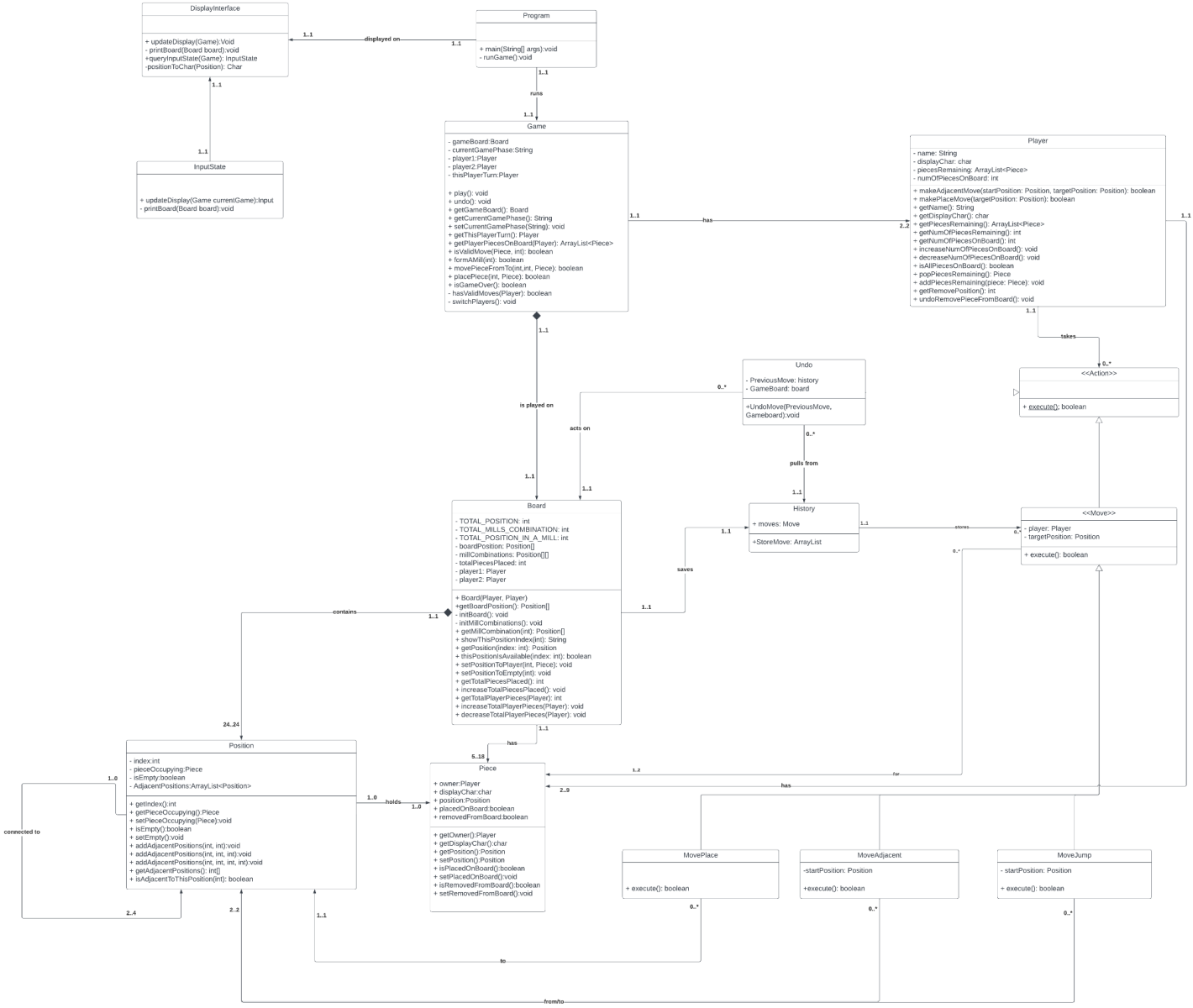
FIT3077 - 9MM Sprint 2 - Group 42

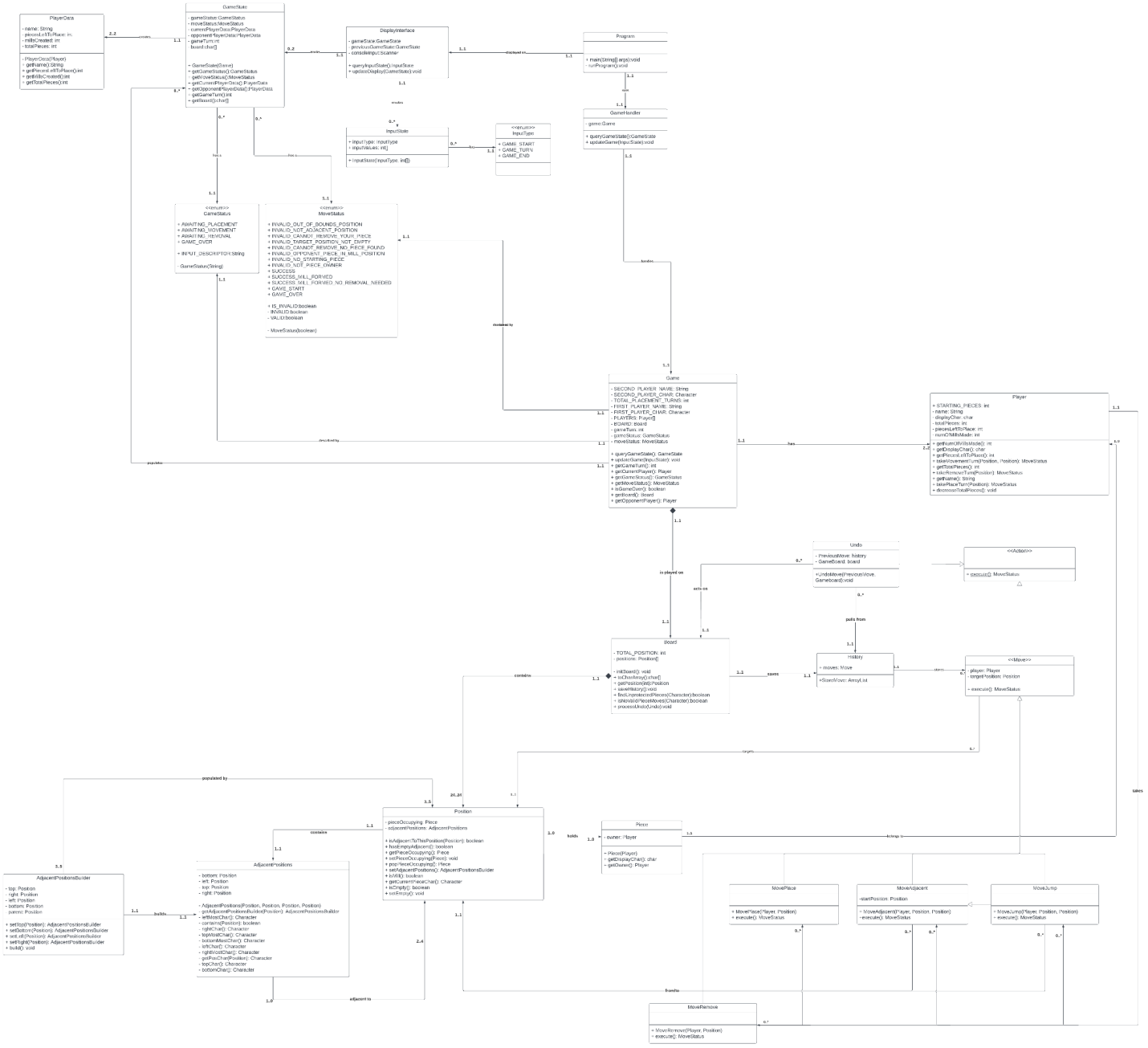
1. Table of Content

2. Class Diagram

UML Class Diagram - Sprint 2 (Before)



UML Class Diagram - Sprint 3 (After)



Class Diagram Changes and Architecture Rationale

Changes to the class structure were generally split into two categories of restructure, the first being separation and definition of a view layer, and the second being consistency of program flow and dependency reduction.

Separation and definition of view layer occurred mostly in the top section of the new diagram where there now exists numerous data classes such as GameState, a nested PlayerData object which exists within GameState, and to compliment these also a series of static enum classes. These allow for the displayinterface (which handles displaying the game to the system and input technicalities to operate fully separated and unable to impact the logic layer of the game, since these data classes only provide the bare necessities of information and only store primitives and select enumerations. This provides not only a level of data security but also integrity as no artefacts in the display can affect anything but the display.

This same reasoning was applied to the inclusion of a GameHandler class, except instead in the interest of separating the program technical runtime itself from the logical game class, this is important as many functions which the player takes may not occur within the game itself and instead are acting on an instance or instances of a game, such as restarting or exiting the game. These shouldn't be handled within the game itself and instead should be interpreted by a handler class and applied to the game to avoid cyclic redundancy and as said previously preserve the logical integrity of the game flow.

