FIT3077: Software Engineering: Architecture & Design

Nine Men's Morris - Sprint 3

Tech-based Basic Software Prototype, Architecture and Design Rationales, and Video Demonstration

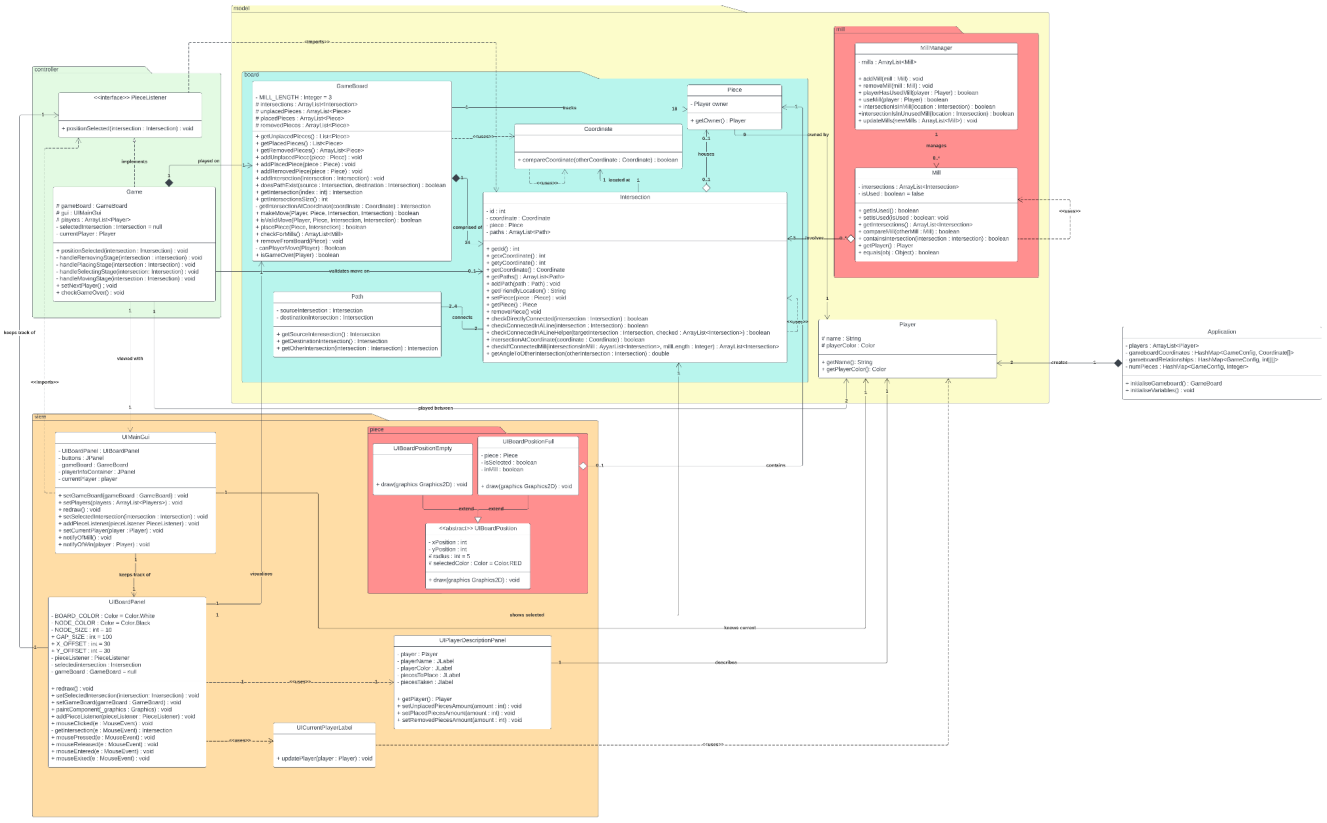
---

Group: CL_Monday6pm_Team38
Team Members: *Rebekah Fullard, Nikola Mutic, Joshua Van Der Veen*

