The Java Debugger – A Short Guide

1. Getting Started

In order to invoke the debug mode in Eclipse, we ... add a linebreak and select run > debug.

To show the variables of interest, we ... select windows > variables and view the variables window.

2. Breakpoints

A breakpoint is set by double clicking on the line number on the left hand side and can be removed by double clicking on it again.

In the debug mode, a program runs as usual until it reaches a break point where it will pause and allow the user to run through the program step by step.

When program execution stops at a breakpoint, Eclipse highlights the state before the execution of the line.

If execution has paused during debugging, the resume button will continue the program to the end or the next breakpoint.

Breakpoints can have conditions attached. Such a condition can be entered by right clicking the break point, selecting “breakpoint properties” and writing the condition code in the box.

A breakpoint with a condition has the effect that it will pause when the condition is met. It is useful when testing if code ever reaches a specific condition.

3. Stepping Through Code

There are three buttons that can be used to step through code. These are “step into”, “step over” and “step return”.

The main difference between them is their behaviour when executing methods.

Assume you have a “simple” method call, .e.g.,

```
i = m(42);
```

or

```
myMethod();
```

with a breakpoint on the line. The three buttons behave as follows:

- The first behaves as a step a method to be run, so if used on myMethod it will step to the code in the method instead of just running the method.
- The second behaves as an advance button, executing the current line of code then highlighting the next. If the line is a method call it will run the method without stepping into it.
- The third is not available. Now assume that you press the first button, then the third button becomes available. Pressing the third button will now return you to the line prior to pressing the first button.

Assume you have a "complex" method call, e.g.,

```
i = f(g(42));
```

with a breakpoint on the line. This line contains *method calls* to the methods f and g. The *method declarations* of f and g are defined elsewhere in the source code, they especially include the *method body* of f and g, respectively.

The three buttons now have more complex behaviour, following the evaluation order that Java defines for expressions: First evaluate method g, and then evaluate method f (using the returned value from g as a parameter).

- The second behaves as usual, executing the line and moving on
- The first moves the focus into the first method called which is g, then it will return to the method and then step into the second method; f.
- The third is not available. Now assume that you press the first button, and then the third button becomes available. Pressing the third button will now return to the line where the method is called in the main method. Now you can press the first button again which will move the focus into the second method called. Finally, pressing the third button again, will return to the method call in the main method having executed all method calls.
- In general, the third button when available, will skip ahead executing the rest of the method and return the pointer to the method call in the main method.