This brings us to the second main motivator for major changes in the class structure, keeping a consistent and simple program flow which focuses on allowing easy and clear extendability while minimizing dependency between classes. A big part of this is change some dependencies around, removing the dependency between piece and board, and inverting the dependency between player and piece. While this seems somewhat counter intuitive, it actually gives a lot of flexibility to the design since now moves, through position, are able to handle advanced checking such as mill checking directly instead of relying on passing long chains or parameters or data objects through the program flow. This also keeps all of the error handling cases in one place and able to be easily represented through a enum namely MoveStatus. This allows for error handling to occur on the way down through the program flow, and move resolution to occur on the way up, when seen in code this is even more intuitive than simply having player own pieces and having to do error checking and resolution both on the way down and up which would have both increased the chance of artifacts and decreased maintainability due to increased complexity in program flow. When combined with the fact that board is unable to access pieces directly this also removes the cyclic dependency which would occur between piece and position whilst allowing for flexibility and ease of modification in move functions, since they are the core areas of mutation in the program, and isolating them should be a key thought in the design of the program.

Notice also that player now instantiates concrete move classes directly, this really improve clarity in the player class but also help further in isolating the mutating and moving parts of the program, and in the end allows for game to be represented by a defined set of states which decreases overall mutability and increase predictability.

