M. Roggenbach, CS-135 – Lab Class 5 – 12.3.

- **To be solved in groups of two.**
- **To be submitted to blackboard by Monday, 19.3., 11am.**
- **Submission: Every student individually, one pdf file.**
- **Please mark clearly with whom you are working together: there should be 2 student numbers on top of your submissions.**
- **No other formats than .pdf will be accepted!!!**

□ Task 5.1

The purpose of this task is to use and understand the basic debugging features in Eclipse. To this end, you will write a short guide explaining how to use the debugger.

Download the `DebuggerGuideTemplateDocument.doc` file from
`http://www.cs.swan.ac.uk/~csmarkus/Tools/`.

This file is a word document that provides a template for the user guide for this lab. Your job for this task is to understand the debugger and fill out this template as you go (replace the "..." with appropriate text). As you perform each experiment below, you will gain knowledge that you can add to the user guide. You may be as creative as you like with the user guide.

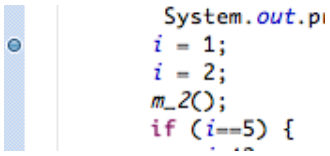
Experiment 1 part i: Download the program `Debug.java` from
`http://www.cs.swan.ac.uk/~csmarkus/Tools/`.

Setup a new project in Eclipse, import the java file, and run it.

Experiment 1 part ii: Start the debugger via the pull-down menu "Run", by choosing the option "Debug as" (or by clicking the debug button in the toolbar).

Effect: Your program should run as in the usual mode.

Now you set a "breakpoint" in one of the methods by double clicking at the left of your program code. When you have set the breakpoint then you can see it as a dot in the left column:



Run the debugger again.

Effect: The program runs up to the breakpoint.

Experiment 1 part iii: You set a breakpoint left to the line `i = 13`.

Question 1: Does the breakpoint interrupt before the line is executed, or after the line has been executed? (You should provide this information in your user guide).

Hint: In order to answer this question, you might want to add the "Variable view" by choosing in the "Window" menu the option "Show View", and within this the item "Variables".

Experiment 1 part iv: Set some extra breakpoints on the lines `i = 12` and `i = 14`. Consider the green "resume button" – at the left of the picture below:

Question 2: What does the resume button do? (You should provide this information in your user guide).

Hint: you can now fill out the Section 2 "BreakPoints" in your user guide (except for the part on breakpoints with conditions – That part will be covered a little later).

Experiment 2a: It is now time to step through code. Clear all your breakpoints. Place a breakpoint on the line `i = m(42);`. Run the debugger. You can now Use the yellow "arrow" buttons ("step into", "step over", "step return") at the right of the picture above to step through the code. Understand what the three buttons do.

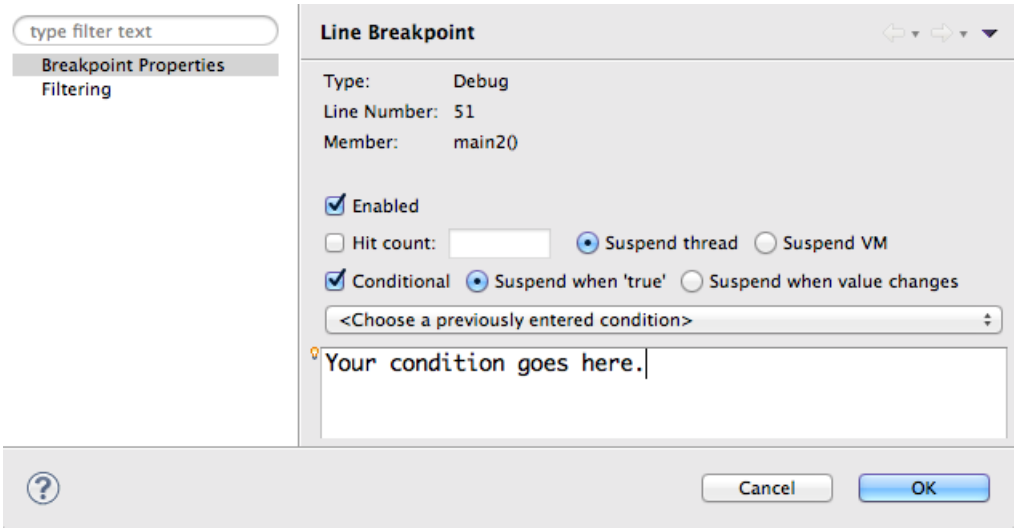
Now clear the breakpoint and set a new one on the line `myMethod();` Try use the step buttons again to step through the code.

