement, Eclipse will mark the line with a red X symbol on the left
   edge of the code window. To see what the mistake is, move the cursor over the X. A
   message will appear that tells you what is wrong. For example:
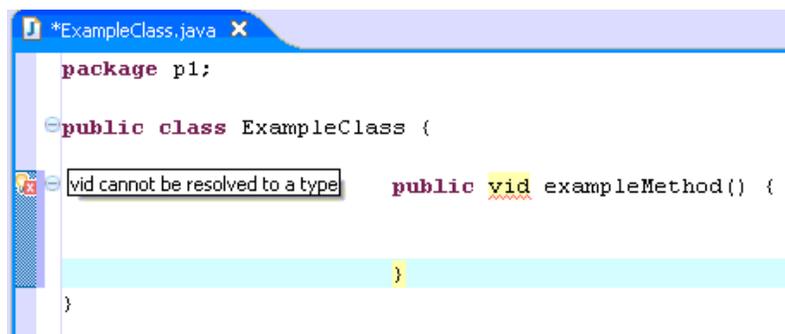
   Eclipse can sometimes propose a fix. To see possible fixes, click on the red X:
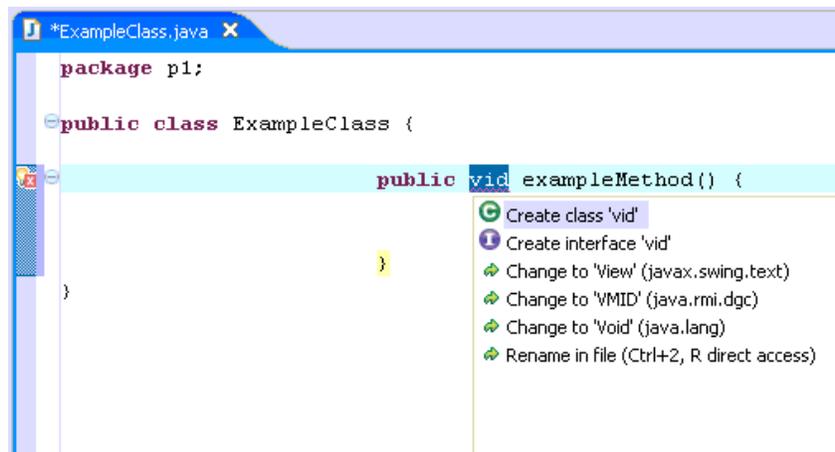
Select the one which will fix the problem (in this case, "Void").

6. Debugging

   You can use Eclipse's debugger to stop execution of your program at a particular place in your code, and then inspect the program's data (objects) at that point.

   a. To specify where your program should stop, you need to set a "breakpoint". To do this, first position the cursor on the line of code where you wish to stop. Then move the mouse to the left edge of the code window, and right click. You should see a blue circle next to the line, indicating that a breakpoint has been set:
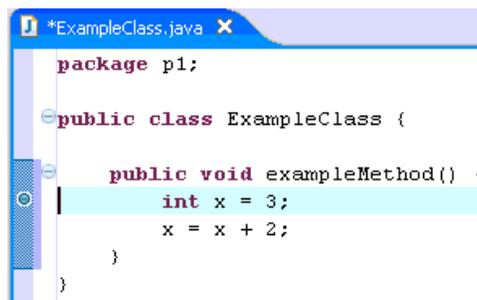
   b. Start your program, by select "Debug" from the Run menu. Start the MASON process as usual.
   c. When your program reaches the breakpoint, Eclipse will either automatically switch to the Debug perspective, or the Eclipse icon on the Taskbar (on the bottom of your computer screen) will start to blink, indicating that Eclipse would like to switch to this perspective. Once you do, you will see a screen that looks like this:
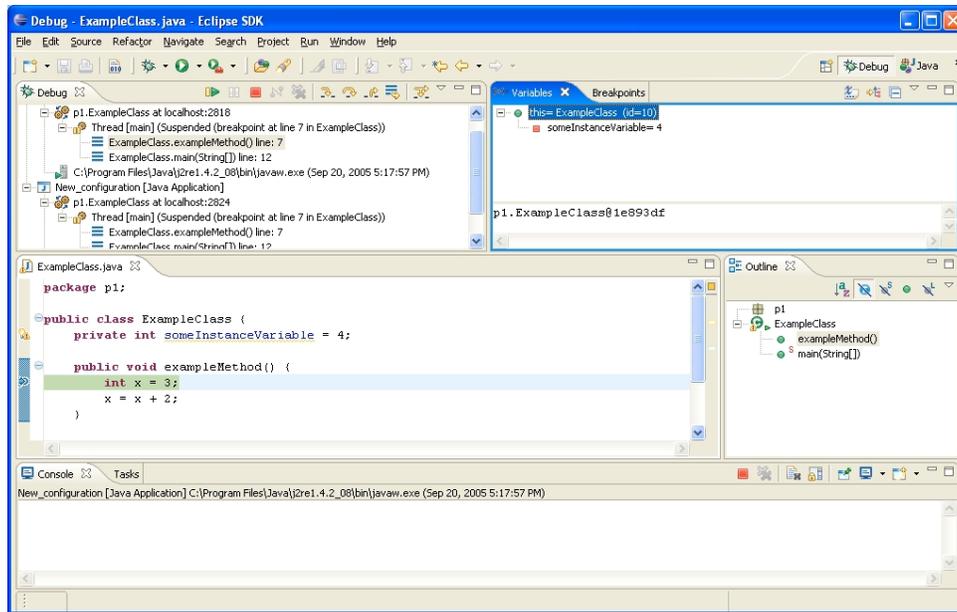
d. In the "Variables" window (upper right of the screen), you can inspect the current object and its instance variables. You can also "step" through your program by using the "Step into", "Step over", or "Step return" buttons ⯮ ⯮ ⯮.

e. To resume execution, press the "Resume" button ▷.

f. To terminate, click the Stop button ◼; then you will also need to remove the Debug environment by clicking "Remove all terminated launches" ✖.