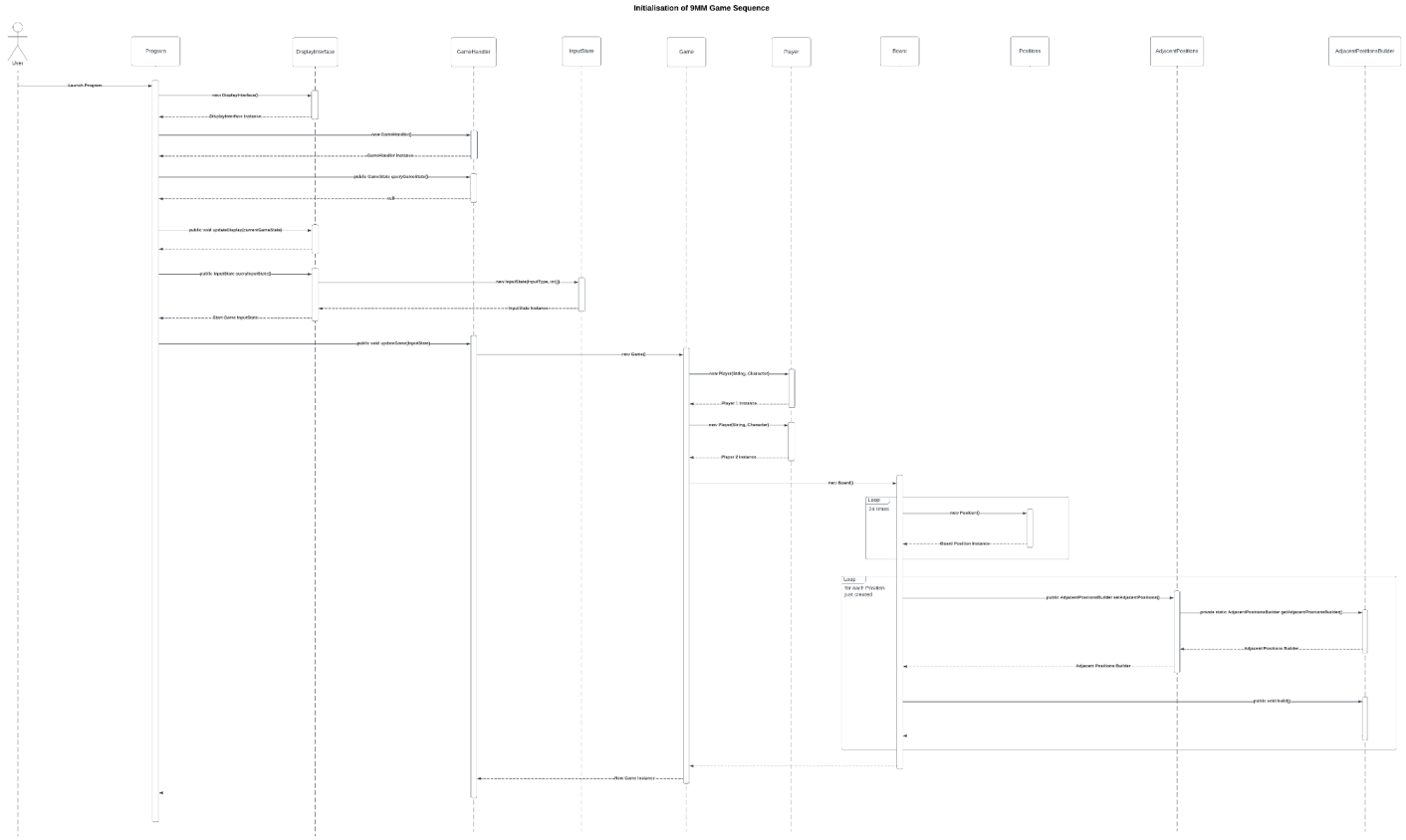
There is also the inclusion of the adjacent position nested class in positions, this was mainly added to decrease the coupling between position and the rest of the game since a lot of checking occurs here especially that used by move classes it is important that positions do not couple with higher parts of the game and act as the low level functions they should be. This builder class means that the board class can easily and clearly define the board layout with respect to adjacent positions with minimum coupling to positions themselves and more importantly without the ability to accidentally mutate them substantially since about adjacency construction adjacency positions are private to even the position class itself.

Finally is the addition of the RemovalMove which was an oversight in the original class diagram and added to aid in maintaining a predictable game state whilst also affirming the disparity between input and game logic without needed to add additional gamestate mutability.