Question 3: What is the difference between the behaviour of the three step buttons now? (You should provide this information in your user guide)

Experiment 2b: Clear your breakpoint and move it to the line `i = f(g(42));`.

Question 4: How do the three step buttons behave now? (You should provide this information in your user guide)

Experiment 3: Clear all breakpoints and set a new breakpoint on the line `z = (z + i * 4) % 345;`. We can add conditions to the break point. Right click it and choose "Break Point properties". Add a suitable condition so that you pause execution when the variable `i` has a value of 3000. The image below shows what options should be selected.



Question 5: What is the value of variable `z` when variable `i` has a value of 3000? (This information probably does **not** belong in your user guide, but it will help you fill out the part about conditional break points.)

Submission: Your user guide for debugging.

□ Task 5.2

Debugging an Index Function

Download the program `Search1.java` from
`http://www.cs.swan.ac.uk/~csmarkus/Tools/`. Create a new project in Eclipse, import the java file and, run it. Verify that the program gives a wrong result without the `break` statement, however, runs correctly with the `break` statement.

Consider the program without the `break` statement.

Think how

- to set break points and
- to use the "step into", "step over", "step return" buttons

in order find out why the program does not work correctly without the `break` statement.

Question: Explain the steps that you performed to find out that the program does not work correctly without the `break` statement. Your answer should explain how you systematically found the bug.

Submission: Your answer to the above Question (5 lines of text).

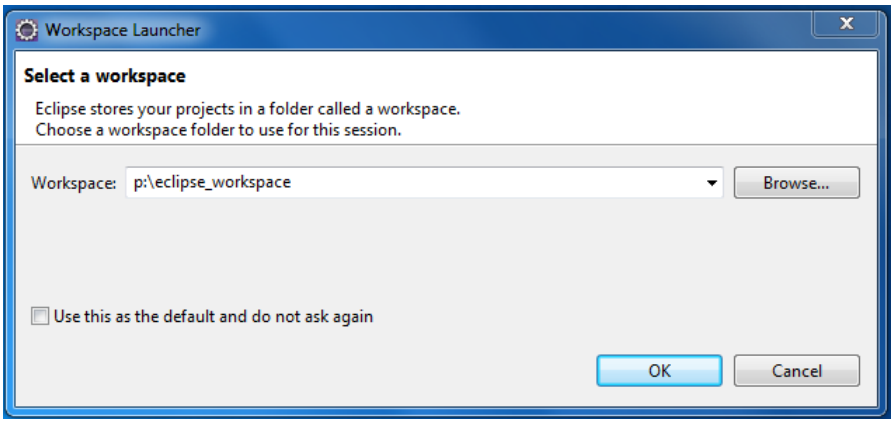
Computer Instructions

1 Making a screen-shot

Click on 'Start', type 'Snipping Tool' in the search field, press 'enter'. Use the tool.

2 Eclipse

Under the "Specialist Apps", open the folder "College of Science". Within this folder, open the folder "Computer Science". There, you find the program "Eclipse". When you start Eclipse you might be asked for the workspace path. This path should be set as follows:



2.1 Making a new project

1. Click `File` → `New` → `Project` → `Java Project`.
2. Typing a good project name i.e. `Sphinx`.
3. Click `Finish`.

2.2 Importing a file into a project

1. Expand your project, say `Sphinx` in the left hand panel (Package Explorer),
2. Right click the `src` folder, click import.
3. Select `File System` under `General`, click `Next`.
4. Locate the directory containing the `Sphinx.java` file, click `OK`.
5. Check the file, e.g. `Sphinx.java`, in the right hand list, Click `Finish`.

2.3 Running a program

You run a program, e.g., `Sphinx.java`, by clicking the play icon. This may bring up a wizard where you need to select to run a `Java Application`. You may need to show the `Console` view by clicking `Window` → `Show View` → `Console`.

2.4 Activating JUnit4 for a project

1. Right-click on your project and select `Properties`.
2. Click on `Java Build Path`.
3. Select `Libraries`
4. Select `Add Library`.
5. Select `Junit`.
6. Click on next, select the Junit Version `JUnit 4`.
7. Click `Finish.`
8. Click `OK.`