TCP2201 Project

Trimester 2310

by Fiji

Team Leader	Chan Kar Kin	0169209773	1211102630@student.mmu.edu.my
Team Member	Ang Jin Nan	0162601883	1211100925@student.mmu.edu.my
	Ng Yun Shi	0102799327	1211103311@student.mmu.edu.my
	Chee Weng Kee	01136538947	1211304009@student.mmu.edu.my
	Tai Qi Tong	0105610242	1211102777@student.mmu.edu.my

Compile and Run Instructions

1. Open a Terminal or Command Prompt:
    - On Windows, you can use the Command Prompt (cmd).
    - On Mac and Linux, you can use the Terminal.

2. Navigate to the directory containing your code:
    - Use the cd command to change to the directory where this unzipped ‘Fiji’ folder is located.

3. Compile the Java code:

```
Java
javac Board.java BoardView.java Game.java GameController.java
Hourglass.java HourglassFactory.java ImageInverter.java
Main.java Piece.java PieceFactory.java PieceType.java
Player.java Plus.java PlusFactory.java Point.java
PointFactory.java Square.java Sun.java SunFactory.java
Time.java TimeFactory.java
```

4. Run the compiled Java program:

```
Java
java Main
```

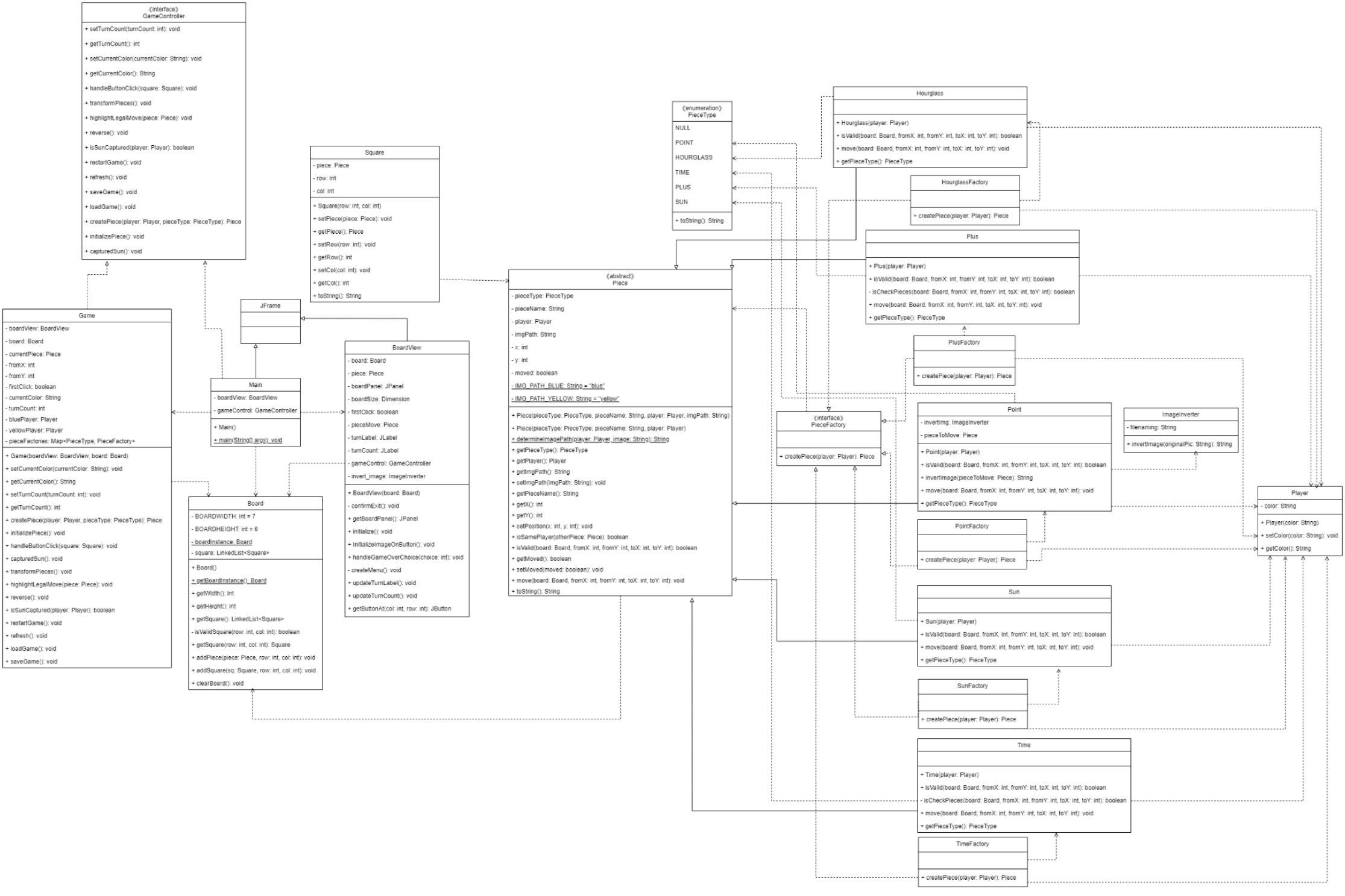
UML Class Diagram

https://drive.google.com/file/d/1PUXuz9ctMqRTw_3xoKclIWI6HlAOz-Hg/view?usp=sharing

Board class:
Singleton pattern is used at the getBoardInstance() method. This ensures the Board class has only one instance and provides a global access point to the instance.

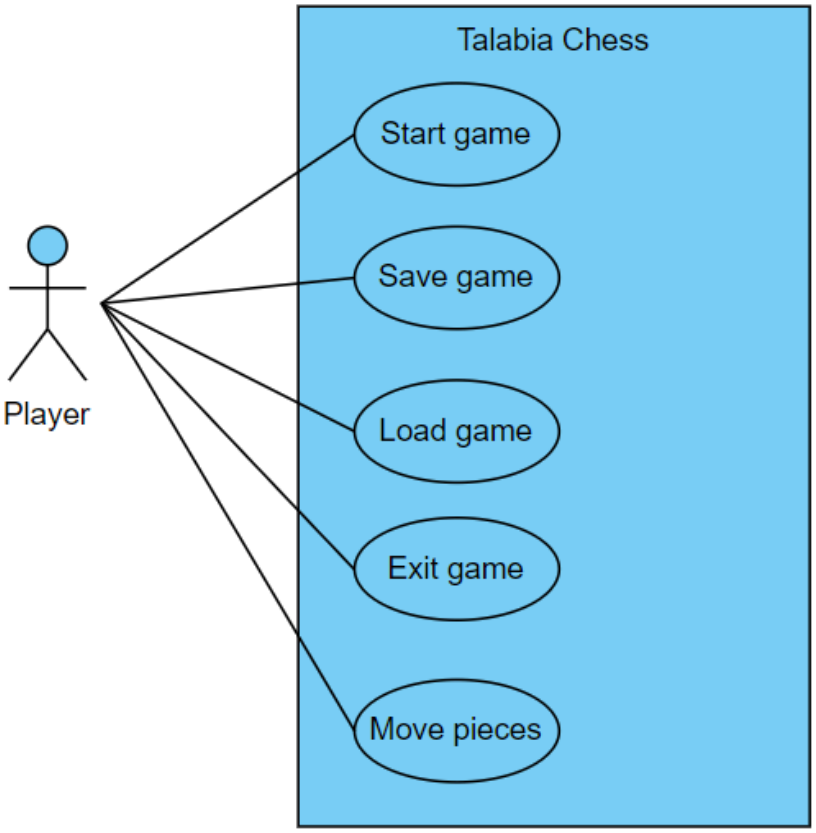
BoardView class:
Event handling such as WindowListener, ActionListener, and ComponentListener were used in this class.

