The Jungle Game

Chesshire Coders

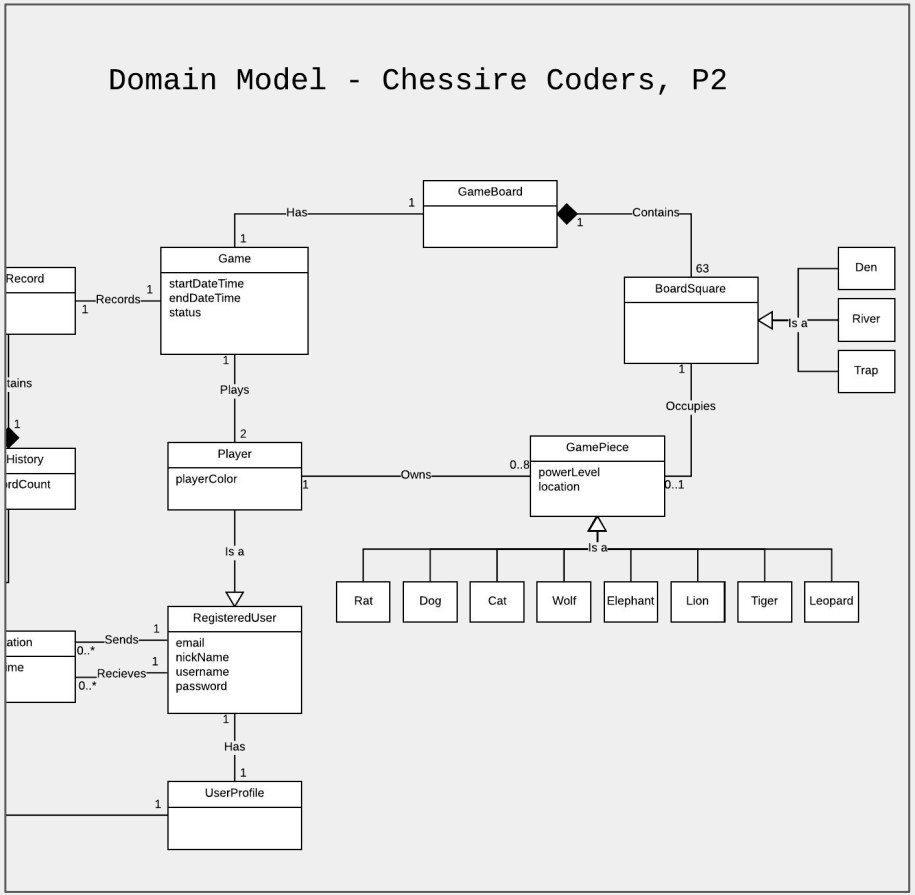
- Angélica Fallas
- Taner King
- Adam Gundem
- Alexander Hennings
- Cameron Ackerman

Image from: http://www.tattoodaze.com/tattoo-images/79/th_tiger-art-decor-mountain-tiger-tiger-hand-painting-original-bkLbwV.jpg

Table of contents

Domain Model Changes

- Removed *piece* from **BoardSquare** and *players* from **Game**.
  - To keep notation consistent (association vs attribute vs both).



Glossary Changes

- Added entries for attributes.
- Sorted items alphabetically.

Glossary

BoardSquare: A representation of a single square on the Jungle board. A square has an attribute piece.

Game: An instance of a game of Jungle.
-**endDateTime:** Date and time when a game ended.
-**startDateTime:** Date and time when a game started.
-**status:** Status of a specific game (ongoing, completed, abandoned, etc)

GameBoard: A representation of the Jungle board that contains the current state of a game. The game board contains the different squares of Jungle, and any uncaptured Jungle pieces.

GameHistory: The game history is shown on each registered user's profiles. It includes a brief synopsis of each game played by that user.
-**gameRecordCount:** Represents the average score for a certain player.

GamePiece: A representation of a single Jungle piece. It is required that a game piece must be one of its eight different specialization types (i.e. if GamePiece were a Java class, it would be abstract). And there may be no more than one of each piece type per player.
-**location:** represents where a GamePiece is located withing a GameBoard.
-**powerLevel:**Represents the current level that a certain GamePiece has in a given state of the game.

GameRecord: A game record is the outcome of a single game of jungle.
-**outcome:** represents the final result of a certain GameRecord.

