FIT3077: Software Engineering: Architecture & Design

Nine Men's Morris - Sprint 4

Revised User Stories, Final Game, Architecture and Design Rationales, and Video Demonstration

---

Group: CL_Monday6pm_Team38

Team Members: *Rebekah Fullard, Nikola Mutic, Joshua Van Der Veen*

New advanced requirement:

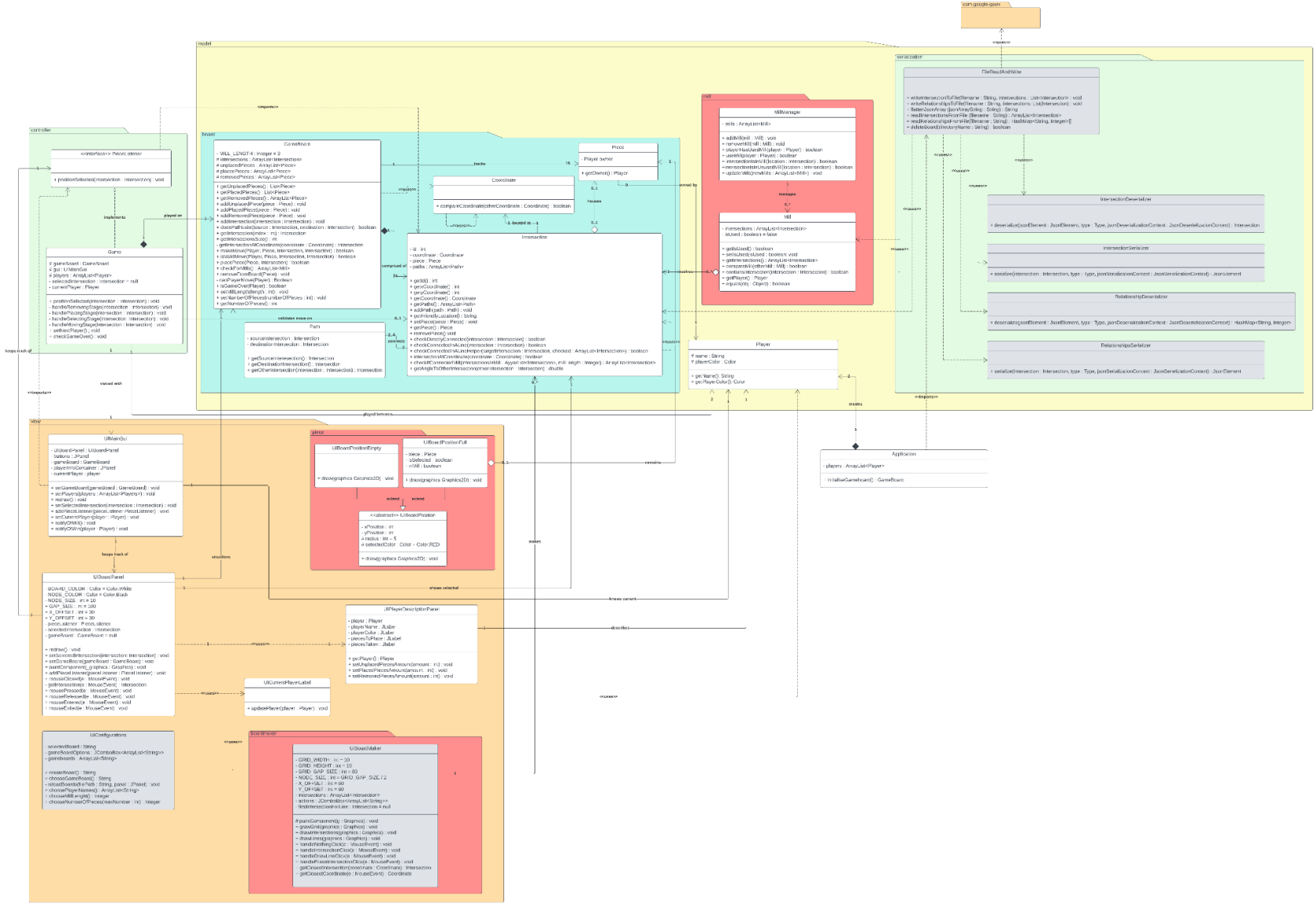
Variable size/shape gameboard. The user can create gameboards of a custom size, with custom placement and connection of intersections, as well as the number of pieces in a row required to form a Mill. They can also choose the number of players and the number of pieces that each player starts the game with.