Factory method is used to create game pieces. The HourglassFactory, PlusFactory, PointFactory, SunFactory, and TimeFactory classes implement PieceFactory interface to create game pieces.


{interface}
GameController
+ setTurnCount(turnCount: int): void
+ getTurnCount(): int
+ setCurrentColor(currentColor: String): void
+ getCurrentColor(): String
+ handleButtonClick(square: Square): void
+ transformPieces(): void
+ highlightLegalMove(piece: Piece): void
+ reverse(): void
+ isSunCaptured(player: Player): boolean
+ restartGame(): void
+ refresh(): void
+ saveGame(): void
+ loadGame(): void
+ createPiece(player: Player, pieceType: PieceType): Piece
+ initializePiece(): void
+ capturedSun(): void
Square
- piece: Piece
- row: int
- col: int
+ Square(row: int, col: int)
+ setPiece(piece: Piece): void
+ getPiece(): Piece
+ setRow(row: int): void
+ getRow(): int
+ setCol(col: int): void
+ getCol(): int
+ toString(): String
{enumeration}
PieceType
NULL
POINT
HOURGLASS
TIME
PLUS
SUN
+ toString(): String
Hourglass
+ Hourglass(player: Player)
+ isValid(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
+ move(board: Board, fromX: int, fromY: int, toX: int, toY: int): void
+ getPieceType(): PieceType
HourglassFactory
+ createPiece(player: Player): Piece
Plus
+ Plus(player: Player)
+ isValid(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
- isCheckPieces(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
+ move(board: Board, fromX: int, fromY: int, toX: int, toY: int): void
+ getPieceType(): PieceType
PlusFactory
+ createPiece(player: Player): Piece
{abstract}
Piece
- pieceType: PieceType
- pieceName: String
- player: Player
- imgPath: String
- x: int
- y: int
- moved: boolean
- IMG_PATH_BLUE: String = "blue"
- IMG_PATH_YELLOW: String = "yellow"
+ Piece(pieceType: PieceType, pieceName: String, player: Player, imgPath: String)
+ Piece(pieceType: PieceType, pieceName: String, player: Player)
+ determineImagePath(player: Player, image: String): String
+ getPieceType(): PieceType
+ getPlayer(): Player
+ getImgPath(): String
+ setImgPath(imgPath: String): void
+ getPieceName(): String
+ getX(): int
+ getY(): int
+ setPosition(x: int, y: int): void
+ isSamePlayer(otherPiece: Piece): boolean
+ isValid(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
+ getMoved(): boolean
+ setMoved(moved: boolean): void
+ move(board: Board, fromX: int, fromY: int, toX: int, toY: int): void
+ toString(): String
{interface}
PieceFactory
+ createPiece(player: Player): Piece
Point
- invertImg: ImageInverter
- pieceToMove: Piece
+ Point(player: Player)
+ isValid(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
+ invertImage(pieceToMove: Piece): String
+ move(board: Board, fromX: int, fromY: int, toX: int, toY: int): void
+ getPieceType(): PieceType
ImageInverter
- filenaming: String
+ invertImage(originalPic: String): String
PointFactory
+ createPiece(player: Player): Piece
Player
- color: String
+ Player(color: String)
+ setColor(color: String): void
+ getColor(): String
Sun
+ Sun(player: Player)
+ isValid(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
+ move(board: Board, fromX: int, fromY: int, toX: int, toY: int): void
+ getPieceType(): PieceType
SunFactory
+ createPiece(player: Player): Piece
Time
+ Time(player: Player)
+ isValid(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
- isCheckPieces(board: Board, fromX: int, fromY: int, toX: int, toY: int): boolean
+ move(board: Board, fromX: int, fromY: int, toX: int, toY: int): void
+ getPieceType(): PieceType
TimeFactory
+ createPiece(player: Player): Piece
Game
- boardView: BoardView
- board: Board
- currentPiece: Piece
- fromX: int
- fromY: int
- firstClick: boolean
- currentColor: String
- turnCount: int
- bluePlayer: Player
- yellowPlayer: Player
- pieceFactories: Map<PieceType, PieceFactory>
+ Game(boardView: BoardView, board: Board)
+ setCurrentColor(currentColor: String): void
+ getCurrentColor(): String
+ setTurnCount(turnCount: int): void
+ getTurnCount(): int
+ createPiece(player: Player, pieceType: PieceType): Piece
+ initializePiece(): void
+ handleButtonClick(square: Square): void
+ capturedSun(): void
+ transformPieces(): void
+ highlightLegalMove(piece: Piece): void
+ reverse(): void
+ isSunCaptured(player: Player): boolean
+ restartGame(): void
+ refresh(): void
+ loadGame(): void
+ saveGame(): void
JFrame
Main
- boardView: BoardView
- gameControl: GameController
+ Main()
+ main(String[] args): void
BoardView
- board: Board
- piece: Piece
- boardPanel: JPanel
- boardSize: Dimension
- firstClick: boolean
- pieceMove: Piece
- turnLabel: JLabel
- turnCount: JLabel
- gameControl: GameController
- invert_image: ImageInverter
+ BoardView(board: Board)
- confirmExit(): void
+ getBoardPanel(): JPanel
+ initialize(): void
+ InitializeImageOnButton(): void
+ handleGameOverChoice(choice: int): void
- createMenu(): void
+ updateTurnLabel(): void
+ updateTurnCount(): void
+ getButtonAt(col: int, row: int): JButton
Board
- BOARDWIDTH: int = 7
- BOARDHEIGHT: int = 6
- boardInstance: Board
- square: LinkedList<Square>
+ Board()
+ getBoardInstance(): Board
+ getWidth(): int
+ getHeight(): int
+ getSquare(): LinkedList<Square>
- isValidSquare(row: int, col: int): boolean
+ getSquare(row: int, col: int): Square
+ addPiece(piece: Piece, row: int, col: int): void
+ addSquare(sq: Square, row: int, col: int): void
+ clearBoard(): void

