

Debug with Eclipse

Cody Henrichsen
03/04/15

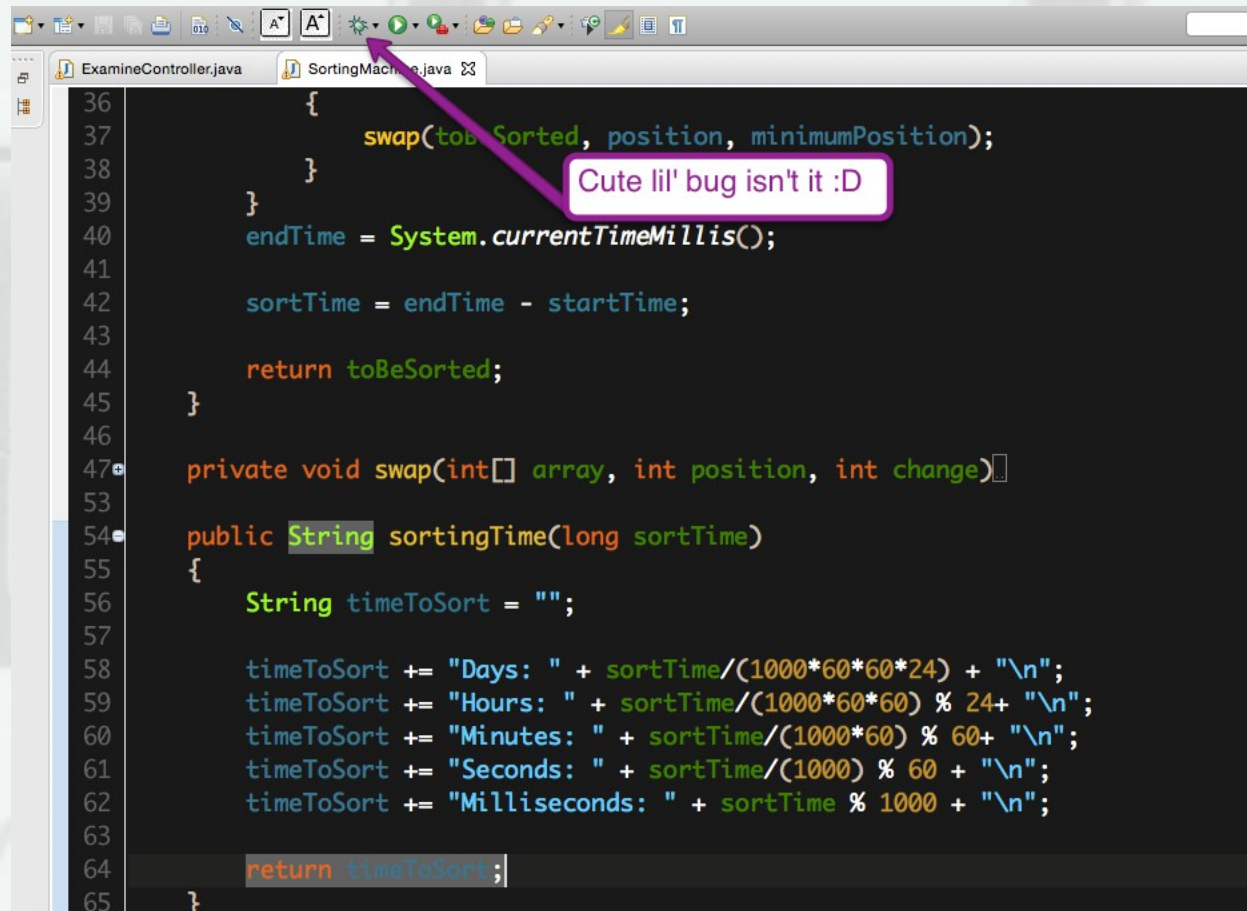
Debug Process

- Build project - :P
- Run project and find where you want to start looking
 - Generally where it breaks or causes errors
- Set a breakpoint there
 - Look at the console window for the line number

Debug Process

- Debug project
- Step-Into lines/methods you want to examine closely
- Step-Over lines/methods you have already resolved
- Step-Return to get back to where this line was called