Architecture and Design Rationales

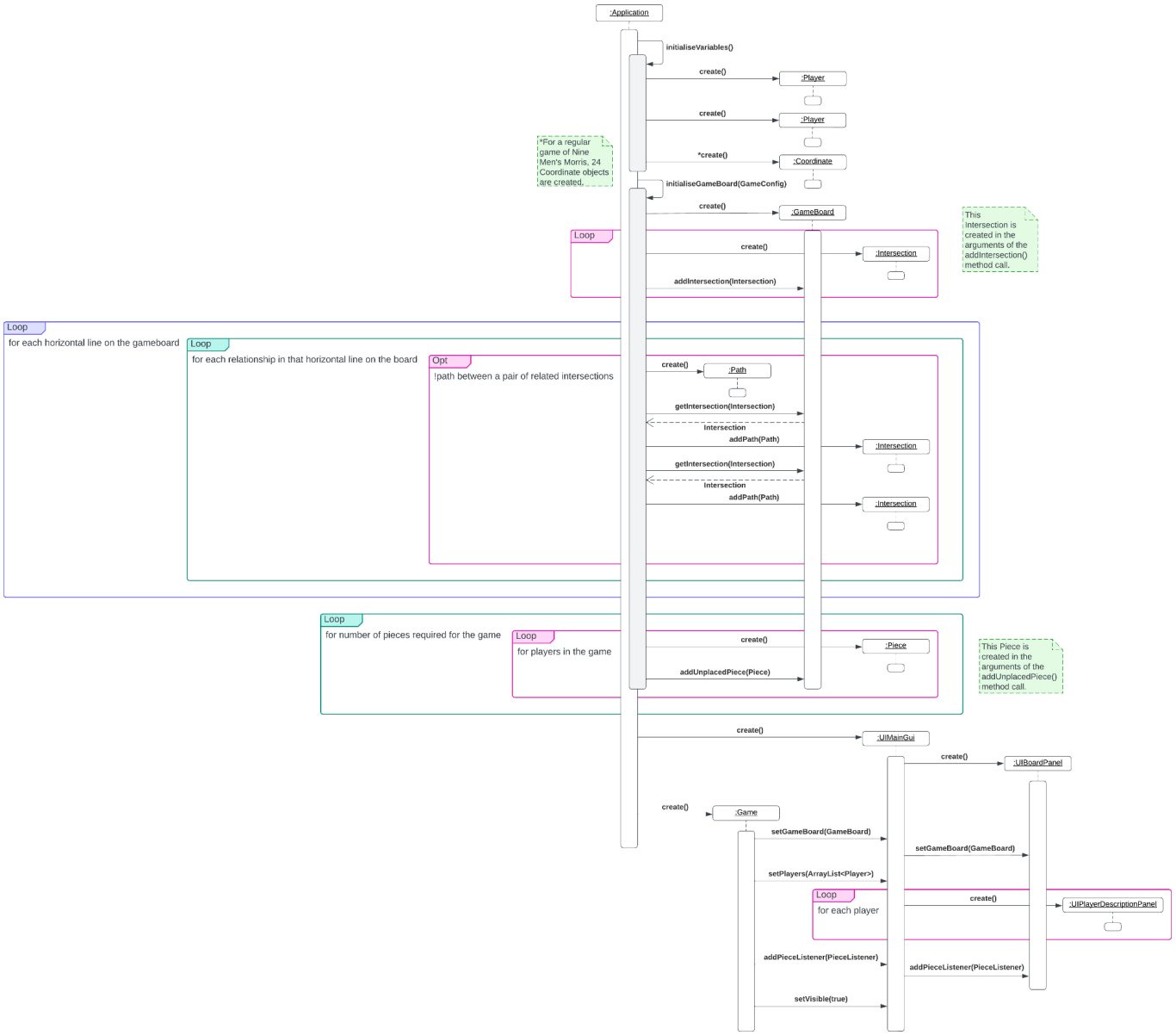
High quality versions of each image can be found in the repository/zip in PNG format in the docs/sprint3 folder.