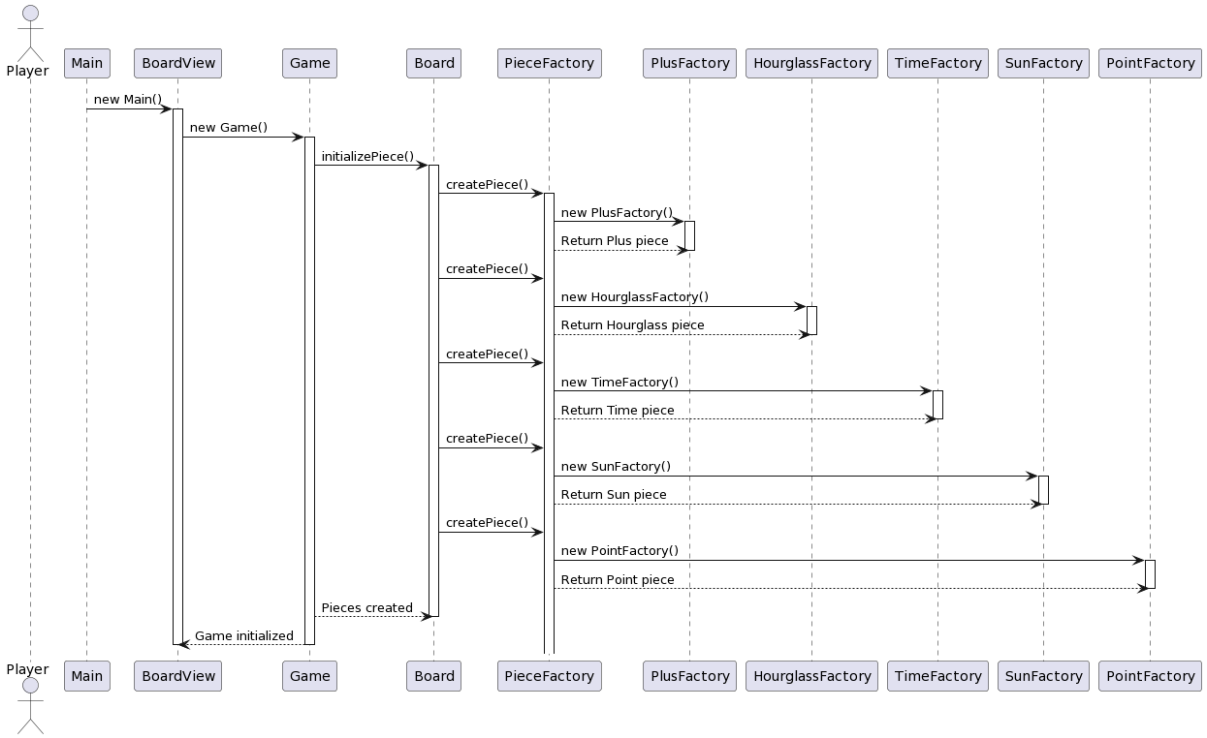
Use Case Diagram

The use case diagram below shows the main functions of our system. The player can start game, save game, load game, exit game and move pieces.

