TCP2201 Project

Trimester 2310

by Team 1 (Argentina)

Team Leader: Ashley Sim Ci Hui, 014-9322912, 1211101022@student.mmu.edu.my

Team members:

Muhammad Zafran Bin Mohd Anuar, 018-3834584, 1211101321@student.mmu.edu.my

Javier Austin Anak Jawa, 019-3669511, 1211100857@student.mmu.edu.my

Thasharn Surein A/L Vinod Anand, 017-3455687, 1211101584@student.mmu.edu.my

Shashikaran A/L Kalaiarasu, 012-6101778, 1211102028@student.mmu.edu.my

A. Compile and run instructions
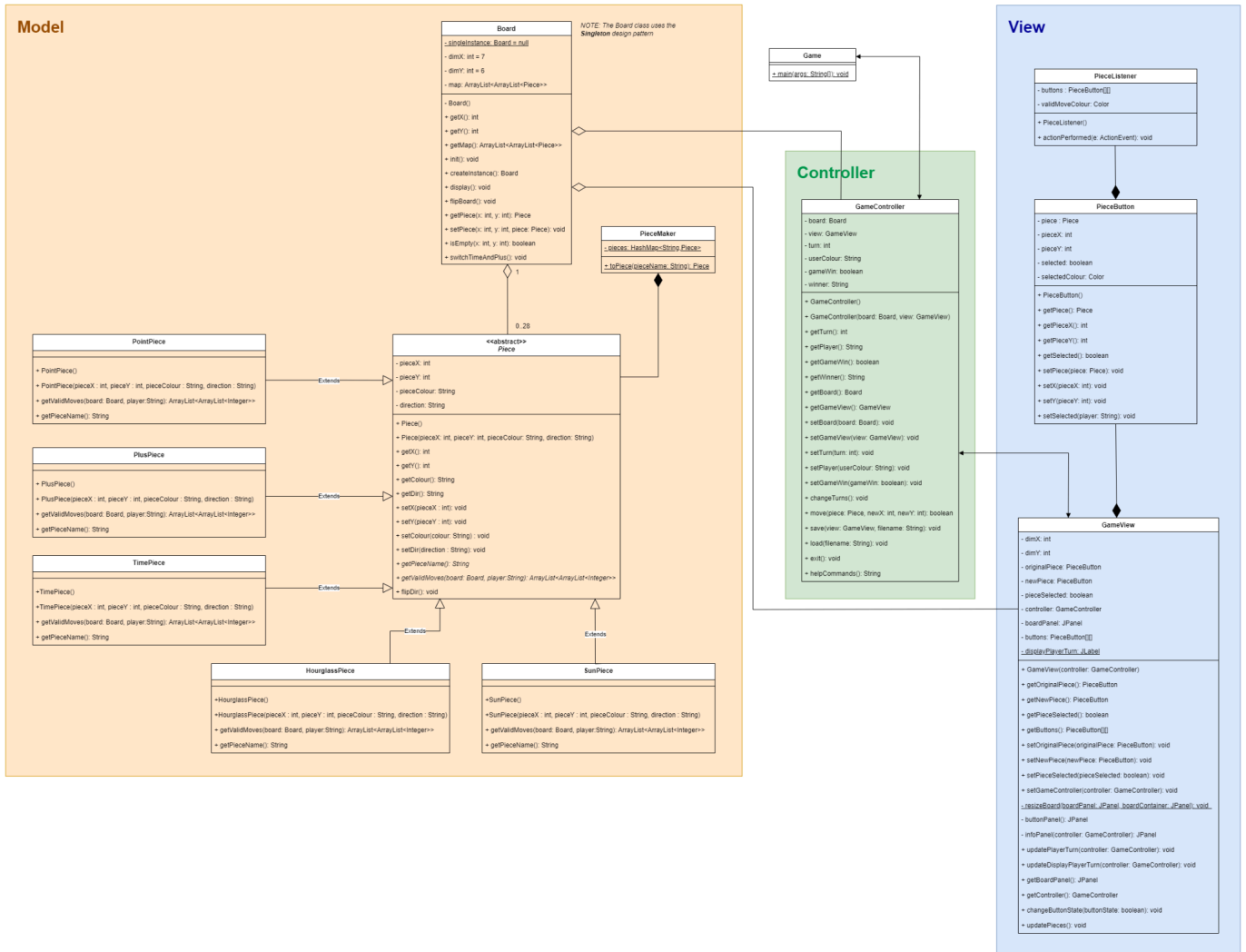
The following instructions assume that the user already has Java installed on their respective operating system. Instructions are the same for Windows, Mac and Linux operating systems.