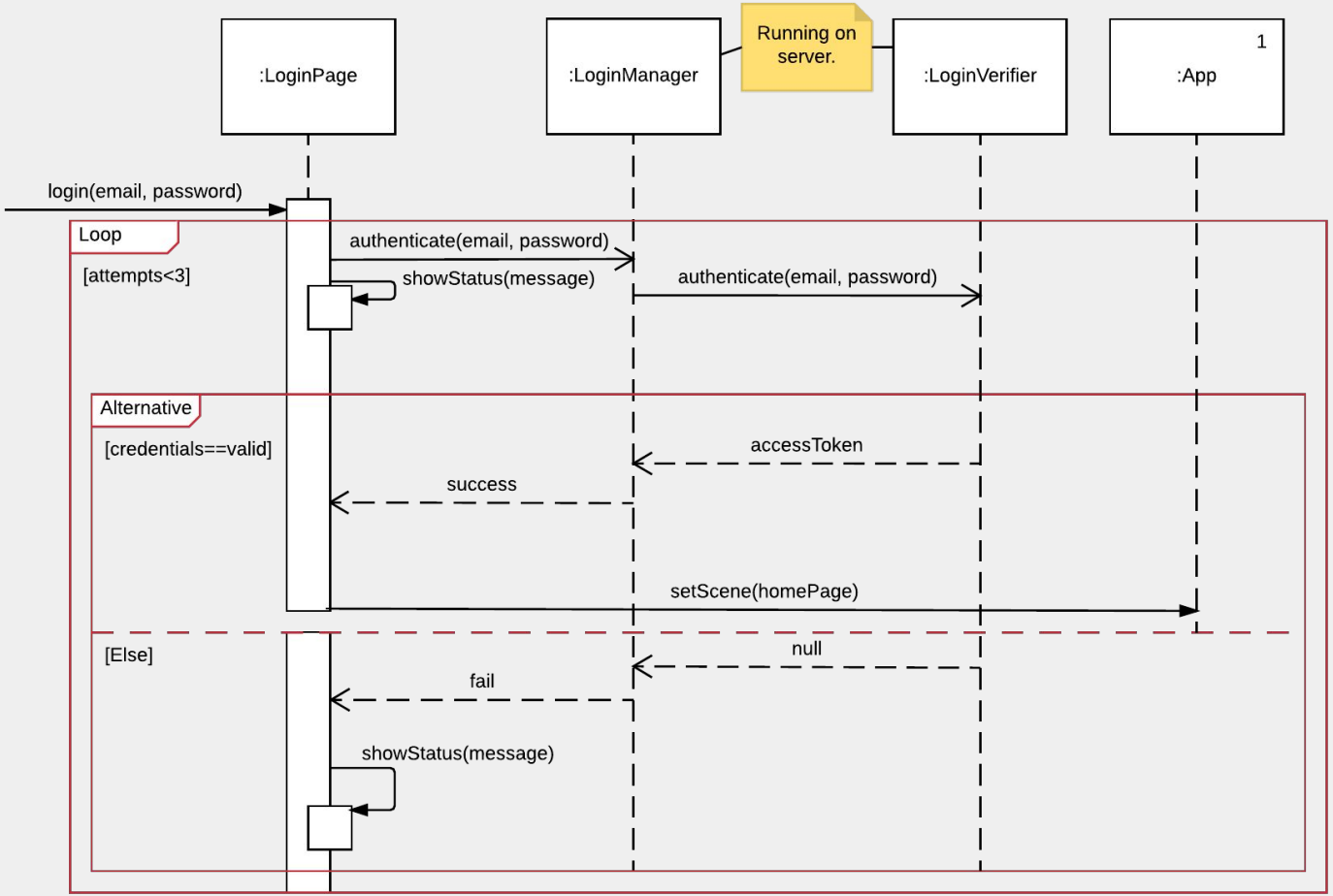
Invitation: An invitation is a request for another registered user to play a game with the sending user. Each invitation has one sender and one receiver.
-**sentDateTime:** specific date time value that represent when an invitation was sent.

Player: An extension of a registered user. They may make moves, capture pieces, and perform other actions that the registered user entity cannot. Each player owns 0-8 game pieces(depending on how many have been captured by an opposing player) that they may control.
-**playerColor:** indicates what team the player is on.

