The Jungle Game

Chesshire Coders

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

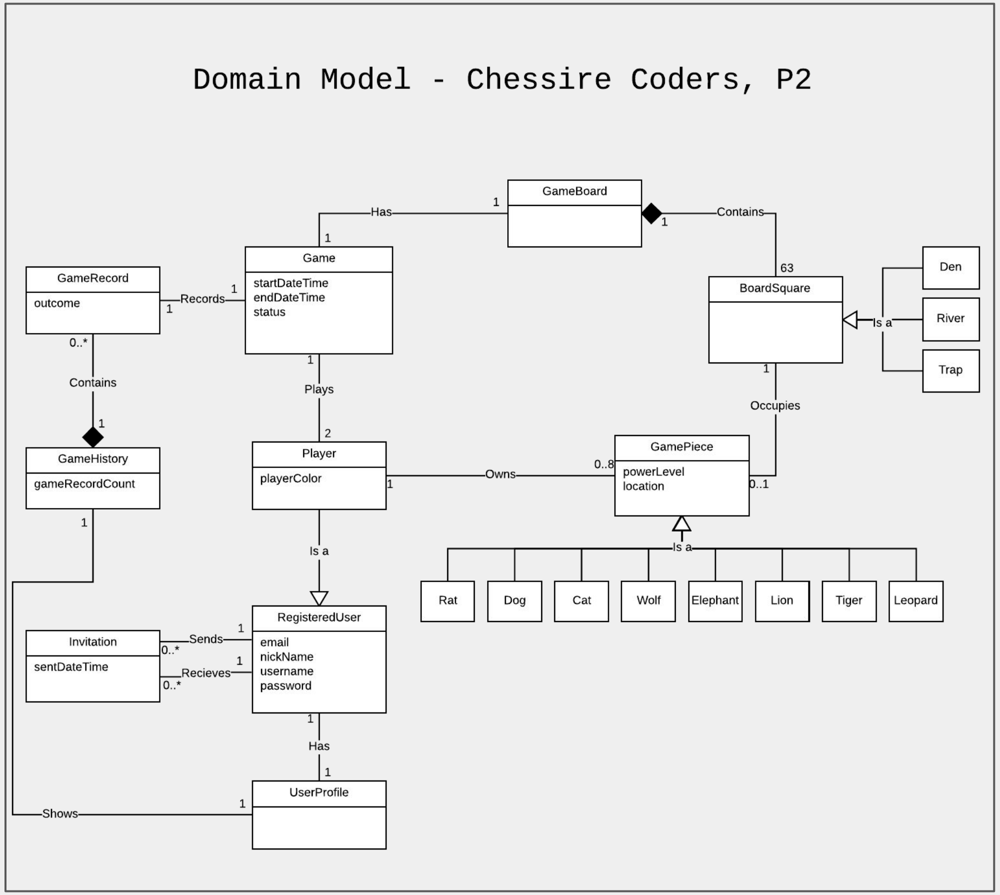
Image from: http://www.tattoodaze.com/tattoo-images/79/th_tiger-art-decor-mountain-tiger-tiger-hand-painting-original-bkLbwV.jpg

Presentation Overview

- ➢ Changes from last iteration
- ➢ Sequence diagrams
- ➢ Class diagrams
- ➢ Test Case Summary
- ➢ Project Tools
- ➢ Traceability Matrix & Use Case Progress
- ➢ Demo

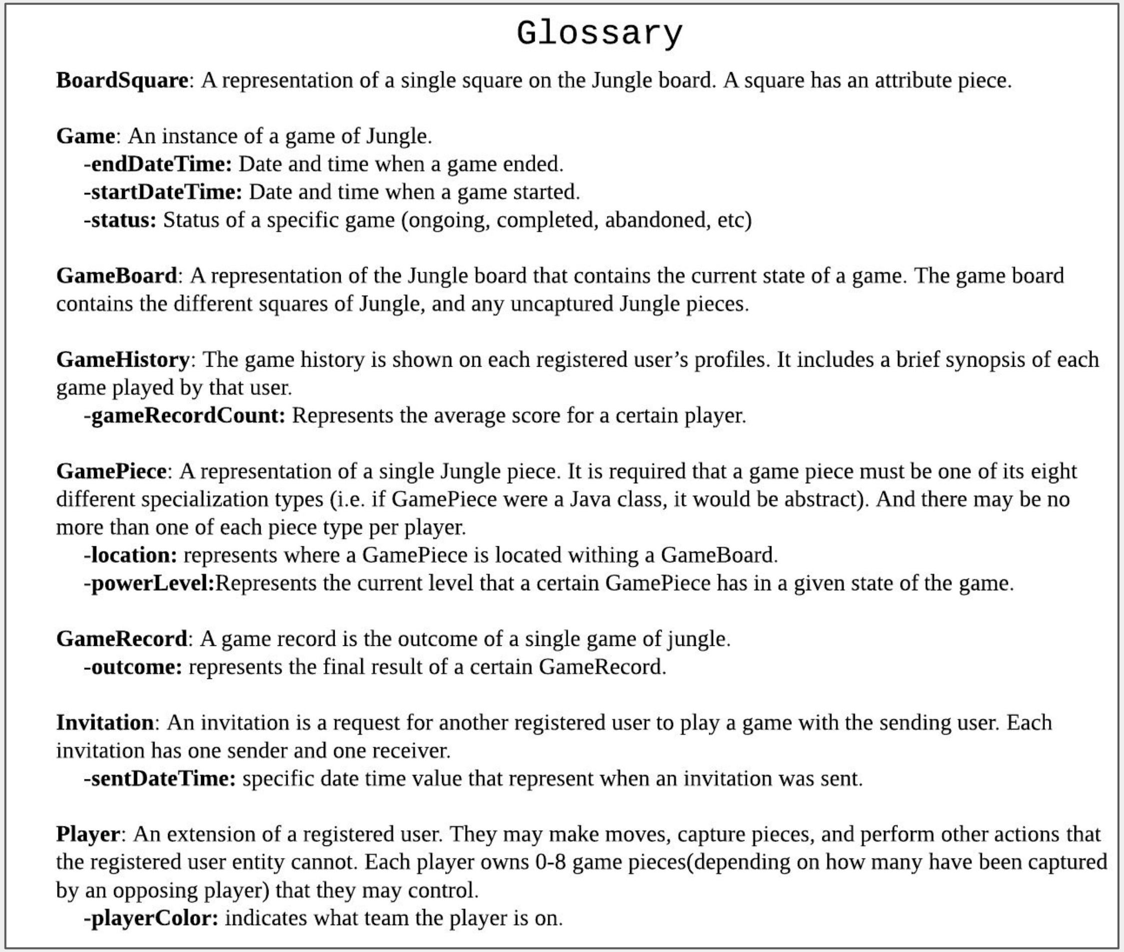
Domain Model Changes

Removed *piece* from **BoardSquare** and *players* from **Game**.



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.

