I appreciate this tool, it fits naturally to the debugging process, i'm definitely using this from now on.

I don't find the question "I would prefer *name of the debugging technique* to the traditional debugger" a valid question. Object centric debugging still relies and only complements traditional debugging. e.g. I still need to add a traditional breakpoint to enable object centric debugging, or I still use normal stepping operations once I hit a breakpoint.

---

a bit confused by the exercise scenarios
what helped most is the inspect menu in ammolite (getting at an arbitrary domain object)
For ammolite I saw the wrong value in the inspector. For warmup 1 it showed that = was not called on the personOfInterest (so it's not included in the array). So beyond that it was mostly normal debugging and while I used halt on halt on access and halt on call to understand the code, they didn't really pinpoint the bug.
For ammolite I would have removed the space from the data string, but I guess that's cheating.

I find myself falling back to other ways of problem solving a lot. That will take some time to get used to

---

While i understood the concept I found the first example with lightOn not to be an obvious one for object centric debugging - I wanted to set a conditional breakpoint for setting the colour attribute to grey for all instances - and this isn't available as an option (I don't understand why? Given I didn't know which insance was likey to go wrong, it wasn't obvious where set an object centric breakpoint, so I wanted to set a write condition on color - I think this is an obvious omission. Writing now - I guess I could have inspected all 4 corners and set a write breakpoint for all 4 instances but that seems wasteful. In essence, I didn't find it a compelling example? I also find it deeply annoying that in the Pharo debugger you can't add new breakpoints while debugging - why? (not an object centric issue though). I also found that object centric breakpoints don't always seem to work if you set one on an instance and then press restart in the debugger? The persons example with the do loop - when I got a test failure I set an OC breakpoint, pressed restart on the test (in the debugger) and it didn't work. I had to set a breakpoint before the do loop (and coldn't do that in the debugger - although realised I could have restarted and stepped up to the point I needed and then set an OC breakpoiint - but it seems like a bug?

No specific comment. Was easy to follow

As I am not an experimented Pharo users, it took a bit more time than what was intended. The experiment was fun and the exercices were clear. I may have issues understanding the debugger (which steps get a certain method called in a loop for example), that's why it took me this much time.

I probably missedfilled the first form, because it didn't scroll as expected, my appologies.
Cliffnotes are that it was tedious to find without OCDdebugger.
I had to find a discriminating factor, to be able to basically do OCD debugging.

I had issues triggering the object-centric debugging breakpoints. While they seemed to work in the tutorial, I could only trigger sometimes in the warmup and led to an error message in the ammolite example.

However, I enjoyed the process and would like to use objcdg more!!

---

i didn't understand well the 2nd task to complete... i didn't locate the good object with the problem... i tried to solve a problem that was not a problem... when i have identified the correct problem, it was easy to locate and solve the problem... the time pressure added by the lunch didnt permit to solve it before the lunch... but immediately at the return i found a correction granted by the efficiency of the tools...

Not really a comment on the experiment, but I have a few minor remarks regarding the ergonomics of Pharo which I find a bit annoying (note that I use Pharo very rarely so maybe I have missed some settings, or some other things that I am not familiar with) :

- the closing button of the window on the left side while it is on the right side in my OS (Ubuntu 23.10) which almost lead me to close the whole pharo window several times.
- the fact that you need to click on the title bar of a window to put it in the foreground.
- the fact that some windows (I think it only happened with the window of the programs (ammolite and light one) get behind menu bar, making it difficult to move it.
- In the debugging window, there is no elevator for the methods in the meta tab, making it difficult to see all the methods and select the one you want to debug (e.g. the registerOn: method in the warm up exercise)

I'm now coding in Pharo for around 1 year, so I'm still learning what some methods do and what's Pharo's behaviour in different situations, so I believe this may have impacted my performance negatively in the experiment. Besides that, I understood how the features you suggested work, even though I feel my lack of knowledge in Pharo in general made me had trouble applying your feature to solve real problems.

One of the most enjoyable and well-documented experiments that I have participated in!
Thank you

I'm not the sure the tasks (including the last one) really highlight the features of OCD.

I had one crash

I liked the experiment, with the lights out introduction and tutorial. I'm surprised this tool existed and I don't use it... and I debug everyday! thanks for your work.

- I did put breakoints on state read/access and it froze the system
- I feel that finding objects of interest uses very traditional "put break, inspect", Could there be more ways?

It seems pretty easy to forget all the breakpoints added during the experiment. It would be nice to have some way to quickly get a reminder for such state.

I think OCDeb would help me a lot with some bugs, but I did not find it that usefull for the last exercice of the experience. I used it once then the normal debugging was better. But I can see how it can help me more with other bugs.

I had one crash when performing steps in the LightsOut task.

My image freezed when putting object-centric breakpoints (read or state-access) on the `marker` instance variable of `AMStudent`.