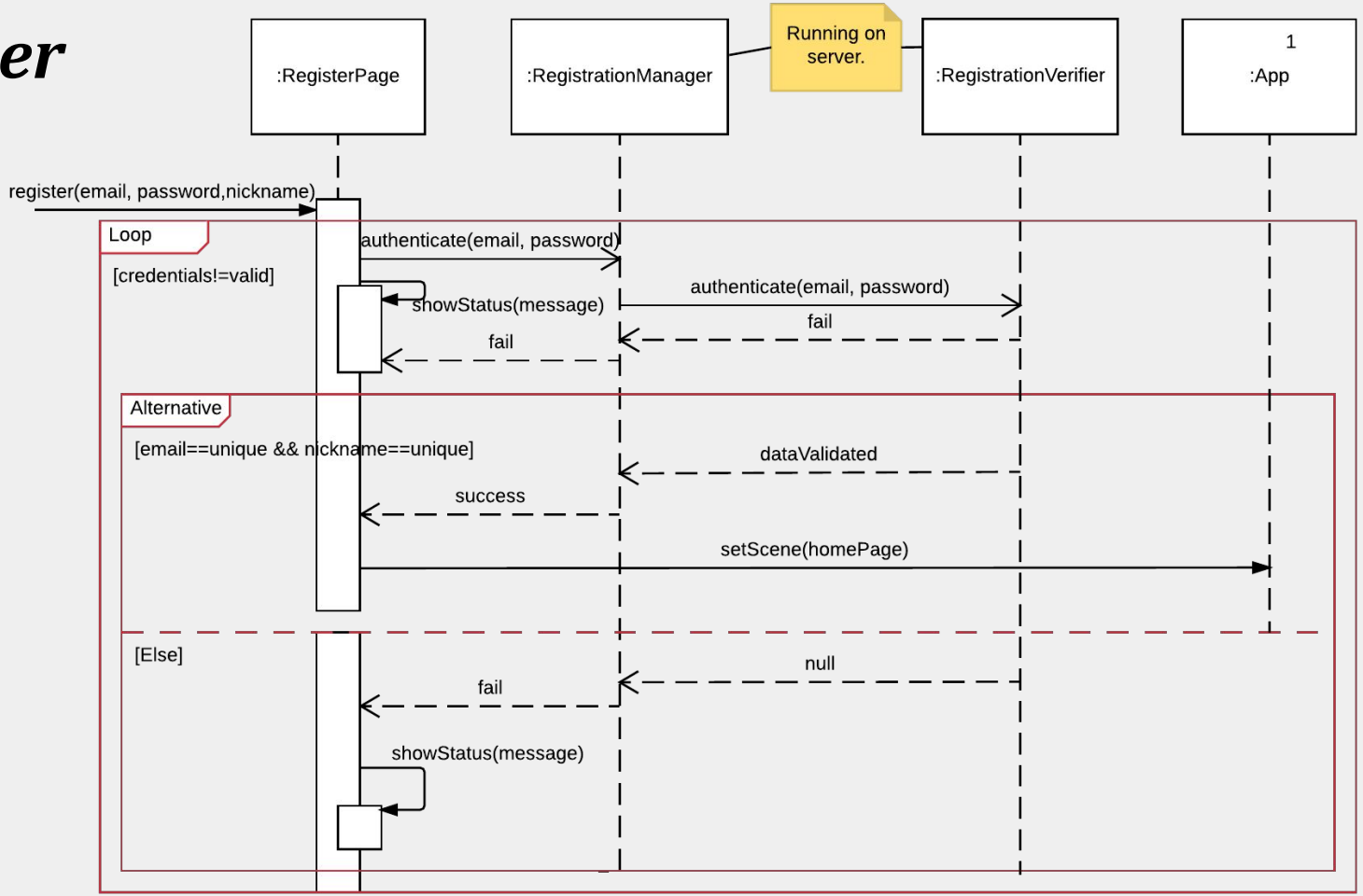
Sequence diagrams

- Login
- Game Invite
- Game Create
- Register
- Take Turn

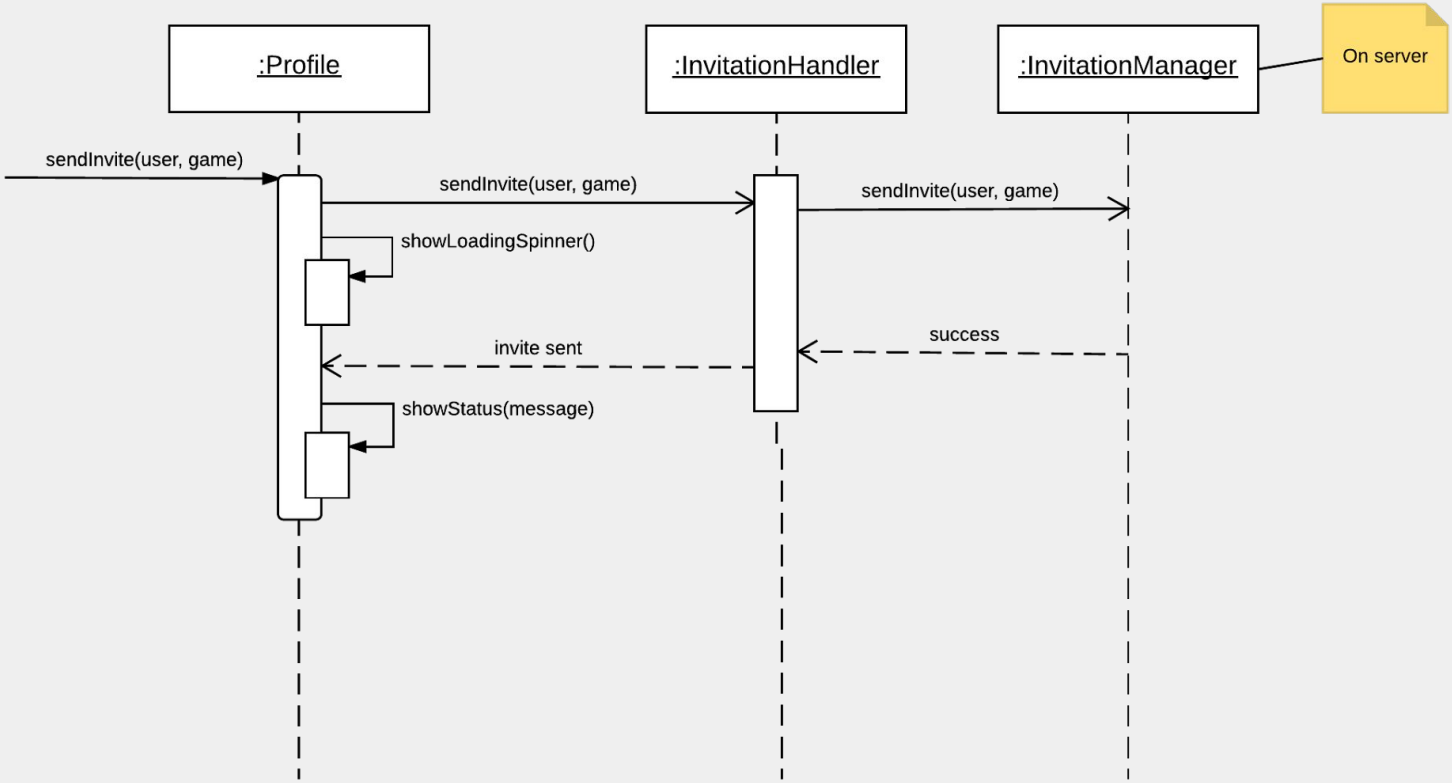
Login


:LoginPage
:LoginManager
Running on server.
:LoginVerifier
1
:App
login(email, password)
Loop
[attempts<3]
authenticate(email, password)
showStatus(message)
authenticate(email, password)
Alternative
[credentials==valid]
accessToken
success
setScene(homePage)
[Else]
null
fail
showStatus(message)