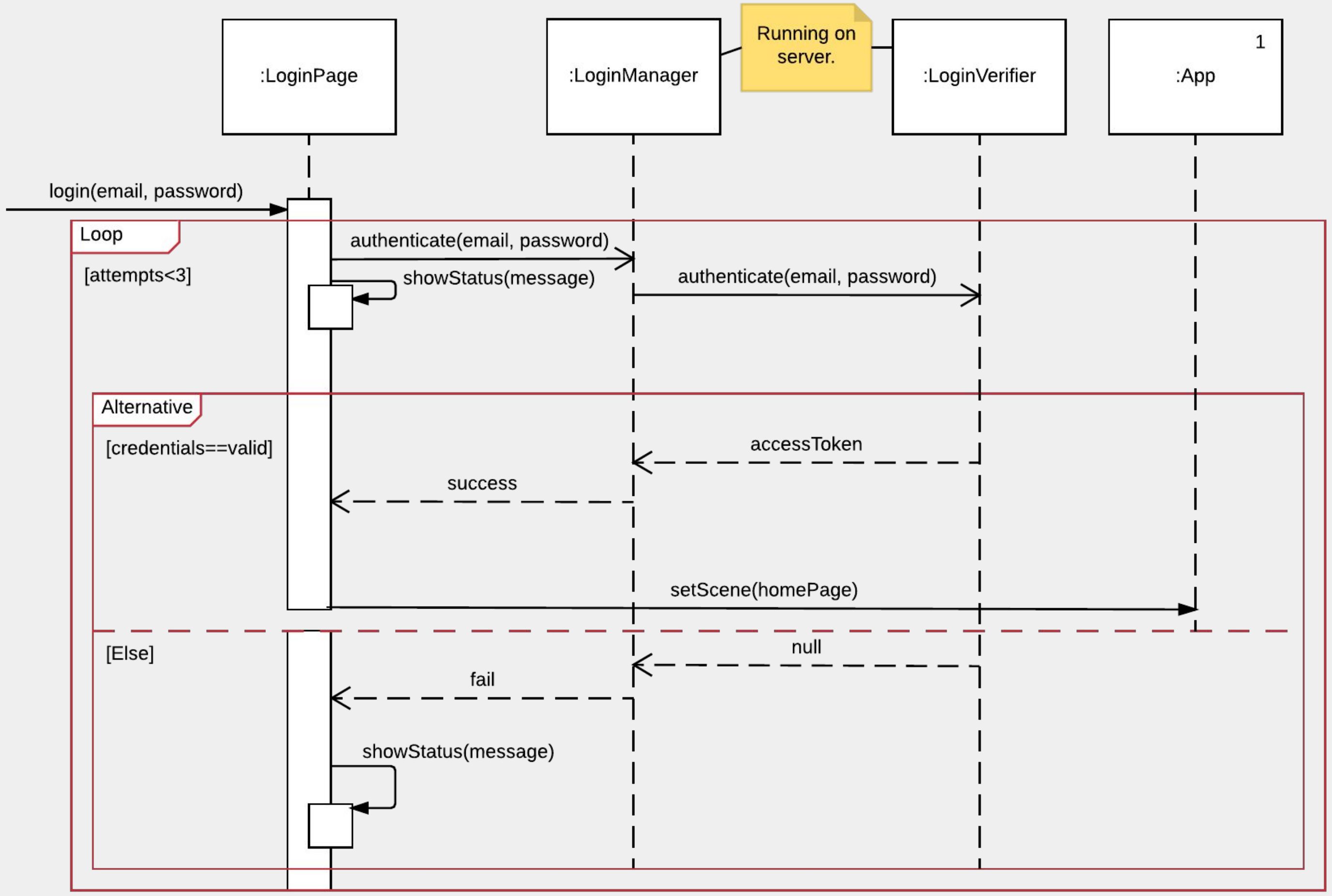
Invitation: An invitation is a request for another registered user to play a game with the sending user. Each invitation has one sender and one receiver.
-**sentDateTime:** specific date time value that represent when an invitation was sent.

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.

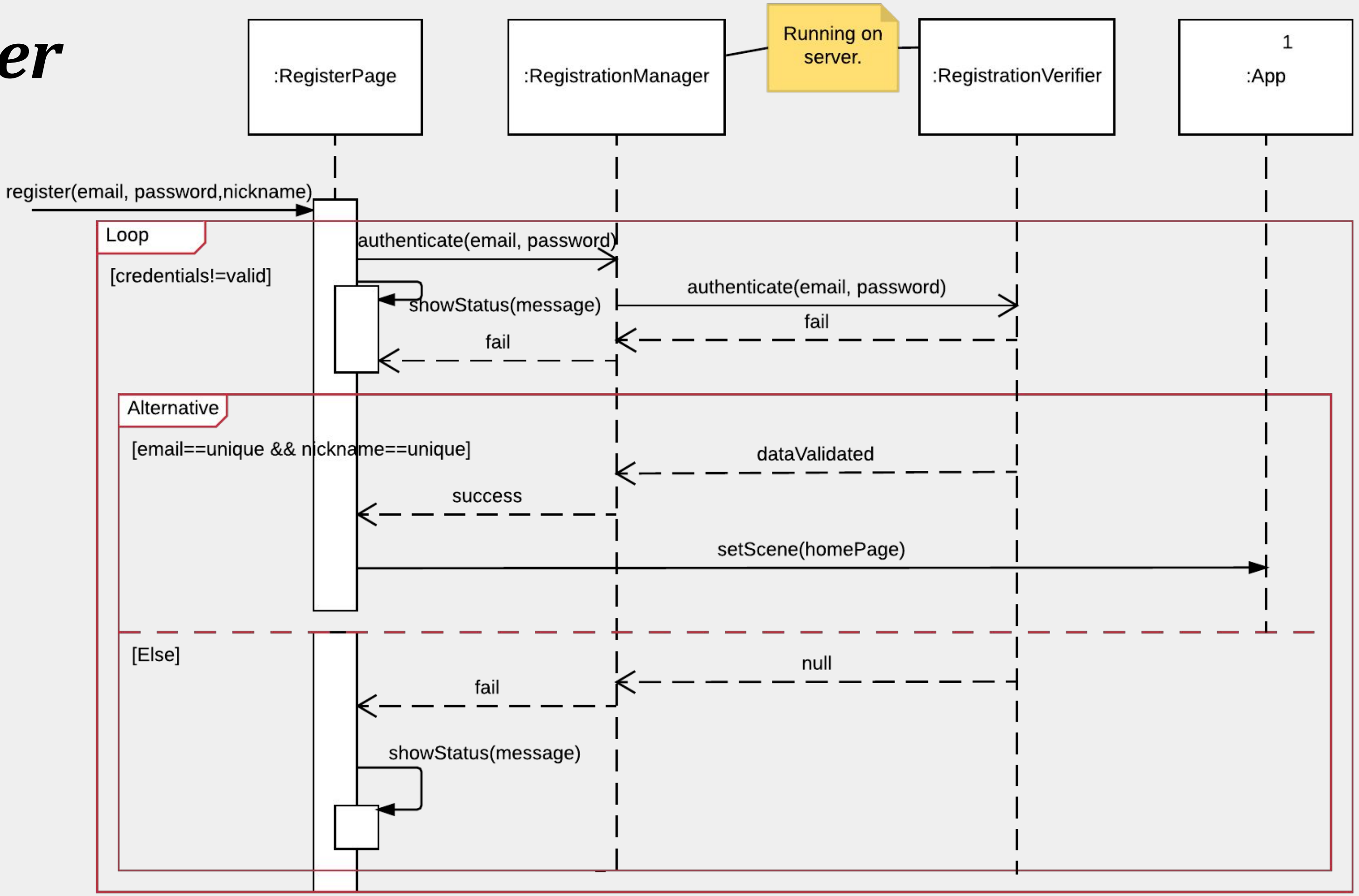
Sequence diagrams

- ➢ Login
- ➢ Game Invite
- ➢ Game Create
- ➢ Register

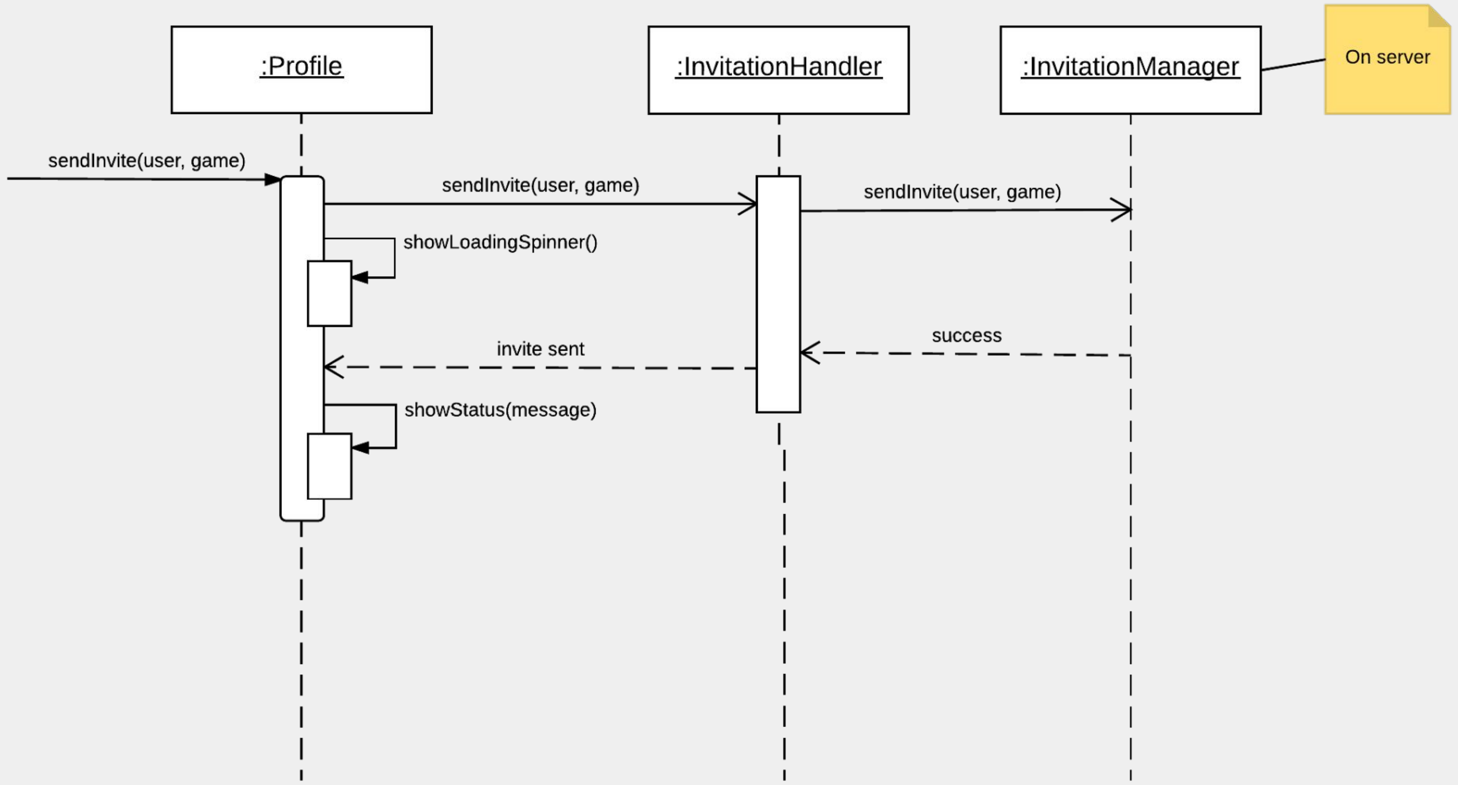
Login


: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)