1. Open up the command line interface (CLI).
2. Navigate to the directory where the game files are located.
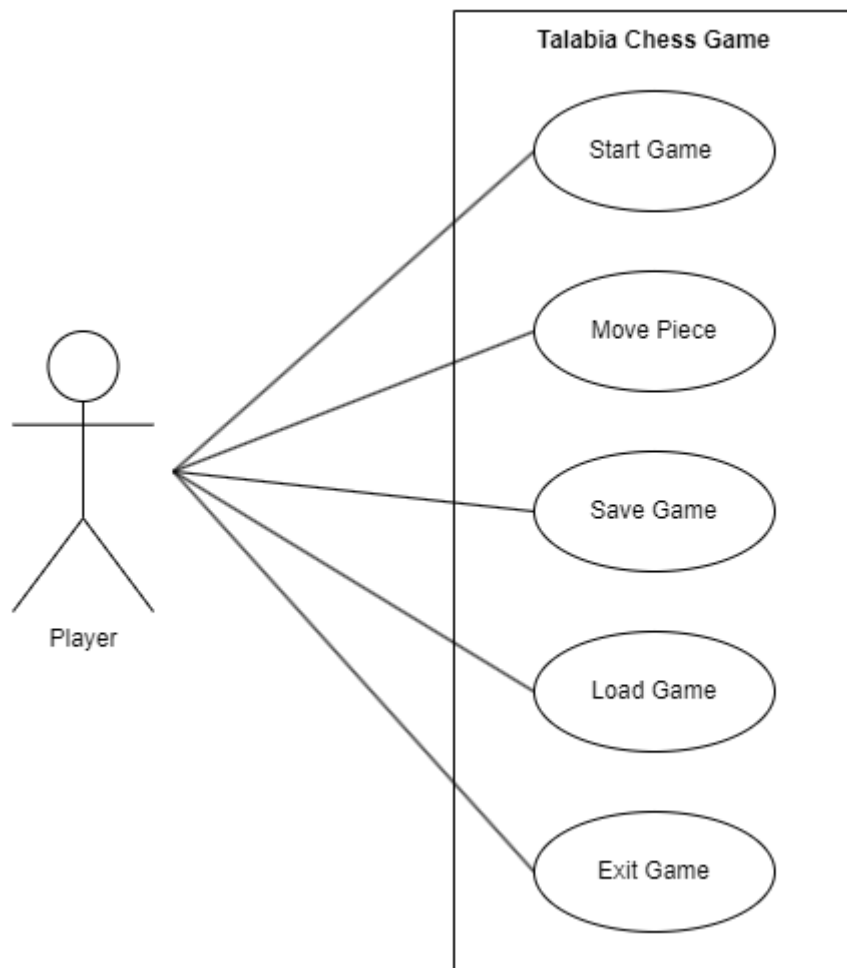3. Run the following commands:

```
> javac Game.java
> java Game
```