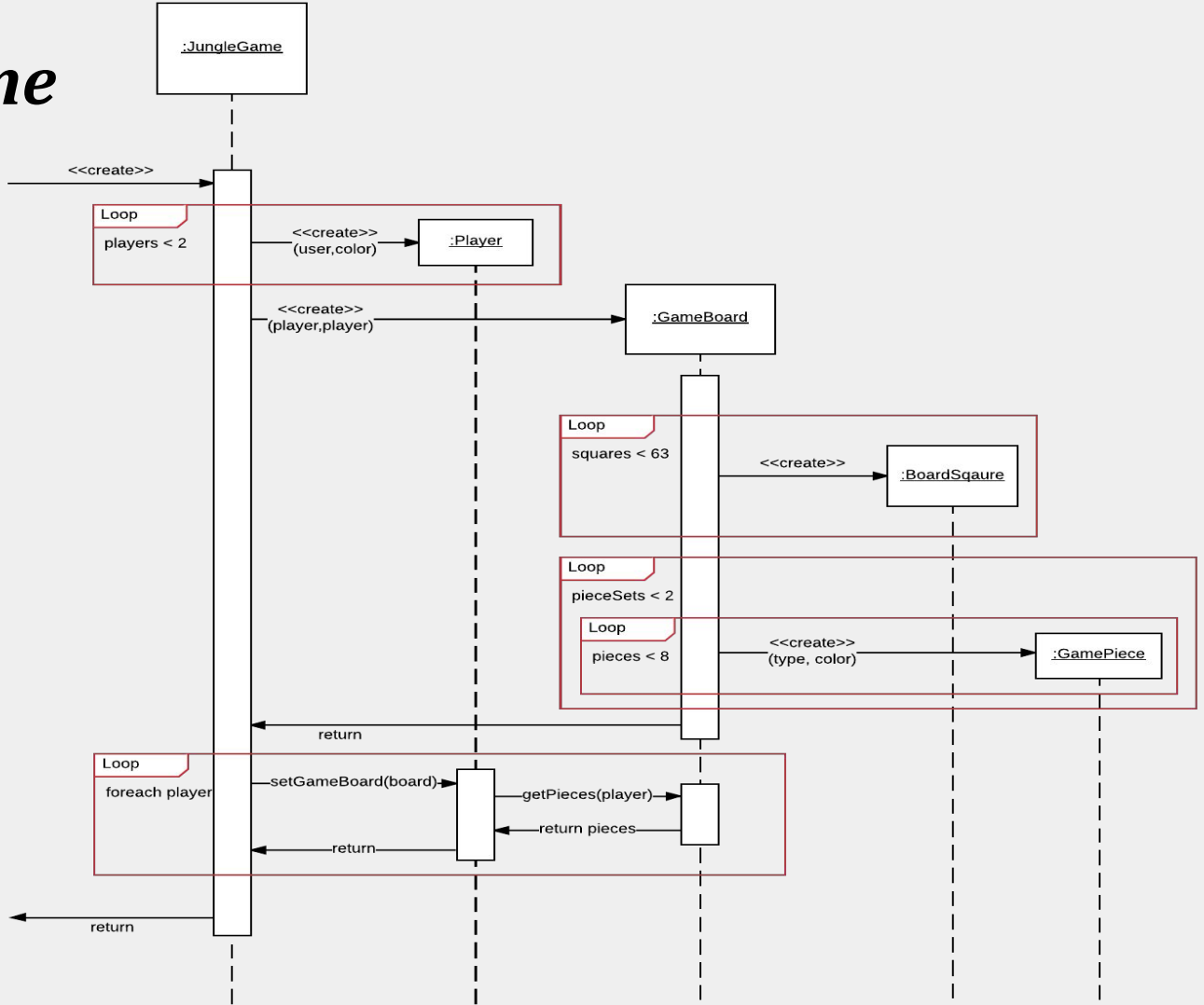
Register


:RegisterPage
:RegistrationManager
Running on server.
:RegistrationVerifier
1
:App
register(email, password,nickname)
Loop
[credentials!=valid]
authenticate(email, password)
showStatus(message)
authenticate(email, password)
fail
fail
Alternative
[email==unique && nickname==unique]
dataValidated
success
setScene(homePage)
[Else]
null
fail
showStatus(message)