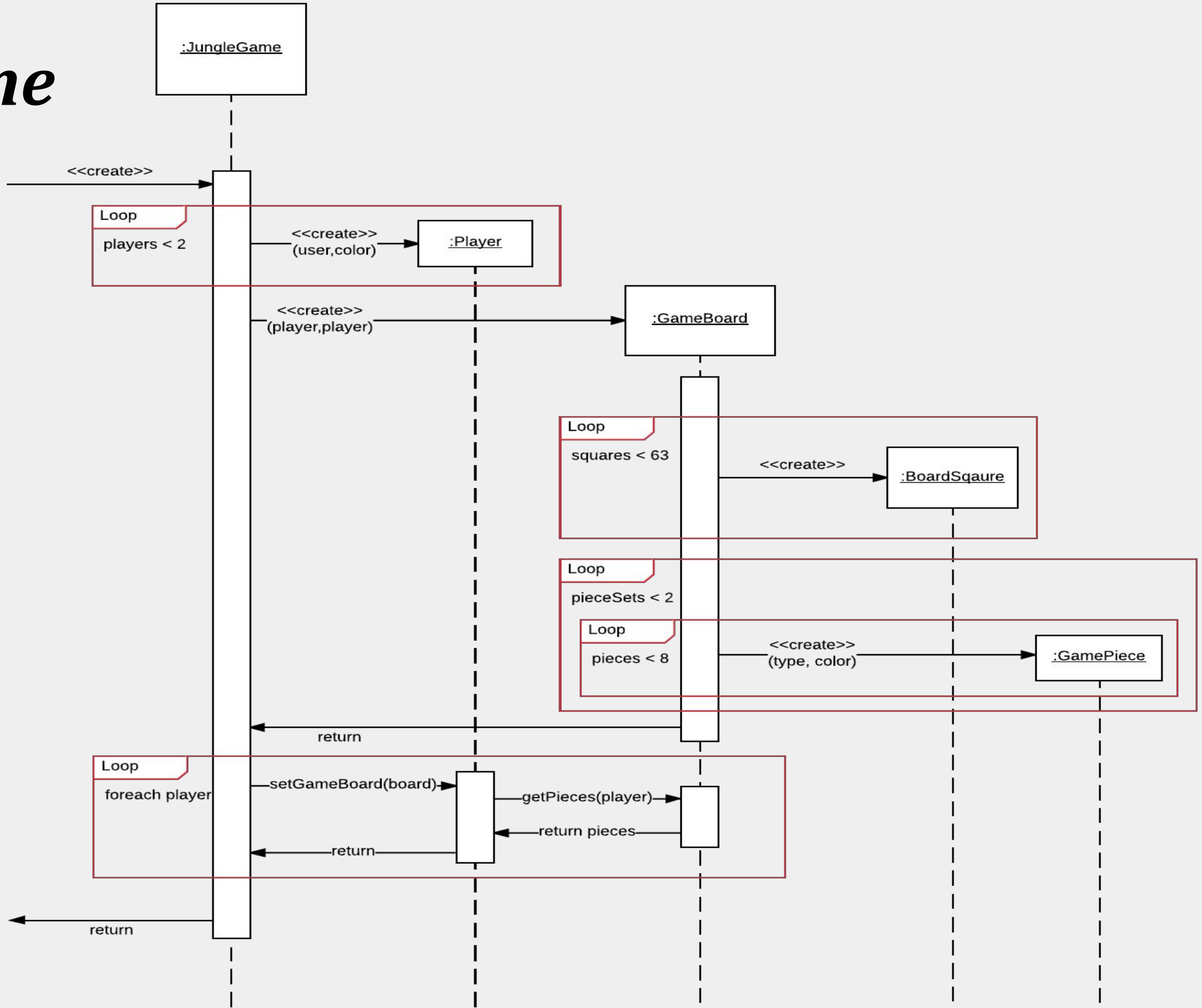
Register


: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

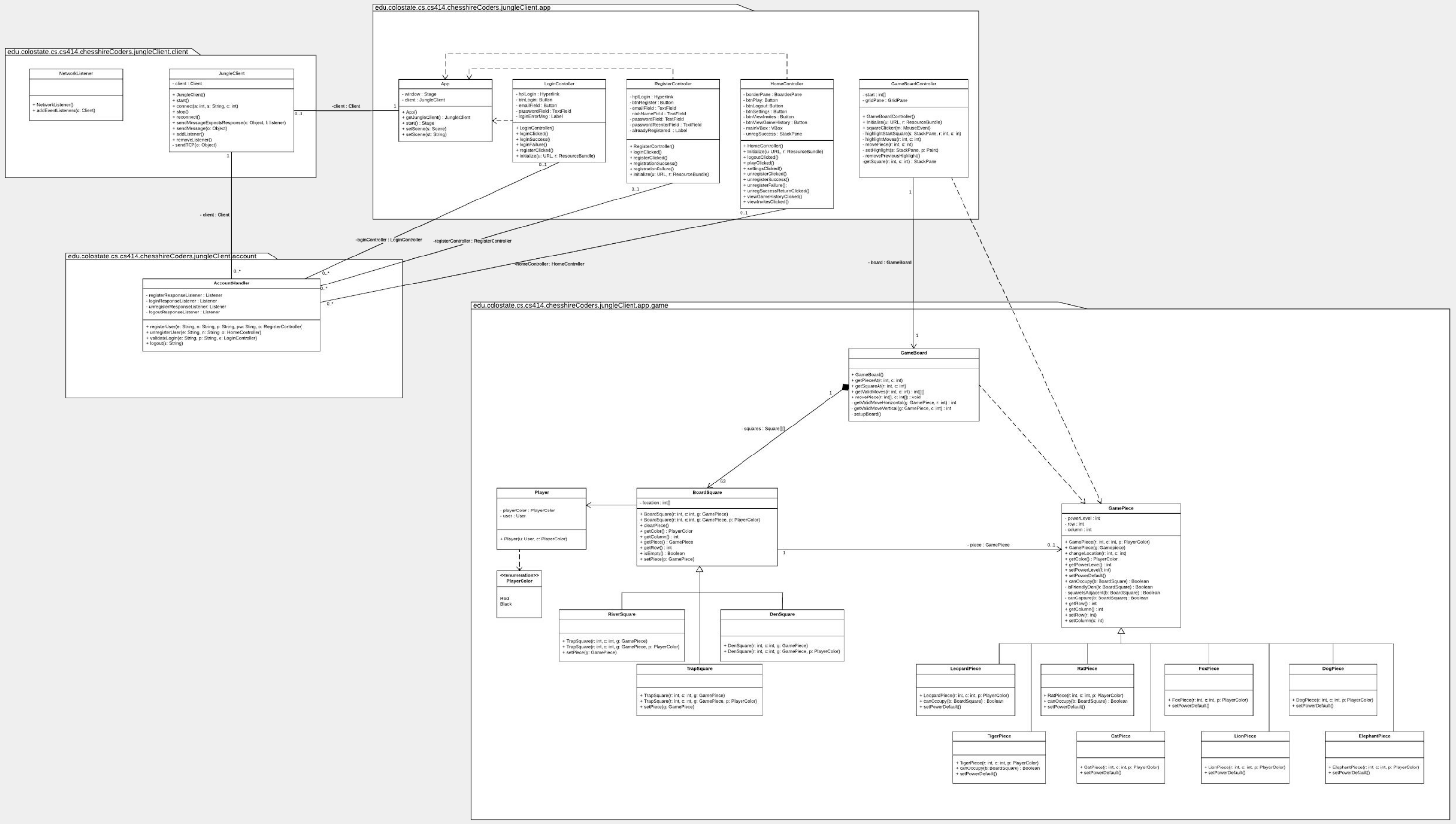

:Profile
:InvitationHandler
:InvitationManager
On server
sendInvite(user, game)
sendInvite(user, game)
sendInvite(user, game)
showLoadingSpinner()
success
invite sent
showStatus(message)