Class Diagram

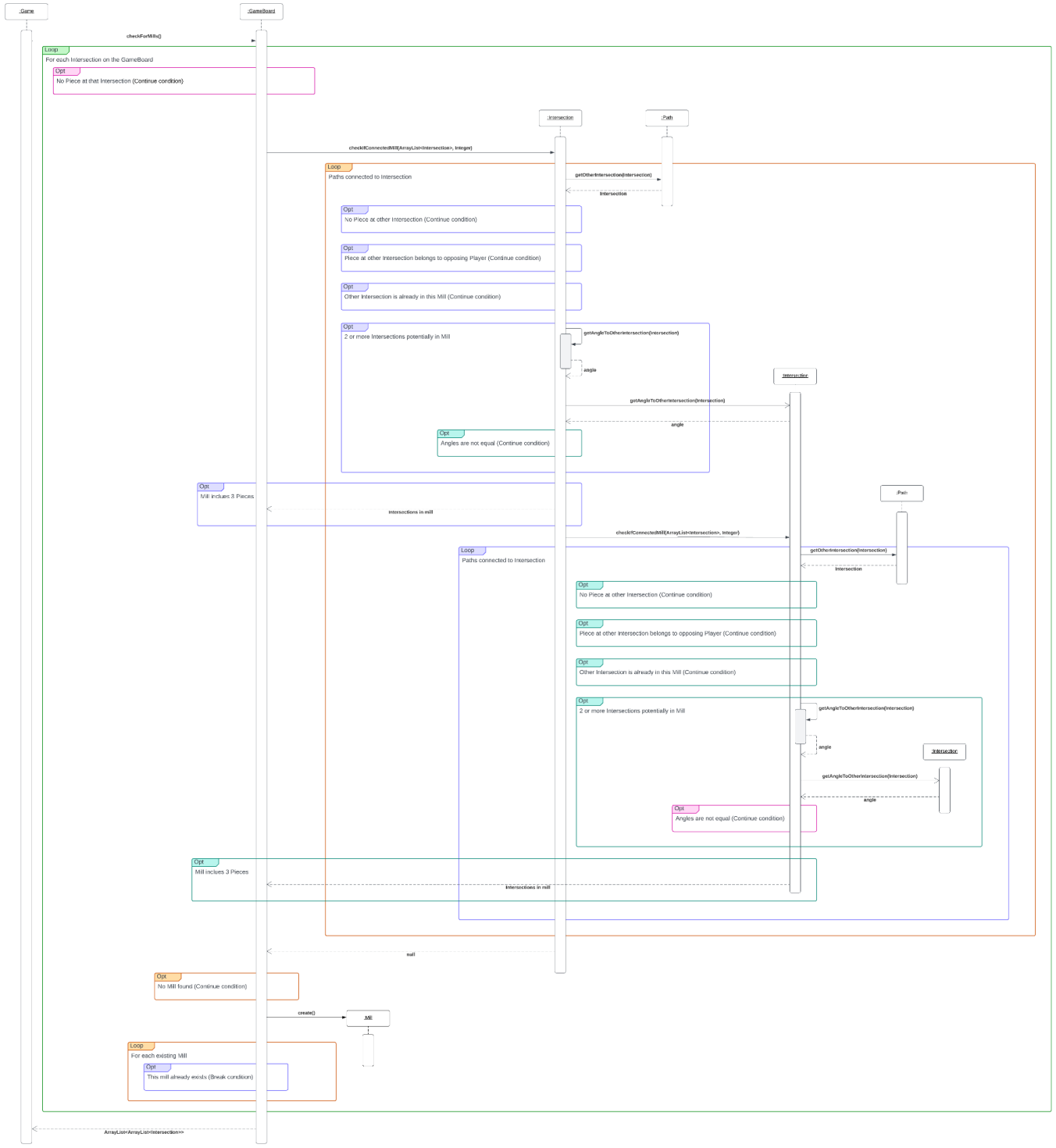


Sequence Diagram(s)