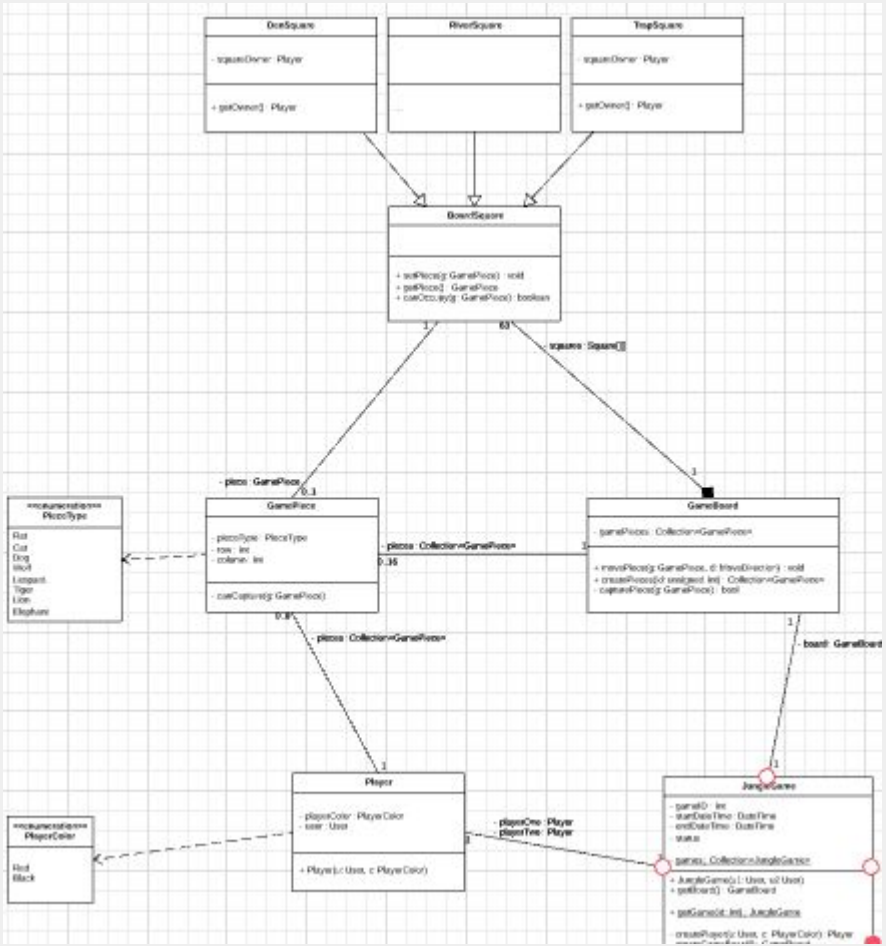
Game Invite


:Profile
:InvitationHandler
:InvitationManager
On server
sendInvite(user, game)
sendInvite(user, game)
sendInvite(user, game)
showLoadingSpinner()
success
invite sent
showStatus(message)

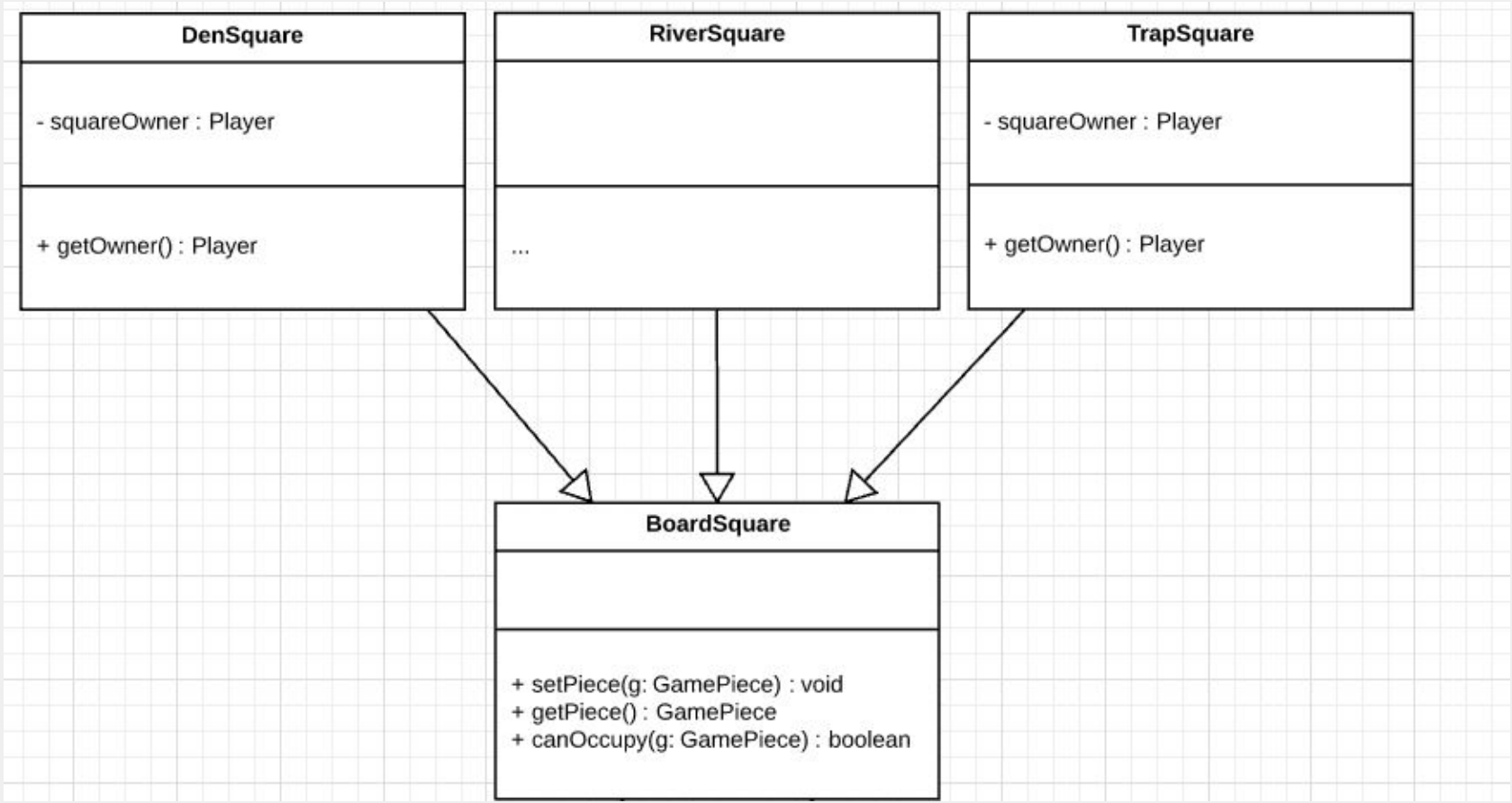
Create Game


:JungleGame
<<create>>
Loop
players < 2
<<create>>
(user,color)
:Player
<<create>>
(player,player)
:GameBoard
Loop
squares < 63
<<create>>
:BoardSqaure
Loop
pieceSets < 2
Loop
pieces < 8
<<create>>
(type, color)
:GamePiece
return
Loop
foreach player
setGameBoard(board)
getPieces(player)
return pieces
return
return