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

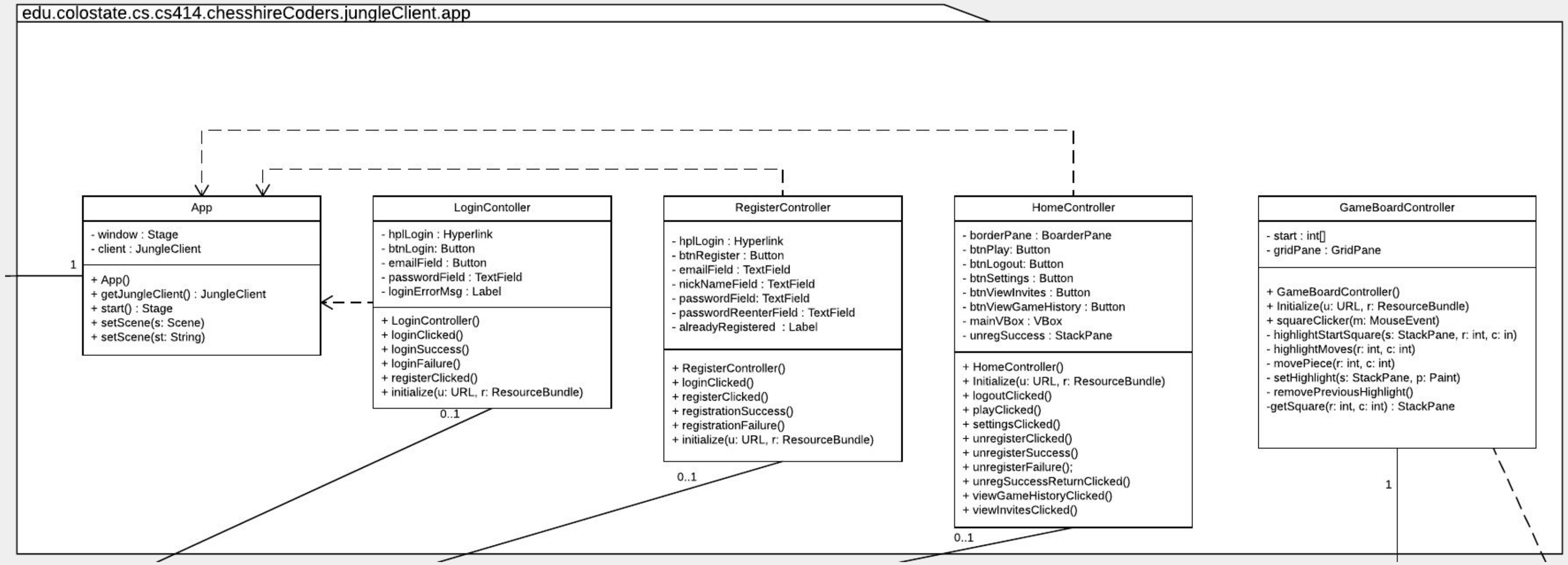
Class Diagram

Client Logic - Overview

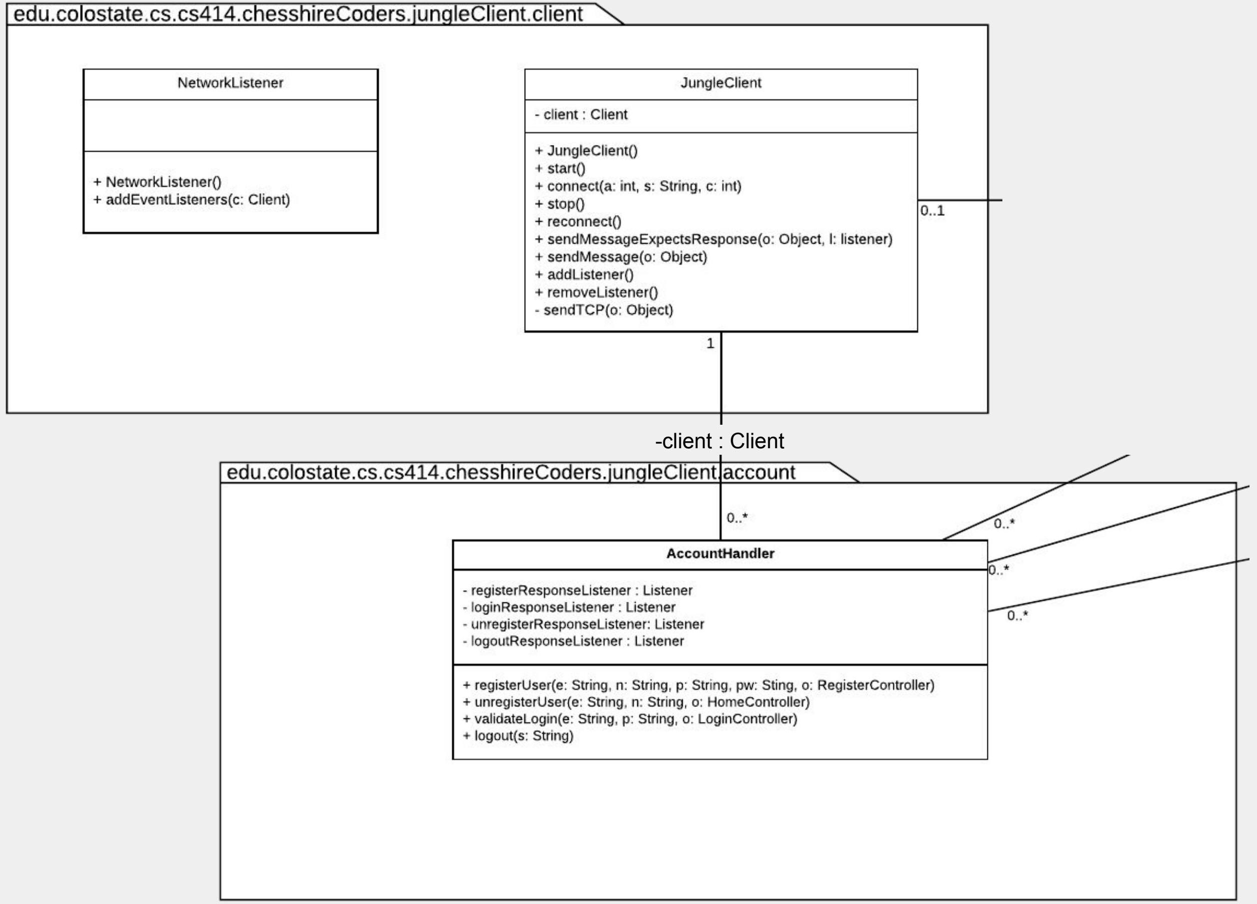

edu.colostate.cs.cs414.chesshireCoders.jungleClient.client
NetworkListener
+ NetworkListener()
+ addEventListeners(c: Client)
JungleClient
- client : Client
+ JungleClient()
+ start()
+ connect(a: int, s: String, c: int)
+ stop()
+ reconnect()
+ sendMessageExpectsResponse(o: Object, l: listener)
+ sendMessage(o: Object)
+ addListener()
+ removeListener()
- sendTCP(o: Object)
-client : Client
0..1
1
- client : Client
edu.colostate.cs.cs414.chesshireCoders.jungleClient.app
App
- window : Stage
- client : JungleClient
+ App()
+ getJungleClient() : JungleClient
+ start() : Stage
+ setScene(s: Scene)
+ setScene(st: String)
LoginContoller
- hplLogin : Hyperlink
- btnLogin: Button
- emailField : Button
- passwordField : TextField
- loginErrorMsg : Label
+ LoginController()
+ loginClicked()
+ loginSuccess()
+ loginFailure()
+ registerClicked()
+ initialize(u: URL, r: ResourceBundle)
0..1
RegisterController
- hplLogin : Hyperlink
- btnRegister : Button
- emailField : TextField
- nickNameField : TextField
- passwordField: TextField
- passwordReenterField : TextField
- alreadyRegistered : Label
+ RegisterController()
+ loginClicked()
+ registerClicked()
+ registrationSuccess()
+ registrationFailure()
+ initialize(u: URL, r: ResourceBundle)
0..1
HomeController
- borderPane : BoarderPane
- btnPlay: Button
- btnLogout: Button
- btnSettings : Button
- btnViewInvites : Button
- btnViewGameHistory : Button
- mainVBox : VBox
- unregSuccess : StackPane
+ HomeController()
+ Initialize(u: URL, r: ResourceBundle)
+ logoutClicked()
+ playClicked()
+ settingsClicked()
+ unregisterClicked()
+ unregisterSuccess()
+ unregisterFailure();
+ unregSuccessReturnClicked()
+ viewGameHistoryClicked()
+ viewInvitesClicked()
0..1
GameBoardController
- start : int[]
- gridPane : GridPane
+ GameBoardController()
+ Initialize(u: URL, r: ResourceBundle)
+ squareClicker(m: MouseEvent)
- highlightStartSquare(s: StackPane, r: int, c: in)
- highlightMoves(r: int, c: int)
- movePiece(r: int, c: int)
- setHighlight(s: StackPane, p: Paint)
- removePreviousHighlight()
-getSquare(r: int, c: int) : StackPane
1
- board : GameBoard
-loginController : LoginController
-registerController : RegisterController
-homeController : HomeController
edu.colostate.cs.cs414.chesshireCoders.jungleClient.account
0..*
AccountHandler
- registerResponseListener : Listener
- loginResponseListener : Listener
- unregisterResponseListener: Listener
- logoutResponseListener : Listener
+ registerUser(e: String, n: String, p: String, pw: Sting, o: RegisterController)
+ unregisterUser(e: String, n: String, o: HomeController)
+ validateLogin(e: String, p: String, o: LoginController)
+ logout(s: String)
0..*
0..*
0..*
edu.colostate.cs.cs414.chesshireCoders.jungleClient.app.game
1
GameBoard
+ GameBoard()
+ getPieceAt(r: int, c: int)
+ getSquareAt(r: int, c: int)
+ getValidMoves(r: int, c: int) : int[][]
+ movePiece(r: int[], c: int[]) : void