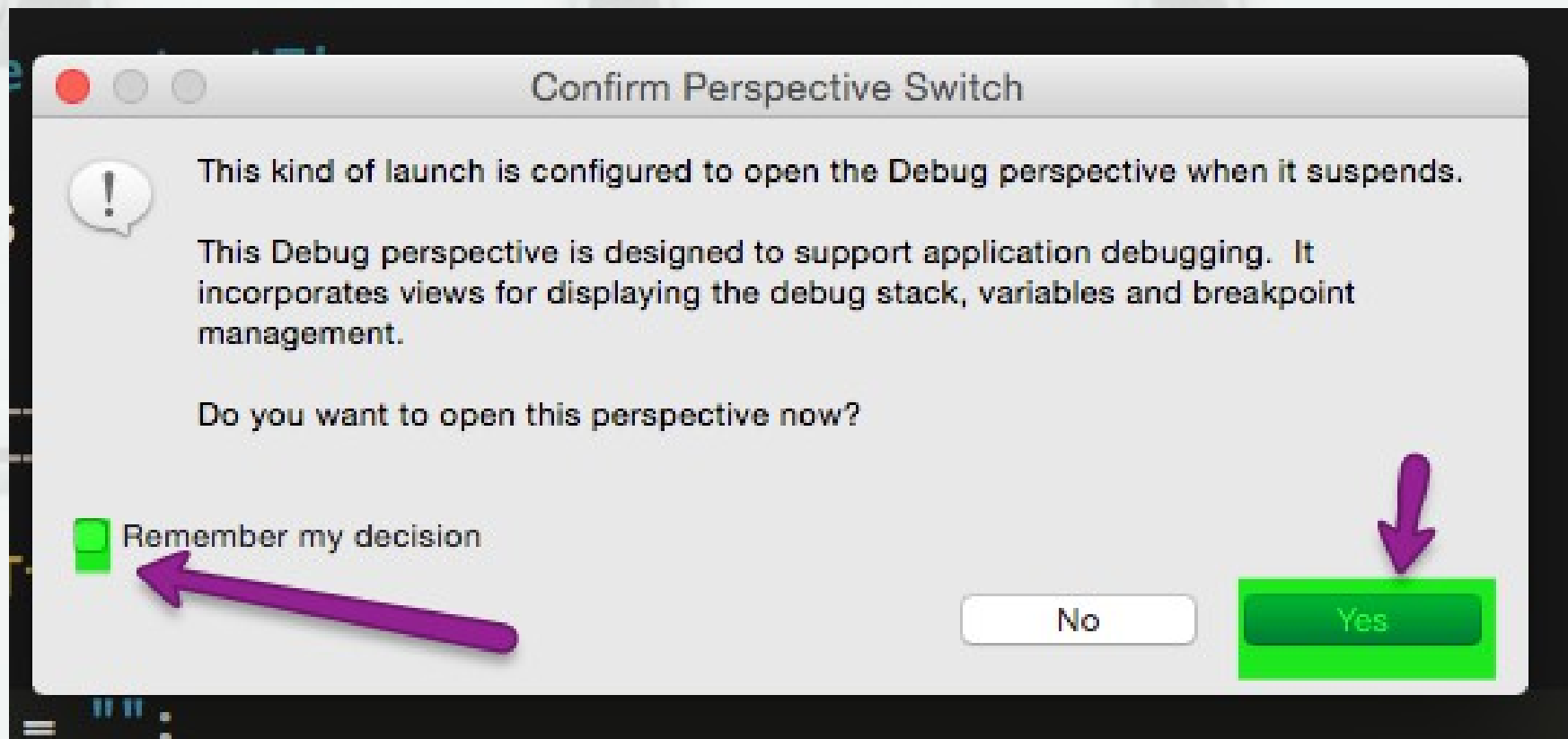
Debug



The screenshot shows an IDE window with two tabs: 'ExamineController.java' and 'SortingMachine.java'. The 'SortingMachine.java' tab is active, displaying Java code. A purple arrow points to the debug icon (a green bug) in the IDE's toolbar. A purple text box with the text 'Cute lil' bug isn't it :D' is positioned next to the arrow. The code in the editor is as follows:

```
36     {
37         swap(toBeSorted, position, minimumPosition);
38     }
39 }
40 endTime = System.currentTimeMillis();
41
42 sortTime = endTime - startTime;
43
44 return toBeSorted;
45 }
46
47 private void swap(int[] array, int position, int change) {
48
49 }
50
51 public String sortingTime(long sortTime)
52 {
53     String timeToSort = "";
54
55     timeToSort += "Days: " + sortTime/(1000*60*60*24) + "\n";
56     timeToSort += "Hours: " + sortTime/(1000*60*60) % 24 + "\n";
57     timeToSort += "Minutes: " + sortTime/(1000*60) % 60 + "\n";
58     timeToSort += "Seconds: " + sortTime/(1000) % 60 + "\n";
59     timeToSort += "Milliseconds: " + sortTime % 1000 + "\n";
60
61     return timeToSort;
62 }
```


