# The Problem

* Apply Object Oriented (OO) design principles and patterns to develop a modified chess game.
* Chess game is similar to a normal game:
  + 6 x 6 size board
  + Three pieces:
    - Bishop, Rook and Knight.
    - move characteristics are identical to the game of chess.
    - Each pieces can also combine with other piece of the same colour but different type.
    - Combining can only occur when the pieces reside on the same tile with a combination piece taking on the move characteristics of each combined piece.
    - The pieces are taken by the opponent in the normal chess manner, for a taken piece five points is awarded.
  + In addition to the standard chess pieces, barriers are placed to limit the initial movements. A barrier is automatically removed when a piece lands over it and the player collects 1 point.”
* The flow of the game is such that a game will run for n number of moves ().
* For each turn the active player is limited to 30 seconds to make a move, upon the time limit being reached the turn will be forfeited. At the end of n turn the player with the most points will be declared the winner.
* In the development of the above problems the development team are to demonstrate the use SOLID and Grasp principles, refactoring to achieve code quality guidelines, design by contract, and separate the design into Model, View and Controller.

# Assumptions

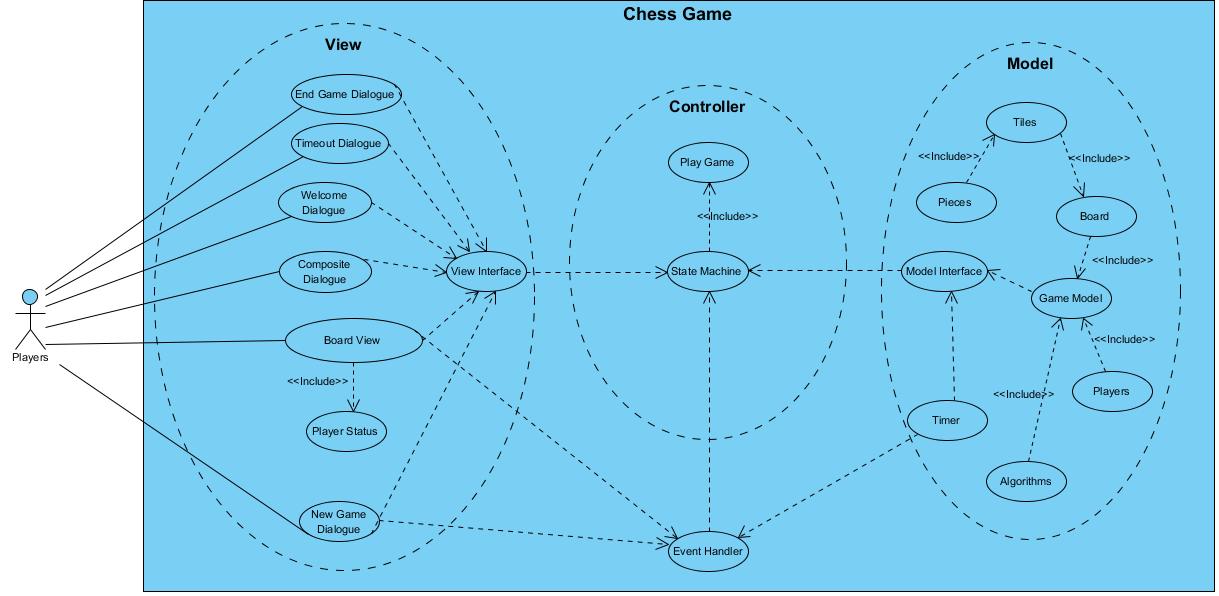
# Specifications

* **One Chess Board**
  + Characteristics:
  + Shape: Square
  + Design: 6 columns by 6 rows with a checkers patterns using black and white
  + Output: When a player has a turn and selects one of their pieces all valid moves are to be highlighted for the individual or combined piece.

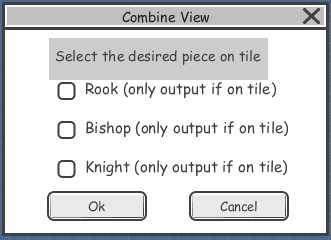
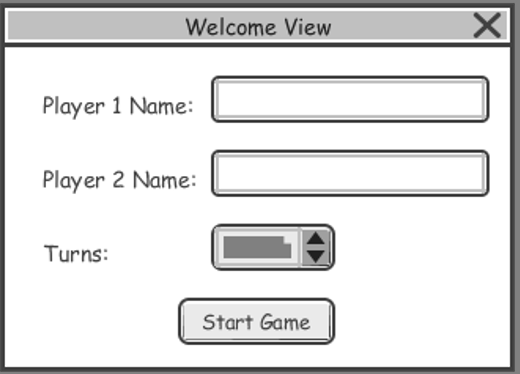
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Rook** | **Bishop** | **Knight** | **Barrier** |
| **Shape** | Castle | Bishop's Hat | Horse's Head | Star |
| **Movement** | May move Vertically or Horizontally n co-ordinates until the end of the board is reached or until another piece is reached, in which case move into that pieces co-ordinate. | May move Diagonally n co-ordinates until the end of the board is reached or until another piece is reached, in which case move into that pieces co-ordinate. | May only move two co-ordinates vertical and horizontal and one co-ordinate perpendicular (L), only if result is still on the chess board. | Nil |
| **Combine** | When moved to another piece of the same colour and the piece does not consist of any variation of castle, combine. | When moved to another piece of the same colour and the piece does not consist of any variation of Bishop, combine. | When moved to another piece of the same colour and the piece does not consist of any variation of Knight, combine | Nil |
| **When Taken** | update the opponents score by 5, remove castle from chess board (maybe place in some container?) | update the opponents score by 5, remove Bishop from chess board (maybe place in some container?) | update the opponents score by 5, remove Knight from chess board (maybe place in some container?) | update the opponents score by 1, remove castle from board (maybe place in some container?) |
| **Colour** | May be black or white | May be black or white | May be black or white | Blue |
| **Quantity** | 2 Black, 2 White | 2 Black, 2 White | 2 Black, 2 White | 12 |
|  |  |  |  |  |
| **Start Location** | Black, one in each corner of the top of Chess Board, (0,0) and (5,0) White, one in each corner of the bottom of Chess Board, (0,5) and (5,5) | Black, one in each in the co-ordinate one in from the corner at the top of Chess Board, (1,0) and (4,0) White, one in each in the co-ordinate one in from the corner at the bottom of Chess Board, (1,5) and (4,5) | Black, one on the black centre co-ordinate, one on the white centre co-ordinate, at the top of Chess Board, (2,0) and (3,0) White, one on the black centre co-ordinate, one on the white centre co-ordinate, at the bottom of Chess Board, (2,5) and (3,5) | Fill rows 2 and 3 (initial row = 0) |

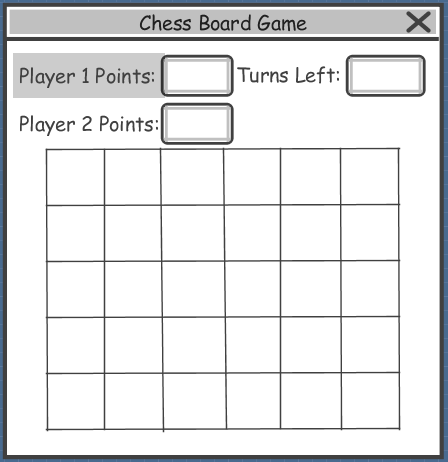
* **Combine:**
  + **Occurrence**: When a piece moves into a co-ordinate occupied by a piece of the same colour which is not the same type as itself they are known as combined.
  + **Selecting Combination/Spliting**: For a players turn if they select a combined piece they will be prompted to move with a single piece of the combination, or the entire combination
  + **Movement:** For the selected combination, moves of that type are valid
  + **When taken:** update opponent score by killing individual pieces.
  + **Shape:** the shapes are reduced in size and painted onto co-ordinate
  + **Colour:** Same as Piece
* **Game Rules**
  + **Number of Turns:** 
    - Select: User input before start of game, locked when game starts
    - Range: 10 - 50
  + **Playing Order**
    - Start: The player with the white pieces always moves first (chess rules)
    - Continued play: After start every turn is alternate between black and white pieces
  + **Game End**
    - End: When both players have had n turns
    - Winner: Player with the most points wins
* **Visual**
  + Must have a Chess board
  + Pieces
  + output number of turns
  + Input to select number of turns
  + Button to start game
  + Score Display

