Eclipse Tutorial

Running Eclipse in the microcomputer center:

**Step 1: LOGGING IN**

On the computer screen you should see the login dialog box as shown below. If it is not visible, wiggle the mouse or hit one of the arrow keys to "wake up" the computer.



**Use your E-mail Account to login.** (You may or may not have a USM email account.)

* + Enter your user id in the Username field (e.g. joe.student)
  + Next enter your email password in the Password field.

**To use your Student ID and PIN to login** . (All students should have a Student ID and PIN.)

* + Enter your ID number in the Username field.
  + Next enter your PIN in the Password field. If you have not changed your PIN, then it will be your six-digit birth date. For example, November 3, 1965 would be 110365. Your PIN is the same one that you use to access DSIS and IVR.

**Step 2: STARTING ECLIPSE** (with Windows XP)

CLICK on Start /All Programs/ Programming Applications / Eclipse/ Eclipse (For computers in the lab).  The computers in the lab will not save your work on the hard drive after you log off your current session so remember to have a floppy disk or other storage media handy to save your project (Floppy disks are far from perfect, you could also email your project  to yourself. Sending it as an attachment often works well.)

The first screen that comes up after opening Eclipse is the Workspace Launcher. This screen allows you to choose where the projects you create will be saved. You may use the default as shown below, or specify a location on the hard drive or a removable disk.

.

You should now see this window



**Step 3:** **CREATING A NEW JAVA PROJECT**

###### Click on **File** on the menu bar, choose **New** and go to **Project**... as follows:



        After choosing Project, you will arrive at the following screen:



        In the New Project Wizard, select Java and click **Next**. The following screen allows you to name the project and specify a location to save it.



You must type in the name of the project you want to create. By default Eclipse saves all projects in D:\myworkspace (for computers in the lab). Alternatively, you may uncheck the "Use default" checkbox and specify a location to save it. It is recommended that you save your project to a floppy disk or other storage media, the computers will not save your work to the hard drive when you end your current session.



Once a name is selected and a location to save the project is specified click **Finish** to exit the wizard and create the project. As the wizard creates your project it may ask if you want to switch to the "Java Perspective", click yes to switch perspectives. A perspective in Eclipse defines the initial set and layout of views in the Workbench window. They provide a set of capabilities aimed at accomplishing a specific type of task or working with specific types of resources. We are interested in the Java Perspective which gives us a convenient way to organize Java files and execute Java programs.



**Step 4:** **ADDING FILES TO THE PROJECT**

###### You have successfully started a new Java project, now you need to add files to the project. We will add a Class to the project and execute it. Once again, go to **File**, choose **New** then **Class** as follows

###### 

The following wizard will appear

### 

You will need to enter the name of the new class, lets call it HelloWorld. Also, since the HelloWorld class will contain the main method, check the box labeled "public static void main(String args[])". Now click **Finish**, the wizard creates the new class and puts it in your project folder. The window (workspace) now contains the newly created Class, notice the auto generated comments and class declaration...



As you can see by checking the box labeled "public static void main(String args[])" on the wizard, Eclipse has generated the main method for us. Eclipse also generated comments at the top of the program that identify the creation date and author - you should delete these comments and replace them with your own.

**Step 5: ENTERING JAVA CODE**

Now we are ready to enter the Java code for our simple HelloWorld class. Type or copy and paste the following code into the Eclipse editor. Since some code was generated for you by Eclipse it will be necessary to replace that code with the new code.

/\*  
 Name: <your name>  
 Course: COS 160 - MW 1:15 - 2:30 pm  
 Assignment: Test Program 1  
 File Name: HelloWorld.java  
   
 This program just displays a greeting window.  
 \*/  
   
 **import** javax.swing.JOptionPane;  
 **public class** HelloWorld {  
 **public static void** main(String args[]) {  
 JOptionPane.showMessageDialog(**null**, "Hello World!!");  
 System.exit(0);  
 }  
 }

The comments at the beginning of the program represent a nice format for identifying yourself, the course the assignment is for and the name of the assignment, the name of the file, and a brief description of the program's functionality.

You should now have a file that looks similar to the following...



**Step 6:** **EXECUTING THE PROJECT**

To run the project, go to **Run** on the menu bar and select Run...



The following window will appear.



If  "Java Application" is not highlighted in the Configurations view on the left (as above), select it and click **New**. Eclipse will respond by creating a new configuration and filling in the appropriate fields. The fields that Eclipse fills in are **Name**: the name of the new configuration, **Project**: the name of the project, and **Main class**: the class that contains the main method to execute.



