Assignment: CSE503 HW1
Name: Volodymyr Loyko
NetID: vloyko

Hours spent: 2

Problem 1 (Testing)

I ran into this problem while grading CSE331 assignments last quarter so while it may not be strictly a problem that I discovered while developing on my own, I still think it is relevant and related. To give some background, one of the requirements for their testing suites was that all of the methods that they declare have at least one test attached to them, and in the best case more than one covering various edge conditions. To keep track of this requirement I saw lots of students applying the following approach: in their test class they define a large full line spanning comment with the methods name that labels all of the following tests as belonging to that method and provides a visual separator from the rest of the tests. Other approaches included simply having all of the tests unlabeled with only the name indicating which method they belong to, or not including separate tests for each method but rather performing multiple scenarios with assertion for each in one test, which was just wrong.

I think the underlying issue here is something I have encountered on my own when writing tests during my internship for already developed code(I know, horrible, but what could I do?). It is that there is no obvious and clean way to discover all of the tests along with the conditions that they cover that belong to a single method in a class. Checking whether a method had tests attached to it would involve finding the corresponding testing class and then finding the tests that were relevant through name search.

A quick search showed that there are some ways, specifically in IntelliJ IDEA to run tests based on a pattern supplied, which I assume means that it would allow to run the tests via a name pattern, but this does not solve my problem. What I think would be helpful and have the smallest learning curve and highest usability is just a info tab when hovered over a method declaration that tells you the attached tests and their conditions and oracles.

I think there are a few reasons why something like this is not done or at least is not popular if it has been done. One is the general issue of adding plugins to an IDE like Eclipse or IntelliJ IDEA. I think most students and casual developers do not like installing extra plugins to their IDE's. If that is overcome though, another more technical issue is that something like this would need your tests to be organized in a certain way and follow a specific format so the info tips could be derived. This in itself could undo the usefulness of the solution I mentioned. Nevertheless I think there is a good way of doing this out there.

Problem 2 (Working together)

Pair programming is not great and one of the reasons is that it's hard to not just sit back and watch what the person in front of you is writing while your mind goes off on some other tangent. I think making you watch somebody write code for more than three minutes is a crime. To be able to write code together in a collaborative style is a great idea but I think it could be improved upon but performing something similar to what Google Docs is doing. There are ways

out there right now to write code together, for example Visual Studio's Live Share. It does almost what I would want but not quite. The problem there is that both people are working on a live document and that puts pressure on you to write code that does not break the other person's document. You could use version control to work on your separate versions and then merge things together but it takes too long to see what the other person is working on and merging later could be a nightmare because of that.

My solution to this is essentially allowing you to see and comment on live what the other developer is working on while that code being completely separate from yours. This way you could run your code without worrying whether your partners code is ready yet still see what they are planning on doing or what their current state is. This would allow you to also work on avoiding big merge conflicts in the future is you can adapt live or tell your partner that the code that their trying to add does not work with yours.

I think VS Live Share comes close to what I want but misses those few ideas. The reason I think is because they want it to be truly collaborative experience which is different from what I am pitching, which is more of a live update on what the other developer is doing with the file both of you are working on. I think the Live Share should have a mode that allows for switching to the behavior that I described and back to normal.

Problem 3 (Reading Code)

I really enjoy the way Jupiter Notebook presents code and I think other programming languages or IDE's should allow for similar approaches. Reading comments in Java code could be a hit or miss experience. Sometime they are overly descriptive for just a single line of code where you feel it would've been better understood if it was shown with graphics or better edited text. Sometimes you are not sure whether a comment refers to just the next line, a whole loop, or somewhere in-between. Some comments are simple facts to keep you straight as you are reading code and some are more conceptual descriptions of the behaviors implemented. Things like JavaDocs attempt to put some comments into a different category with strict guidelines, which I think works really well. But anything outside of that seems to be just whatever developer comes up with. Especially frustrating for me personally are the ASCII diagrams and pictures that are included in the comments.

A solution for this I think would be to define a more general commenting guideline with abilities to embed images, etc. Something similar to LateX commands that can be included in the comments and later read by the IDE and interpreted as not just blocks of text, but text with attached styles and maybe even diagrams.

Currently JavaDoc does actually allow for attachment of images, which is fantastic and I did not know about this before writing this, yet I think a more general solution for all of the comments in you Java files needs to exist and be included with your choice of an IDE.