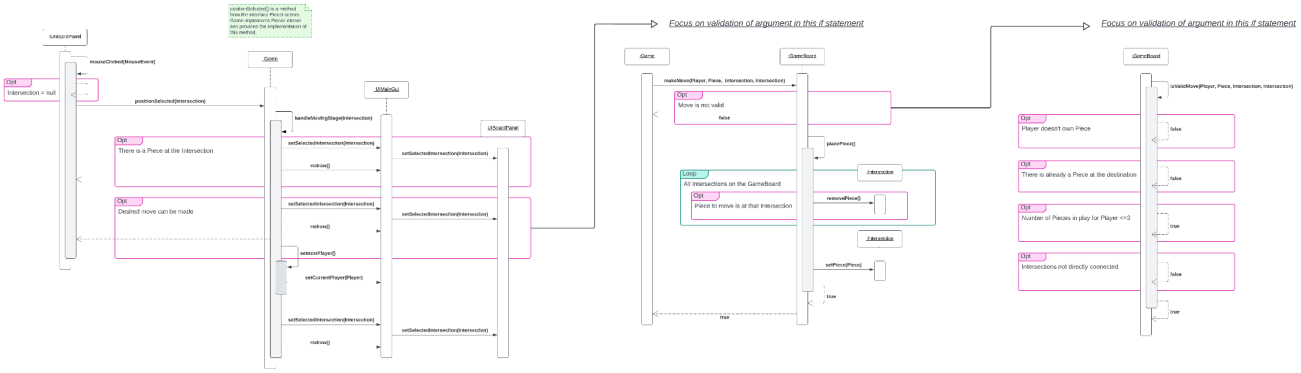
- *Initialisation (bootstrap)*



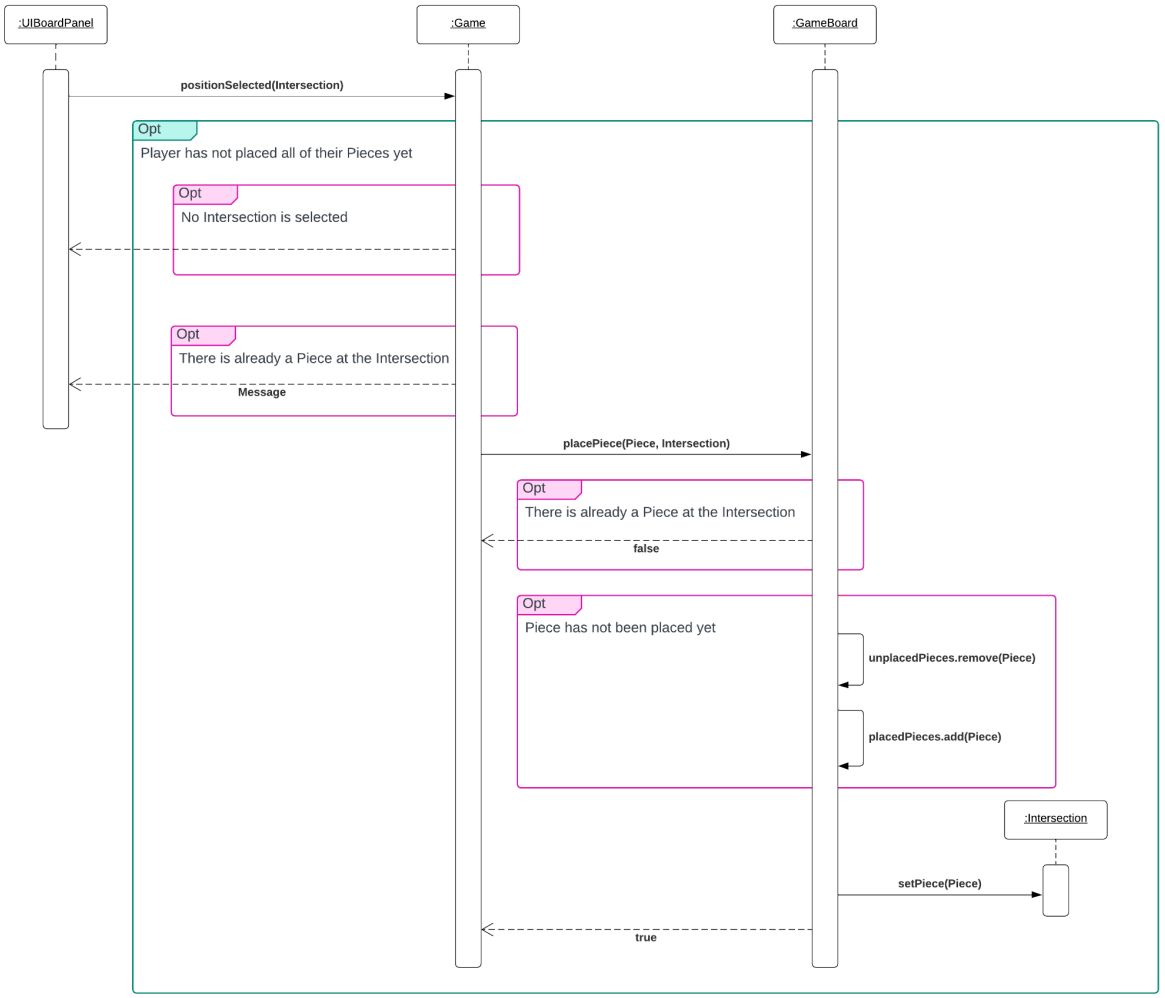
- *Mill recognition*



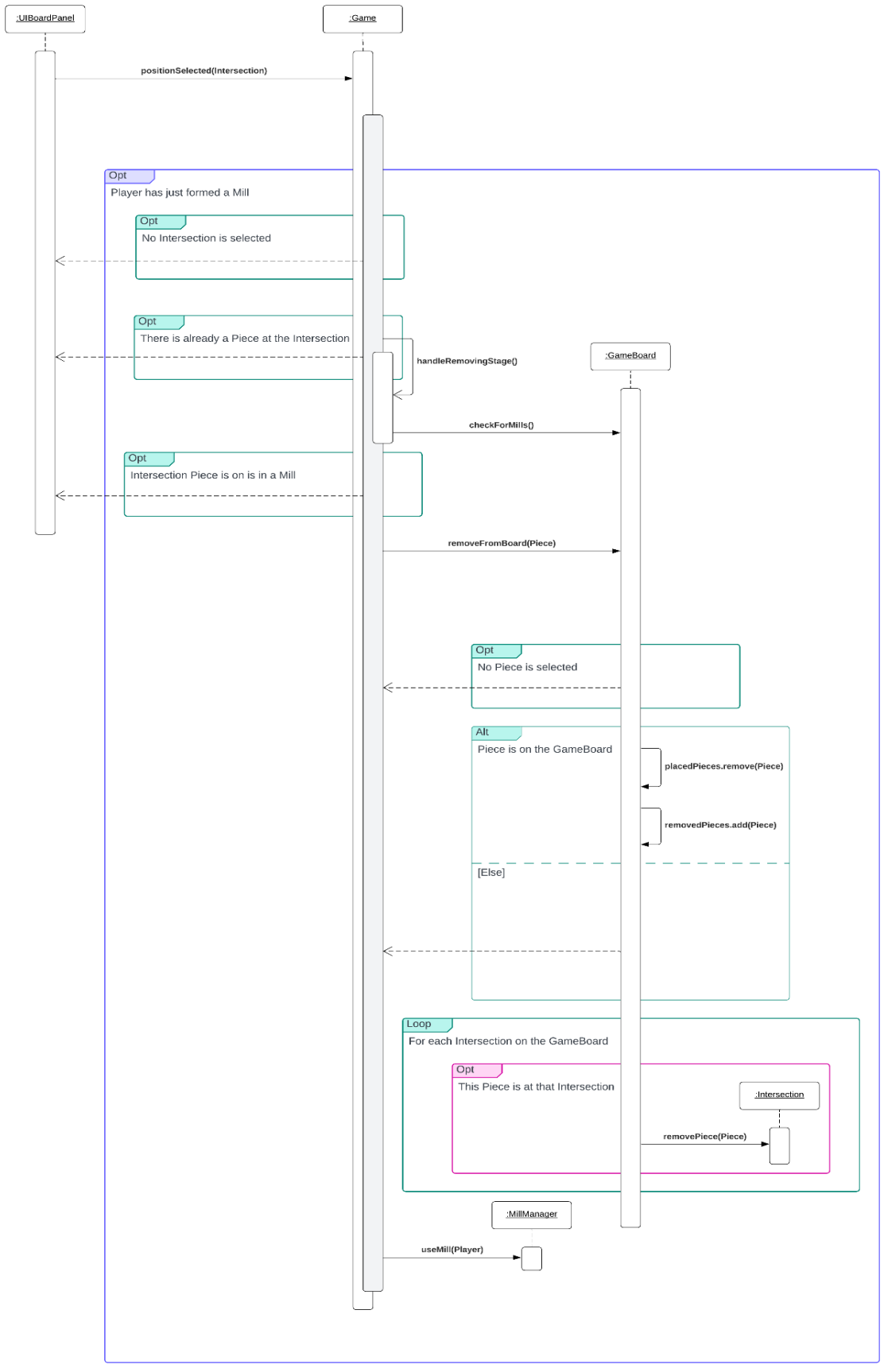
- *Move validation*



- *Piece placing*



- *Removing a piece*



Design Rationales

Reasons for architecture revisions

- One of the architectural revisions that we made was to implement a MillManager class. The idea behind this was to move the responsibility of keeping track of Mills out of the Game or GameBoard. The Game does not need to have the responsibility of keeping track of Mills since we have defined it to validate moves and not to keep track of the state of the Game. Similarly, the GameBoard should not have this responsibility either. The GameBoard has the responsibility of making moves, placing pieces, etc. This helps our design to remain true to the single responsibility principle.
- We decided to make MillManager a final class (java) which has static methods. The MillManager methods include adding a Mill, checking if intersections are part of a mill, using mills (to remove an opposing players' piece), and checking if a player has unused mills.
- Architecturally, we have not changed much. We are still using an MVC architectural pattern. The GameBoard as the Model, the GUI as the View and Game as the Controller. We did not see any need to change our architecture too much considering:
    - The progress we have made so far, especially regarding mill checking and our methods to create game boards of any dimension. We have put considerable effort in ensuring correct logic for mill checking which can be extended to any dimension game board.
    - Our familiarity with the code base.
    - We have had a good start and were happy with the direction of the project and felt no need to start fresh or change architecture this late into the project.

Application of quality attributes