# Use Case Diagram



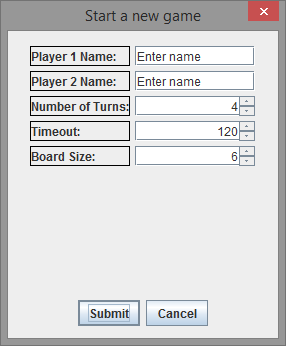
# View wireframes

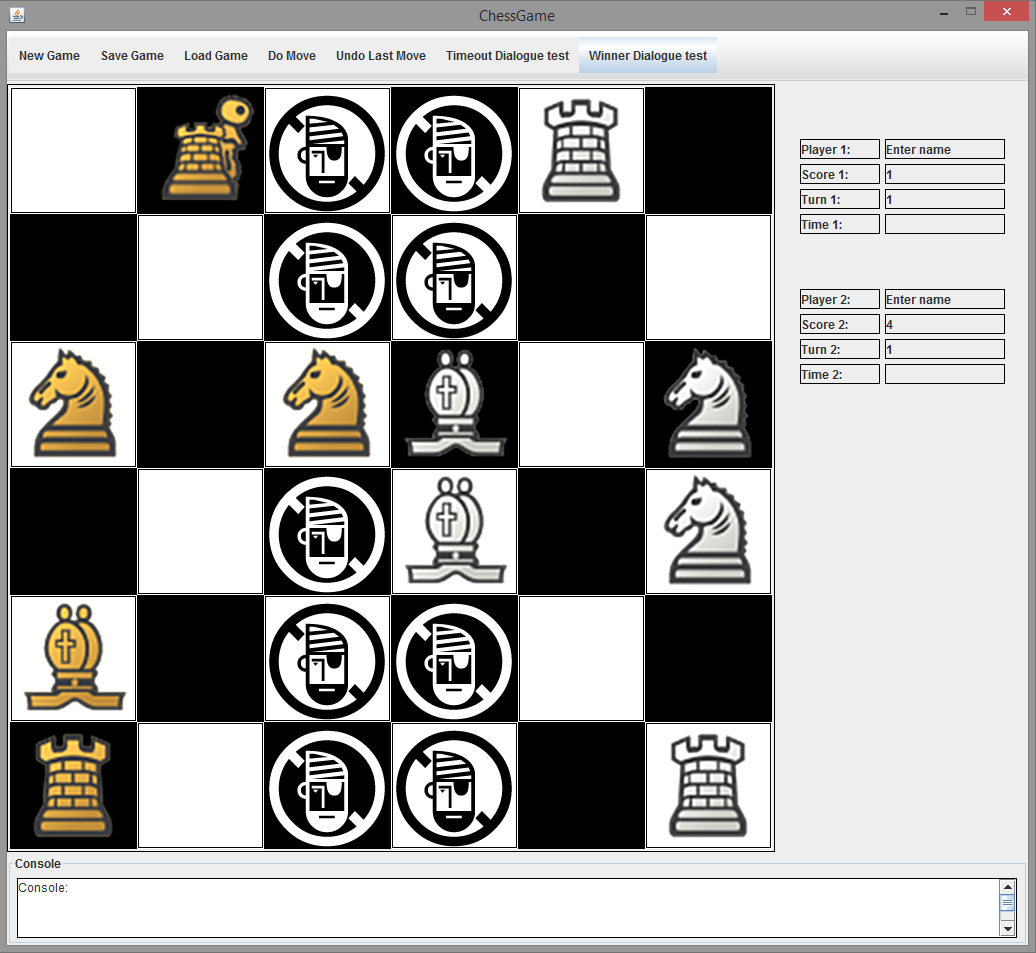


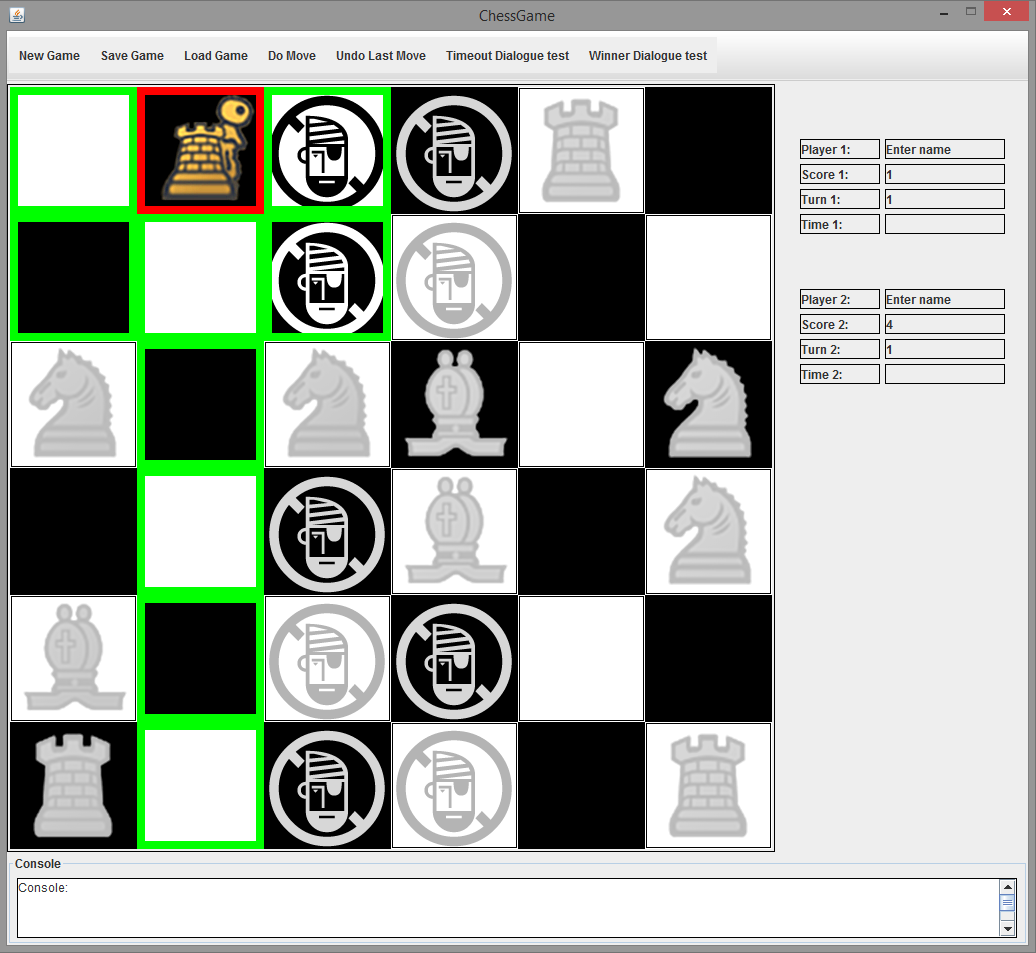
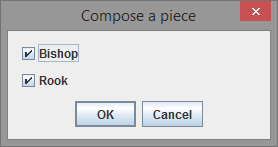