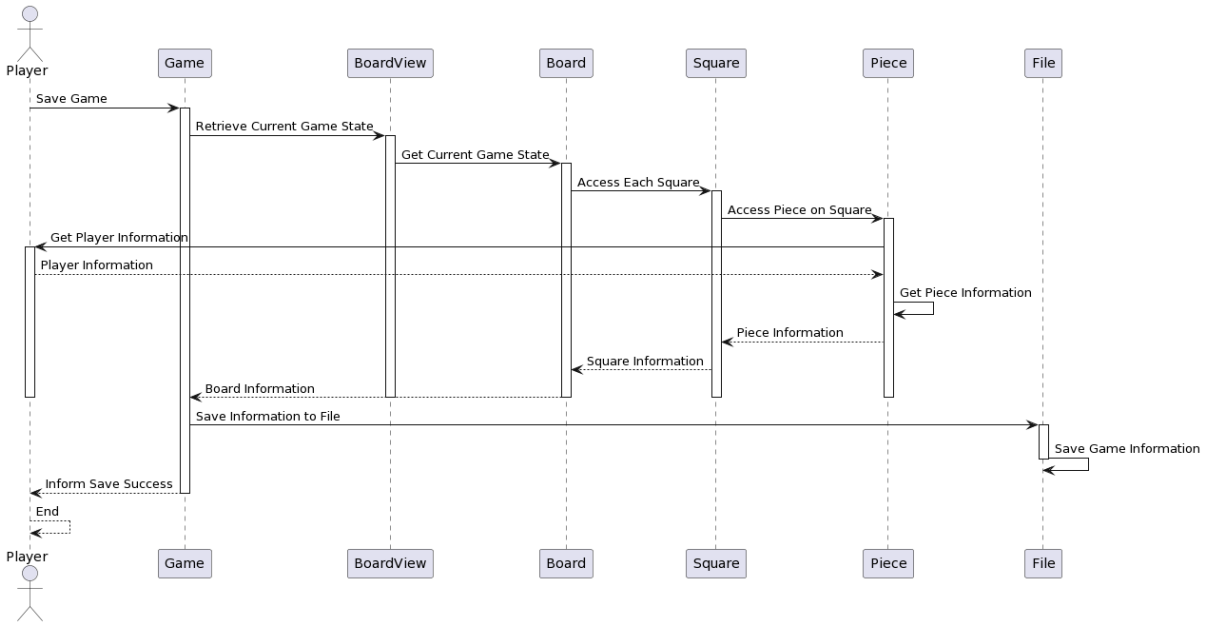


Sequence Diagram

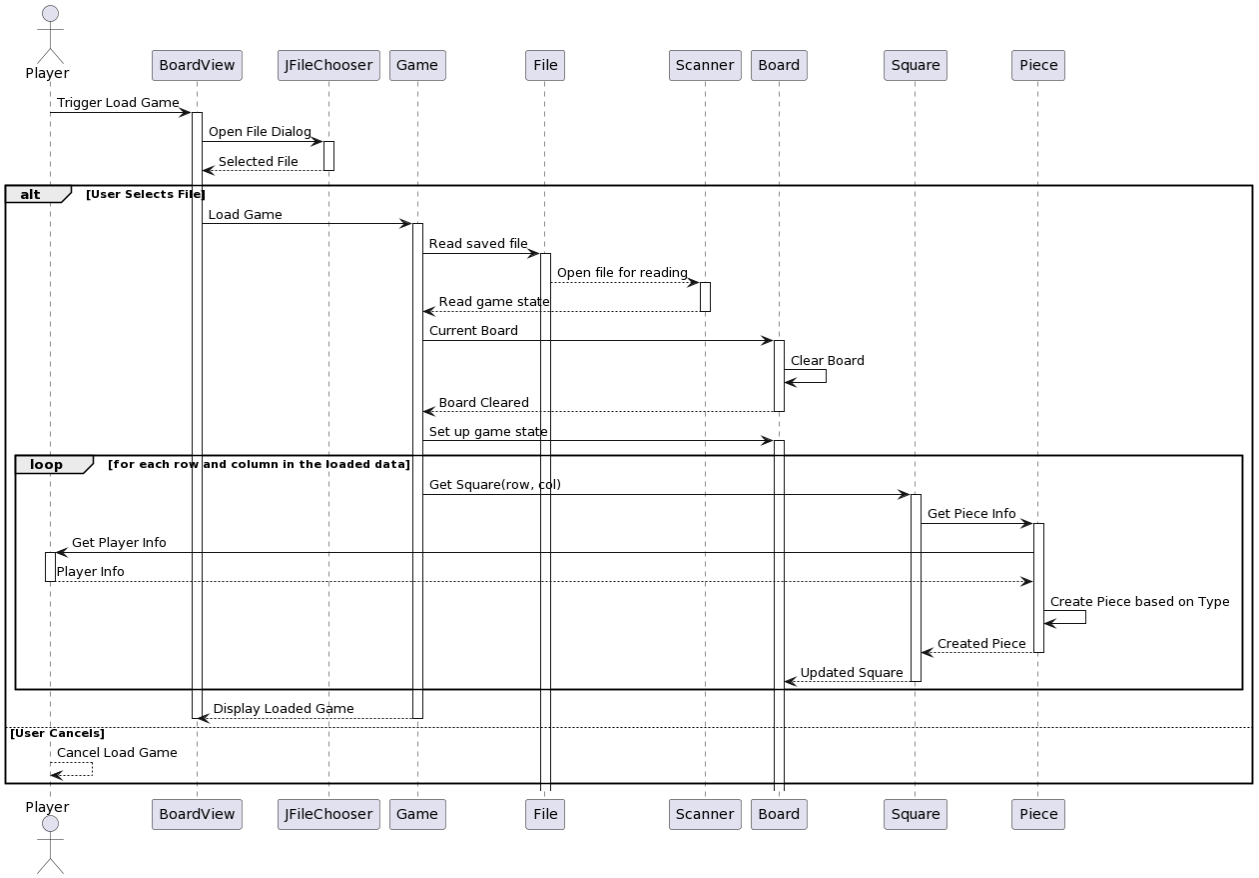
1. Start Game



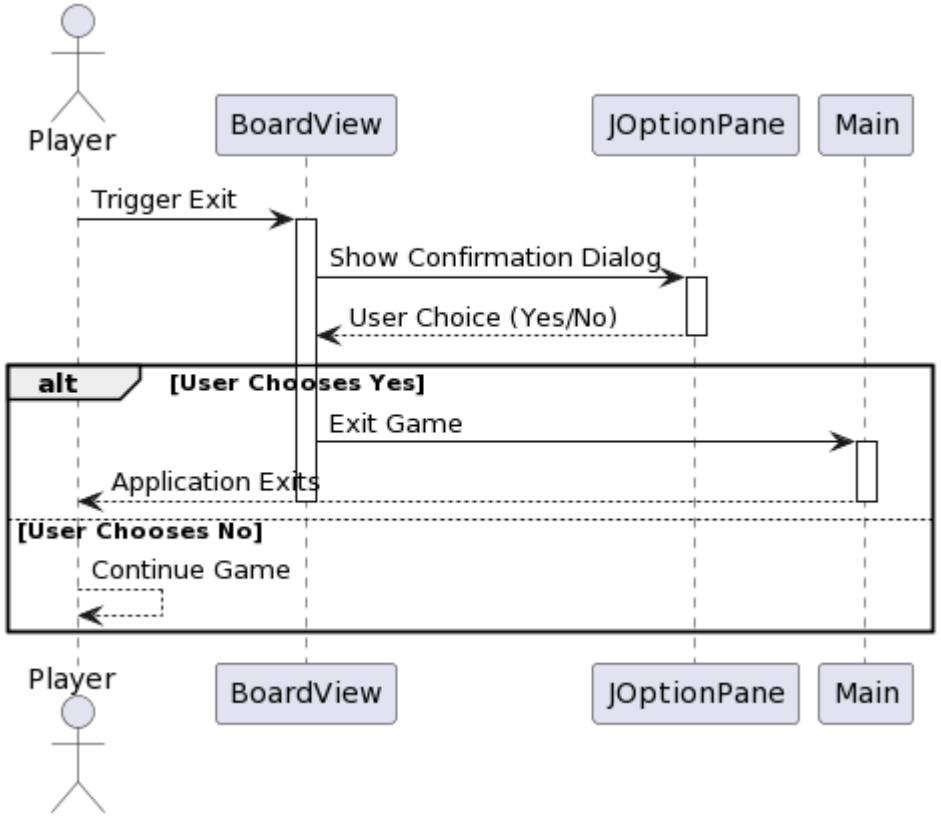
2. Save Game



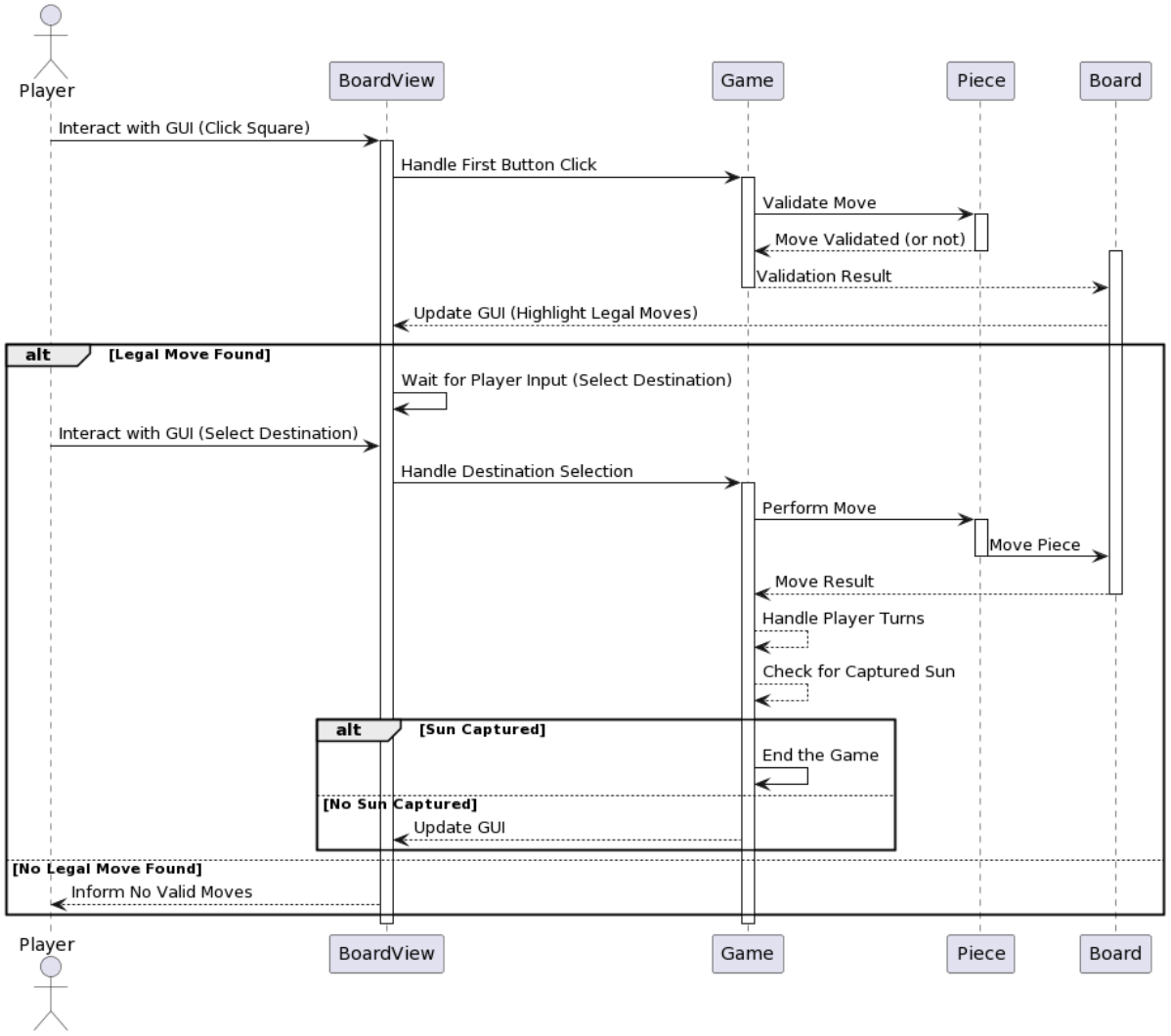
3. Load Game



4. Exit Game

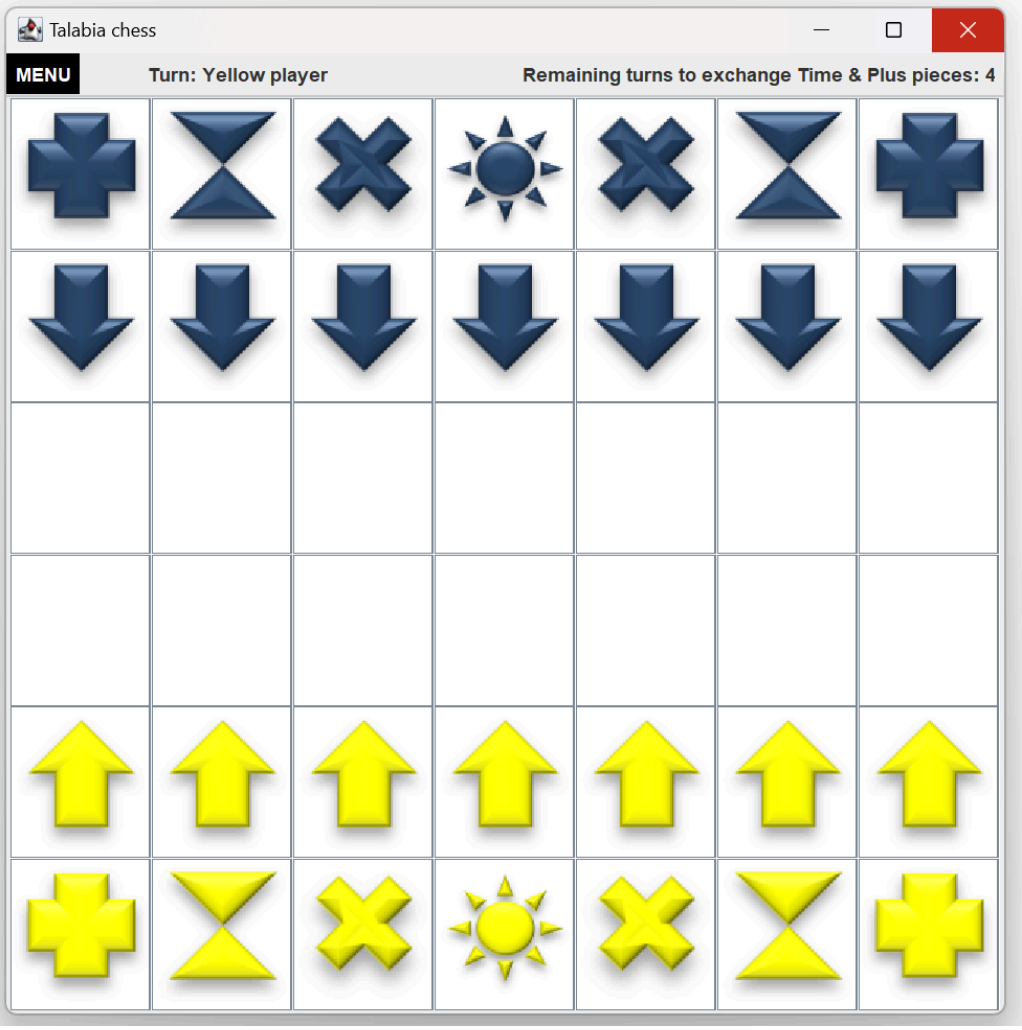


5. Move Pieces

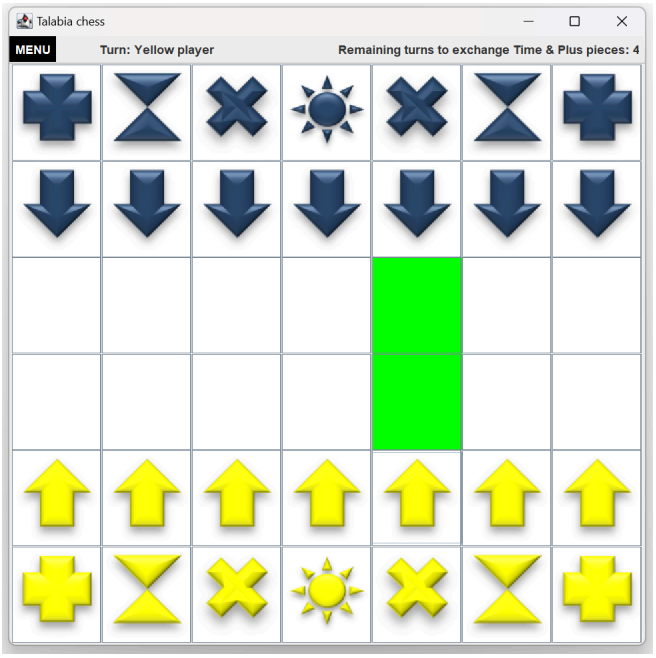


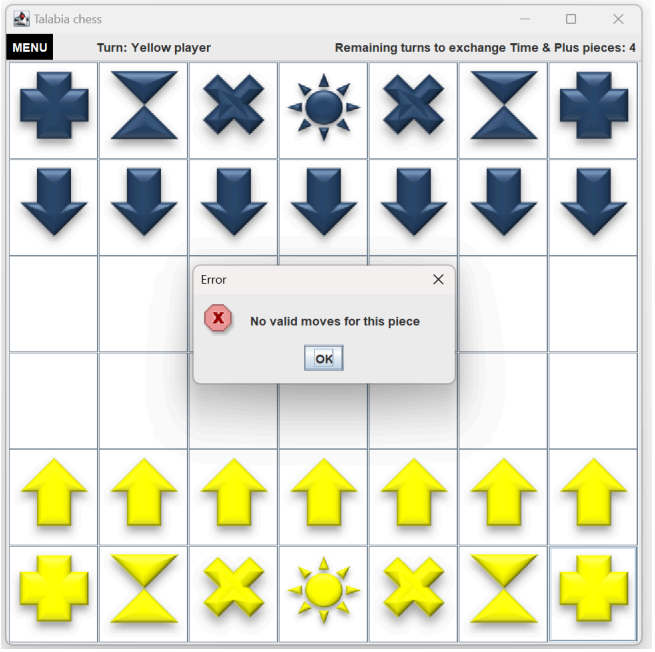
User Documentation

This is the GUI of our Talabia Chess game during runtime. The 7x6 board showcases different game pieces. The player can