Once all this is done, you can click **Run** and Eclipse will execute the project. You should see a dialog box like the following upon success.



**IMPORTING EXISTING RESOURCES:**

Another way to add files to a project is to import them from a disk, or the local file system. This is done when you already have files you've written and want to incorporate them into the project or have files provided to you by a professor. First, create a new project as described earlier. Go to **File** and click on **Import.** The import wizard appears and gives you many different ways to import resources, we are interested in importing from the file system.



Choose "File System" in the window and click **Next.** The window that appears will allow you to select a folder to retrieve files from and a folder to place files in. Click the **Browse** button to search for the folder containing your files.

### **EXPORTING FILES TO A DISK:**

To save a project to a disk or into a folder in the file system we must Export the project. Lets try exporting the "My First Project" to a floppy disk. The first step is to right click on the project you wish to export in the package explorer. A pop up menu appears, choose **Export...**

****

In the window that comes up choose File System and click **Next**

****

The export wizard appears allowing you to choose the entire project by checking the box on the left hand side next to the name of the project, or only selected files by un-checking the box on the left and choosing the desired files from the right hand side. Once you have decided which files to take click the **Browse** button to specify the location that the project will be exported to.



Scroll through the list of locations and choose the 3 1/2 Floppy Drive, Click **OK.**

****

The Export wizard now has the A:\ drive specified in the "To Directory" text field. Click **Finish** to save all of the files to your floppy disk.



### **DEBUGGING:**

 There are two types of errors that are commonly made when writing a program. The first are syntax errors. These are errors that occur when the syntax of your program is incorrect or inconsistent, for example, a variable is declared as **myVariable**, but later misspelled as **myVarable**. Eclipse finds these types of errors on its own and highlights them by underlining the error in red, here is an example:



Notice that the error is highlighted and in the margin there is a symbol in the margin on the line corresponding to the syntax error that looks like this:



If you click on this symbol, a list of suggested fixes will come up, simply choose the one that fits and click on it. The code will automatically be fixed. If no fixes are listed, read over the code and try to determine what the problem is.

The second type of error is a logical error. These errors are not detected by Eclipse and can be very difficult to locate in the code. Before resorting to the debugger it is wise to read through the code carefully, try to find the error by tracing the logic of your program by hand. Debugging can help but there is no replacement for human thought.

Lets begin by closing any open projects in the workbench by right clicking on an open project in the package explorer and choosing "Close Project" from the pop up menu. Repeat this step for all open projects in the workbench. 

The next step is to start a new project named "Debugging" as described in the section titled [Creating A New Java Project](http://www.cs.usm.maine.edu/~welty/New%20Eclipse%20tutorial/Eclipse_new_Version/new_versionWelty.htm#CREATING). Add a class to the project called "Debug" as was done previously in the [Adding Files To The Project](http://www.cs.usm.maine.edu/~welty/New%20Eclipse%20tutorial/Eclipse_new_Version/new_versionWelty.htm#ADDING%20FILES) section. Your workbench should now be similar to the following:



 Finally, copy and paste the following code into the workbench, replacing all of the existing code.

/\*  
 Name: <your name>  
 Course: COS 160 - MW 1:15 - 2:30 pm  
 Assignment: Test Program 1  
 File Name: Debug.java

This program sums the numbers from 1 to 5 and outputs the   
 results to the console.  
\*/  
  
**public class** Debug {  
  
 **public static void** main(String[] args) {  
  
 **int** myVariable = 0;  
 **int** num = 1;  
  
 **while** (num <= 5) {  
 myVariable += num;  
 num++;  
 }  
  
 System.out.println("The results are: " + myVariable);  
 }  
}

Resulting in the following:



On the line that has "while(num <= 5)" right click in the margin of the workspace and choose Add Breakpoint. You should now have something like the following, (the blue dot indicates a break point in the program).



Before debugging, you must create a new launch configuration that Eclipse uses to run the new project. This was covered in "[Executing The Project](http://www.cs.usm.maine.edu/~welty/New%20Eclipse%20tutorial/Eclipse_new_Version/new_versionWelty.htm#EXECUTING%20THE%20PROJECT)".

The next step is to select the icon on the toolbar that looks like a bug.    Eclipse launches the project and automatically opens the Debug perspective.  The program will run to the point at which you have set your breakpoint, highlight that line, and stop just BEFORE executing that line of code. We will use the "[step over](http://www.cs.usm.maine.edu/~welty/New%20Eclipse%20tutorial/Eclipse_new_Version/new_versionWelty.htm#STEPOVER)" button to debug our small program.