3. Sequence/Communication Diagrams

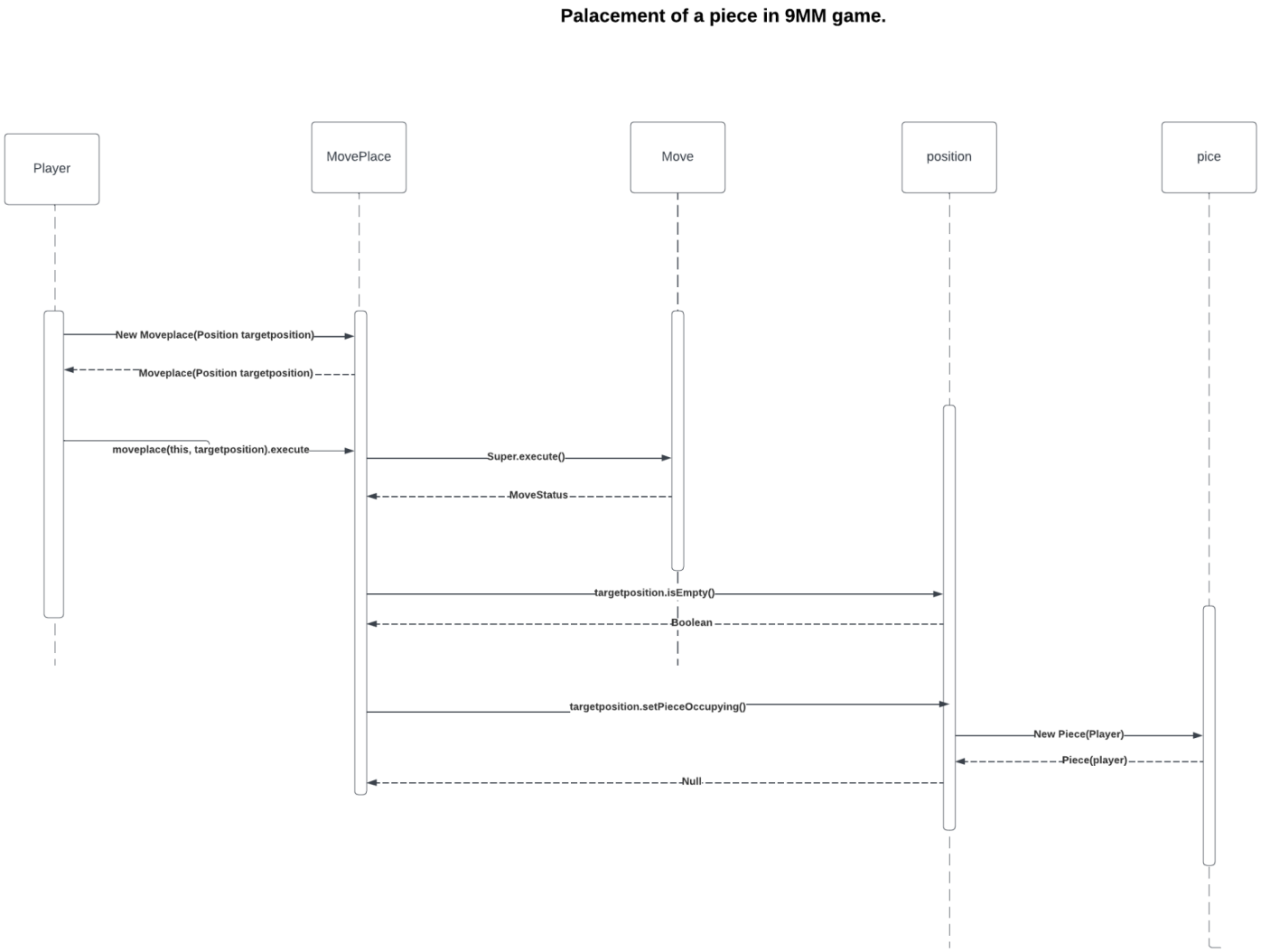
Initialisation of 9MM Game Sequence

A sequence diagram detailing the initialization of the 9MM Game (Start Game Input - Single Cycle)



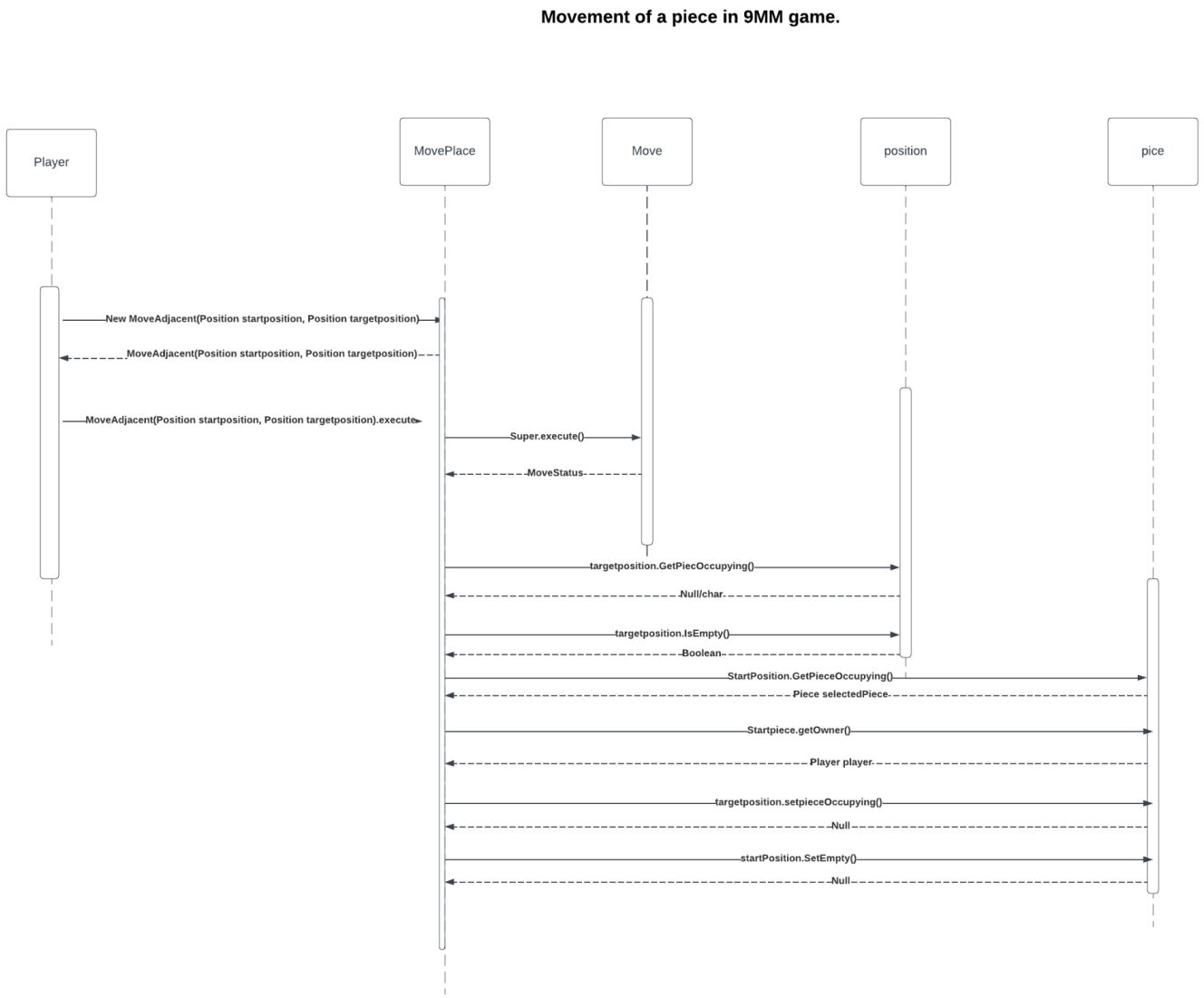
Place Piece Input

A sequence diagram detailing the placing of a piece.



