SWEN225 – Self Reflection

Niraj Gandhi – 300564849

Description	*Commit URL*	*Issue URL*
This is where I implemented saving of the Maze object to an XML file using the JDOM2 library.	*https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t16/chaps/-/commit/eb0fd6cfb5b3af45318bb86ebf4113f6458e7eea*	*https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t16/chaps/-/merge_requests/60*
This is where I implemented the loading logic of loading the XML save file into the game and returning a Maze object for APP to use.	*https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t16/chaps/-/issues/30*	*https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t16/chaps/-/commit/139ac0d1b26906945286b3fe889e5f6c1f87778e*
Here I created many unit tests for my modules, reaching 85% branch coverage and 100% class coverage, and having various tests aimed towards the main components of the program.	*https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t16/chaps/-/commit/12ba55aae9f359141599d215efb07d618746a67f*	*https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t16/chaps/-/issues/63*

This project was fully managed by the use of creating issues, associating a branch to the issue and then each commit having a dedicated server compile and run check style and spot bugs to ensure the commits are fitting the requirements. Each merge request to the main branch was approved by two appointed group members of which checked to ensure that each merge was not causing game breaking issues.

In this process some commits may have lost tracking data due to GitLab by default deleting branches when making a merge request that gets approved. The newer branches intentionally disabled this feature; however this is why some of the initial branches are hard to find even though all tracking data showing that the commits were made by me are still available within GitLab.

Team Contribution Score Board

Name	***Email***	***Score***
Niraj Gandhi (myself)	***gandhinira@ecs.vuw.ac.nz***	***5***
Sam Redmon	***redmonsam@ecs.vuw.ac.nz***	***5***
Lawrence Schwabe	***schwablawr@ecs.vuw.ac.nz***	***5***
Jonty Morris	***morrisjont@ecs.vuw.ac.nz***	***5***
Shae West	***westshae@ecs.vuw.ac.nz***	***4***
George North	***northgeor@ecs.vuw.ac.nz***	***3***

Reflection

What knowledge and/or experience have I gained?

One of the main takeaways from this project was the use of git, especially in a team based environment. I always had a basic understanding of git and how it worked, as I had used it on personal projects although more so as a form of file hosting with GitHub instead of fully utilizing the features it provides.

With this project I have learned how to use git, and GitLab which is the service we used for this project. The features we made use of were raising issues, committing, rebasing, merging, creating merge requests and using pipelines to automate and ensure that all code being merged into main is quality code that has been checked over for spotbug and check style errors.

What were the major challenges, and how did we solve them?

I think one of the major challenges was that most of us had never really done a project like this before where essentially, we could do anything and implement everything anyway that we wanted to implement it. We were not restricted to a templated assignment and got to develop the code from the ground up. This was challenging because it poses problems such as “Where do we start?” however we managed to work around this by joining regular Discord calls and talking through

the various things we would need to do, this helped immensely as we were able to put multiple minds towards the problem and assign everyone's task.

Another challenge was that a couple members weren't active with us, we all had a plethora of assignments from other courses due around the same time which caused some "last minute" work from us, however in these times we worked diligently and efficiently. However still in this case as reflected by my scoreboard, some members hardly responded or participated in group meetings, making it very difficult for other members to work around their assigned modules which were required to run the game.

Which technologies and methods worked for me and the team, and which didn't, and why?

I personally used IntelliJ for the assignment, this is particularly because I am very familiar with it. Others in the group such as Sam used Eclipse with Jonty and Lawrence using VSCode. These IDE's worked for us simply because we had become familiar with them causing us to be more efficient with our tasks.

We made use of maven as a way to manage all the libraries we needed, which worked well and efficiently.

Gitlabs issues was a great feature, we ended up having two members of our group appointed towards reviewing each merge request, ensuring that it did not cause errors. This process worked well and is commonplace in a real working environment, however one of the problems with this was that by default GitLab deleted the old branches, essentially making it very difficult to show our commit counts accurately, and with the two members reviewing having their commit counts be inflated due to this.

Another technology that we used was a pipeline within gitlab, essentially we had a VPS server check our code for compile errors, spotbug errors, checkstyle errors and in turn maven errors. These checks were made each time a commit was made and a merge request to main could not be made unless these conditions were

made, in the end this made our code very uniform tidy. This is something that was definitely helpful and something that I will be using again in future projects.

However overall the method that we used “Issue -> branch -> commit -> merge” worked very efficiently, and realistically made the project manageable with a group of people working on it, if we were all just working on the main branch, it would become a mess very quickly and we would not have been able to work that well as a team.

Discuss how you used one particular design pattern or code contract in your module. What were the pros and cons of using the design pattern or contract in the context of this project?

A design pattern that I have used in my module, was the factory pattern. This pattern allows to generate a document object file that has been parsed.

This essentially allows any module to make use of the function by providing an XML file with the factory generating a document object which can then be traversed.

This design was also prevalent within my actor generation functions, with the use of three functions generating and returning an object. We can see this is the case with “loadCustomActorClass()” where it generates and returns a class based on the custom actor class, which is then used by “newCustomActor()”

The design pattern / contract wasn’t that significant in my module, simply due to the nature of my module being more utilitarian with functions that complete a specific task, the other design patterns available to my use did not apply significantly in the context of the persistency module or atleast was not within the scope of the current projects persistency module. Other modules such as the domain or app made more uses of these patterns as they greatly benefitted from them. For example the domain made use of the Facade structural design pattern with the use of a Tile interface which greatly simplifies the program as a whole and allows other classes such as my module (persistency.java) make use of the Tile object to generate and cast to classes that implement the interface. So the

usage of design patterns in this context can only be seen as a pro as it removes the need for unnecessary code.

What would I do differently if I had to do this project again?

If I had to do this project again, I would create the unit tests alongside development of the functions. This is simply because in that moment I will know the different potential points of failure that should be targeted within the tests.

I would have also modularized the code a little bit more. For example, having a function to read in the inventory and having the function return an array of inventory items.

In terms of management, I think I would have liked to set deadlines for everyone to follow, to ensure we have certain aspects of our program completed before hand so that we are not rushing to get everything done last minute.

What should the team do differently if we had to do this project again?

What we would do differently is mainly manage our time better and have better communication with all members of the team. This refers to the people that weren't very present in our meetings and in the end caused us to not have a functioning program due to their modules not being fully developed.

Elsewise I think the team could use GitLab even more effectively as we are now all very confident in how it works, unlike previously where we were quite new to the whole concept.

Lastly I believe, even though there were uses of design patterns, the code could have been modularized a bit more to ensure that other classes could potentially use the functions rather than repeating lines of code. The use of more design patterns in this case would create a better basis for our code to work upon alongside take advantage of dynamic dispatch to make my code and the teams code as whole as compact as possible.