Debug Reminder



Debug Commands

The screenshot displays an IDE interface with several panels. The top toolbar contains various icons, with a green box highlighting the 'Debug' icon. Below this, a purple box labeled 'debug commands' points to the 'Debug' tab in the left sidebar. The 'Debug' tab shows a list of threads, with 'Thread [main] (Suspended (breakpoint at line 56 in SortingMachine))' selected. The 'Variables' panel on the right shows the current state of variables: 'this' is 'SortingMachine (id=17)' and 'sortTime' is '0'. The 'Outline' panel on the right shows the class hierarchy, with 'SortingMachine' selected. The 'LogCat' panel at the bottom left shows a table of log messages. The 'Console' panel at the bottom right shows the output of the application.

debug commands

LookInsideRunner [Java Application]

- examine.controller.LookInsideRunner at localhost:54619
 - Thread [main] (Suspended (breakpoint at line 56 in SortingMachine))
 - SortingMachine.sortTime(long) line: 56
 - LookInsideController.start() line: 21
 - LookInsideRunner.main(String[]) line: 9

/Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (Mar 4, 2015, 8:43:34 AM)

Variables

Name	Value
this	SortingMachine (id=17)
sortTime	0

Outline

- examine.model
 - SortingMachine
 - startTime : long
 - endTime : long
 - sortTime : long
 - getSortTime() : long
 - selectionSort(int[]) : int[]
 - swap(int[], int, int) : void
 - sortingTime(long) : String
 - bogoSort(int[], int[]) : int[]
 - shuffle(int[]) : void
 - isSorted(int[], int[]) : boolean

LogCat

Saved Filters

All messages (n)

Le Time	PID	TID	Application	Tag
---------	-----	-----	-------------	-----

Console

LookInsideRunner [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (Mar 4, 2015, 8:43:34 AM)

Writable Smart Insert 56 : 1

Debug Perspective

The screenshot displays the Eclipse IDE in the Debug Perspective. The interface is divided into several panels:

- Debug Console:** Shows the execution stack. The top frame is `LookInsideRunner [Java Application]`. Below it is `examine.controller.LookInsideRunner at localhost:54619`. The current thread is `Thread [main] (Suspended (breakpoint at line 56 in SortingMachine))`. The stack trace includes:
 - `SortingMachine.sortingTime(long) line: 56`
 - `LookInsideController.start() line: 21`
 - `LookInsideRunner.main(String[]) line: 9`
- Variables View:** Displays the current state of variables. The variable `this` is of type `SortingMachine` with value `(id=17)`. The variable `sortTime` is of type `long` with value `0`.
- Source Editor:** Shows the source code of `SortingMachine.java`. The current line is 56, which is the `sortingTime` method. The code is as follows:

```
36 {
37     swap(toBeSorted, position, minimumPosition);
38 }
39 }
40 endTime = System.currentTimeMillis();
41
42 sortTime = endTime - startTime;
43
44 return toBeSorted;
45 }
46 }
```
- Outline View:** Shows the project structure. The `SortingMachine` class is selected. The methods listed are:
 - `start() : void`
 - `endTime : long`
 - `sortTime : long`
 - `getSortTime() : long`
 - `selectionSort(int[]) : int[]`
 - `swap(int[], int, int) : void`
 - `sortingTime(long) : String` (selected)
 - `bogoSort(int[], int[]) : int[]`
 - `shuffle(int[]) : void`
 - `isSorted(int[], int[]) : boolean`
- LogCat View:** Shows the log messages. The filter is set to `verbose`. The log messages are as follows:

Level	Time	PID	TID	Application	Tag
Verbose	2015-03-04 08:43:34 AM	1234	1234	LookInsideRunner	main
- Console View:** Shows the output of the application. The output is as follows:

```
LookInsideRunner [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (Mar 4, 2015, 8:43:34 AM)
```


Stacktrace

The screenshot displays an IDE interface with several panels. The top-left panel, titled 'Debug', shows a stacktrace for a Java application. The stacktrace is as follows:

```
LookInsideRunner [Java Application]
  examine.controller.LookInsideRunner at localhost:54619
    Thread (main) (Suspended (breakpoint at line 56 in SortingMachine))
      SortingMachine.sortTime(long) line: 56
        LookInsideController.start() line: 21
          LookInsideRunner.main(String[]) line: 9
            /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (Mar 4, 2015, 8:43:34 AM)
```

A purple box with the text 'Stacktrace of the current program' is overlaid on the stacktrace.

