

# Re: Setting breakpoint on the end of the method in Eclipse

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2007-05/msg00572.html>

---

- *From:* Lew <[lew@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:lew@xxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 08 May 2007 09:23:00 -0400
- 

dt wrote:

Thanks Patricia, this worked. Two things – this is "indirect", as you said. It will certainly help me do what I wanted (my case is simple enough for the above technique to be applied), but it still sucks

Well, that certainly is an opinion.

There is no instruction on '}' so there is no place to stop. You whine that the IDE does not add a "nop" (is there even such an instruction in the JVM?). I would whine if the IDE changed my code. I prefer breakpoints to exist only where there is a place to stop, so /my/ opinion is that it does not suck.

and this will not let you do the other thing – stop on the end of the while loop. To make it a little clearer:

```
void test() {
boolean a = true;
while(a) {
if(otherMethodReturningBoolean())
a = false;
// other code here
boolean a = true;
while(b) {
if(otherMethodReturningBoolean())
b = false;
} // <---- Breakpoint here
} // <---- Breakpoint here
} // <---- Breakpoint here
```

Consider the two extra breakpoint possibilities (excluding method end). How would you debug the end of while(b), while still staying in while(a) iteration?

Put the breakpoint on the "while ( b )" line.

## Re: Setting breakpoint on the end of the method in Eclipse

Right before the first instruction of a loop is the same spot as right after the last.

You get one extra stop, which can be eliminated with a conditional breakpoint if it bothers you that much.

You will also need to set a breakpoint on the first instruction after the loop, which in your case is the "while ( a )" instruction.

With the breakpoint on method exit you now have everything you asked for. It no longer sucks, does it?

--

Lew

.