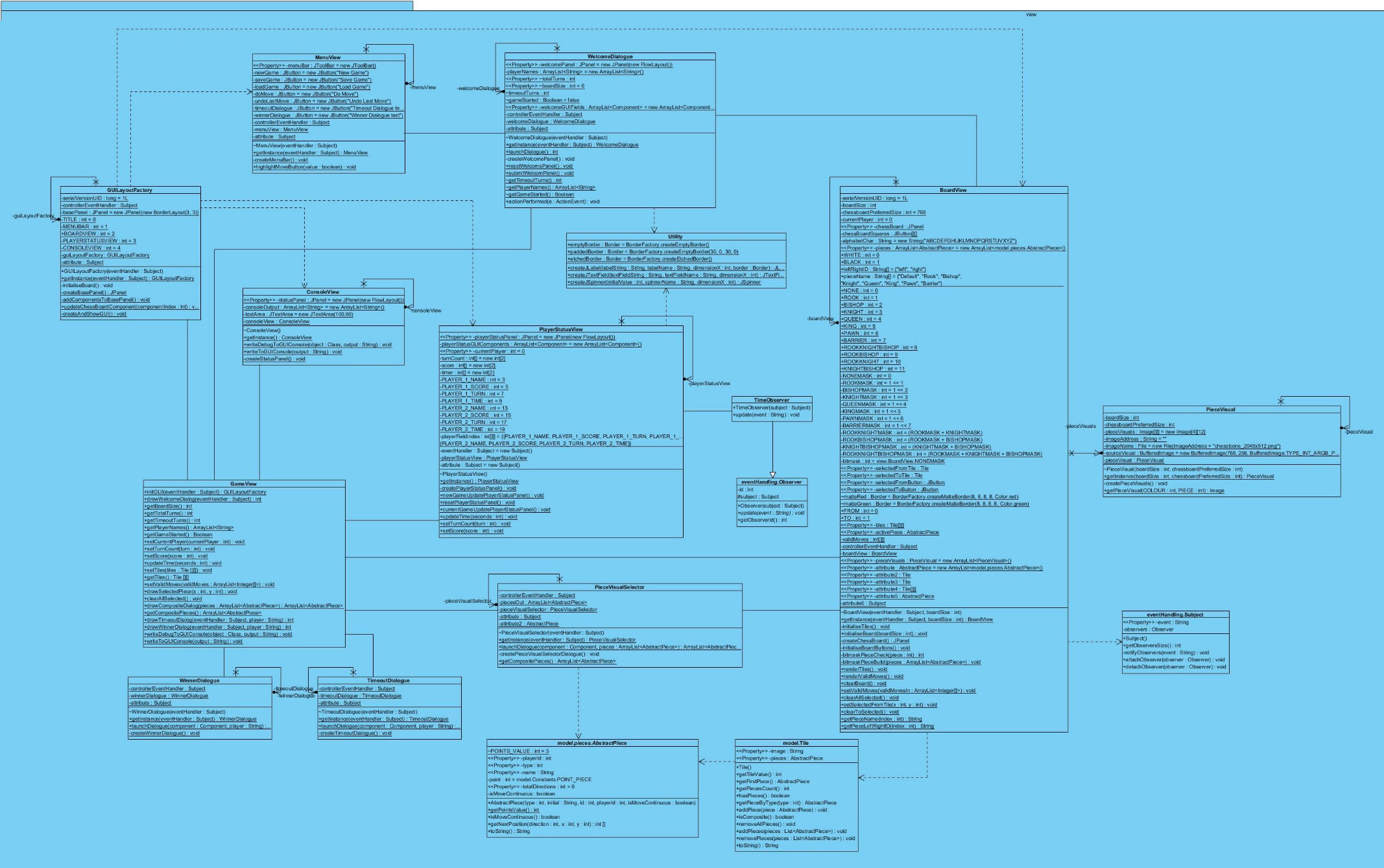
# View

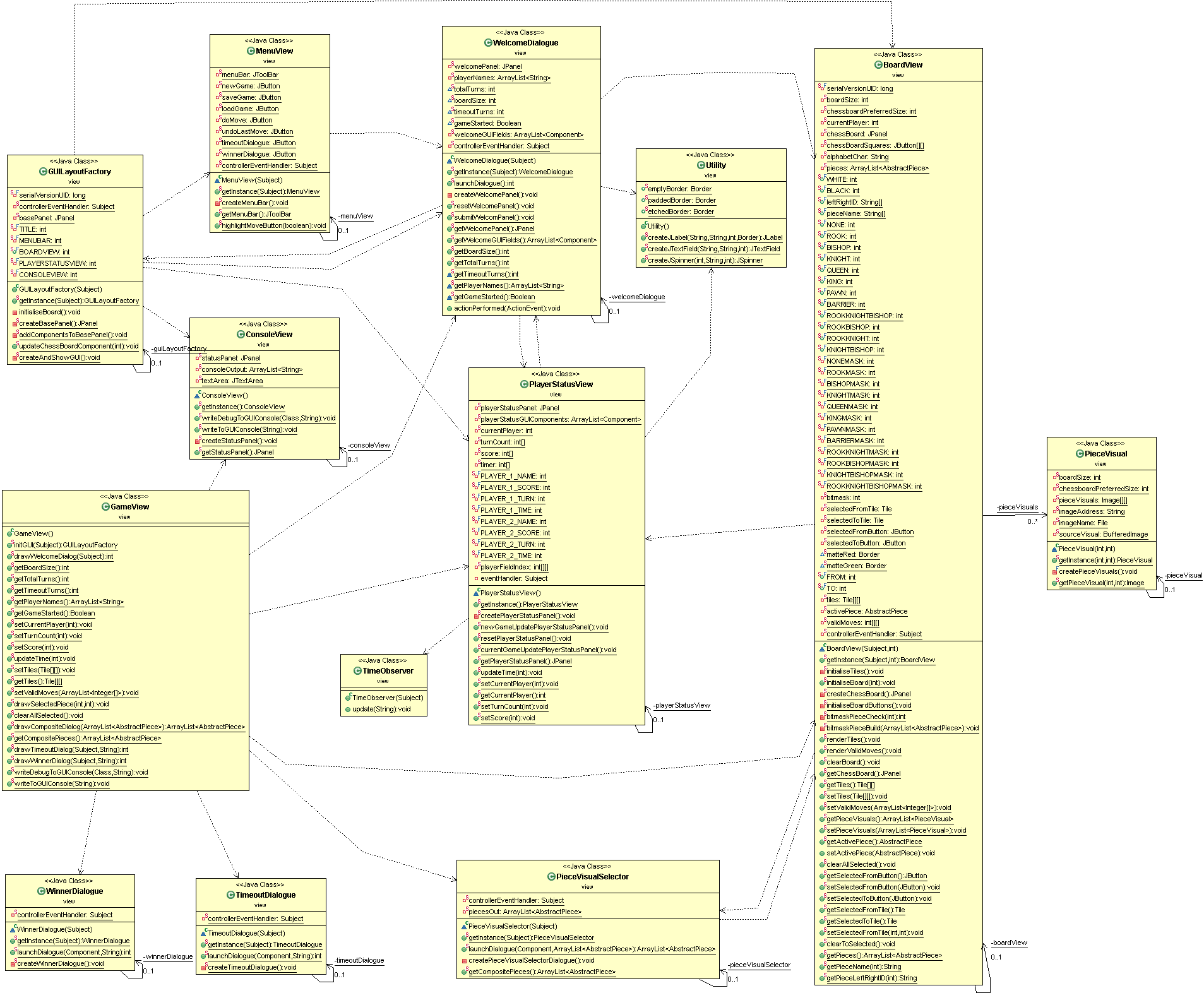




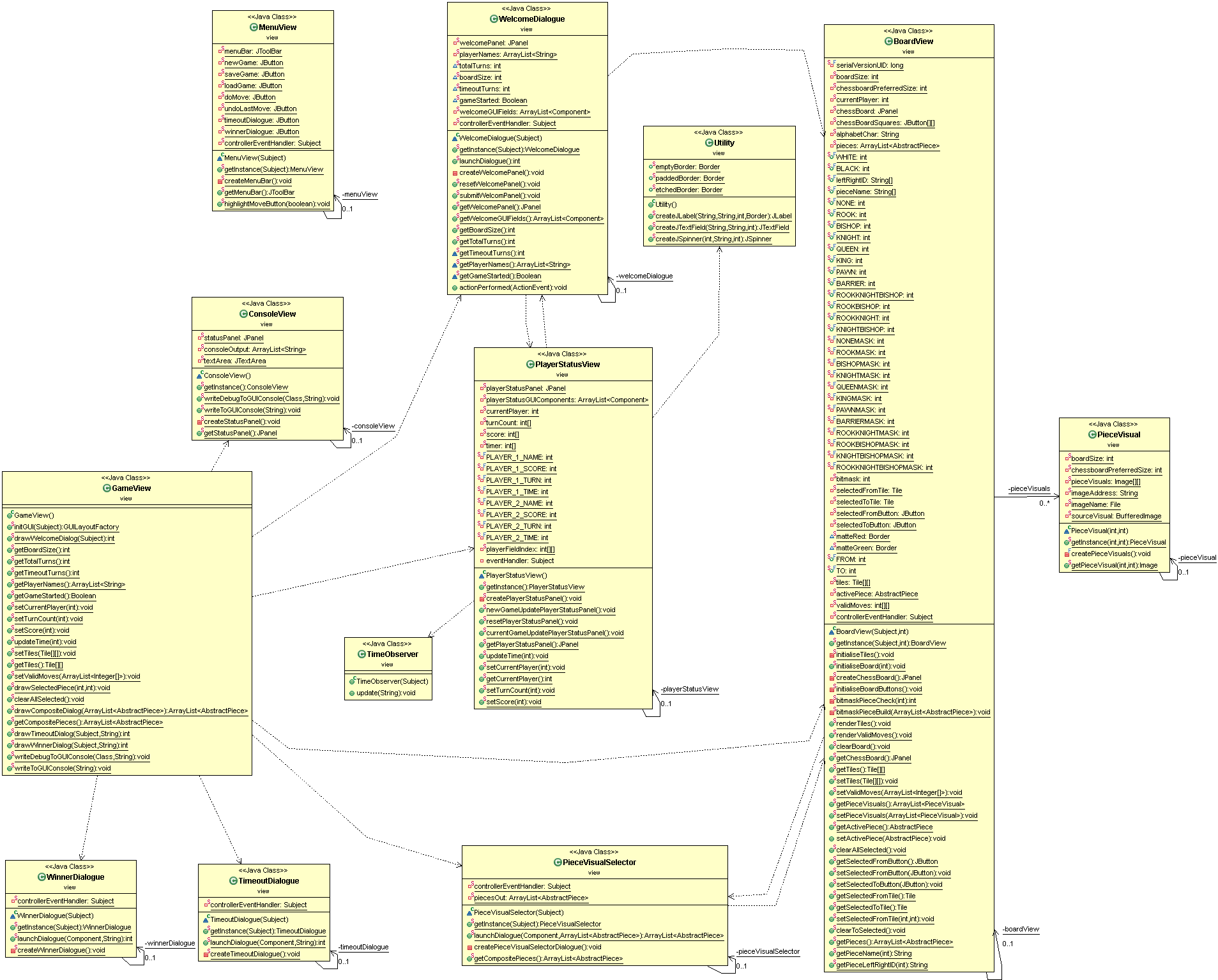


# View Class Diagram

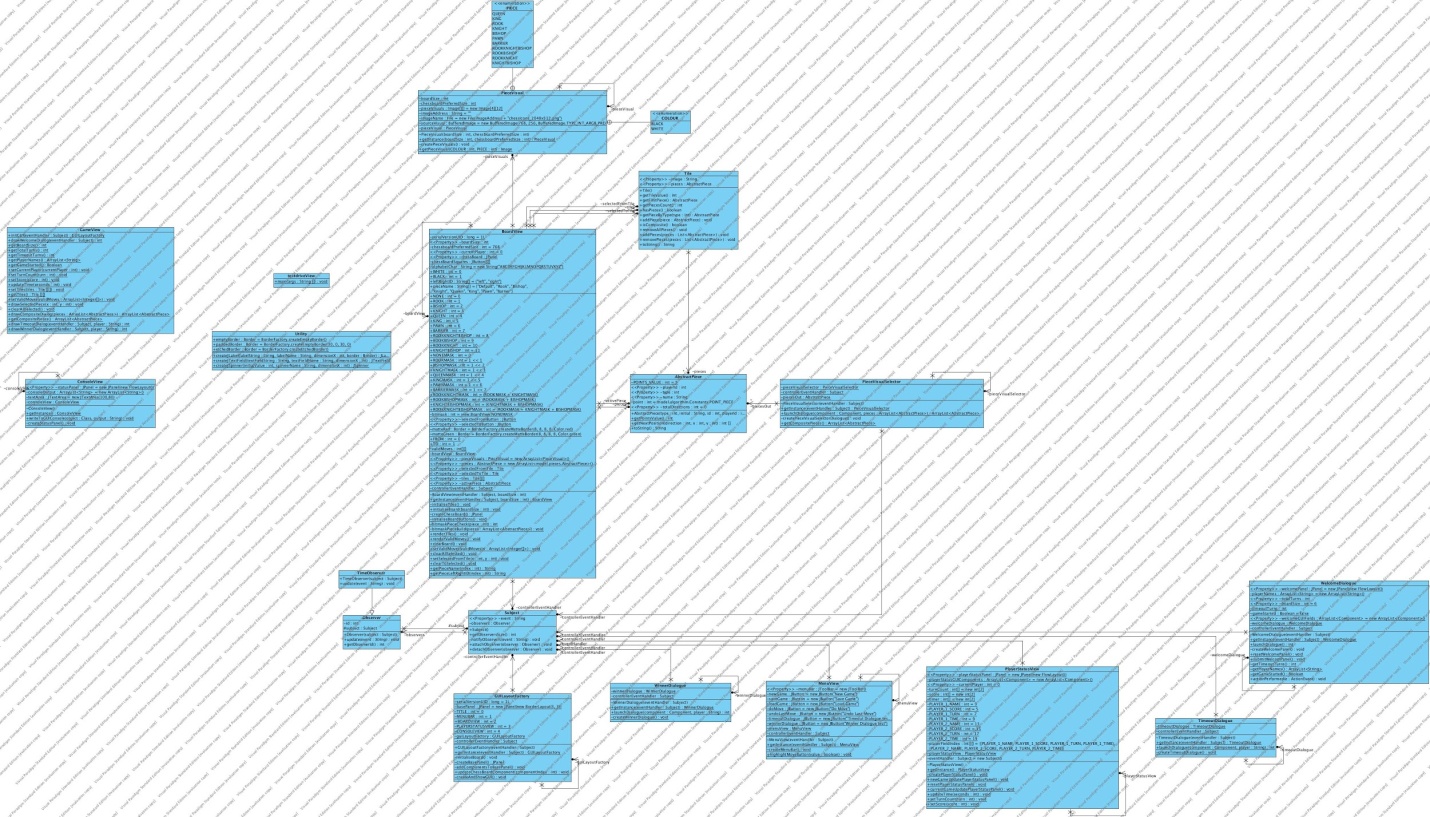




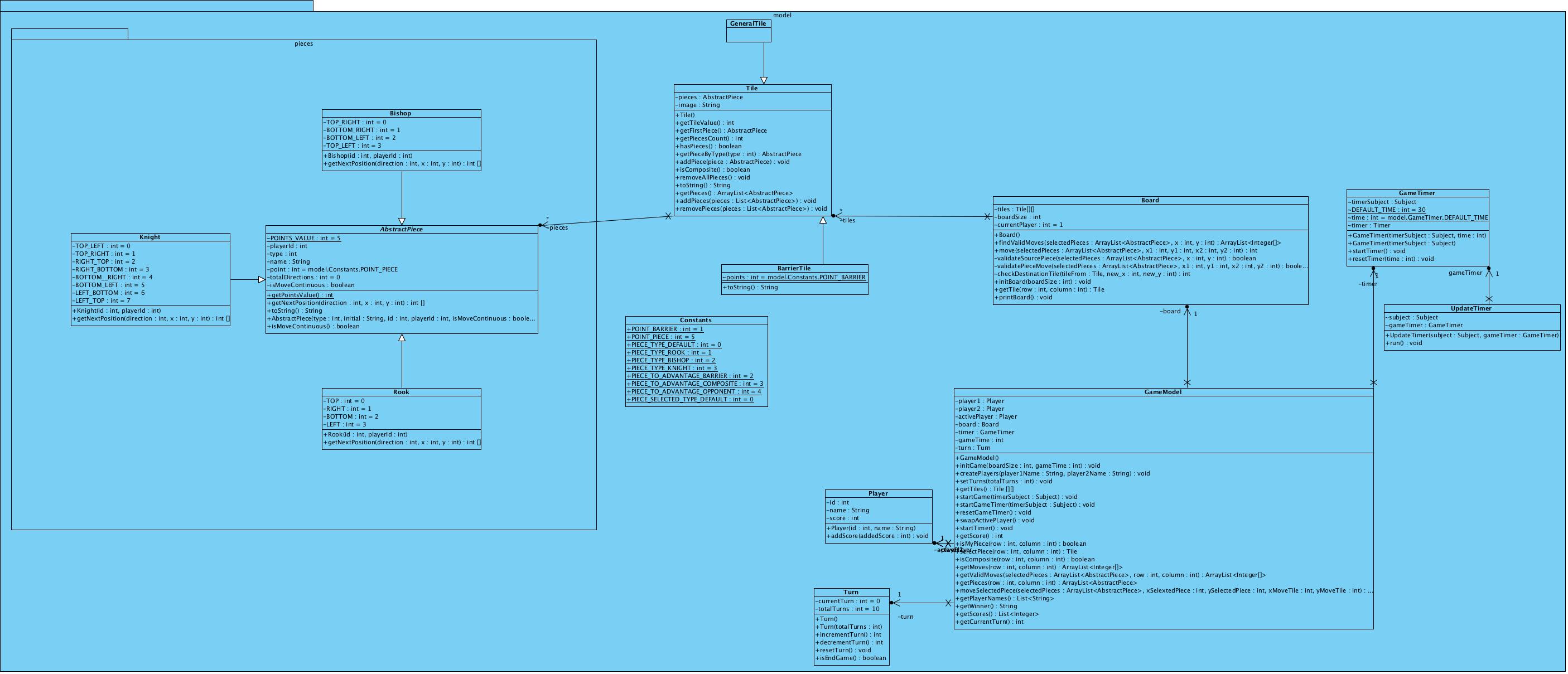
