**Module 02 Lab Assignment - Bug Hunt!**

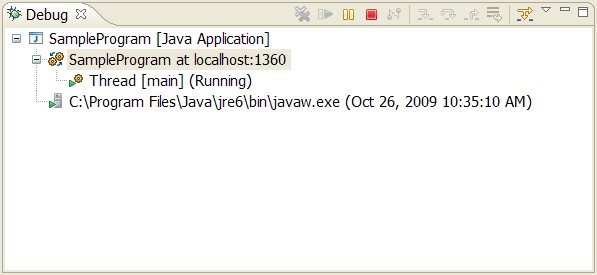
1. Open **SampleProgram.Java** in your IDE. This file has been provided to you.
2. Preview the programming in your debugging perspective.
3. With a **Word** document open, take a screen shot (use the snipping tool) of your open IDE then paste it into the Word document. Label it – **Java IDE Open**. It should look similar to the one above – but not exact. **Name your Word file Module 2**.
4. Switch back to your regular view.

**To change the perspective windows:**

* View the upper-right corner of the Eclipse window and click on the icon: http://www.cs.wisc.edu/~cs302/labs/lab07/labFiles/java_perspective_icon.png
* To the left of that icon is another unlabeled icon that looks like: http://www.cs.wisc.edu/~cs302/labs/lab07/labFiles/open_perspective_icon.png. This is the perspectives icon so click it to open the debug perspective.
* Now you should see and Icon named Debug http://www.cs.wisc.edu/~cs302/labs/lab07/labFiles/debug_perspective_icon.png
* Reverse this process to get back to the main window.

**Working with Suspending, Resuming and Terminating a Program**

1. With the **SampleProgram** open, select **Debug** as and then select **Java** application. You should see that sample program is printing output to the **Console** window.
2. Switch to the **Debug** **perspective**, and then find the window labeled **Debug** in the upper left of Eclipse.



In the Debug window at the top is a toolbar of icons that includes three that we'll use: http://www.cs.wisc.edu/~cs302/labs/lab07/labFiles/resume_suspend_terminate_icons.png. These icons are the Resume, Suspend and Terminate buttons (from left to right).

1. Try the **Suspend** button (looks like a pause button), and notice its effect on the program. **Take a screen shot and add it to the Word document**.
2. Now press the **Resume** button (a yellow bar and a green arrow) and see how it effects the program.
3. Try suspending and resuming a few times.
4. Finally, press the **Terminate** button (**red square**) to end the program.

**This is how you use the program’s general controls over the debugger.**

In the Debug perspective, look over to the left and you will notice the grayish vertical bar at the far left of that window. To set a break point, you simply double click this bar at the line where you want to stop the program.

1. Set a break point at the System.out.println().

Note: You will see a bulling point appearing at the bar indicating that you've set a breakpoint at this line of code (circled in red below):

http://www.cs.wisc.edu/~cs302/labs/lab07/labFiles/breakpoint_icon.jpg

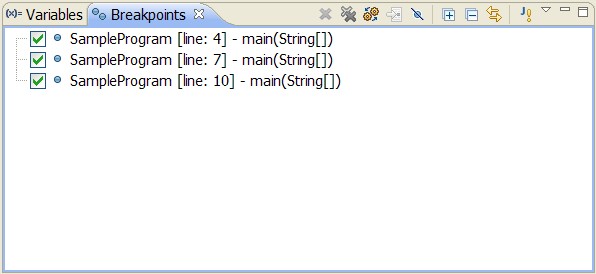
1. **Take a screen shot and add it to your Word document – calling it break point**.
2. Run the program in debug mode again (Run → Debug As → Java Application), and you'll see that the program stops at the breakpoint.

Note: the line will be highlighted in green and the icon in the bar changes to show that you've suspended at that breakpoint (circled in red below):

http://www.cs.wisc.edu/~cs302/labs/lab07/labFiles/breakpoint_icon2.jpg

1. Insert another breakpoint at the method call to **innerLoop**() method
2. Insert another breakpoint at the final call to **System.out.println**()
3. **Take a screen shot and add it to the Word file with a paragraph describing what you think it is doing**.