The bottom-left panel shows the source code of 'SortingMachine.java' at line 40:

```
36
37     swap(toBeSorted, position, minimumPosition);
38
39 }
40 endTime = System.currentTimeMillis();
41
42 sortTime = endTime - startTime;
43
44 return toBeSorted;
45
46 }
```

The bottom-right panel shows the 'Outline' view with a list of methods in the 'SortingMachine' class:

- startTime : long
- endTime : long
- sortTime : long
- getSortTime() : long
- selectionSort(int[]) : int[]
- swap(int[], int, int) : void
- sortTime(long) : String
- bogoSort(int[], int[]) : int[]
- shuffle(int[]) : void
- isSorted(int[], int[]) : boolean

The bottom-most panel is the 'LogCat' view, which is currently empty.

Breakpoints

- This is what you set to find a specific spot to go to when debugging the project
- If you press resume it will go to the next breakpoint or occurrence of this breakpoint if in a loop
- Double click to the left of the line number to set or remove a breakpoint
 - There must be code there for a breakpoint

Variables

The screenshot displays an IDE interface with several components:

- Variables window (top right):** A purple box highlights this window, which shows the current state of variables. It contains a table with two columns: Name and Value.
- Code Editor (middle):** Displays the source code for `SortingMachine.java`. The visible code includes a `swap` method call, a `System.currentTimeMillis()` call, and a calculation for `sortTime`.
- LogCat (bottom left):** Shows a table of log messages with columns for Level, Time, PID, TID, Application, and Tag.
- Console (bottom right):** Displays the output of the application, showing the path to the Java application.

Name	Value
this	SortingMachine (id=17)
sortTime	0

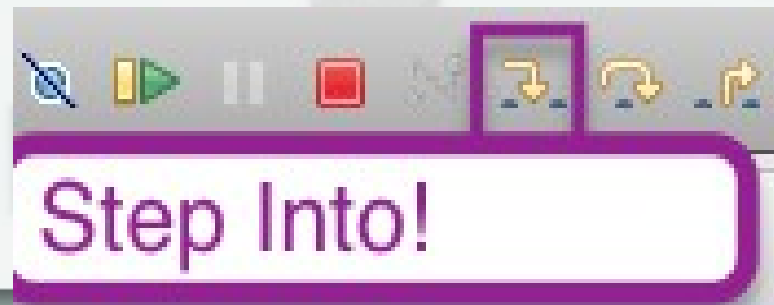
Level	Time	PID	TID	Application	Tag
-------	------	-----	-----	-------------	-----

Variables Window

- Check the value of the components within the program
- Very useful to see what happens at every step of the program
- Use this to see where a logic error occurs
- Use this to see why a NullPointerException occurs

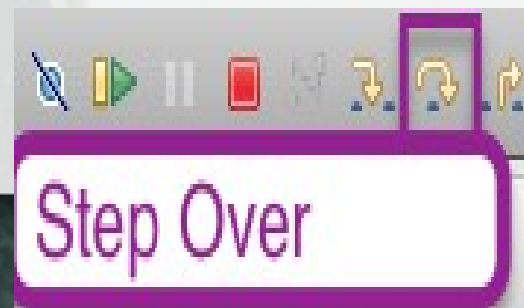
Step Into

- Check every step of this line of code
- This is what most people think of for debugging code
- Look at each line of code and watch the value of the variables in the variables window



Step Over

- Do this one line and stop
- If you know that this is not needed to check but part of a larger sequence
 - Calling a getter
 - Assigning a simple value
 - You have already gone through that method entirely



Step Return

- Finish running the current method and go back to where it was called
- Further proof that one giant method is a bad idea
- Useful especially if you want to continue debugging just not this specific method anymore