resize the game window by dragging the edges of the window. The game will start with the Yellow player's turn and the board will flip according to the player's turn.

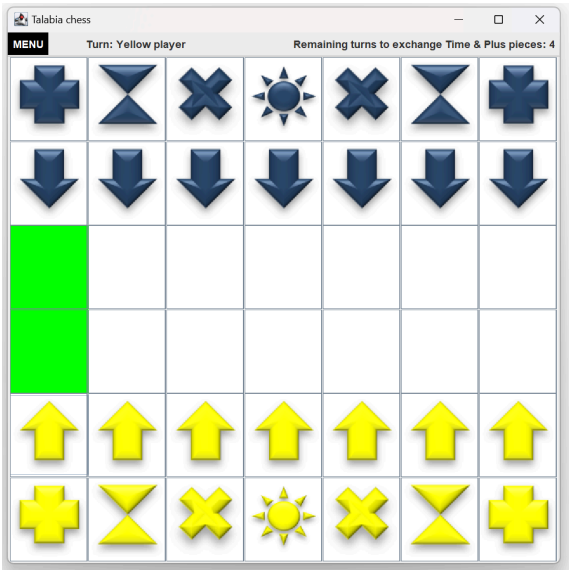


The player can view the legal moves of his game pieces by clicking them. The squares highlighted in green are the legal moves that the player can move the piece to. If there are no legal moves for the piece, no squares will be highlighted and a message box will appear, prompting the player to select other pieces to move.