# View Class Diagram - post init



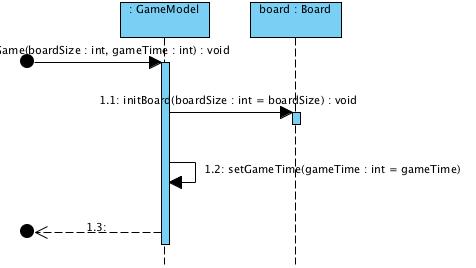
View Class Diagram



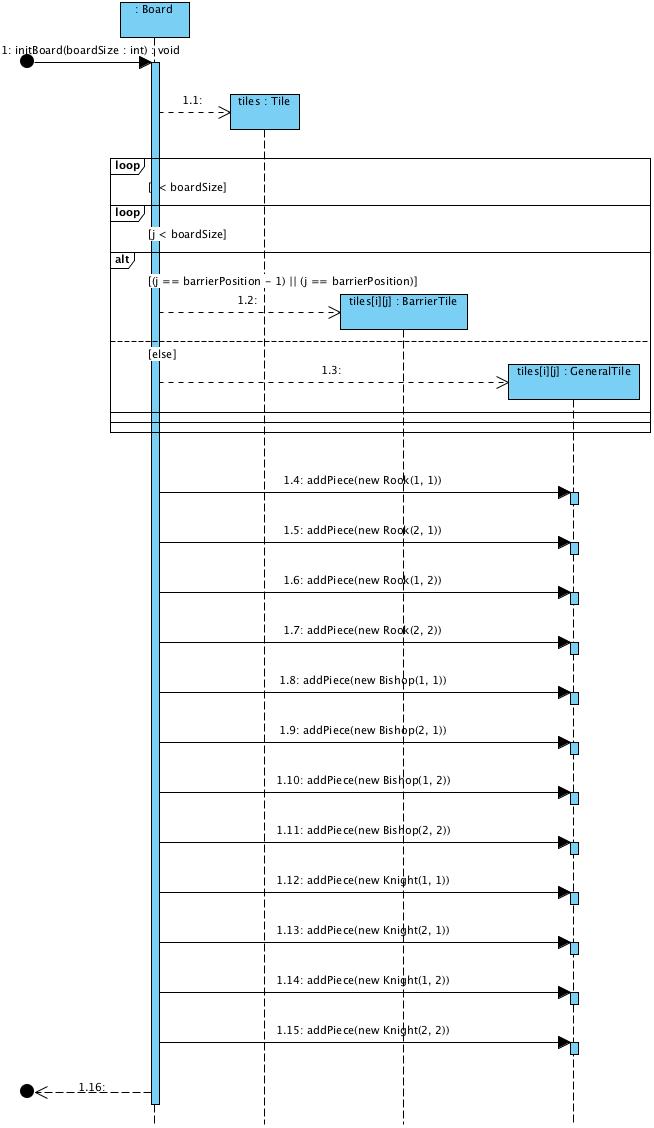
# Model – Class Diagram



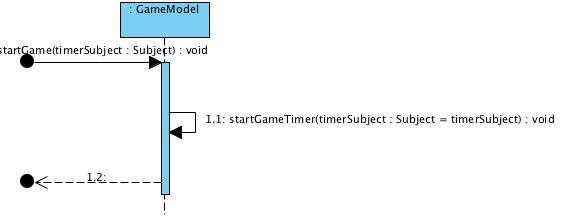
# Initialise Game



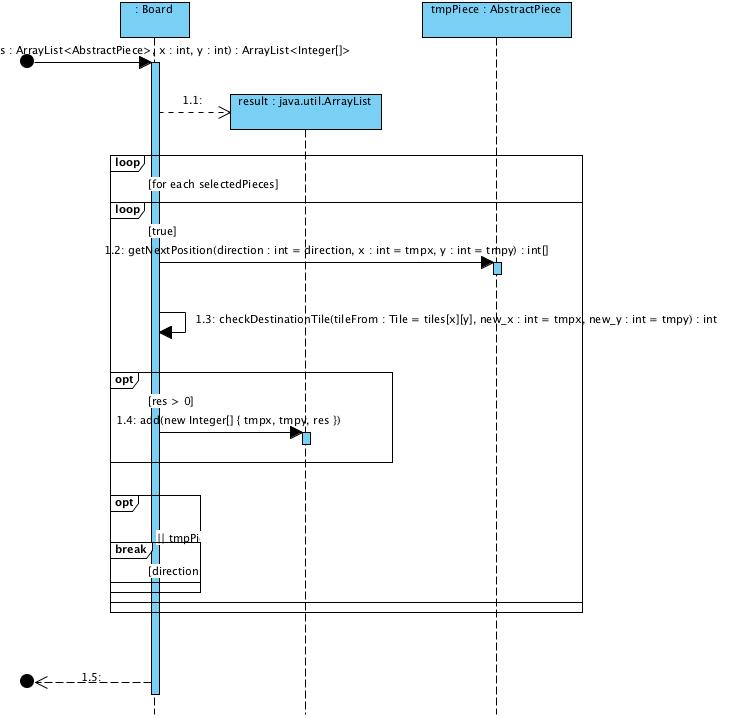