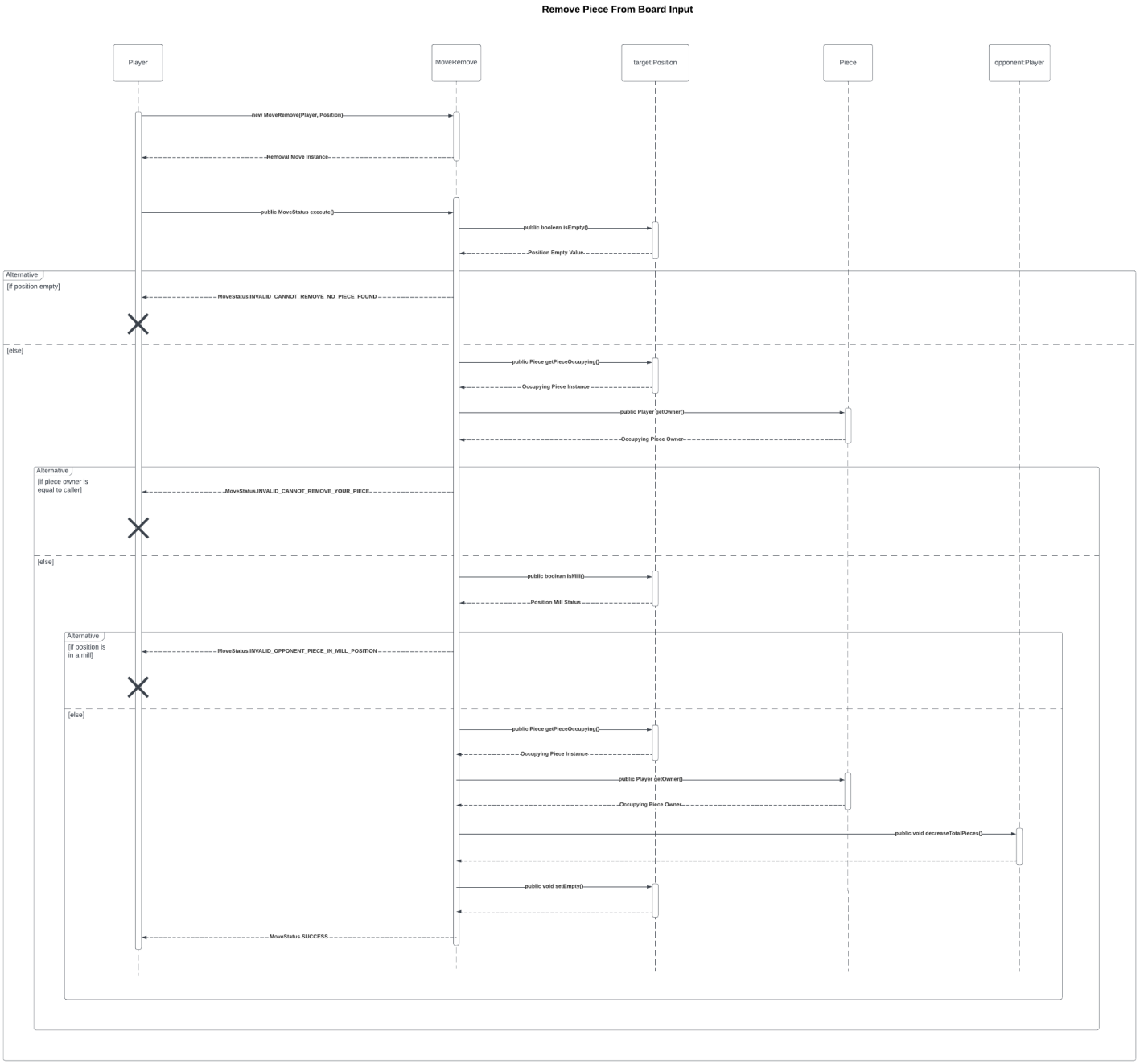
Move Piece Into Mill Input

A sequence diagram detailing the moving of a piece into a mill.