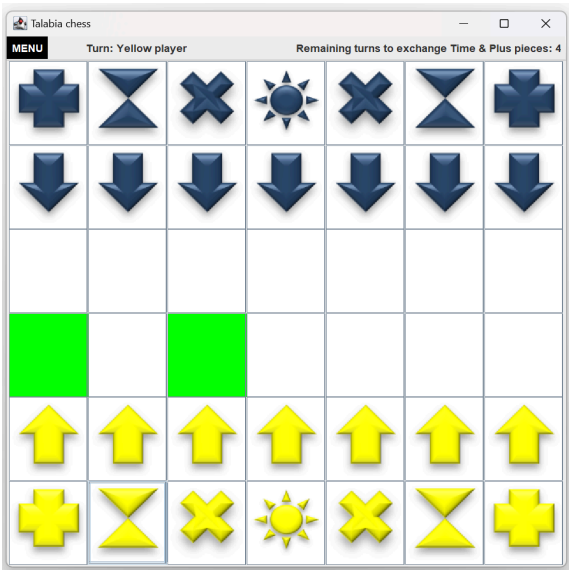




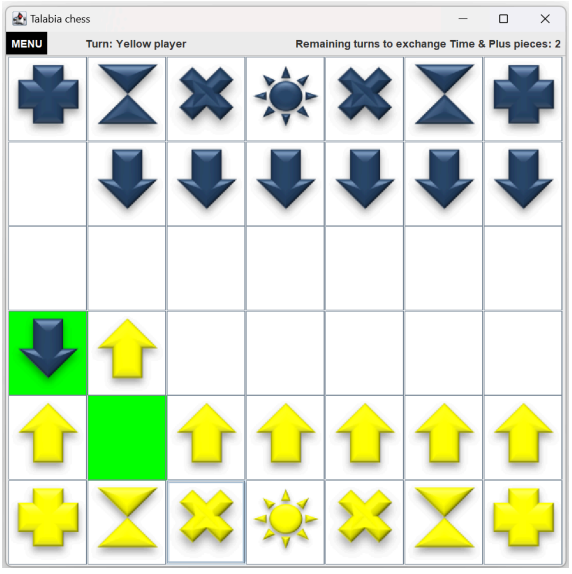
The rules of the game are simple. The Point piece moves forward 1 or 2 steps and changes direction when it reaches the board's end. The Hourglass piece moves in a 3x2 L shape. The Time piece moves diagonally at any distance, while the Plus piece moves horizontally or vertically at any distance. The Sun piece moves only 1 step in any direction. No skipping is allowed for any piece, except for the Hourglass piece. After 2 turns (having one yellow move and one blue move as one turn), all Time pieces will turn into Plus pieces, and all Plus pieces will turn into Time pieces. The game ends when a Sun piece is captured.



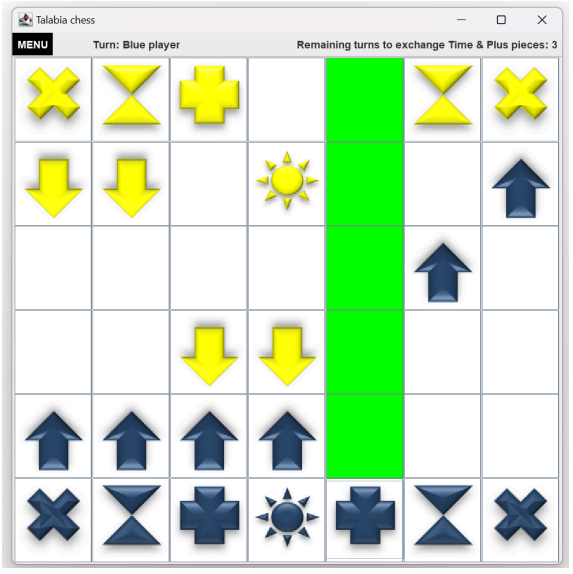
Point piece movement



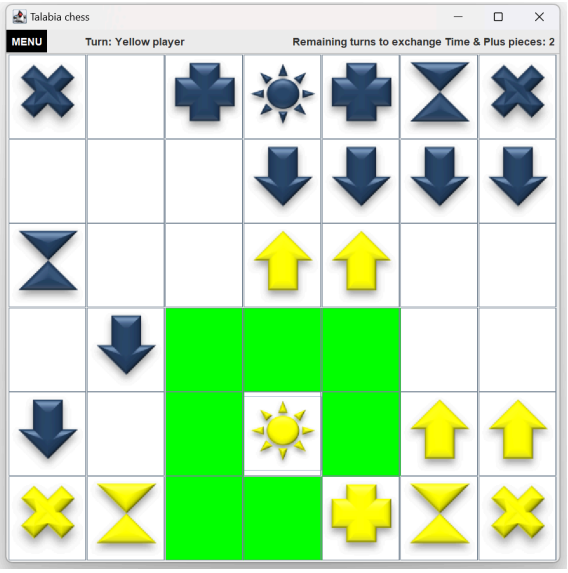
Hourglass piece movement



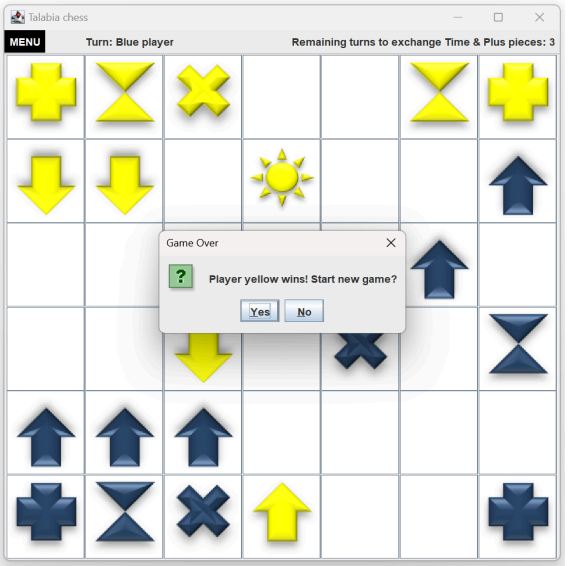
Time piece movement



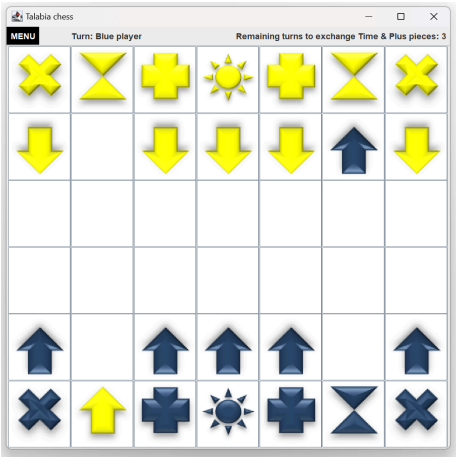
Plus piece movement



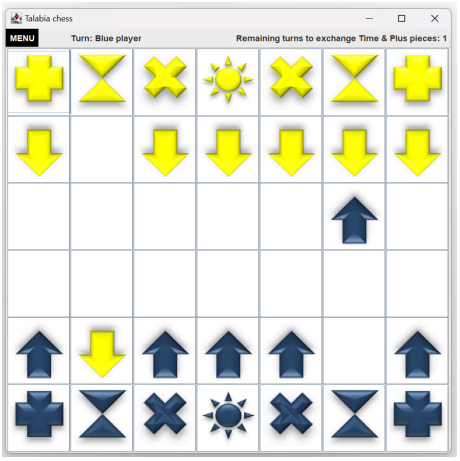
Sun piece movement



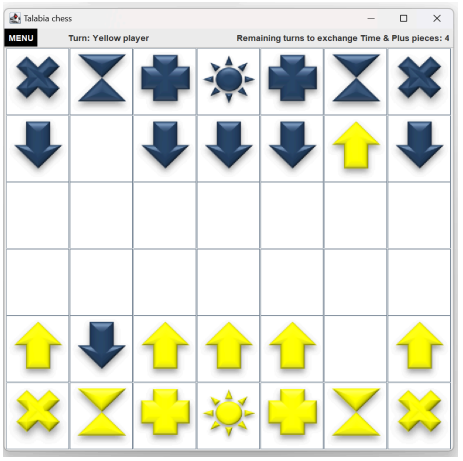
The game ends when a Sun piece is captured