- getValidMoveHorizontal(g: GamePiece, r: int) : int
- getValidMoveVertical(g: GamePiece, c: int) : int
- setupBoard()
1
- squares : Square[][]
63
Player
- playerColor : PlayerColor
- user : User
+ Player(u: User, c: PlayerColor)
<<enumeration>>
PlayerColor
Red
Black
BoardSquare
- location : int[]
+ BoardSquare(r: int, c: int, g: GamePiece)
+ BoardSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
+ clearPiece()
+ getColor() : PlayerColor
+ getColumn() : int
+ getPiece() : GamePiece
+ getRow() : int
+ isEmpty() : Boolean
+ setPiece(p: GamePiece)
1
- piece : GamePiece
0..1
RiverSquare
+ TrapSquare(r: int, c: int, g: GamePiece)
+ TrapSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
+ setPiece(g: GamePiece)
DenSquare
+ DenSquare(r: int, c: int, g: GamePiece)
+ DenSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
TrapSquare
+ TrapSquare(r: int, c: int, g: GamePiece)
+ TrapSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
+ setPiece(g: GamePiece)
GamePiece
- powerLevel : int
- row : int
- column : int
+ GamePiece(r: int, c: int, p: PlayerColor)
+ GamePiece(g: Gamepiece)
+ changeLocation(r: int, c: int)
+ getColor() : PlayerColor
+ getPowerLevel() : int
+ setPowerLevel(l: int)
+ setPowerDefault()
+ canOccupy(b: BoardSquare) : Boolean
- isFriendlyDen(b: BoardSquare) : Boolean
- squareIsAdjacent(b: BoardSquare) : Boolean
- canCapture(b: BoardSquare) : Boolean
+ getRow() : int
+ getColumn() : int
+ setRow(r: int)
+ setColumn(c: int)
LeopardPiece
+ LeopardPiece(r: int, c: int, p: PlayerColor)
+ canOccupy(b: BoardSquare) : Boolean
+ setPowerDefault()
RatPiece
+ RatPiece(r: int, c: int, p: PlayerColor)
+ canOccupy(b: BoardSquare) : Boolean
+ setPowerDefault()
FoxPiece
+ FoxPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
DogPiece
+ DogPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
TigerPiece
+ TigerPiece(r: int, c: int, p: PlayerColor)
+ canOccupy(b: BoardSquare) : Boolean
+ setPowerDefault()
CatPiece
+ CatPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
LionPiece
+ LionPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
ElephantPiece
+ ElephantPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()

Class Diagram

Client Logic



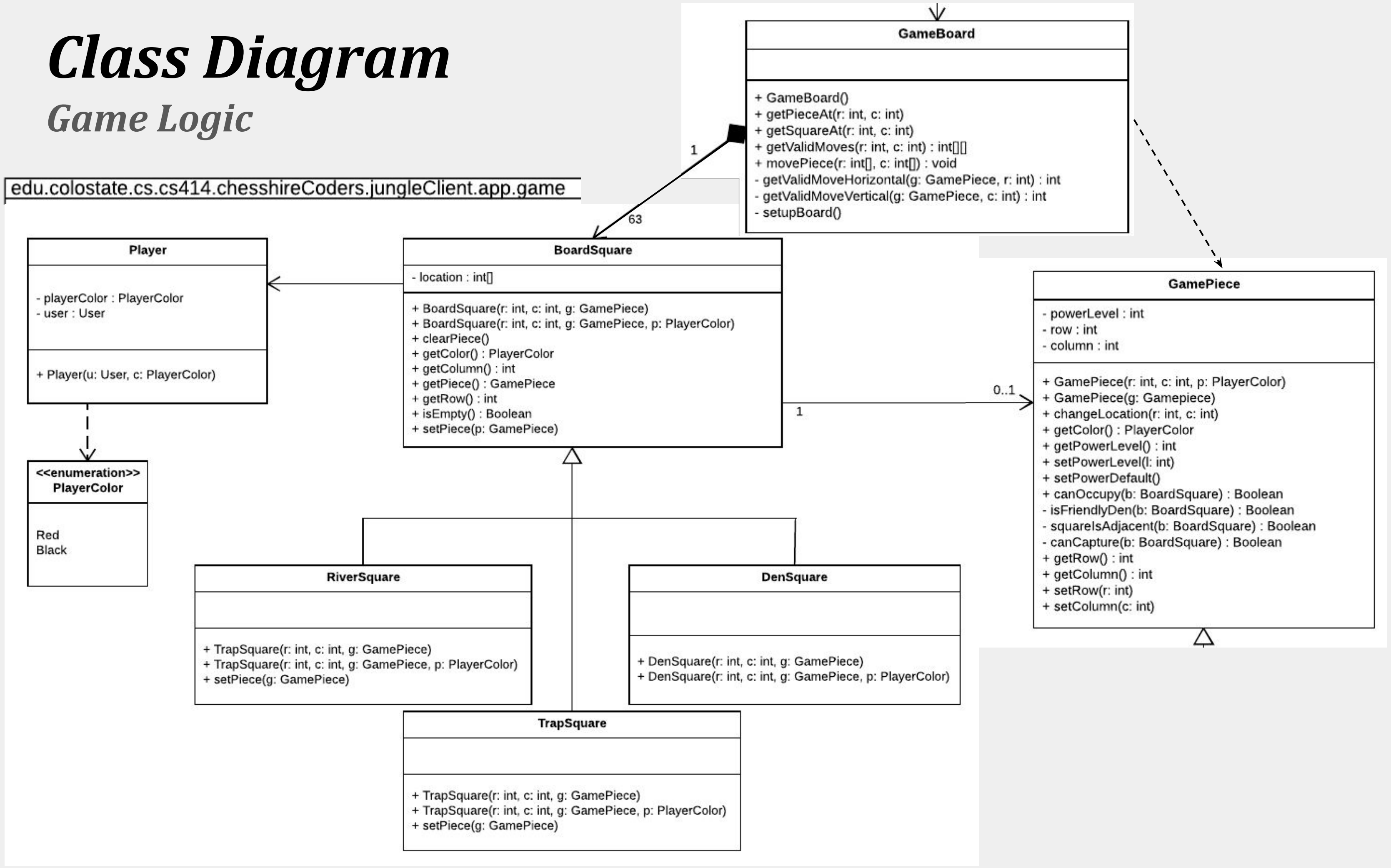
Class Diagrams - Client cont.



edu.colostate.cs.cs414.chesshireCoders.jungleClient.client
NetworkListener
+ NetworkListener()
+ addEventListeners(c: Client)
JungleClient
- client : Client
+ JungleClient()
+ start()
+ connect(a: int, s: String, c: int)
+ stop()
+ reconnect()
+ sendMessageExpectsResponse(o: Object, l: listener)
+ sendMessage(o: Object)
+ addListener()
+ removeListener()
- sendTCP(o: Object)
0..1
1
-client : Client
edu.colostate.cs.cs414.chesshireCoders.jungleClient.account
0..*
0..*
0..*
0..*
AccountHandler
- registerResponseListener : Listener
- loginResponseListener : Listener
- unregisterResponseListener: Listener
- logoutResponseListener : Listener
+ registerUser(e: String, n: String, p: String, pw: Sting, o: RegisterController)
+ unregisterUser(e: String, n: String, o: HomeController)
+ validateLogin(e: String, p: String, o: LoginController)
+ logout(s: String)