- **Usability**: Approximately 1 in 12 males and 1 in 200 females experience colour-blindness. We have utilised a colour-scheme that is colour-blind friendly. Rather than using a colour scheme such as blue and red - which may decrease usability for colour-blind users - we have implemented a blue and orange/yellow colour scheme. By designing for people with colour-blindness, we are including that group of people as potential users of the game in our design. This adheres to the accessibility guideline of perceivability by providing content that is easy to see for a wider range of users. This is relevant to our game because we want to design something that can be used and enjoyed by the greatest number of people.
- **Feedback:** Along with our intentional use of a high contrasting colour scheme (discussed above), we have tried to make the application provide the user with good UI/UX. Through the use of 'Rules', 'Controls' and 'Exit' buttons we are providing an easy to use interface that is predictable and provides relevant information and feedback to the user. Furthermore, during in-game situations, such as forming a mill, we provide more

feedback by highlighting the intersections which are part of the Mill, as well as providing a pop up that informs the user that they have formed a Mill. The same applies for an end game situation where the user has made a winning move, the game will recognise this and present them with a pop up to inform them that they have won the game.

- **Flexibility:** Our initial goal when designing this application was to be able to support any size board. Although this was not explicitly stated as a requirement, we made it a challenge to ourselves to ensure that our implementation was able to properly follow the rules of Nine Men's Morris, regardless of the board dimensions (and intersection relationships). Although there is no way to select board layouts or change them via the GUI, we have done thorough testing on another board layout which is vastly different to the conventional 9MM board and our implementation of mill checking and game rules work flawlessly. This is evidence of good design.

Application of human value

- **Capable**: We have considered the human value that people want to feel capable when designing our game. When people play games, they get a sense of satisfaction from the feeling of capability that comes with doing well or succeeding at something within that game. Nine Men's Morris is no different in this case. If the language used to communicate with the viewer was simply dry and emotionless, then when players achieved something in-game (such as forming a Mill or winning the game) then they would experience a dampener on this feeling of capability. Our design manifests this value in particular with the language that it uses to inform players that they have formed a Mill or won the game. When they form a Mill they are presented with a message that says "Well Done! You have formed a mill. You can now remove one of your opponent's pieces.", and when they win they are presented with a message that says "Congratulations!". This positive language that communicates approval of their actions and that they are doing a good job lets the player feel this sense of accomplishment and capability - which also appeals to the human desire to feel successful.

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***

Alternatively, if you would like to compile and run without the IntelliJ IDE, you can navigate to the src folder and type `javac Application.java` and then run it `java Application`. This requires the Java SDK to be installed. We recommend to use IntelliJ since that is what we know will provide good results.

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 3.