4. Play the game.

B. UML Class Diagram
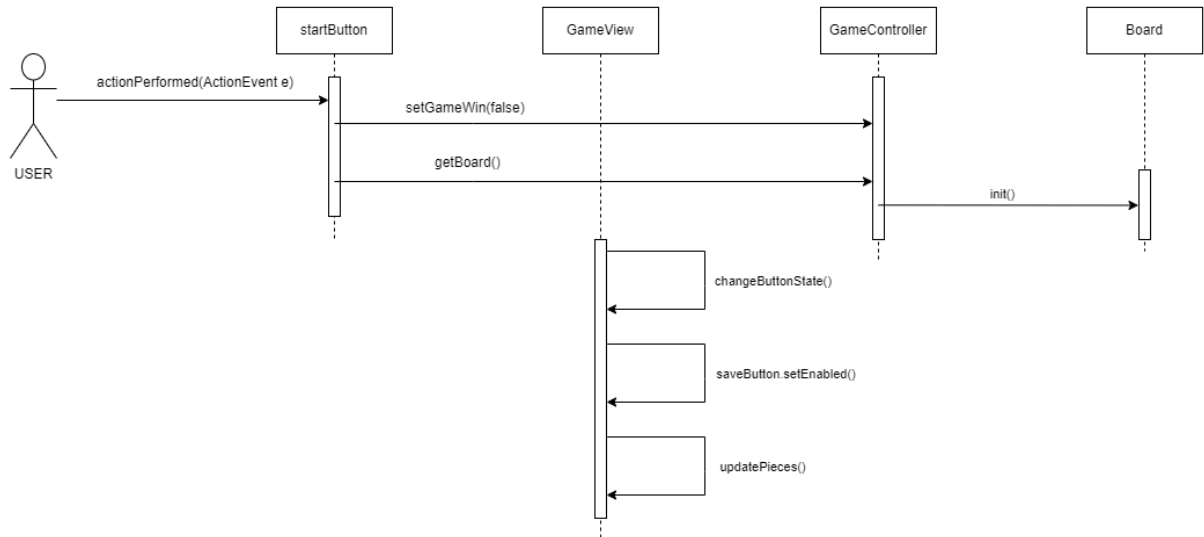
Model

Board
*NOTE: The Board class uses the **Singleton** design pattern*

- singleInstance: Board = null
- dimX: int = 7
- dimY: int = 6
- map: ArrayList<ArrayList<Piece>>

- Board()
+ getX(): int
+ getY(): int
+ getMap(): ArrayList<ArrayList<Piece>>
+ init(): void
+ createInstance(): Board
+ display(): void
+ flipBoard(): void
+ getPiece(x: int, y: int): Piece
+ setPiece(x: int, y: int, piece: Piece): void
+ isEmpty(x: int, y: int): boolean
+ switchTimeAndPlus(): void

Game
+ main(args: String[]): void

PieceMaker
- pieces: HashMap<String, Piece>
+ toPiece(pieceName: String): Piece

PointPiece
+ PointPiece()
+ PointPiece(pieceX : int, pieceY : int, pieceColour : String, direction : String)
+ getValidMoves(board: Board, player:String): ArrayList<ArrayList<Integer>>
+ getPieceName(): String

PlusPiece
+ PlusPiece()
+ PlusPiece(pieceX : int, pieceY : int, pieceColour : String, direction : String)
+ getValidMoves(board: Board, player:String): ArrayList<ArrayList<Integer>>
+ getPieceName(): String

TimePiece
+ TimePiece()
+ TimePiece(pieceX : int, pieceY : int, pieceColour : String, direction : String)
+ getValidMoves(board: Board, player:String): ArrayList<ArrayList<Integer>>
+ getPieceName(): String

<> Piece
- pieceX: int
- pieceY: int
- pieceColour: String
- direction: String

+ Piece()
+ Piece(pieceX: int, pieceY: int, pieceColour: String, direction: String)
+ getX(): int
+ getY(): int
+ getColour(): String
+ getDir(): String
+ setX(pieceX : int): void
+ setY(pieceY : int): void
+ setColour(colour: String) : void
+ setDir(direction : String): void
+ getPieceName(): String
+ getValidMoves(board: Board, player:String): ArrayList<ArrayList<Integer>>
+ flipDir(): void