Point piece changes direction after reaching board's end

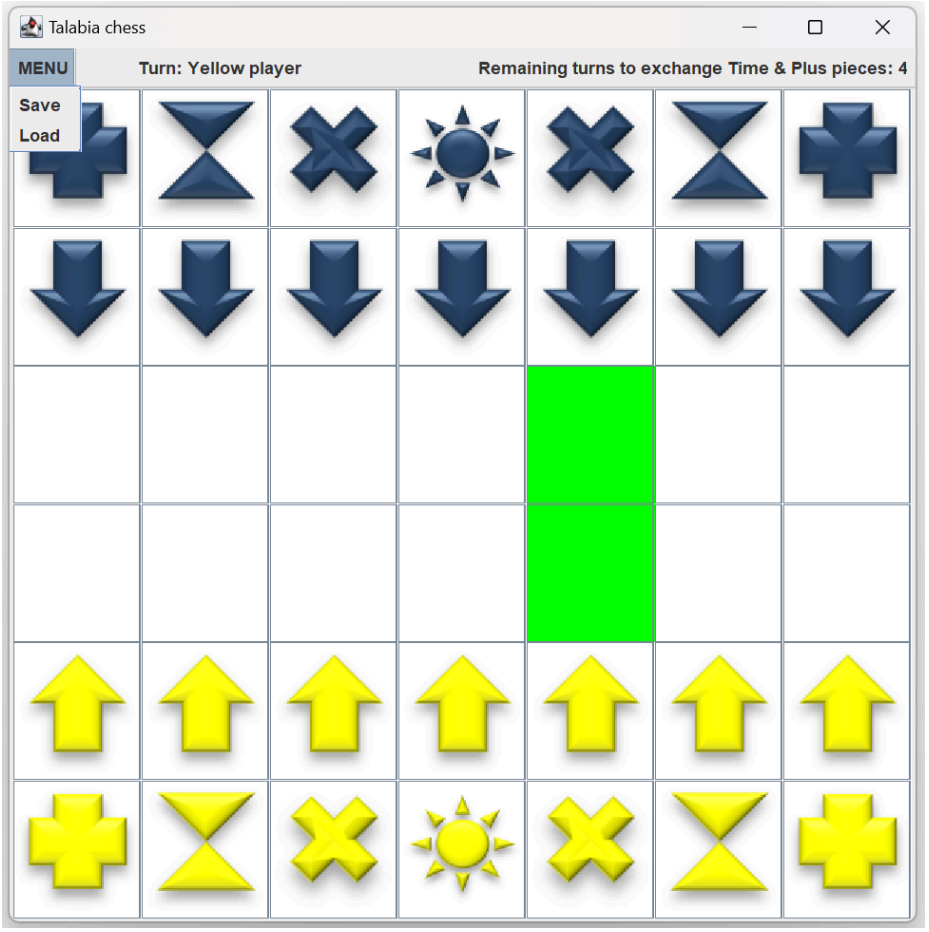


Before exchanging Time and Plus pieces

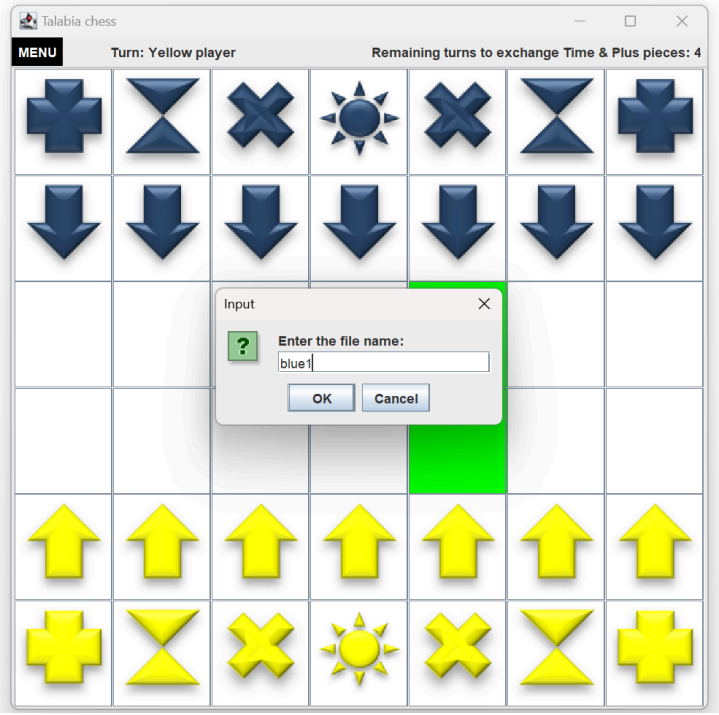


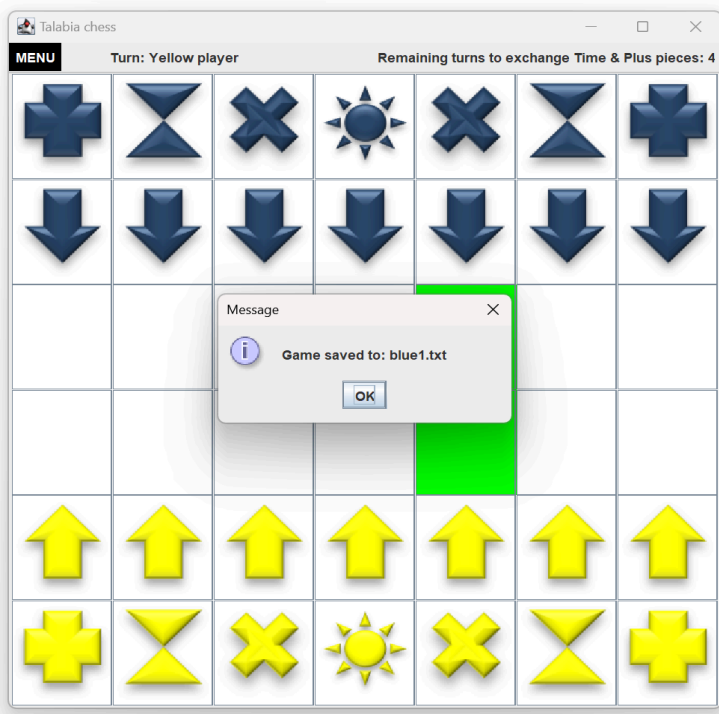
After exchanging Time and Plus pieces

There is a menu and two labels on top of the board. The leftmost “MENU” is the board's menu where the player can choose to save or load the game. The left label shows the current player’s turn, while the right label shows how many remaining turns to exchange the Time and Plus pieces.