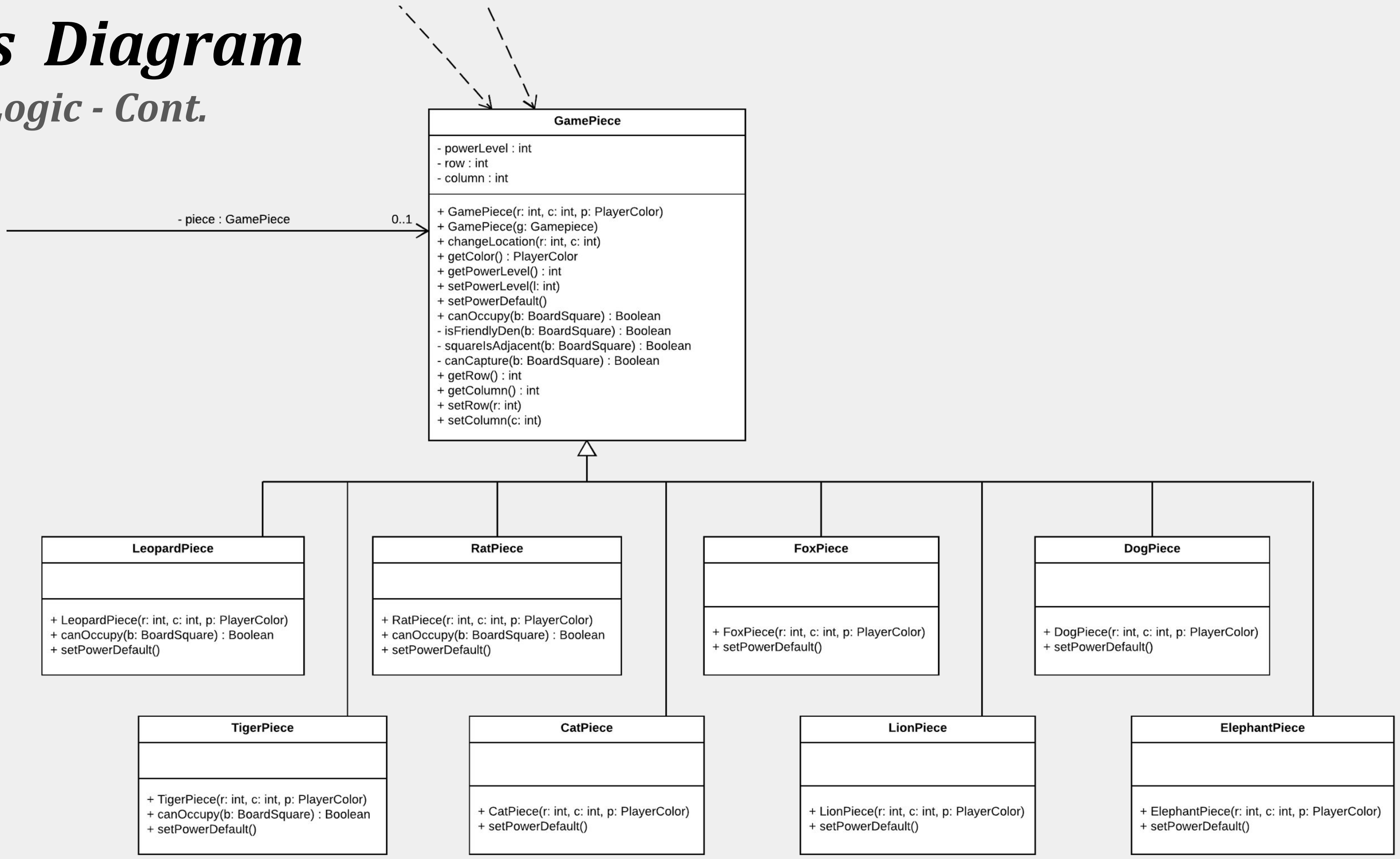
Class Diagram

Game Logic

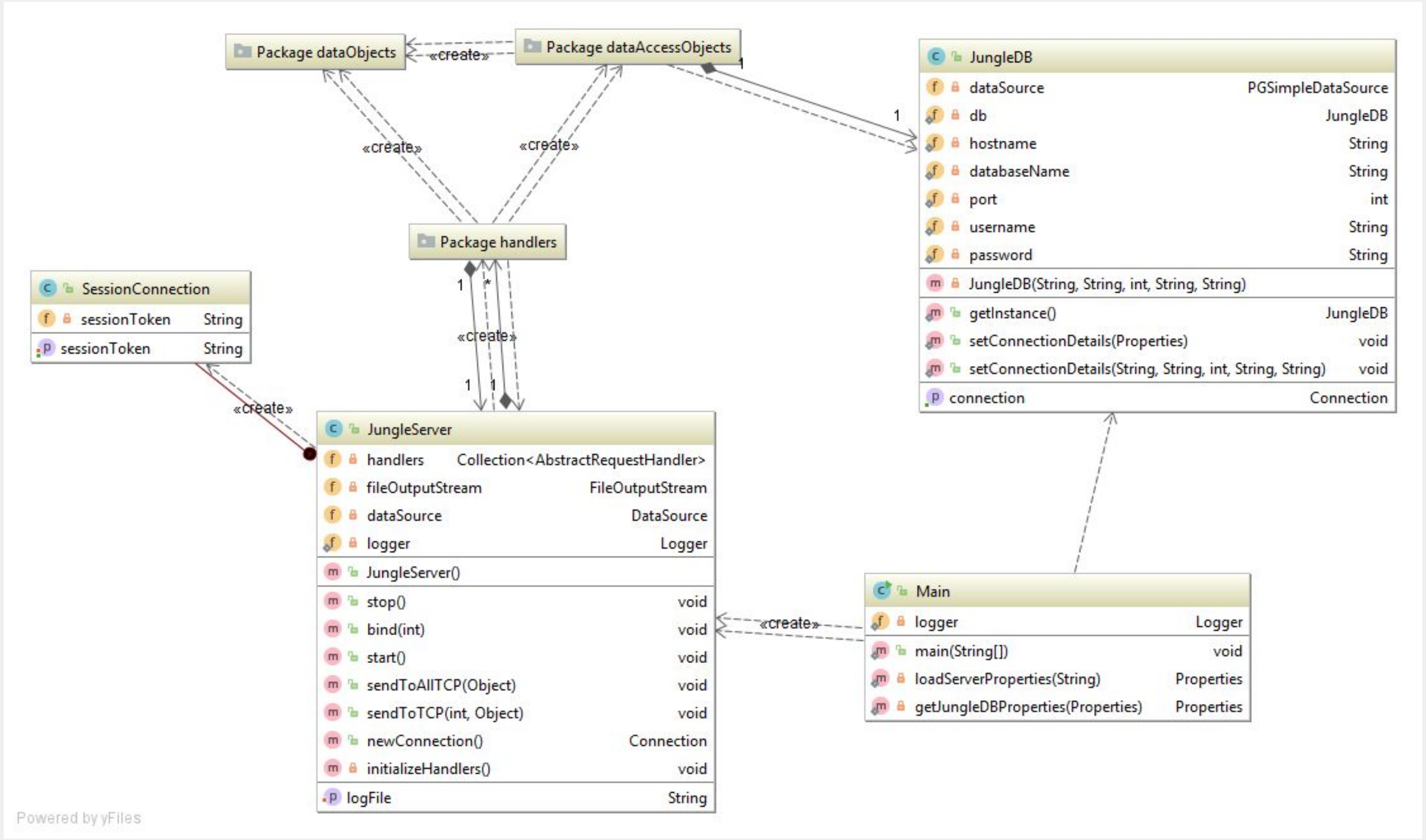

edu.colostate.cs.cs414.chesshireCoders.jungleClient.app.game
GameBoard
+ GameBoard()
+ getPieceAt(r: int, c: int)
+ getSquareAt(r: int, c: int)
+ getValidMoves(r: int, c: int) : int[][]
+ movePiece(r: int[], c: int[]) : void
- getValidMoveHorizontal(g: GamePiece, r: int) : int
- getValidMoveVertical(g: GamePiece, c: int) : int
- setupBoard()
1
63
Player
- playerColor : PlayerColor
- user : User
+ Player(u: User, c: PlayerColor)
<<enumeration>>
PlayerColor
Red
Black
BoardSquare
- location : int[]
+ BoardSquare(r: int, c: int, g: GamePiece)
+ BoardSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
+ clearPiece()
+ getColor() : PlayerColor
+ getColumn() : int
+ getPiece() : GamePiece
+ getRow() : int
+ isEmpty() : Boolean
+ setPiece(p: GamePiece)
1
0..1
GamePiece
- powerLevel : int
- row : int
- column : int
+ GamePiece(r: int, c: int, p: PlayerColor)
+ GamePiece(g: Gamepiece)
+ changeLocation(r: int, c: int)
+ getColor() : PlayerColor
+ getPowerLevel() : int
+ setPowerLevel(l: int)
+ setPowerDefault()
+ canOccupy(b: BoardSquare) : Boolean
- isFriendlyDen(b: BoardSquare) : Boolean
- squareIsAdjacent(b: BoardSquare) : Boolean
- canCapture(b: BoardSquare) : Boolean
+ getRow() : int
+ getColumn() : int
+ setRow(r: int)
+ setColumn(c: int)
RiverSquare
+ TrapSquare(r: int, c: int, g: GamePiece)
+ TrapSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
+ setPiece(g: GamePiece)
DenSquare
+ DenSquare(r: int, c: int, g: GamePiece)
+ DenSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
TrapSquare
+ TrapSquare(r: int, c: int, g: GamePiece)
+ TrapSquare(r: int, c: int, g: GamePiece, p: PlayerColor)
+ setPiece(g: GamePiece)

Class Diagram

Game Logic - Cont.