Find the Breakpoints window in the upper right of the Eclipse window, it looks like the screen shot below. Be aware you may have to click on the Breakpoints tab in order to display.



The three breakpoints you have set should display in this window and it will provide you with information about each. It can include line number, class name, method name, etc. and it is designed for you to be able to double click on the entry and it will take you to that line within your code. For small amounts of code, you may not need this but for larger amounts, it may be easier for you to utilize this feature then to scroll through the lines to find it.

To disable breakpoints, you simply uncheck them in the Breakpoints window or by right-clicking on its listing in the Breakpoints window and the selecting Disable.

1. Disable the **innerloop**, and then **Resume** execution of your program and notice the effect.
2. **Take a screen shot and add it to the Word document**.

**Stepping Control**

Once we've stopped at a breakpoint, often you'll want to be able to control the execution of the subsequent lines of code. The debugger has buttons that provide us "Step Into" , "Step Over" and "Step Return" control that look like: http://www.cs.wisc.edu/~cs302/labs/lab07/labFiles/step.png. They are located in the tool bar at the top of the debug window to the right of the buttons we used to resume and suspend execution.

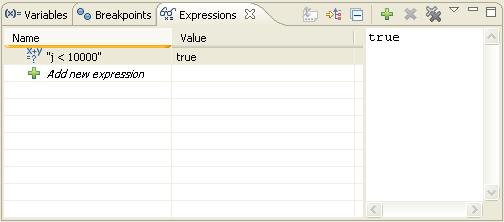
1. Try working with all three. **Take a screen shot of onethen add to the Word document with an** **explain the differences in what you are seeing from one to the other**.

**Evaluating Expressions**

Conditional expressions can allow you to view how expressions can change.

1. Open the **Expressions** window by selecting the menu **Window → Show View and select Expressions**.
2. Now highlight and copy (ctrl-C) in the source code displayed in the editor window the expression j < 10000 in the for loop of the **innerLoop** method.
3. Right click on the **Expressions window** and click on **Add Watch Expression**.
4. **Paste** (ctrl-V) the copied expression into the text area of the pop-up window that appears and click **Ok**.
5. Step through your program until your expression widow matches what is shown below.
6. **Take a screen shot and add it to your Word document**.

Here's what the Expressions window looks like:



**Fixing Buggy String Methods using the Eclipse Debugger**

**StringBugs.java:**

Run the above-referenced program and watch how it behaves. You could probably debug it by view but use the methods above to find the bugs.

1. Set up break points
2. Step through the code
3. Correct any bugs that you find while you are stepping through the code.
4. **Screen shot the errors you find and add them to the Word file**.
5. **Screen shot the fully functioning program and add it to the Word file**.
6. While working on the **reverseString** method use the debugger to view the swap array contents by clicking on the plus symbol next to the array's name in the **Variables** window.

**Guess2.java:**

Run the above-referenced program and watch how it behaves. You could probably debug it by view but use the methods above to find the bugs. These bugs will be harder to find then the previous.

1. Take a moment to determine how you wish to approach debugging this program.
2. Utilize the variety of tools that have been discussed in this lesson.
3. **Take screen shots of the errors you found and add them to the Word document**.
4. **Take screen shots of the final working code**.
5. **Include what features of Eclipse's debugger you use to find the errors in the Word document**.

**BuggyQuilt:**

Run the above-referenced program and watch how it behaves. You could probably debug it by view but use the methods above to find the bugs. These bugs will be harder to find then the previous.

1. Take a moment to determine how you wish to approach debugging this program.
2. Utilize the variety of tools that have been discussed in this lesson.
3. **Take screen shots of the errors you found and add them to the Word document**.
4. **Take screen shots of the final working code**.
5. **Include what features of Eclipse's debugger you use to find the errors in the Word document**.