# Initialise Board



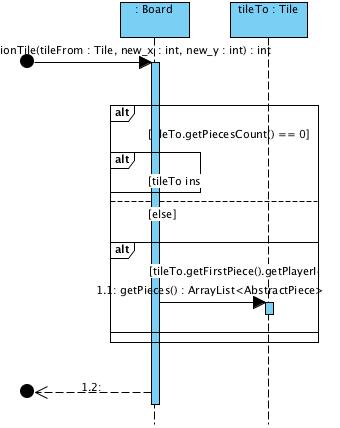
# Start Game



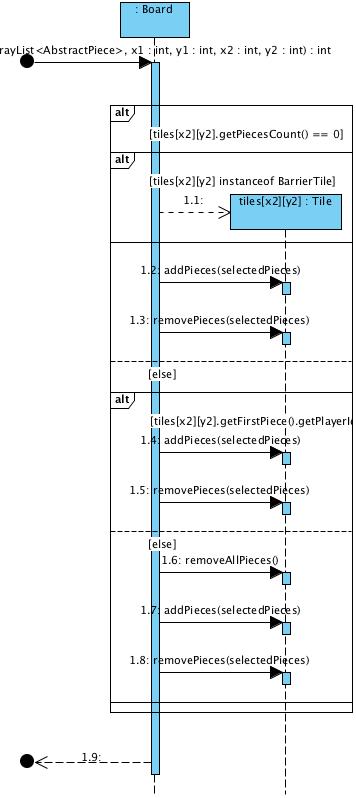
# Find Valid Moves



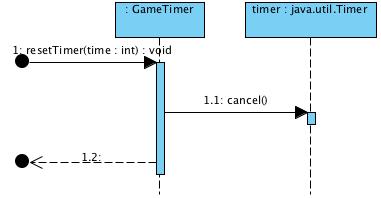
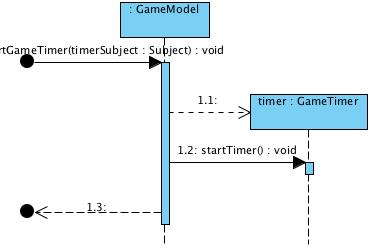
# Check Destination