HourglassPiece
+HourglassPiece()
+HourglassPiece(pieceX : int, pieceY : int, pieceColour : String, direction : String)
+ getValidMoves(board: Board, player:String): ArrayList<ArrayList<Integer>>
+ getPieceName(): String

SunPiece
+SunPiece()
+SunPiece(pieceX : int, pieceY : int, pieceColour : String, direction : String)
+ getValidMoves(board: Board, player:String): ArrayList<ArrayList<Integer>>
+ getPieceName(): String

Controller

GameController
- board: Board
- view: GameView
- turn: int
- userColour: String
- gameWin: boolean
- winner: String

+ GameController()
+ GameController(board: Board, view: GameView)
+ getTurn(): int
+ getPlayer(): String
+ getGameWin(): boolean
+ getWinner(): String
+ getBoard(): Board
+ getGameView(): GameView
+ setBoard(board: Board): void
+ setGameView(view: GameView): void
+ setTurn(turn: int): void
+ setPlayer(userColour: String): void
+ setGameWin(gameWin: boolean): void
+ changeTurns(): void
+ move(piece: Piece, newX: int, newY: int): boolean
+ save(view: GameView, filename: String): void
+ load(filename: String): void
+ exit(): void
+ helpCommands(): String

View

PieceListener
- buttons : PieceButton[][]
- validMoveColour: Color
+ PieceListener()
+ actionPerformed(e: ActionEvent): void

PieceButton
- piece : Piece
- pieceX: int
- pieceY: int
- selected: boolean
- selectedColour: Color

+ PieceButton()
+ getPiece(): Piece
+ getPieceX(): int
+ getPieceY(): int
+ getSelected(): boolean
+ setPiece(piece: Piece): void
+ setX(pieceX: int): void
+ setY(pieceY: int): void
+ setSelected(player: String): void

GameView
- dimX: int
- dimY: int
- originalPiece: PieceButton
- newPiece: PieceButton
- pieceSelected: boolean
- controller: GameController
- boardPanel: JPanel
- buttons: PieceButton[][]
- displayPlayerTurn: JLabel

+ GameView(controller: GameController)
+ getOriginalPiece(): PieceButton
+ getNewPiece(): PieceButton
+ getPieceSelected(): boolean
+ getButtons(): PieceButton[][]
+ setOriginalPiece(originalPiece: PieceButton): void
+ setNewPiece(newPiece: PieceButton): void
+ setPieceSelected(pieceSelected: boolean): void
+ setGameController(controller: GameController): void
+ resizeBoard(boardPanel: JPanel, boardContainer: JPanel): void
- buttonPanel(): JPanel
- infoPanel(controller: GameController): JPanel
+ updatePlayerTurn(controller: GameController): void
+ updateDisplayPlayerTurn(controller: GameController): void
+ getBoardPanel(): JPanel
+ getController(): GameController
+ changeButtonState(buttonState: boolean): void
+ updatePieces(): void