- piece : GamePiece
0..1
GamePiece
- powerLevel : int
- row : int
- column : int
+ GamePiece(r: int, c: int, p: PlayerColor)
+ GamePiece(g: Gamepiece)
+ changeLocation(r: int, c: int)
+ getColor() : PlayerColor
+ getPowerLevel() : int
+ setPowerLevel(l: int)
+ setPowerDefault()
+ canOccupy(b: BoardSquare) : Boolean
- isFriendlyDen(b: BoardSquare) : Boolean
- squareIsAdjacent(b: BoardSquare) : Boolean
- canCapture(b: BoardSquare) : Boolean
+ getRow() : int
+ getColumn() : int
+ setRow(r: int)
+ setColumn(c: int)
LeopardPiece
+ LeopardPiece(r: int, c: int, p: PlayerColor)
+ canOccupy(b: BoardSquare) : Boolean
+ setPowerDefault()
RatPiece
+ RatPiece(r: int, c: int, p: PlayerColor)
+ canOccupy(b: BoardSquare) : Boolean
+ setPowerDefault()
FoxPiece
+ FoxPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
DogPiece
+ DogPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
TigerPiece
+ TigerPiece(r: int, c: int, p: PlayerColor)
+ canOccupy(b: BoardSquare) : Boolean
+ setPowerDefault()
CatPiece
+ CatPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
LionPiece
+ LionPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()
ElephantPiece
+ ElephantPiece(r: int, c: int, p: PlayerColor)
+ setPowerDefault()

Class Diagrams - Server



SessionConnection

sessionToken	String
sessionToken	String

JungleDB

dataSource	PGSimpleDataSource
db	JungleDB
hostname	String
databaseName	String
port	int
username	String
password	String
JungleDB(String, String, int, String, String)	
getInstance()	JungleDB
setConnectionDetails(Properties)	void
setConnectionDetails(String, String, int, String, String)	void
connection	Connection

JungleServer

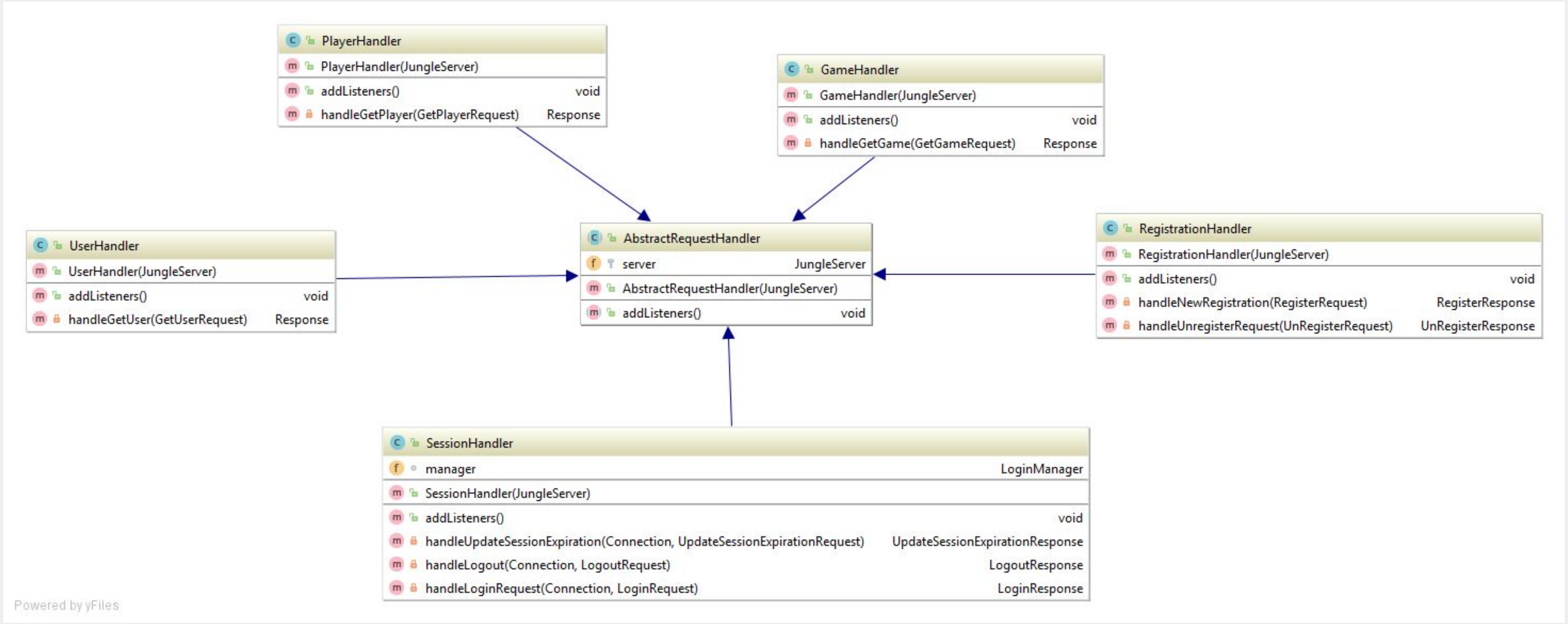
handlers	Collection<AbstractRequestHandler>
fileOutputStream	FileOutputStream
dataSource	DataSource
logger	Logger
JungleServer()	
stop()	void
bind(int)	void
start()	void
sendToAllTCP(Object)	void
sendToTCP(int, Object)	void
newConnection()	Connection
initializeHandlers()	void
logFile	String

Main

logger	Logger
main(String[])	void
loadServerProperties(String)	Properties
getJungleDBProperties(Properties)	Properties

Class Diagrams - Server cont.