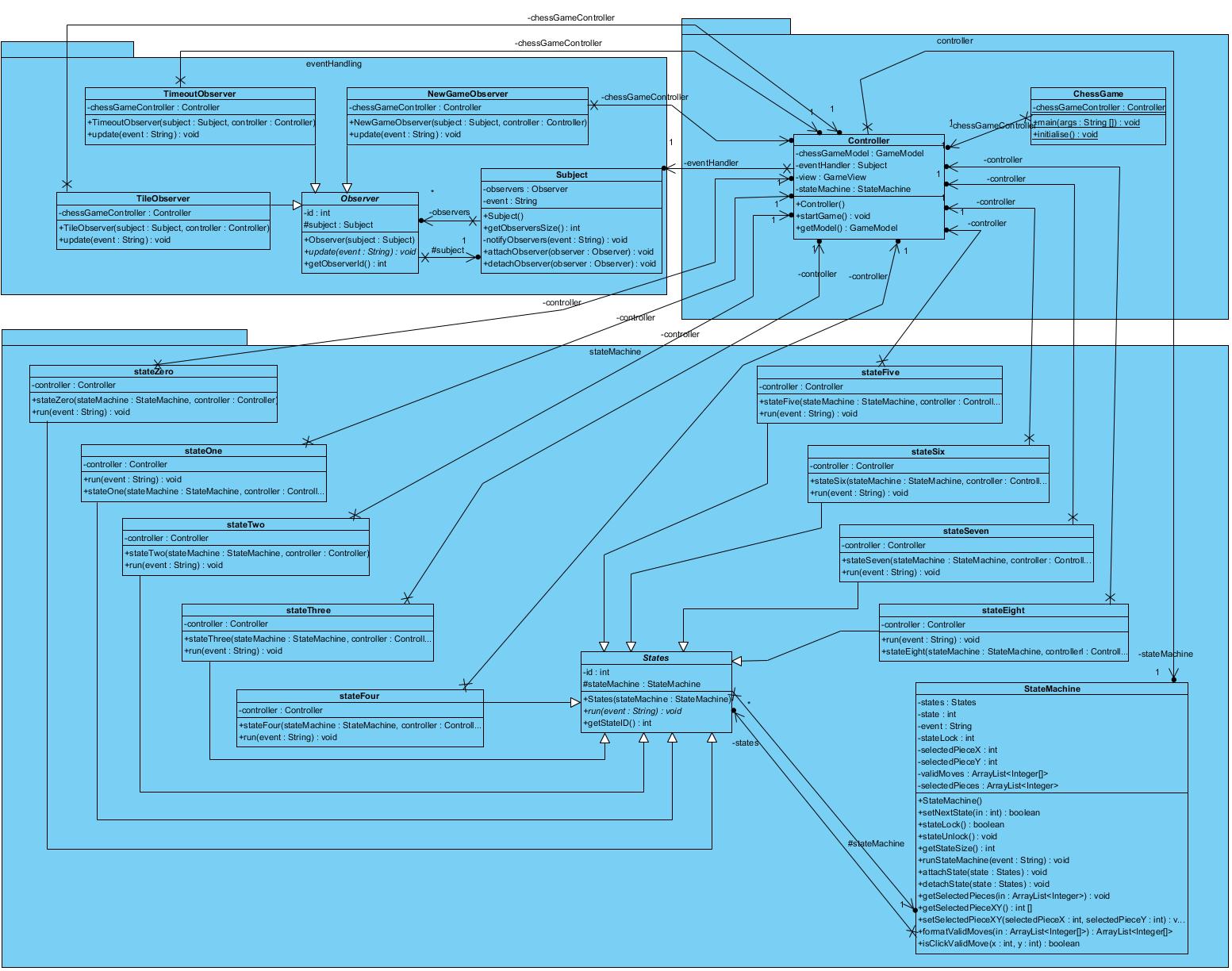
# Move Piece



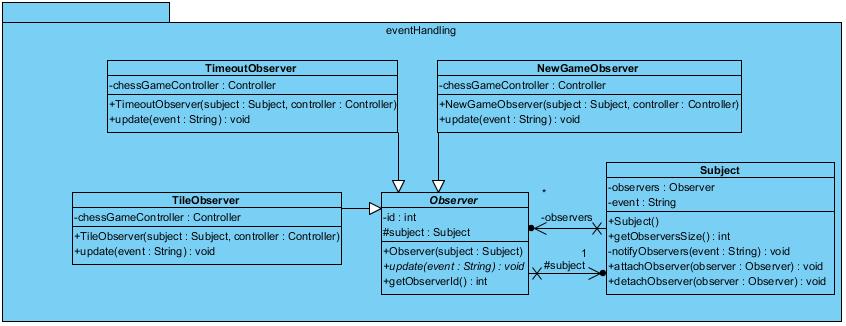
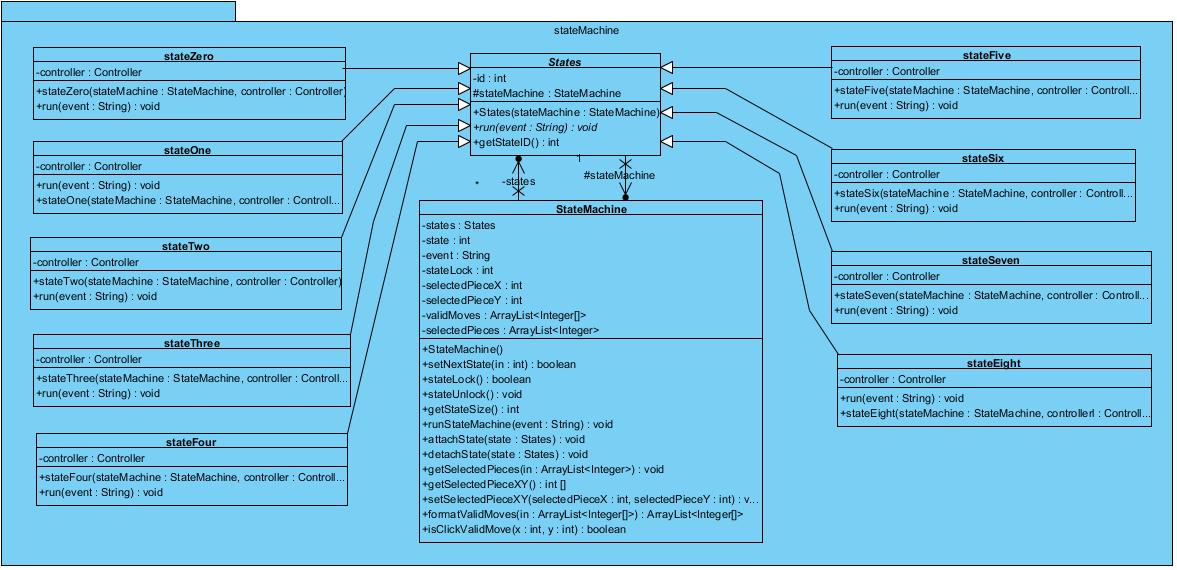
# Start Timer, Reset Timer



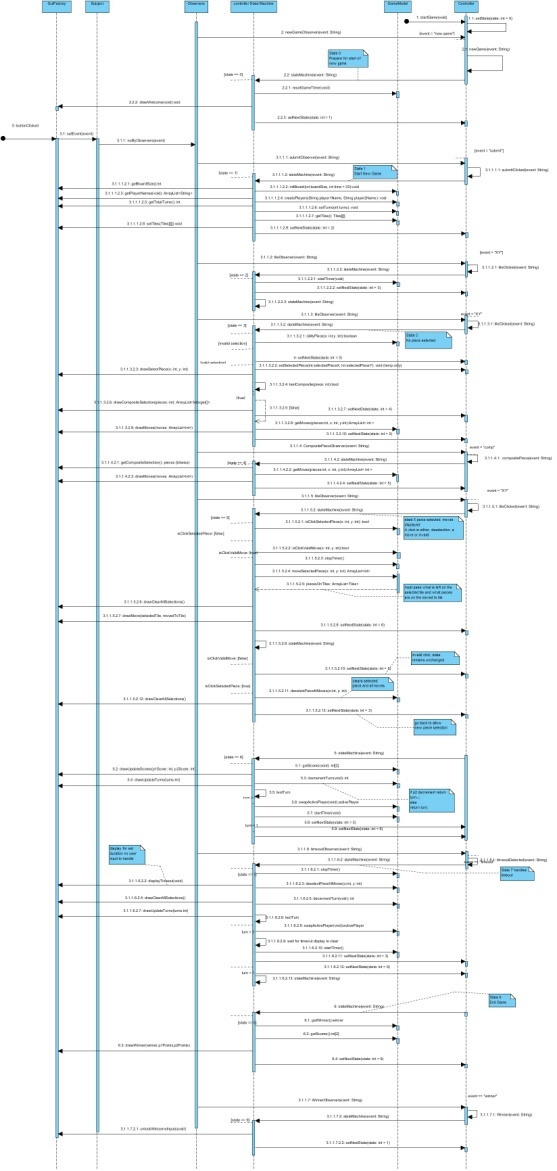
# Controller – Class Diagram



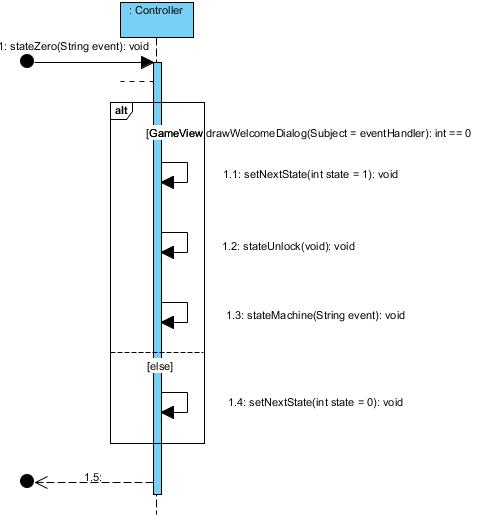
# Controller – State Machine and Event Handler