The debugger is very easy to use and can be controlled using the toolbar above the top left window labeled Debug that looks like this:



Some of the useful features:

  resume - resume the current process.

  suspend - "pause" execution of the process.

  terminate - end the program / debugging session - useful when your program is "stuck" in an infinite loop, or you are through debugging.

  disconnect - disconnect from a VM that was connected to with a Remote Java Application launch configuration.

  remove all terminated launches - discontinue displaying old processes that have ended.

  step into - step into a method or class.

  step over - execute the current line of code.

  step return - This command steps out of the current method. This option stops execution after exiting the current method

The upper right window in the debug perspective labeled Variables lists the variables and their values at each step through. Notice the value of num is 1 and myVariable is 0.

You may now step through your program one line at a time to see exactly what each statement is doing. CLICK the step over button to execute the highlighted line of code. The debugger has allowed the program to execute the highlighted line of code and stopped running the program right before the next line of code. Again, CLICK the step over button to execute the highlighted line of code. Notice that in the variable window that the value of myVariable has been highlighted in red and changed to 1. Continue to step over each statement and watch the variables change. This is a very simple example, used only to acquaint you with the basics of the debugger.

Using judiciously placed breakpoints can greatly help in debugging.  You can spend hours stepping through code to no avail if you do not think about what you are doing.  If you are truly lost, first get away from the machine and think.  You can also utilize the tutors or the course instructor for help.  You may think that you want the person helping you to write the program for you, but you don't.  You need help in understanding what is happening.

To restore the Java perspective, go to **Window** on the toolbar choose **Open Perspective** then **Java** this will restore the Java perspective we use for editing Java programs.

### TIPS AND TRICKS

| **Managing screen real estate with fast views** | Use fast views to free up screen real estate while keeping views easily accessible. Clicking on the icon for a fast view temporarily reveals it over top of the other views. The fast view retracts as soon you click outside of it. The **Fast View** command in the view's system menu toggles whether it is a fast view. You can also create a fast view by dragging a view onto the shortcut bar at the left. |
| --- | --- |
| **Opening editors using drag and drop** | You can open an editor on an item by dragging the item from a view like the Navigator or Package Explorer and dropping it over the editor area. |
| **Incrementally find a string** | Use **Edit > Incremental Find Next** (Ctrl+J) or **Edit > Incremental Find Previous** (Ctrl+Shift+J) to enter incremental find mode, and start typing the string to match. Matches are found incrementally as you type. The search string is shown in the status line. Press Ctrl+J or Ctrl+Shift+J to go to the next or previous match. Press Enter or Esc to exit incremental find mode. |
| **Go to last edit location** | **Navigate > Go to Last Edit Location** (Ctrl+Q) takes you back to the place where you last made a change. A button marked  shows up in the toolbar when the Editor Navigation command is enabled. If this command does not appear in your perspective, you can add it by selecting **Window > Customize Perspective > Other > Editor Navigation**. |
| **Importing files** | You can quickly import files and folders into your workspace by dragging them from the file system (e.g., from a Windows Explorer window) and dropping them into the Navigator view. The files and folder are always copied into the project; the originals are not affected. Copy and paste also work. |
| **Exporting files** | Dragging files and folder from the Navigator view to the file system (e.g., to a Windows Explorer window) exports the files and folders. The files and folder are always copied; workspace resources are not affected. Copy and paste also work. |
| **Restoring a perspective's layout** | Rearranging and closing the views in a perspective can sometimes render it unrecognizable and hard to work with. To return it to a familiar state, use **Window > Reset Perspective**. |
| **Importing an existing project** | If you import an existing project, the resources files for the project are *not* copied. If you check the properties of the project, you'll see that the project's location in the file system is the location you specified. |
| **Reordering editor tabs** | You can rearrange the order of open editors by using drag and drop. Grab the editor tab and drag it to the position you want the editor to appear. When positioning editors, the stack icon  indicates a valid spot to drop. |
| **Maximizing a view or editor** | You can maximize a view or editor by double-clicking on the view's title bar or the editor's tab. Double-click again to restore it to its usual size. |
| **Comparing zip archives with each other or with a folder** | Select two zip archives or one archive and a folder in the resource Navigator view and choose **Compare With > Each Other** from the view's popup menu. Any differences between the two inputs are opened in a Compare editor. The top pane shows all the archive entries that differ. Double clicking on an item performs a content compare in the bottom pane.  This works in any context where a file comparison is involved. So if a CVS Synchronize operation lists an archive in the resource tree, you can double click on it in order to drill down into changes within the archive. |