Extends

1

0..28

C. Use Case Diagram

Talabia Chess Game

- Start Game
- Move Piece
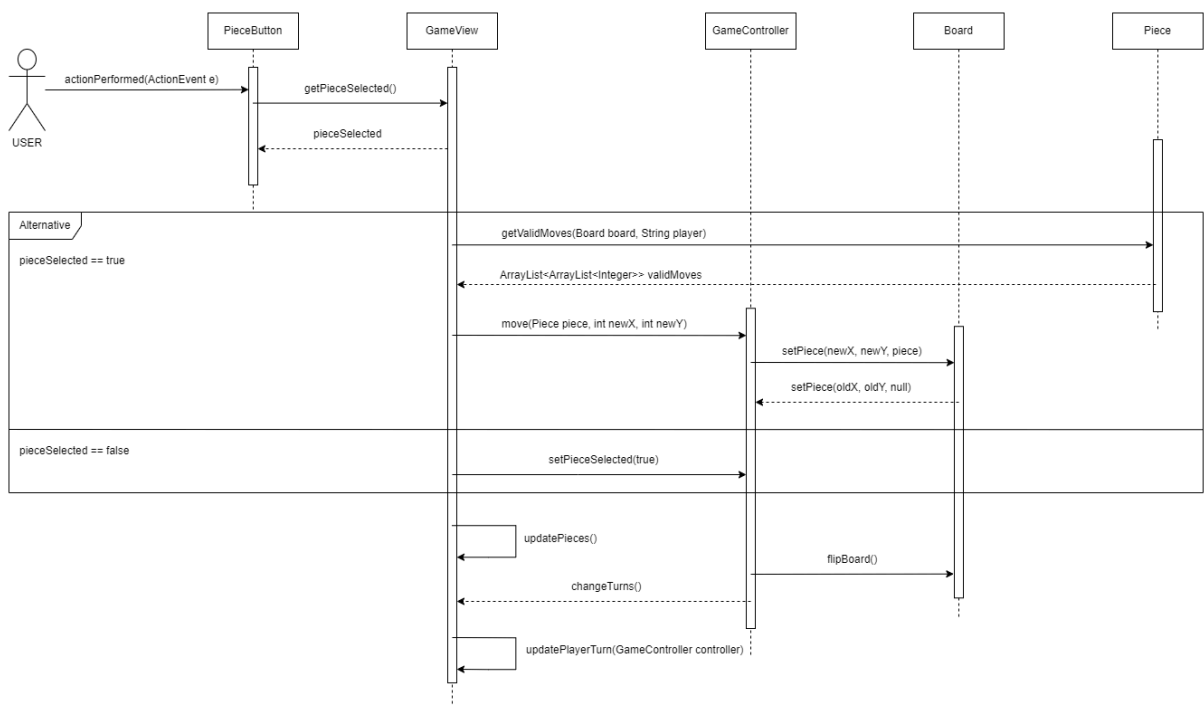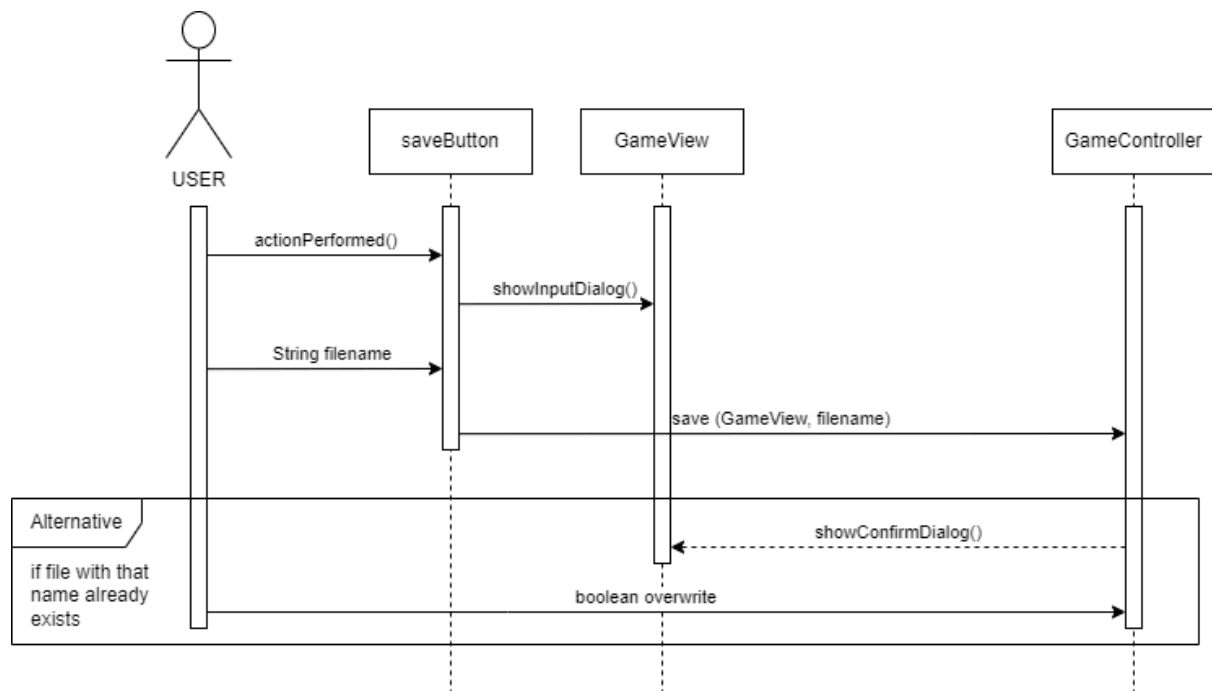- Save Game
- Load Game
- Exit Game