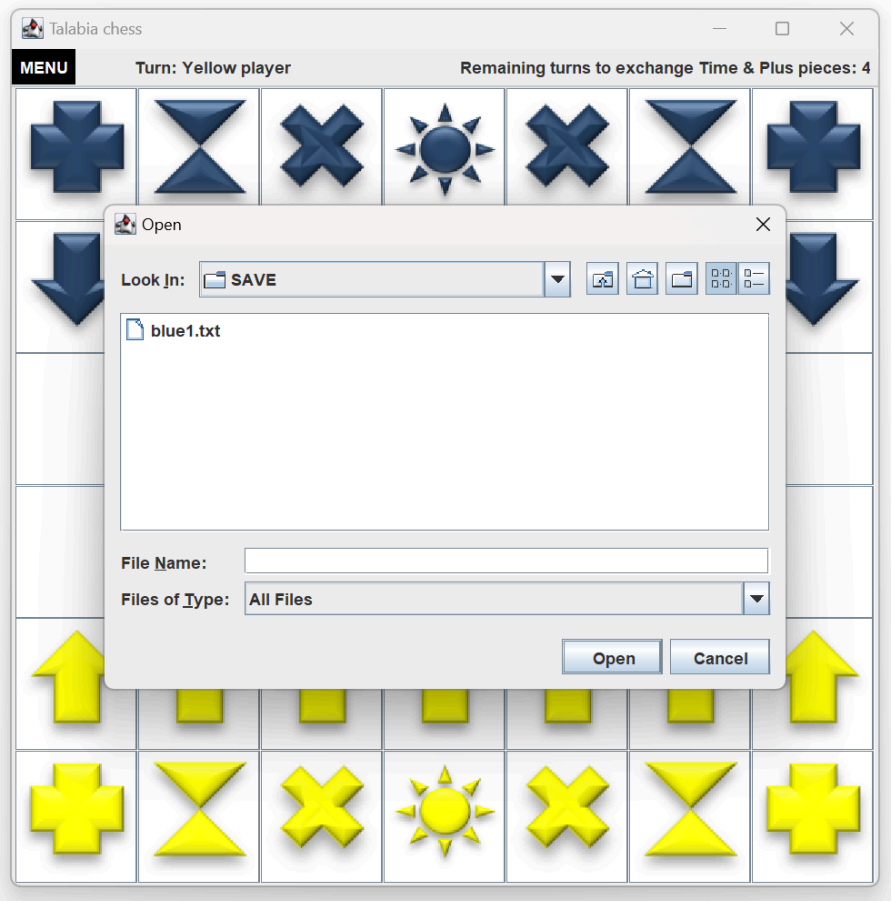


To save the game, the player can select the “MENU” on the top right corner and enter the desired file name for saving the current game. If there is a file name duplication, the player will have to replace it with the current file or enter a new file name. Upon successful file saving, a message will be displayed.

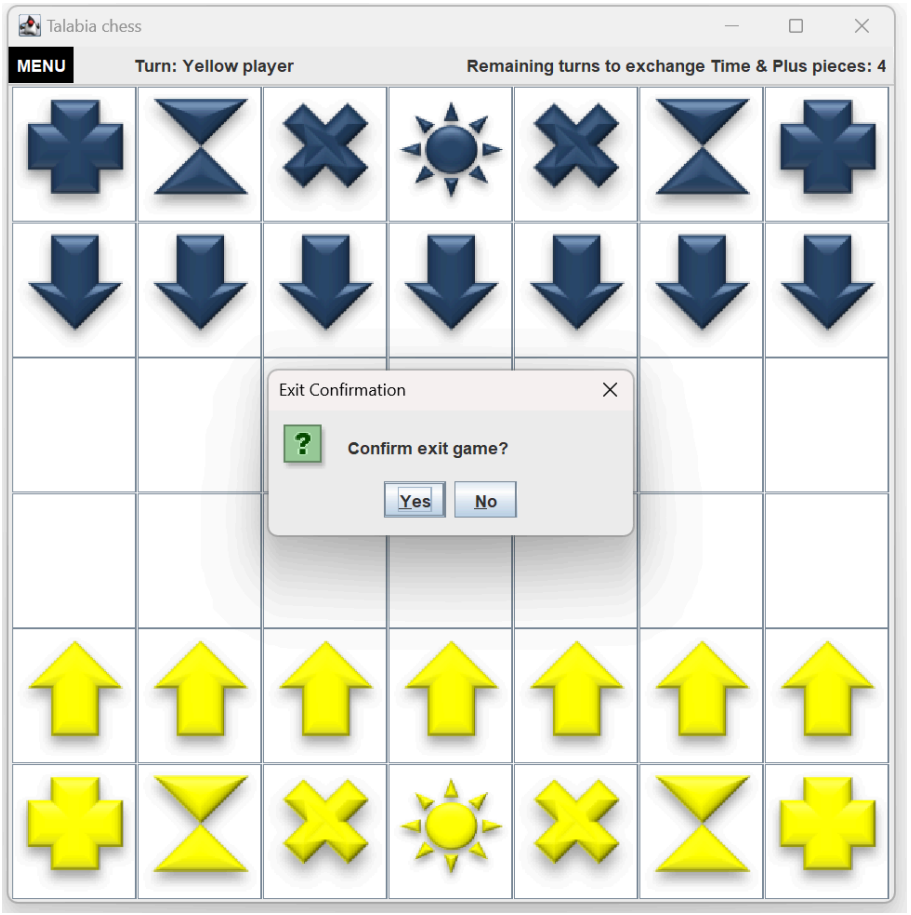




To load the game from a file, there will be a pop-up frame for the user to choose a file from all the saved games. The user will have to either enter the file name or click the corresponding file to load the game.



To exit the game, the player can click the ‘X’ symbol on the top right corner of the board. An exit confirmation message will be displayed before exiting the game.



TCP2201 Project Evaluation Form (40%)

Tutorial Section:	TT4L
Team Name:	Fiji
Group Leader:	Chan Kar Kin
Member	Ang Jin Nan
Member	Ng Yun Shi
Member	Chee Weng Kee
Member	Tai Qi Tong

Prototype and Presentation (20%)

Item	**Maximum marks**	**Actual Marks**
Comments, indentation, following proper Java naming conventions, other Java style issues.	3	
Object-oriented concepts like subclassing, delegation, composition, aggregation, polymorphism, etc.	4	
Appropriate use of Model-View-Controller, and at least one additional Design Pattern.	4	
User friendliness and appropriate GUI components used, windows resize properly and the board scales properly, menus still work during game play, the board flips for each player, etc.	5	
Functional requirements fulfilled, e.g. the board is set up correctly, players can play through a game properly, all the pieces move and transform correctly, winner is declared, save game, load saved game, etc.	9	
Total:	25	

Report (10%)

Item	Maximum marks	Actual Marks
Accurate and complete command line compile and run instructions.	1	
UML Class Diagram done and is coherent with the implementation.	4	
Use Case Diagram done and is coherent with the implementation.	3	
Sequence Diagrams for each use case done and is coherent with the implementation.	4	
User Documentation done and is coherent with the implementation.	3	
Total:	15	

Note: Individual marks will be adjusted after the lecturer interviews you, based on how much work each person did.