Revised user stories (for advanced requirement):

- As a user I want to be able to create a custom gameboard so that I can challenge myself.
- As a user I want to be able to save my custom gameboards so that I can easily use the same gameboard more than once.
- As the gameboard I want to be able to identify Mills on any gameboard at any orientation and with any set of relationships between intersections so that this rule can be enforced on any custom gameboard.
- As a player I want to be able to set the number of pieces that are required to form a Mill so that I can explore different in-game challenges.
- As a player I want to be able to play against more than 2 players so that more of my friends can play Nine Men's Morris at the same time.
- As a user I want to be able to delete gameboards that I no longer use so that it is easier for me to find the gameboard I want to use.
- As a user I want to be able to choose the number of pieces that each player starts with so that I can start with a number of pieces appropriate to the size of the gameboard.
- As the gameboard I want to validate the set number of starting pieces so that players cannot start a game that is impossible to successfully play.

Class diagram

A high quality version of this image can be found in the repository/zip in PNG format in the docs/sprint4 folder.
Please note new classes implemented in this Sprint have been coloured grey.



Design rationales

Explain why you have designed the architecture for the advanced requirement the way that you have.

We designed the architecture to be as dynamic as possible to allow for freedom in the configuration and play of the game. The main things we needed to consider was the drawing of the board, the mill checking and the movement. We used different design methods including recursion and a node graph layout so that these things would be future proof, and would work regardless of the board shape or size.

Explain why you have revised the architecture.

We decided to add the ability to save and load boards to file as an extra requirement in our advanced feature to the game. This meant that we had to create some extra classes/packages in order to implement this functionality. We created a boards folder with 2 json files for storing intersections and paths, and the reason for this is to avoid circular dependency on itself since the path stores the intersections and the intersections store the paths. This meant that our application would convert it into deserialised classes that java can read.

This change meant that we had to refactor the board loading stage of the application, but it opened up many possibilities in saving, loading and creating boards.

We planned ahead in previous sprints, and this meant that we didn't have to completely refactor the project to enable the validation of mill checking and moving pieces around.

Explain when your advanced feature was finalised and how easy/difficult it was to implement.

- The advanced feature was finalised as we worked on Sprint 3. During the time that we were finalising the design of the game we decided to focus on ensuring that the design was dynamic and allowed for future expansions of the game (e.g. different size/shape gameboards). It was from this that we discussed the idea of implementing an advanced feature that provided different gameboards from the traditional board layout. These discussions eventually culminated in the decision to allow users to create and save their own custom gameboards - along with other customisation features such as number of players, number of starting pieces, and number of pieces required in a line to form a Mill.
- We had planned ahead of time for our advanced requirements so as to avoid having to rewrite logic. Surprisingly, we had very little code to rewrite to get it to work with our advanced feature. Our function for mill checking did not

need to change at all from the last sprint. Due to the fact that we used a graph data structure, our algorithm that worked on horizontal and vertical mill checking worked fine on dynamic boards with diagonal mill checking. The only things we needed to change were implementing a new way of alternating between players (since we had a simple solution for the last sprint where there were only two players).

Video Demonstration

Please find our video demonstration submitted through Moodle.

How to create and run an executable

1. Open the repository in Intellij
2. To build the executable, press Build > Build Artifacts > project.jar > Build
3. Then run the file at ***out\artifacts\project_jar\project.jar***
4. *We have compiled using JDK 19. To run in terminal, follow these exact steps*
    a. *ensure you have javac version 19.*
    b. *verify using 'javac -version' command*
    c. *example output: `javac 19.0.2`*
    d. *ensure that the project.jar file is in project/ directory*
    e. *run the program using `java -jar project.jar`*
5. *Alternatively, you can run it through intellij by right clicking and pressing run.*

Notes

- SomeGithubNoob is Josh Van Der Veen's personal Github account, and was accidentally not configured at the start.
- We did many 'Code with Me' sessions which is Intellij's built in collaboration tool, and because of that some of the team members didn't have as many commits. The commits were created by the person who was hosting the 'code with me' session, however we had a great amount of collaboration during the implementation of Sprint 4.
- Built with JRE 19 which is needed to run the executable
- Must be run in root directory otherwise it will not work because of the boards storage and recollection