Player

D. Sequence Diagrams

1. Start Game

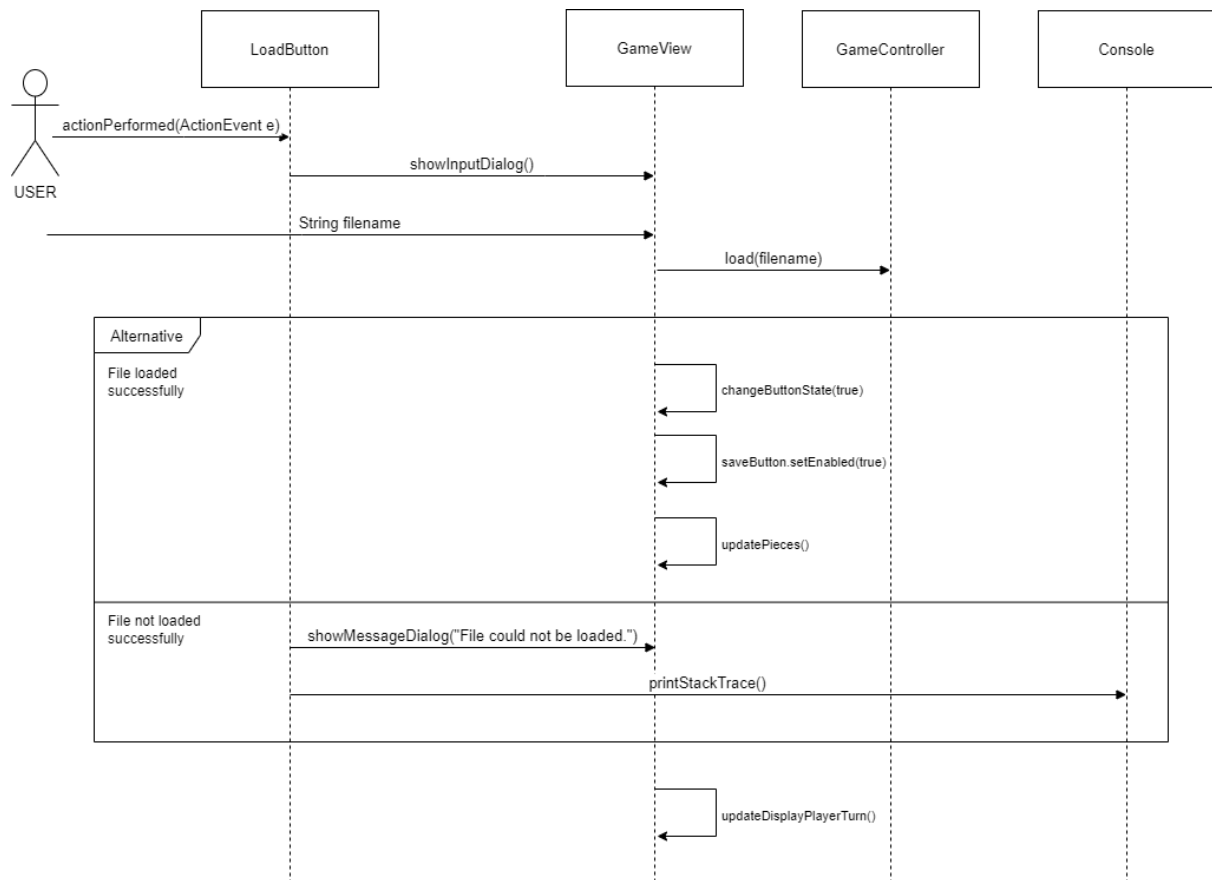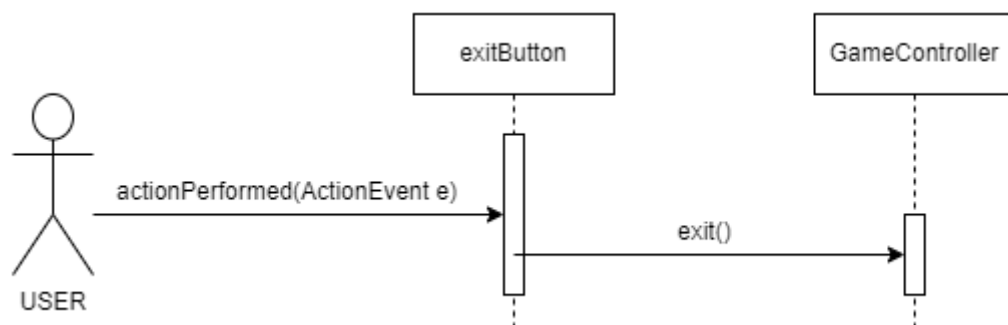
2. Move Piece