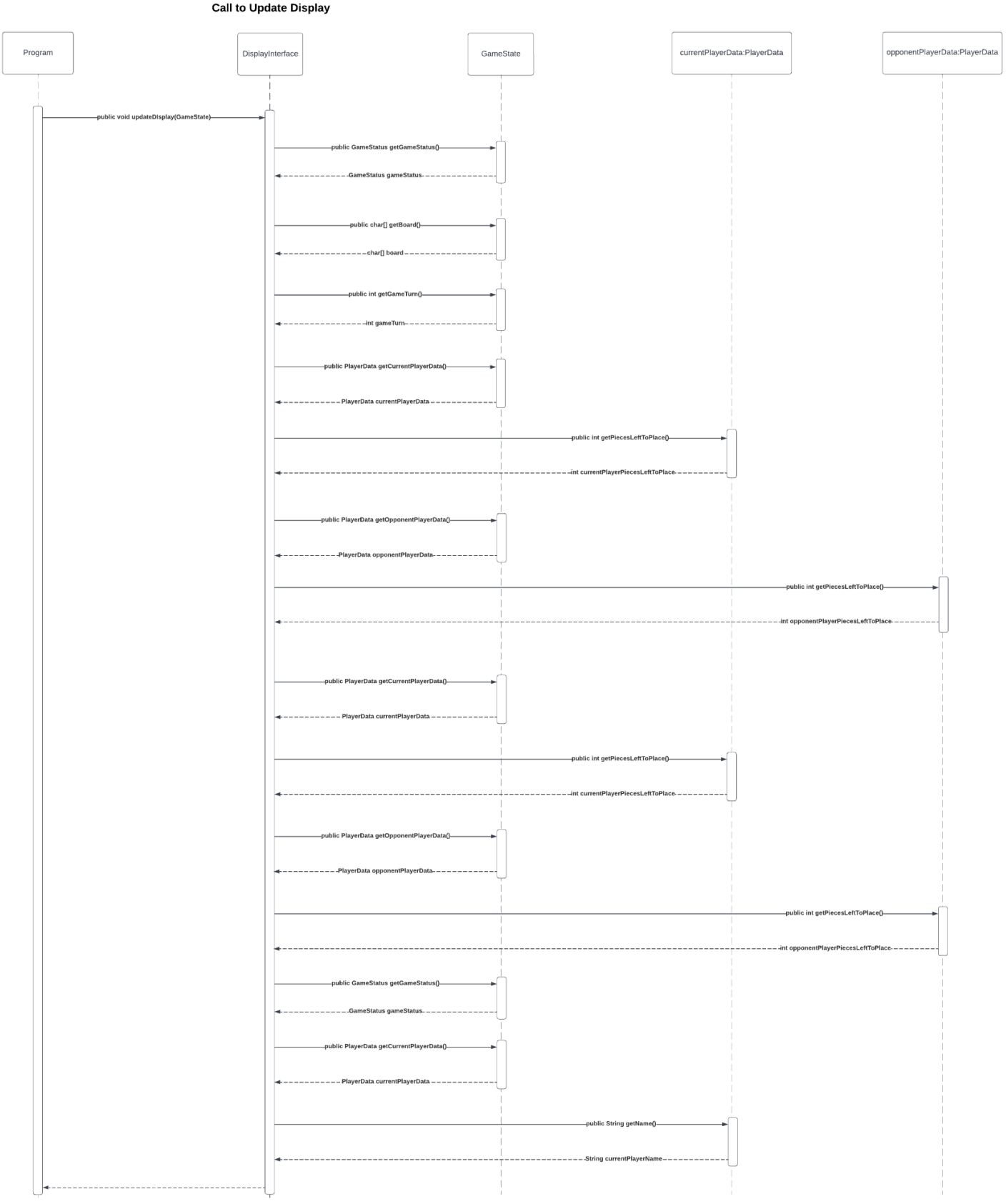
Remove Piece From Board Input

A sequence diagram detailing the removal of a piece. (From Player Move Call Assuming Removal Input)



Update Display Call

A sequence diagram detailing an update to the display. (From Program Call Assuming Game Started)



4. Design Rational

Quality Attribute - Usability

In a game like 9 Men's Morris, usability is paramount as it directly influences the player's experience and satisfaction. In implementing a console-based design, we faced certain usability challenges, as the lack of visual aids can make navigation and interaction more complex for users. However, by adhering to specific design principles, we've ensured the game remains intuitive and user-friendly:

Representation of 9MM Game Board:

While the console-based design limits our ability to create intricate, visually engaging elements that could potentially enhance gameplay, we've nevertheless developed a clean and intuitive game board that doesn't compromise the integrity of the game. We've managed to preserve the game board's original design, comprising a neat configuration of lines and brackets to delineate the board.

To boost the usability of our game design, we've deviated from the traditional grid pair numbering system employed in Chess. Instead, we've introduced an integer-based position numbering system ranging from 0 to 23 alongside the board position. This unconventional yet straightforward modification further enhances the clarity and player-friendliness of our game board design, making it more accessible and easy to use for all players.

Clear Instructions:

The console-based design paradigm primarily restricts us to text input, but we have managed to transform this apparent constraint into a key strength. We have crafted an intuitive and informative input system that provides players with an unambiguous understanding of their in-game actions. Following each modification in the game state, the console prompts the corresponding player to execute their next move.

For improved transparency, we display the identity of the active player and the current stage of the game, thereby providing players with a live comprehension of the unfolding game narrative. At the commencement of the game, each player's pieces are neatly arranged in their respective starting positions and visually depicted via unique display characters.

When a player positions a piece on the board, the count of their available pieces visibly reduces, and the console replaces these with empty "_" symbols. This interactive feature augments the game's clarity by constantly reminding players of their remaining piece count, thus adding an extra dimension of strategy and understanding to the gameplay.

Prompt Feedback:

The constraints of a console-based design limit our ability to provide visual feedback. However, we've mitigated this through the provision of detailed written feedback. Following each user input, an insightful response such as "Invalid Move, Please Try Again" or "Player Placed a Piece" is generated, among others. This immediate feedback mechanism ensures players are never left in the dark regarding the success or recognition of their actions.

Nevertheless, we consciously control the specificity of the feedback, opting for concise over detailed responses. Rather than elaborating with feedback like “Player Placed a Piece at Position X” or “Player Moved A Piece From Position X to Position Y”, we maintain simplicity with straightforward responses. This decision was a strategic choice made by our team to mirror the dynamics of a Chess game in our 9MM game flow. In such a setting, if a player fails to pay attention to the opponent’s move, it puts them at a potential

disadvantage. This introduces an element of strategy and attentiveness into the game, enhancing the overall gaming experience.

Quality Attribute - Flexibility

In the ever-evolving world of gaming, flexibility stands as a pivotal quality attribute to assure the sustainable progress and longevity of a game. This element of flexibility ensures the game's architecture is designed in a way that can seamlessly adapt to advancements, integrate novel features, and scale up according to the evolving demands of the market. Crucially, this architecture must exhibit the capability to function synergistically with an array of other systems, thereby bolstering compatibility across platforms and devices. As such, flexibility not only amplifies the game's current capabilities, but also fortifies its readiness for future expansion, fostering adaptability, scalability, and ultimately, a richer, more immersive gaming experience.