# Controller – State Machine Sequence Diagram



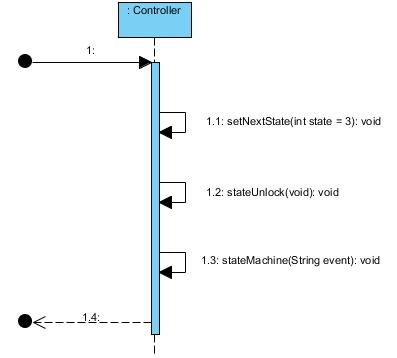
# Controller – State Zero



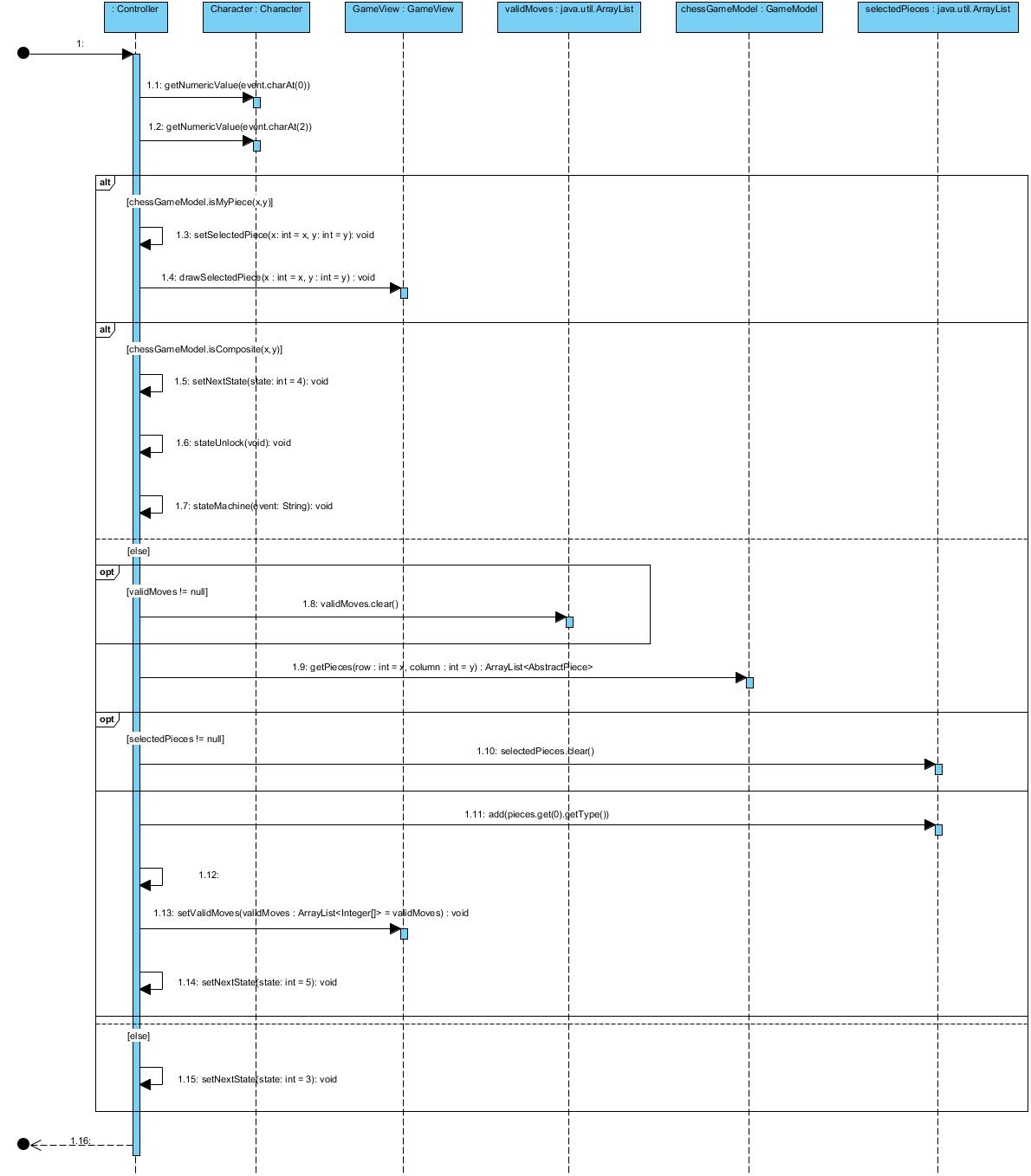
# Controller – State One



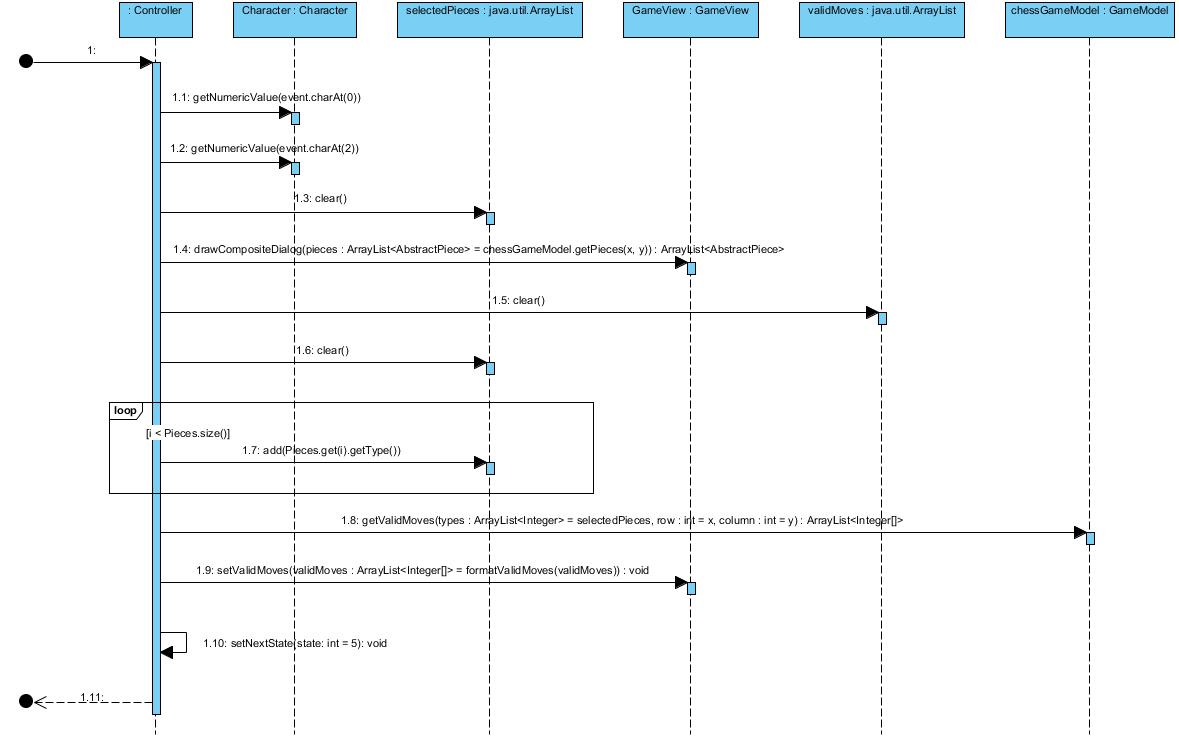
# Controller – State Two



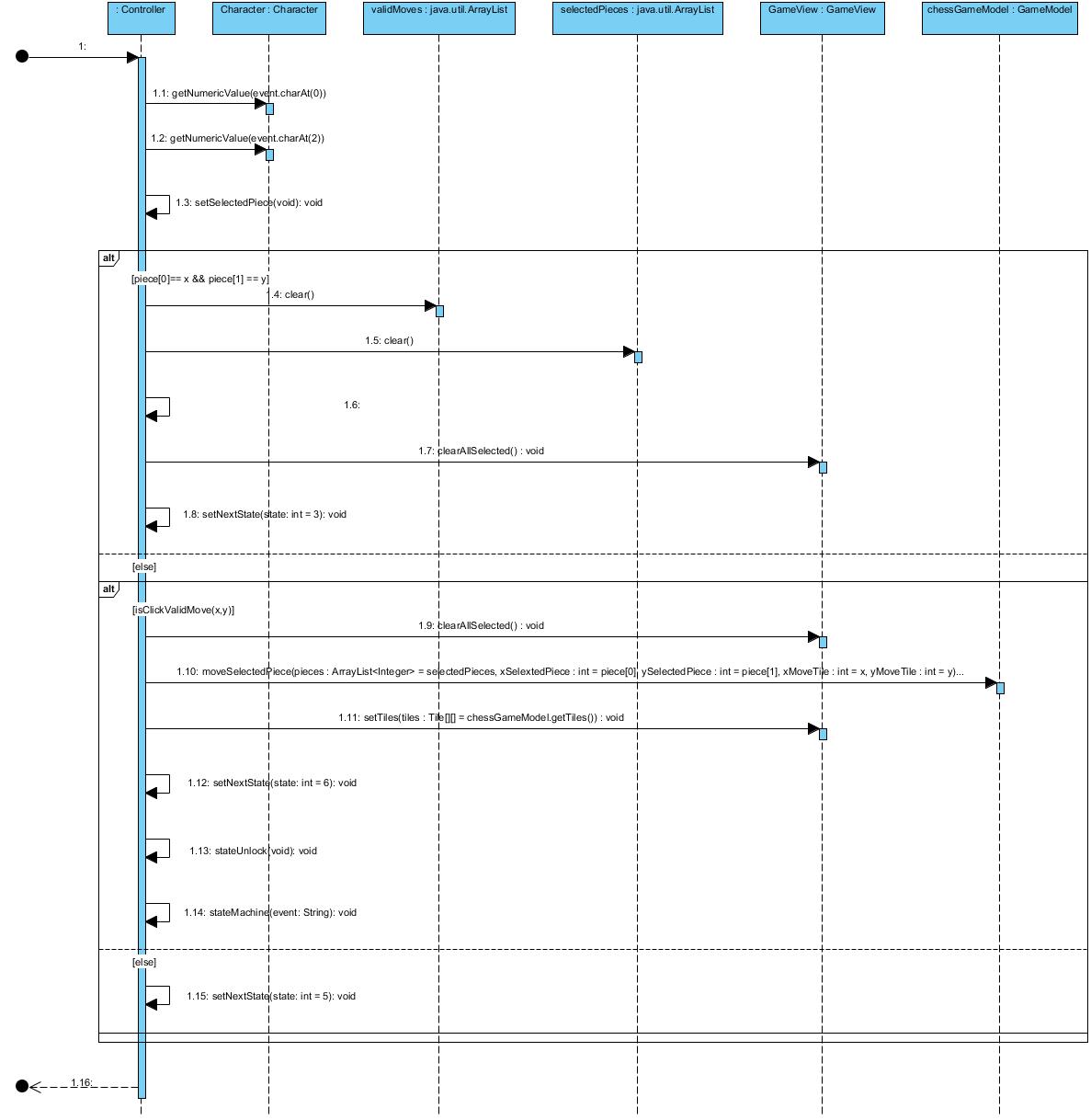
# Controller – State Three



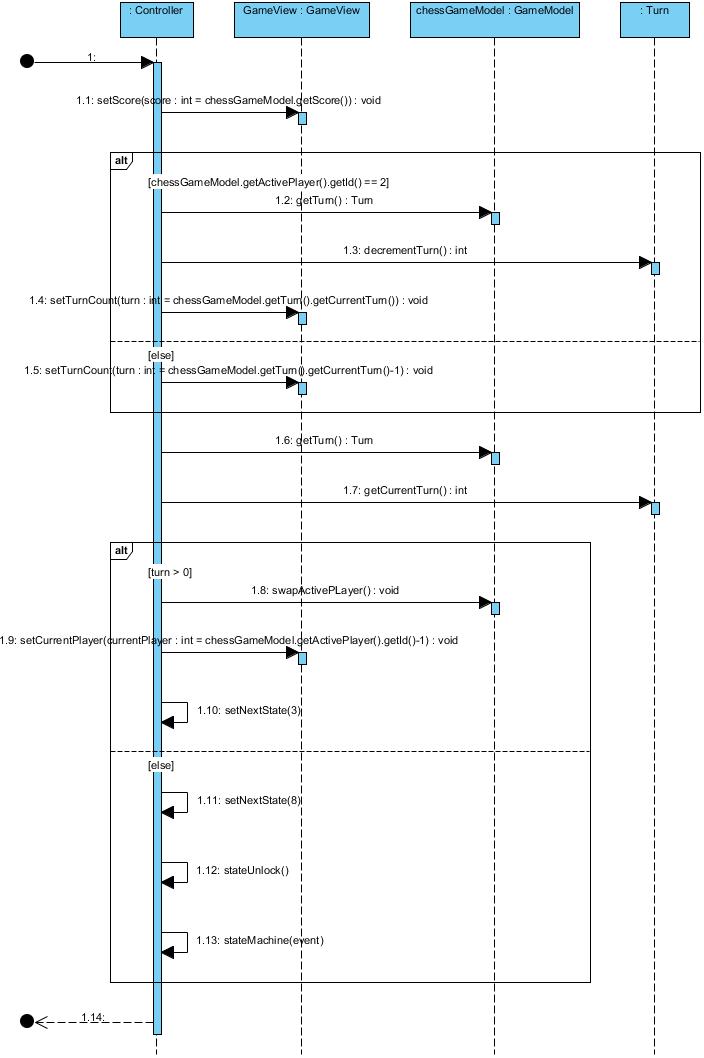
# Controller – State Four



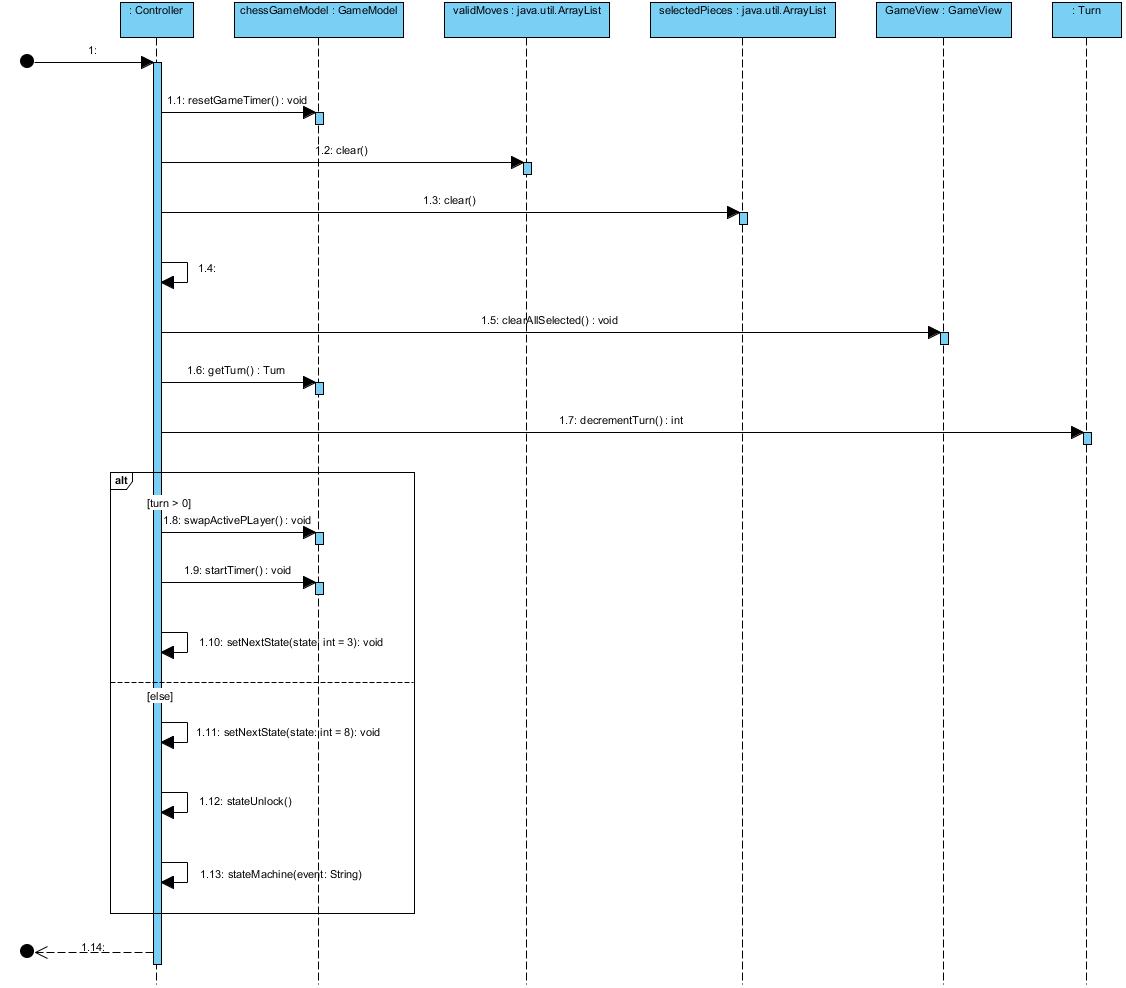
# Controller – State Five



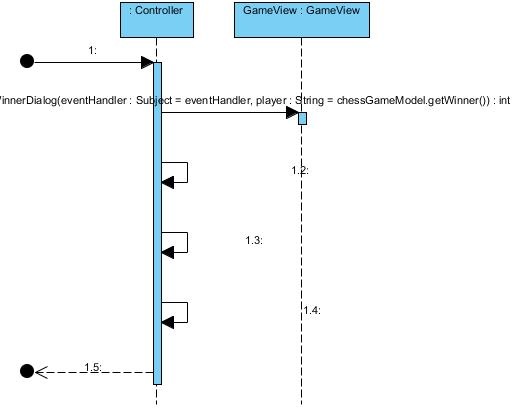
# Controller – State Six



# Controller – State Seven



# Controller – State Eight



Work distribution

Dolly Shah (s3399503)

