This file compiles all experiment feedback that reports specific difficulties on the treatment tasks, using the object-centric breakpoints. It also adds informal feedback (personal messages or emails sent to researchers just after performing the experiment) reporting similar information.

In general

> 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?

> 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'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.

From informal feedback

> tu ne sais pas si tu as déjà rencontré le bug quand tu trouves l'objet

tu ne sais pas quel objet est la source parfois (exemple, on avait un bug graphique avec plein de petits composants visuels imbriqués et on ne savait pas le responsable)

My biggest pain was that I didn't find a good reason to setup object centric breakpoint for case, where object was incorrectly setup during initialization of the object. E.g. When I did exercise from OCD experiment, the error came from early initialization of object (when object was created from some other class iterating over stream). So halt on call, or halt on write was never triggered, since change of inst var did happen during initialization and breakpoint was never invoked again.

With Lights Out task as treatment

First of all - if break is related to object. I need to see hash of this object (in order to visually verify that object is on given instance). Biggest trouble for me was that bug occured during initialization, didn't find the way, when state of this object is wrongly set. When you restart the entire app/or game, breakpoint is no longer valid, because instance is different (new obect is created). So maybe I missed whole point, when to setup "halt on state access". When instance is created and it has wrong name (or color in case of LoCell), i didn't find a way, how OCD can be helpful. Maybe it is just lack of my experience. Also confusing was, when I closed game or test, where instance occured, it didn't remove from breakpoint navigator. It needs to be garbage colllected (image cleanup) and then it is no longer visible. I would really like to see the video, how procedure should be done ideally using OCD.

The operating instructions for object centric debugging failed to inform that the breakpoint on the object instance must be inserted each time we execute the code. This is counterintuitive, as developers are used to adding the breakpoint once and running the code several times to exercise and analyze the piece of code with the breakpoint. The breakpoint functionality on the instance is useful when we have objects of the same class in a collection of objects. This idea works well in Pharo because like the LightsOutGame example, we can inspect the object instance in just one click. However, in traditional languages such as C# and Java, it would be necessary to debug in a traditional way until the object is created, and then add the breakpoint to the instance, it would be very laborious and perhaps not very efficient. In these cases of collections of objects, in traditional debugging tools, we have conditional breakpoint functionality that normally meet this need. For Pharo, I suggest an improvement: making it not necessary to re-add the breakpoint in the new instance of the same object once the previous instance is lost. This may be possible by creating a history of object instances for the class once the instance breakpoint is added to an instance.

Other think to consider is that when you run a test, or open a windows other different objects may be created. So, the debugger will not jump because is other object. So, I need to be able to inspect the object before the method that cause the bad beharvior is executed, and then hope that the break point works in the same execution. Otherwise, a second execution may create another objects that I need to tag again.

I was surprised at the Lights Out task, as OCD does not particularly help there. Once you have the object in question, it is too late to set any breakpoint as the damage has been done. Unless I am missing something, you need to use other techniques to locate the dirty method.

for the last part I got stuck because of a bug which meant that sometimes my 4 sides were colored and sometimes not, whereas there should always have been only 3 of them colored.

I tend to look at bugs as "fault -> error -> failure" where a failure is a service that has failed, the cause of the failure is fault (bad instructions in the code). An error is the manifestation of the fault. I see OCDbug as giving more help to find the objects wher an "error" has occurred. Sadly, the complexity of black-box frameworks (like morph) doesn't really make debugging much easier even if we know the object. The last experiment seemed to have a non-deterministic bug, and that is really hard to trace back to the framework (or it was for me). That is, I suspected onColor was being corrupted, but didn't really have time to go back and find out when it was being called on the 100 squares at initialization (or wherever), especially when that item was in the morphic framework.

Cool experiment.
About the last task: because the bug occurs at initalization of the game, it's not easy to use object-centric debugger.
The 'wrong' state is already here when one can install a breakpoint., so breakpoints only indicate that it *does not* break.
This is an information of course, but then, when one understands this, the object-centric is not useful anymore and one needs to switch to standard debugging and static analysis.

With Ammolite task as treatment

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

- 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?

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`.