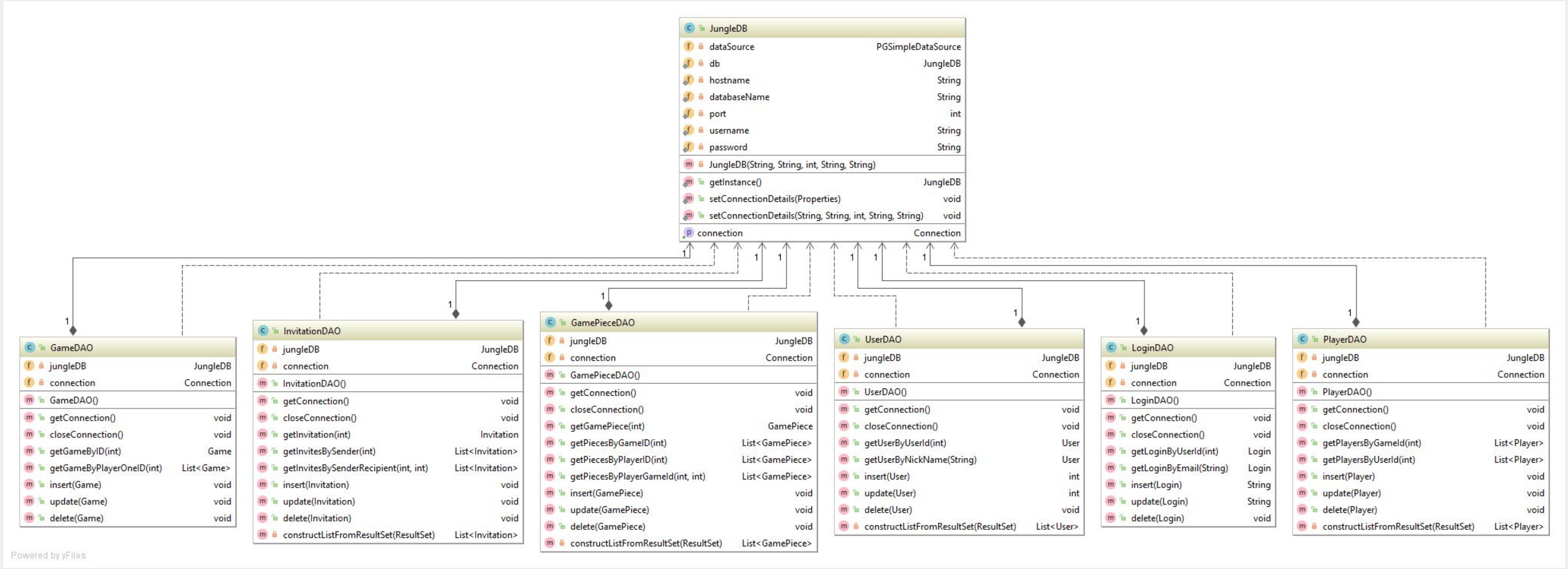
Handlers



PlayerHandler
PlayerHandler(JungleServer)
addListeners() void
handleGetPlayer(GetPlayerRequest) Response
GameHandler
GameHandler(JungleServer)
addListeners() void
handleGetGame(GetGameRequest) Response
UserHandler
UserHandler(JungleServer)
addListeners() void
handleGetUser(GetUserRequest) Response
AbstractRequestHandler
server JungleServer
AbstractRequestHandler(JungleServer)
addListeners() void
RegistrationHandler
RegistrationHandler(JungleServer)
addListeners() void
handleNewRegistration(RegisterRequest) RegisterResponse
handleUnregisterRequest(UnRegisterRequest) UnRegisterResponse
SessionHandler
manager LoginManager
SessionHandler(JungleServer)
addListeners() void
handleUpdateSessionExpiration(Connection, UpdateSessionExpirationRequest) UpdateSessionExpirationResponse
handleLogout(Connection, LogoutRequest) LogoutResponse
handleLoginRequest(Connection, LoginRequest) LoginResponse
Powered by yFiles

Class Diagrams - Server cont.

Data Access Objects



JungleDB	
dataSource	PGSimpleDataSource
db	JungleDB
hostname	String
databaseName	String
port	int
username	String
password	String
JungleDB(String, String, int, String, String)	
getInstance()	JungleDB
setConnectionDetails(Properties)	void
setConnectionDetails(String, String, int, String, String)	void
connection	Connection

GameDAO	
jungleDB	JungleDB
connection	Connection
GameDAO()	
getConnection()	void
closeConnection()	void
getGameByID(int)	Game
getGameByPlayerOneID(int)	List<Game>
insert(Game)	void
update(Game)	void
delete(Game)	void

InvitationDAO	
jungleDB	JungleDB
connection	Connection
InvitationDAO()	
getConnection()	void
closeConnection()	void
getInvitation(int)	Invitation
getInvitesBySender(int)	List<Invitation>
getInvitesBySenderRecipient(int, int)	List<Invitation>
insert(Invitation)	void
update(Invitation)	void
delete(Invitation)	void
constructListFromResultSet(ResultSet)	List<Invitation>

GamePieceDAO	
jungleDB	JungleDB
connection	Connection
GamePieceDAO()	
getConnection()	void
closeConnection()	void
getGamePiece(int)	GamePiece
getPiecesByGameID(int)	List<GamePiece>
getPiecesByPlayerID(int)	List<GamePiece>
getPiecesByPlayerGameId(int, int)	List<GamePiece>
insert(GamePiece)	void
update(GamePiece)	void
delete(GamePiece)	void
constructListFromResultSet(ResultSet)	List<GamePiece>

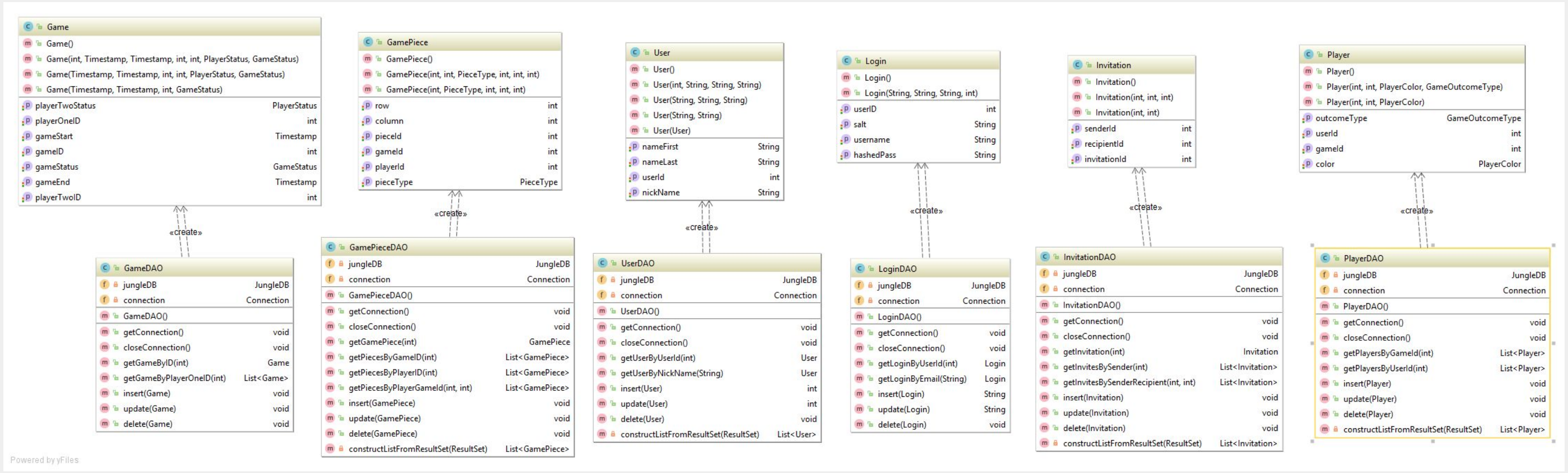
UserDAO	
jungleDB	JungleDB
connection	Connection
UserDAO()	
getConnection()	void
closeConnection()	void
getUserByUserId(int)	User
getUserByNickName(String)	User
insert(User)	int
update(User)	int
delete(User)	void
constructListFromResultSet(ResultSet)	List<User>

LoginDAO	
jungleDB	JungleDB
connection	Connection
LoginDAO()	
getConnection()	void
closeConnection()	void
getLoginByUserId(int)	Login
getLoginByEmail(String)	Login
insert(Login)	String
update(Login)	String
delete(Login)	void

PlayerDAO	
jungleDB	JungleDB
connection	Connection
PlayerDAO()	
getConnection()	void
closeConnection()	void
getPlayersByGameId(int)	List<Player>
getPlayersByUserId(int)	List<Player>
insert(Player)	void
update(Player)	void
delete(Player)	void
constructListFromResultSet(ResultSet)	List<Player>

Class Diagrams - Server cont.

Data Objects/Data Access Objects



Test Case Summary

- ➢ **TestBoardSquare**: 16 tests
- ➢ **TestGameBoard**: 15 tests
- ➢ **TestAccountHandler:** 6 tests
- ➢ **TestBoardSquare**: 6 tests

Project Tools

Development Tools

- ➢ Eclipse
- ➢ IntelliJ IDEA
- ➢ Gluon Scene Builder
- ➢ Maven
- ➢ Git & Github
- ➢ Lucidchart

Libraries & Frameworks

- ➢ JavaFX
- ➢ Mockito
- ➢ JUnit
- ➢ KryoNet

Other

- ➢ DigitalOcean Cloud Hosting
- ➢ PostgreSQL
- ➢ Docker
- ➢ Slack
- ➢ Waffle.io
- ➢ Travis CI

Traceability Link Matrix

	Game	Game Piece	Invitation	Login	Player	User	Jungle Game	Board Square	Game Board
#1: Register to the system			X	X		X			
#2: Create a new game	X		X	X					X
#3: Invite other users to a game	X		X	X					
#4: Respond to Game Invitation	X		X	X	X				X
#5: Quit Game	X			X	X				X
#6: Unregister from System				X		X			
#7: View Player Profile				X	X				
#8: Log in to System				X		X			
#9: Log out of System				X		X			
#10: Move Game Piece		X		X	X			X	X
#11: Switch Game	X			X	X				X

Use Case Completion

Use Case	Progress Notes
#1: Register to the system	Most server-side logic in place, validation, redirection to new pase
#2: Create a new game	Client and GUI logic completed.
#3: Invite other users to a game	Data objects and network handlers in place.
#4: Respond to Game Invitation	Server configuration ready to handle invite send/receive events
#5: Quit Game	Game logic functional, working on server implementation.
#6: Unregister from System	GUI elements done, some server side logic in place
#7: View Player Profile	
#8: Log in to System	GUI elements done, server logic in place
#9: Log out of System	GUI elements done, server logic in place
#10: Move Game Piece	GUI and client logic done, some server logic in place
#11: Switch Game	Basic logic in place, working on server side information

Demo

Questions And Discussion