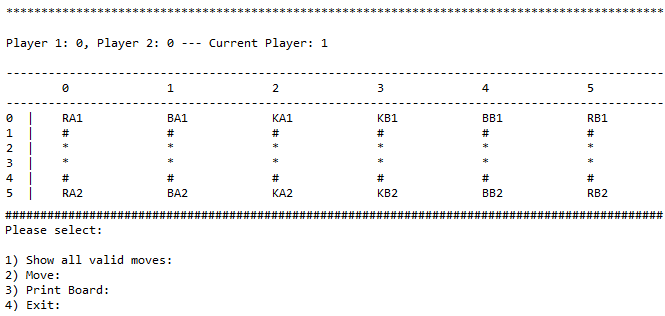
* Implemented game logic for following:

- finding all valid moves for single, composite or part of a composite piece

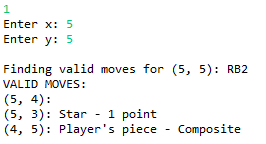
- moving a piece: single piece, joining (composite) or splitting pieces

Console based game implementation

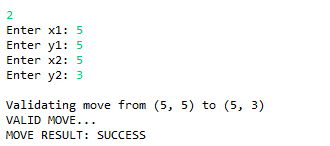
Initial game state as follows. Select one option from 1, 2, or 3 to play the game or option 4 to exit.



Finding valid moves for a RB2 (2nd Rook of player 2) located at (5, 5)



To move the piece RB2 located at (5, 5) to (5, 3):



After moving the piece RB2 located at (5, 5) to (5, 3):