3. Save Game

4. Load Game

5. Exit Game

1. **Piece Rules**

    ● **Point Piece**
    - The Point piece can only move forward by 1 or 2 steps.
    - If it reaches the end of the board, it turns around and starts heading back the other way.
    - It cannot skip over other pieces.

    ● **Hourglass Piece**
    - The Hourglass piece moves in a 3x2 L shape in any orientation.
    - This is the only piece that can skip over other pieces.

    ● **Time Piece**
    - The Time piece can only move diagonally but can go any distance.
    - It cannot skip over other pieces.

    ● **Plus Piece**
    - The Plus piece can move horizontally and vertically only but can go any distance.
    - It cannot skip over other pieces.

    ● **Sun Piece**
    - The Sun piece can move only one step in any direction.
    - The game ends when the Sun piece is captured by the opponent.

    ● **Time and Plus piece switch every two rounds of the game.**
    - After 2 rounds (counting one yellow move and one blue move as one round), all Time pieces will turn into Plus pieces, and all Plus pieces will turn into Time pieces.

2. Game Rules

- **How to start the game?**
  - To start a new game, the user must click on the "Start" button and the game will begin.

- **Who is the first player to start the game?**
  - Upon clicking the "Start" button, player Yellow will automatically be the first player to begin the game. After player Yellow makes their move, the board will flip and it will be player Blue's turn.
  - If the user decides to continue playing from a saved game, the first player will be the player who has not made their move before the game was saved.

- **How to win the game?**
  - Either player Blue or Yellow must capture the opponent's Sun piece.
  - The player who is the first to capture the opponent's Sun piece wins the game.

3. **Button Functionality**

- **Start Button**
  - By clicking the start button, the board will load with the pieces and the game will begin.
  - Clicking the start button after a game has already begun will start a new game.

- **Save Button**
  - By clicking the save button, a pop-up window will appear to ask for user input a file name to name the save file.
  - After the user inputs their preferred file name, the file will then be saved into a text file (.txt) format.

- **Load Button**
  - By clicking the load button, a pop-up window will appear to ask for user input the saved file name.
  - After the user inputs the saved file name, the game will load from its original saved state and the user can continue to play the game until the Sun piece is captured.

- **Help Button**
  - By clicking the help button, there will be a pop up that displays the game rules and how the pieces move.

- **Exit Button**
  - By clicking the exit button, the game will end and the window will close.