M. Roggenbach, CS-135 – Lab Class 5 – 2.3.

- To be solved in groups of two.
- To be ticked off in one of the labs of your house on Monday, 2.3., or Monday, 9.3.
- For being ticked off on Monday, 9.3., you need to have your solution ready at the begin of the lab class.
- You can obtain two marks by solving this sheet.
- Each completed task gives you one mark.
- All group participants need to be present to be ticked off.

This lab is about Debugging.

The purpose of this lab is to get some experience with the Eclipse Debugger.

Note that there are computer instruction at the end of this lab sheet.

☐ Task 5.1

The purpose of this task is to use and understand the basic debugging features in Eclipse.

Experiment 1 part i: Download the program `Debug.java` from

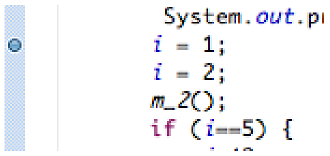
`http://www.cs.swan.ac.uk/~csmarkus/Tools/`.

Set up 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:



```
System.out.p
i = 1;
i = 2;
m_2();
if (i==5) {
```

Run the debugger again.

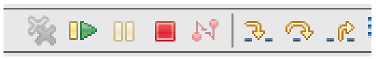
Effect: The program runs up to the breakpoint.

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?

Screen shot 1: Make a screen shot in the "Variable view", showing break point and and the value of the variable. To this end choose 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?

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. 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. Observe that "step return" is not always available.

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

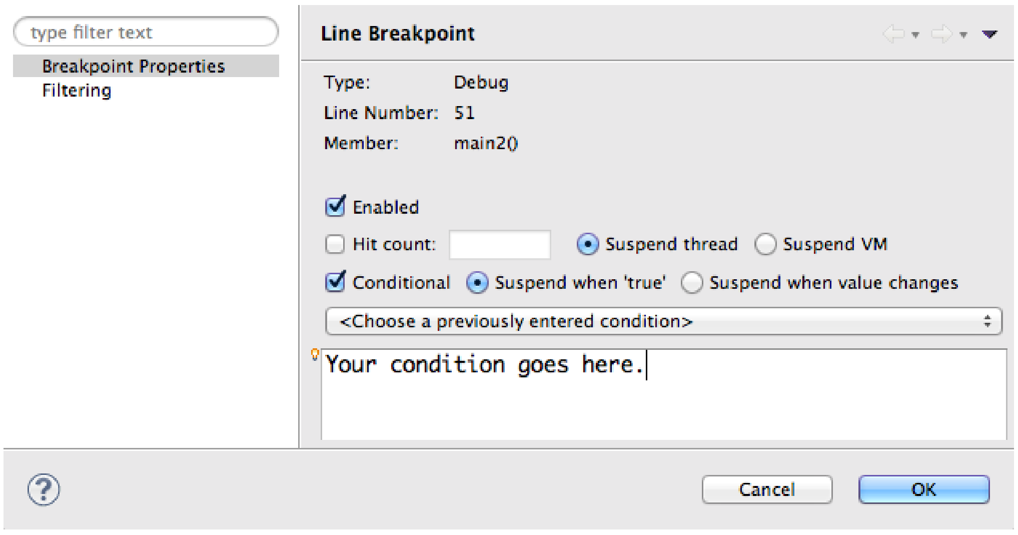
Experiment 2b: Clear your breakpoint and move it to the line `i = f(g(42));`.

Question 3: What is the meaning of the three step buttons?

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 – such as in an if statement or a while loop – so that you pause execution when the variable `i` has a value of 3000. The image below shows what options should be selected.



Line Breakpoint

Type: Debug
Line Number: 51
Member: main2()

[✓] Enabled
[ ] Hit count: (●) Suspend thread ( ) Suspend VM
[✓] Conditional (●) Suspend when 'true' ( ) Suspend when value changes
<Choose a previously entered condition>

Your condition goes here.

Screen shot 2: Make a screen shot in the "Variable view", showing break point, and the value of the variable. To this end choose in the "Window" menu the option "Show View", and within this the item "Variables".

Material to show when getting ticked off: Screen shots 1 & 2; answer Questions 1 – 3.

☐ 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.

The method `search` shall solve the following computational problem:

Index:

Input:	Array *Sequence*; integer *value*
Output:	the smallest index i such that $Sequence[i] = value$ if *Sequence* includes value -1 otherwise

Think how in order find out why the program does not work correctly.

Material to show when getting ticked off: A screen shot demonstrating how you used the debugger (be prepared to explain why you took this specific screen shot); the corrected code of `Search1.java`.

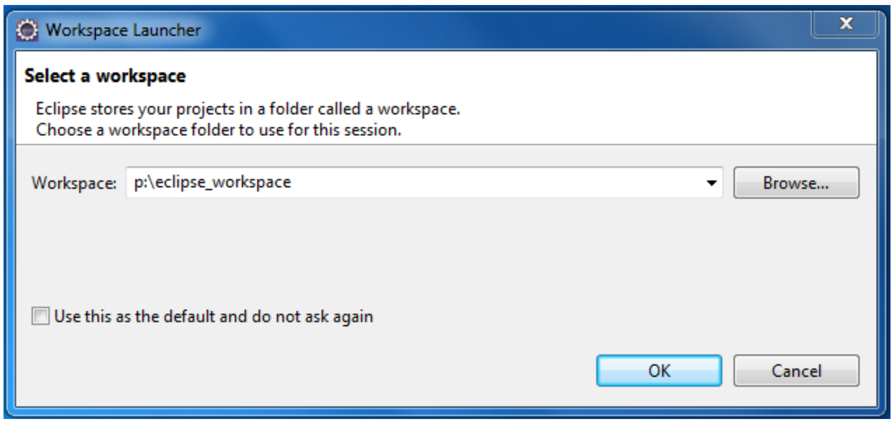
Computer Instructions

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:



Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: p:\eclipse_workspace

Use this as the default and do not ask again

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.`