Enhancing Game Logic via Modularity

Modularity significantly streamlines the game logic of a console-based 9 men's morris game, boosting maintainability, extensibility, and reusability. A modular design approach segments game logic into distinct, manageable components, offering heightened flexibility.This concept is illustrated through inheritance and subclassing. The "Action" class serves as a blueprint for various game actions, while its subclass, "Move," concentrates on piece movements. Specialised subclasses of "Move" further refine different move types, augmenting the game logic's manageability and testability and facilitating system adjustments.

The modular design benefits include enhanced code reusability, improved organisation, and scalability. With inheritance and subclassing, game logic becomes adaptable and can efficiently handle varying move types. Modularity amplifies the game logic's flexibility and maintainability, leading to a robust, customisable gaming experience.

Platform Independence Architecture

Our 9MM game is skillfully crafted to execute flawlessly across a wide spectrum of platforms, completely eliminating the necessity for recompilation or alteration. This impressive versatility is achieved through the utilisation of Java, a platform-independent language renowned for its compatibility with diverse operating systems including Windows, Linux, and macOS.

Cross-platform compatibility stands as a crucial aspect for the 9MM game, as it not only broadens the prospective player base, but also bolsters the game's accessibility. Consequently, players have the freedom to relish the game on their device of choice, resulting in a surge in user satisfaction. The structure of the 9MM game is a testament to this cross-platform compatibility, with its use of object-oriented programming principles being the cornerstone. We encapsulate the game's functionality within a suite of classes and methods, such as Game, GameHandler, and GameState, thereby safeguarding platform-independent functionality. This clever design choice ensures that our game operates seamlessly across a myriad of operating systems, providing a smooth and enjoyable gaming experience for all users.

Scalable System

At the heart of our 9MM game's architecture is a focus on scalability, a design feature specifically incorporated to guarantee that the game can seamlessly evolve and adapt to future requirements, including a variety of different scenarios. Take for example the DisplayInterface class, which deftly manages the console-based user interface.

Conversely, the game engine is crafted using the singleton design pattern. This design choice ensures that there remains just a single instance of the game engine persisting throughout the entirety of the application. Thanks to this flexible design structure, we can independently create multiple instances of the graphical interface classes, such as the GraphicalInterface. These instances can then be used to initialise the game engine with the corresponding interface. Consequently, the game engine has the capability to engage with the selected GUI implementation, whether it's the console-based DisplayInterface or a more graphical interface, all without being restricted or tightly bound to any specific implementation.

Quality Attribute - Reliability

To encourage players to return to our 9 Men's Morris game, we've prioritised creating a reliable, consistent gaming experience. Reliability, the ability of the game to perform dependably under specified conditions, is critical in maintaining smooth gameplay and nurturing player trust and satisfaction. We've ensured reliability in our design through:

Saved Game State:

Our game incorporates a sophisticated mechanism designed to preserve each new game state, effectively capturing the game's essence at each turn and saving it into the input state. This feature offers a historical record of the game, facilitating player understanding of the unfolding strategy.In our upcoming development sprint, we plan to enhance this feature further by introducing an advanced functionality: the 'undo' option. This will grant players the ability to revise their actions, permitting them to retreat from a potentially unfavourable position, or to reevaluate and modify a previous strategy.

This 'undo' feature not only affords players a greater degree of control and flexibility in their gameplay, but also aligns with our overarching goal of making the game as player-centric and intuitive as possible. The ability to correct actions introduces an additional layer of strategy, elevating the cognitive challenge of the game while promoting strategic thinking and calculated risk-taking among players. The blend of past game state preservation and future 'undo' functionality underscores our commitment to delivering an engaging and user-friendly gaming experience.

Robust Error Handling:

Our game is meticulously designed with a sophisticated mechanism to proficiently handle both invalid user inputs and unexpected game states. This robust error management system underscores the resilience of our game engine, allowing it to recover gracefully from errors and maintain unwavering reliability in operation. Consequently, disruptive interruptions are effectively circumvented, preserving the smooth and uninterrupted flow of the game that is so crucial to player immersion and enjoyment.

Our system efficiently manages scenarios where players try to place their pieces beyond permitted areas or in already occupied spots. Instead of disrupting the game, our robust error handling mechanism prompts the user to rectify the action. This not only ensures uninterrupted gameplay but also guides the player back on track. This strategy preserves the game's integrity and fosters a user-friendly environment. Players can rest assured that mistakes won't hinder their experience, as they're guided to correct errors and resume their strategic play in 9 Men's Morris.

Consistent Rules Enforcement:

Our team has constructed a dynamic and dependable game engine that holds the reins of logic for the quintessential strategy board game, 9 Men's Morris.This game engine is an embodiment of consistency, tirelessly enforcing the rules to maintain the essence and integrity of 9 Men's Morris. This unwavering adherence to the game's principles reassures players, as they can trust the system to behave as they anticipate, fostering a deep-rooted confidence in the engine's reliability and fairness.

A player, for instance, can always expect to capture an opponent's piece upon successfully aligning their own pieces to form a 'mill', a crucial tactical manoeuvre in the game. Similarly, they are granted the privilege to 'jump' a piece when they are down to just three pieces on the board. These fundamental mechanics, along with several others, are the bedrock of the game's rules and are tirelessly enforced without fail, in every instance, maintaining the purity of strategy and competition. By upholding these essential aspects of gameplay consistently, our game engine guarantees not only a balanced gaming experience but also preserves the traditional essence of 9 Men's Morris, making every match as thrilling and strategic as the players would expect.

Human Value - Hedonism

In our text-based console implementation of the Nine Men's Morris game, the focus on pleasure and enjoyment remains at the forefront. While visual aesthetics may be absent, the emphasis shifts towards creating a captivating and immersive experience through well-crafted text and engaging gameplay mechanics.