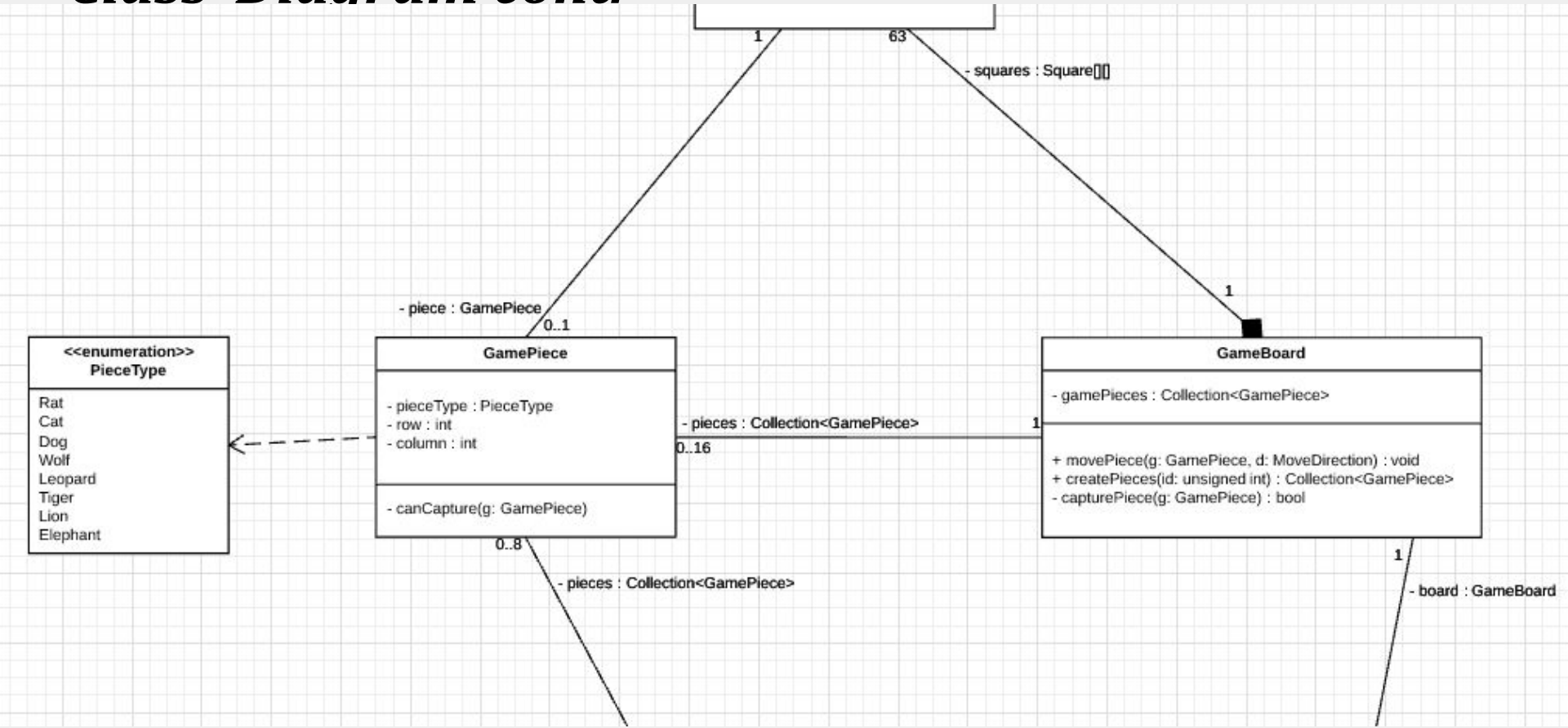
Class Diagram



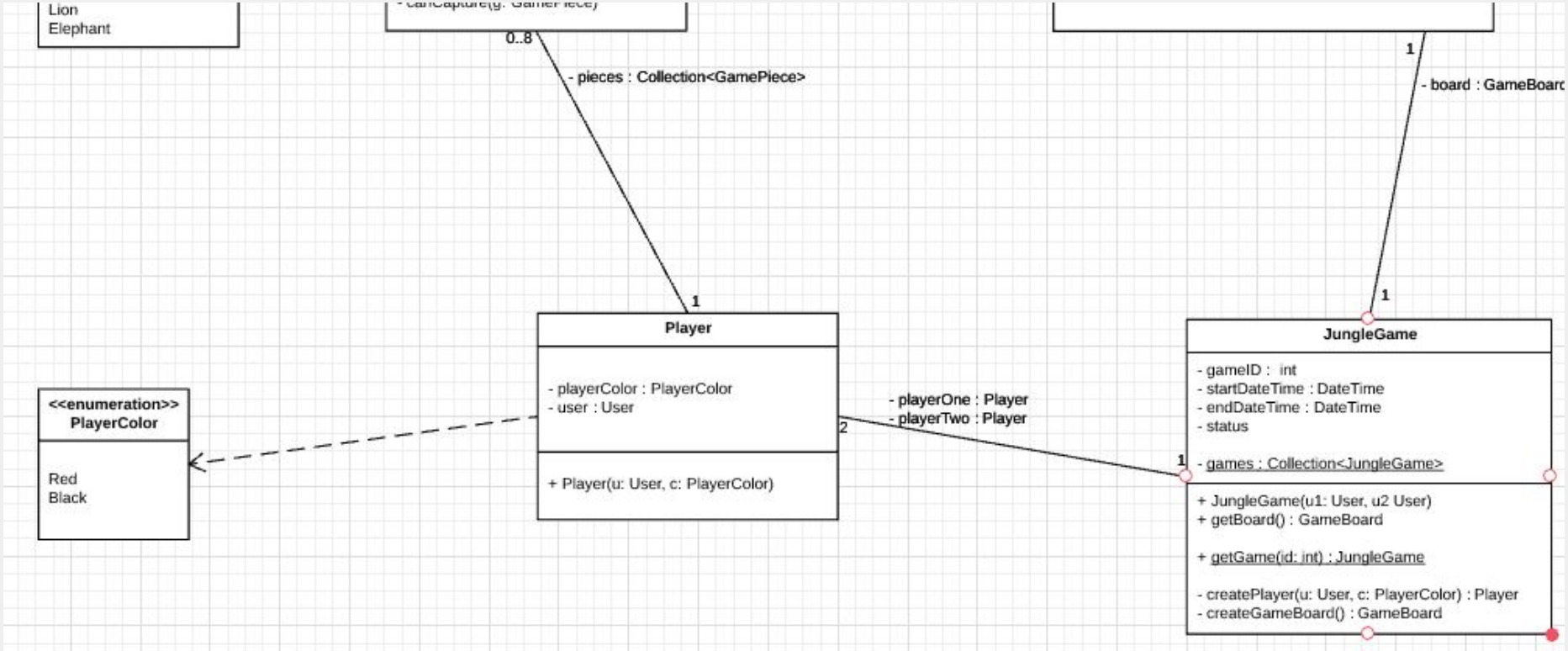
Class Diagram



Class Diagram cont.



Class Diagram cont.



List of test cases:

TestBoardSquare:
- testSetLocation_bottom_right_corner
- testSetLocation_column_too_big
- testSetLocation_column_too_small
- testSetLocation_row_too_big
- testSetLocation_row_too_small
- testSetPowerLevel_zero
- testSetPowerLevel_too_small
- testSetPowerLevel_eight
- testSetPowerLevel_too_big
- testCanOccupy_river
- testCanOccupy_river_rat
- testCanOccupy_friendly_piece
- testCanOccupy_friendly_den
- testCanOccupy_square_not_adjacent_row
- testCanOccupy_square_not_adjacent_column
- testCanOccupy_square_not_adjacent_leopard

TestGameBoard:
- testGetPieceAt_normal
- testGetPieceAt_column_too_big
- testGetPieceAt_column_too_small
- testGetPieceAt_empty_square
- testGetPieceAt_row_too_big
- testGetPieceAt_row_too_small
- testGetSquareAt_top_left_table_edge
- testGetSquareAt_bottom_right_table_edge
- testGetSquareAt_column_too_big
- testGetSquareAt_column_too_small
- testGetSquareAt_row_too_big
- testGetSquareAt_row_too_small
- testGetValidMoves_corner
- testGetValidMoves_leopard
- testMovePiece

TestAccountHandler
- testRegisterUser
- testRegisterUserAlreadyRegistered
- testRegisterUserFailure
- testUnregisterUser
- testValidateLogin
- testLogout

TestBoardSquare:
- testConstructorWithPlayerColor
- testClearPiece
- testIsEmpty
- testIsEmpty_not
- testSetPiece
- testSetPiece_null

Demo

Questions And Discussions