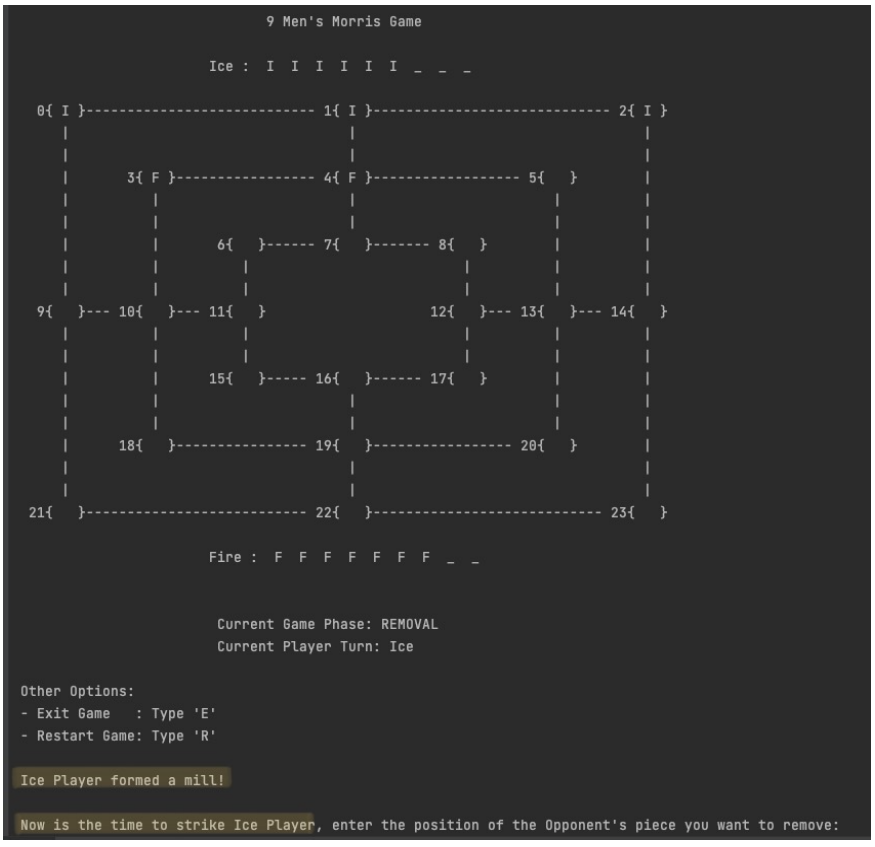
The game employs descriptive and vivid text to paint a detailed picture of the game's progression and strategic maneuvers. Each action and move is communicated effectively, allowing players to visualize the evolving state of the game in their minds. This descriptive language adds depth and richness to the gameplay experience, stimulating the imagination and drawing players into the immersive world of Nine Men's Morris.

Acknowledgment messages play a crucial role in providing positive feedback and instilling a sense of accomplishment. When a player successfully forms a mill, the message "Player has formed a mill!" is displayed, conveying a moment of triumph and celebration. Similarly, when a player emerges victorious, the message "Player has won the game" acts as a conclusive announcement, delivering a surge of satisfaction and fulfillment.

Strategic prompts in the form of text cues contribute to the excitement and anticipation of gameplay. For instance, the prompt "Now is the time to strike Ice/Fire Player" serves as a pivotal moment in the game, signaling players to capitalize on the opportunity and make a decisive move. This text prompt generates a sense of engagement and encourages players to strategize and plan their next move, creating a rewarding and pleasurable experience.

Although lacking visual elements, the text-based console implementation of the Nine Men's Morris game aims to provide an immersive and pleasurable experience through descriptive language, acknowledgment messages, and strategic prompts. By leveraging the power of well-crafted text, the game seeks to engage players' imagination, evoke a sense of accomplishment, and foster enjoyment, ultimately enhancing the hedonistic value of the gaming experience.

Evidence: (highlited in yellow)



Human Value - Achievement

The game of 9 Men's Morris (9MM) is a classic board game with strategic depth, presenting a solid medium for players to embody and express the human value of "Achievement" as postulated by Shalom Schwartz in his theory of basic human values. "Achievement" in this context denotes performing well as per the standards set by the society or group within which one operates. As such, the game of 9MM provides numerous opportunities for players to feel a sense of achievement, thereby enhancing the enjoyment and satisfaction derived from the game. This attainment of achievement is actualized in several ways within the game:

- **Skill Demonstration:** First and foremost, the rules of 9MM necessitate players to exhibit strategic thinking and tactical manoeuvres. It's not just about moving pieces around the board, but carefully planning, predicting the opponent's moves, setting traps, and blocking opponent's potential mills. As such, performing well in the game directly reflects the player's proficiency and tactical acumen, satisfying the achievement value.

- **Performance Metrics:** The game includes an in-built system that tracks a variety of performance measures such as the number of game turns taken, pieces left on the board, pieces captured from the opponent, and the mills made. These metrics provide a tangible record of a player's progress and skill level in the game. When these scores are high, it indicates a player is performing well, thus triggering feelings of achievement.

- **Post-Game Summary:** After each round of play, a summary of the game is presented. This summary, inclusive of scores and key moves, allows players to review their performance, identify their strengths and weaknesses, and appreciate their strategies that worked well. The analysis and understanding gained from these summaries can lead to improvement and a higher sense of achievement in subsequent games.

In these ways, 9 Men's Morris manifests the value of achievement, offering players an opportunity to feel accomplished and satisfied in their gaming pursuits. Thus, beyond being a game of mere entertainment, 9MM serves as an avenue for skill development and achievement-oriented competition.

Evidence: