CS 414 - Object-Oriented Design
# Fall 2020

P1

Colorado State University

Meet the Team!

Unspecified Behavior

- Adam Anderson
- Brent Staab
- Keabeth Gonzales
- Kai Griem
- Maddie Mihevc (Scrum Master)

Description of Portal Chess

Portal chess is a derivative of the classical chess game. The main difference is the introduction of two new 'fairy pieces' called portals which allow the pieces to move in new and exciting ways.

Colorado State University
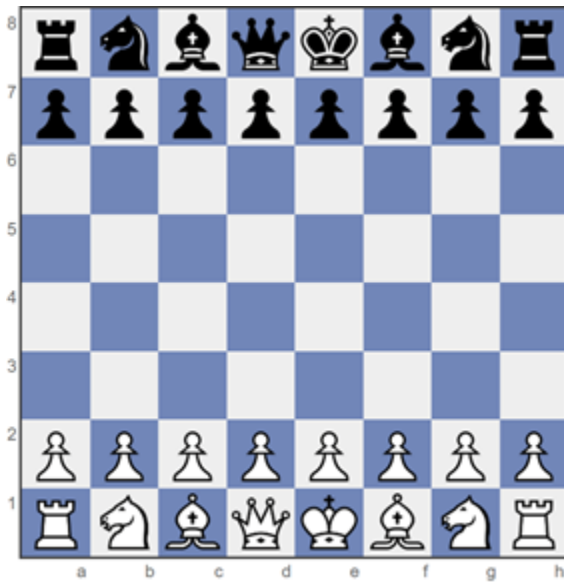
Rules

There are several types of Portal chess, but we are planning on implementing the version described by **wika57** on his YouTube video. https://www.youtube.com/watch?v=Zq4pJw_nk_0

Setup

- The basic rules of classical chess apply
  - Pieces have the same restrictions, and the overall objective is the same (checkmate the King)

- Players place their portals before the game starts
  - White pieces can only go in row 4
  - Black pieces can only go in row 5

- You can only control your own colored portal

Rules - continued

Game Play

- A piece passes through a portal unless the portal is blocked
- A piece can be blocked from entering a portal if a piece of the same color is on the other side.
- A piece can take another piece by using the portal
- Both portals cannot be occupied at the same time
- A piece **can** check/checkmate people using the portal
- A piece must exit the portal in the direction it traveled into the portal
- A portal can be moved onto a piece causing it to teleport. *Unless the other portal is occupied*
- A player has a choice to move a chess piece or the portal piece in one turn, not both.
- If your own chess piece is occupying a portal, you cannot pass another one of your own pieces through the unoccupied portal
- A piece can enter either portal regardless of color
- Portals can **not** occupy the same square
- A portal can be moved regardless of whether it is occupied or not, the occupying piece does not move with the portal
- If a piece occupies one portal and a whole turn phase end without the piece moving from the teleporter it does not teleport.
- A Knight can jump over a portal piece
- A pawn can become a queen by using a portal

Actors and Stakeholders

- Unregistered Users
  - No user information stored
  - Not allowed to play a game

- Registered Users
  - Full user profile stored
    - Includes Name, e-mail, active game(s), g

- Logged in user
  - View profiles
  - Can play one or more games
  - Can invite players to a game
  - Can accept invitations to a game
  - Play/Quit a game
  - Unregister
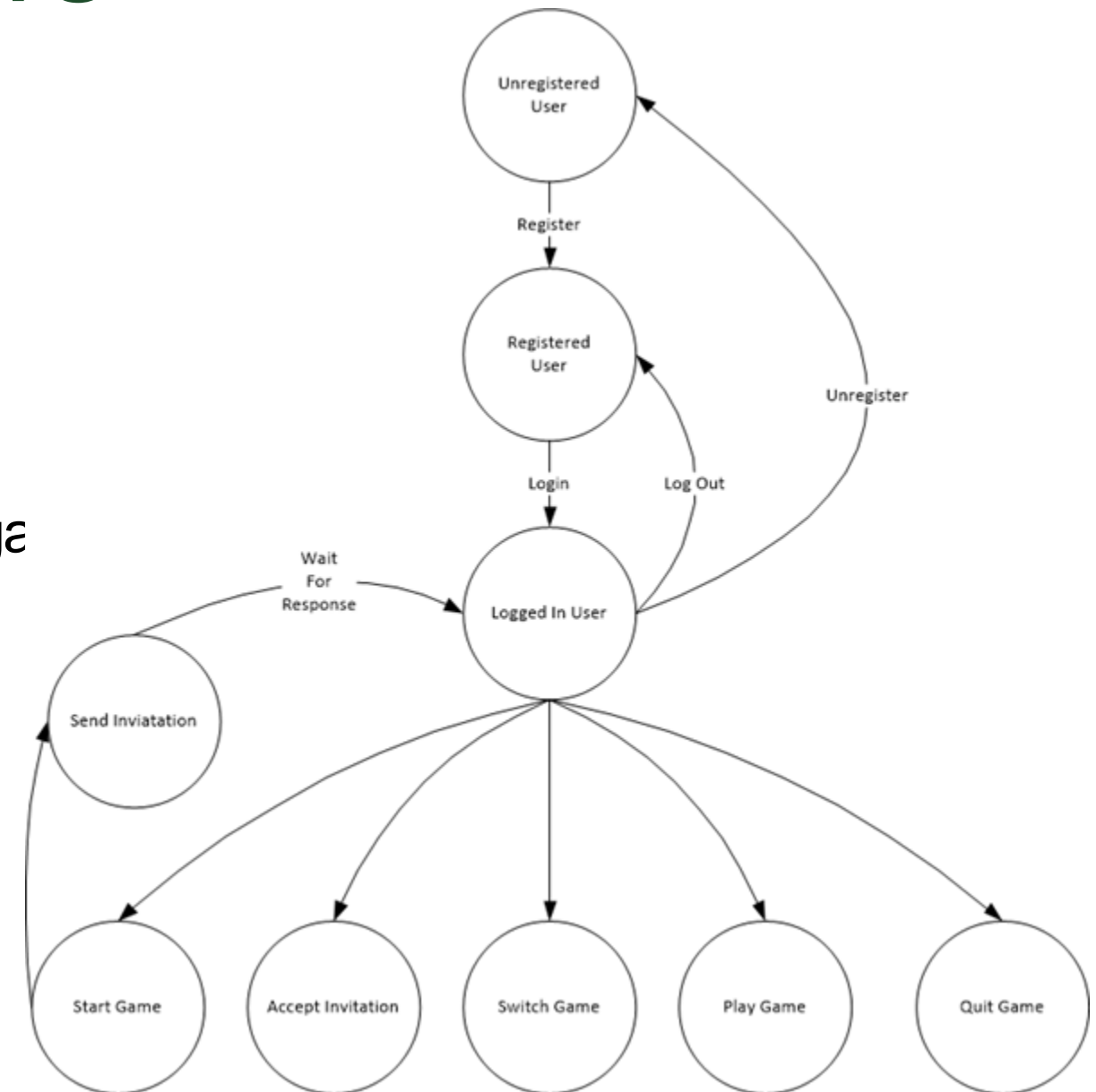
Team Decisions

Tools

- Github
  - Source Code Management (SCM)
  - https://github.com/bstaab/cs414-f20-UnspecifiedBehavior

- Zenhub
  - Kanban board
  - Integrated with Github

- Microsoft Teams

- Java

- JUnit 5

User Stories and Tasks

1. I would like to play by the rules of Portal Chess.
   a. Create chess pieces
      i. Create pieces with movements based on type
      ii. Pieces are: King, Queen, Bishop, Rook, Knight, Pawn
   b. Create Chess board
      i. Chess board will hold chess pieces
      ii. Chessboard will implement portals
2. I would like to register with the system to get access
   a. Register UI
      i. Page users register through
      ii. Contains fields for email, password and username
   b. Registration Verification
      i. Upon submission a prospective user should have their username and email information verified for uniqueness. Either Accept or Reject
   c. Store new user data
      i. If a new user submits unique information their login should be stored

3. I can log into the system to play a game
   a. Login UI
      i. Page for user to log in with their credentials
      ii. Contains username and password fields
      iii. Submit button
   b. Verify Login Fields
      i. The system should check the submitted login fields to determine if this is a valid user and reject/accept the user.
   c. Get User Data
      i. On successful log in get user information
4. I can create a new game
   a. Create a match UI
      i. Create a new game
      ii. invite a user field
   b. Create a new game
      i. Call Chessboard
      ii. Populate chess pieces black/white
   c. Game in progress UI
      i. Where the user will interact with an in-progress game

Acceptance Criteria

- I can create a new game

  - <u>Acceptance Criteria:</u> A user can create a new chess match where the pieces are in their default positions on the board.

- I would like to play by the rules of Portal Chess.

  - <u>Acceptance Criteria:</u> Chess matches can be played, and completed, in which all moves are legal as defined by the Portal Chess rules.

- I would like to register with the system to get access

  - <u>Acceptance Criteria:</u> A user can register to the platform by entering an email, nickname, and password. The email and nickname must be unique at the time of registration.

- I can log into the system to play a game

  - <u>Acceptance Criteria:</u> A registered user can enter their email/nickname along with their password and create/play a game.

Kanban Board

Icebox

---

cs414-f20-Unspecifie... **#52**
Search Users

Filter by Epic Issues

`Epic` `User Story`

---

cs414-f20-Unspecifie... **#14**
Create Subclass for King

`Task`

---

cs414-f20-Unspecifie... **#15**
Create Subclass Queen

`Task`

---

cs414-f20-Unspecifie... **#10**
Create Class ChessPiece

`Task`

Backlog

cs414-f20-Unspecifie... **#28**
Portal Chess

Filter by Epic Issues

`Epic` `User Story`

cs414-f20-Unspecifie... **#56**
Create Chess Pieces

🔖 Portal Chess

`Task`

cs414-f20-Unspecifie... **#44**
Create ChessBoard

🔖 Portal Chess

`Task`

---

cs414-f20-Unspecifie... **#46**
Register

Filter by Epic Issues

`Epic` `User Story`

cs414-f20-Unspecifie... **#57**
Registration UI

🔖 Register

`Task`

cs414-f20-Unspecifie... **#38**
Registration verification

🔖 Register

`Task`

---

cs414-f20-Unspecifie... **#40**
Store new user data

🔖 Register

`Task`

cs414-f20-Unspecifie... **#29**
Login

Filter by Epic Issues

`Epic` `User Story`

cs414-f20-Unspecifie... **#58**
Login UI

🔖 Login

`Task`

---

cs414-f20-Unspecifie... **#33**
Verify login fields

cs414-f20-Unspecifie... **#59**
Get User Data

cs414-f20-Unspecifie... **#47**
Create Game

Output of Scrum Ceremonies

9/14/20

- Summary of Meeting: During our meeting we made several decisions about how we would like to function as a team. We decided to have a scheduled weekly meeting at 3 o'clock on Fridays no matter what. Other meetings times will be flexible and occur when we need to meet to discuss our progress. In order for the Github to be set up we shared our Github usernames and decided on the etiquette we plan on following for merge requests and managing branches. No one felt as though they had a firm understanding of what is expected from them for this project, so we decided to take time until our next meeting to get a greater understanding.
- Tasks:
    - Maddie – email Dr. Cubillos about if we are allowed to use Zenhub, reactstrap, and oper source content, read through description and begin coming up with any user stories/tasks, along with also any questions that may come up
    - Brent – Set up Github repository, read through description and begin coming up with any user stories/tasks, along with also any questions that may come up
    - Kai – read through description and begin coming up with any user stories/tasks, along with also any questions that may come up
    - Keabeth – read through description and begin coming up with any user stories/tasks, along with also any questions that may come up
    - Adam – read through description and begin coming up with any user stories/tasks, along with also any questions that may come up
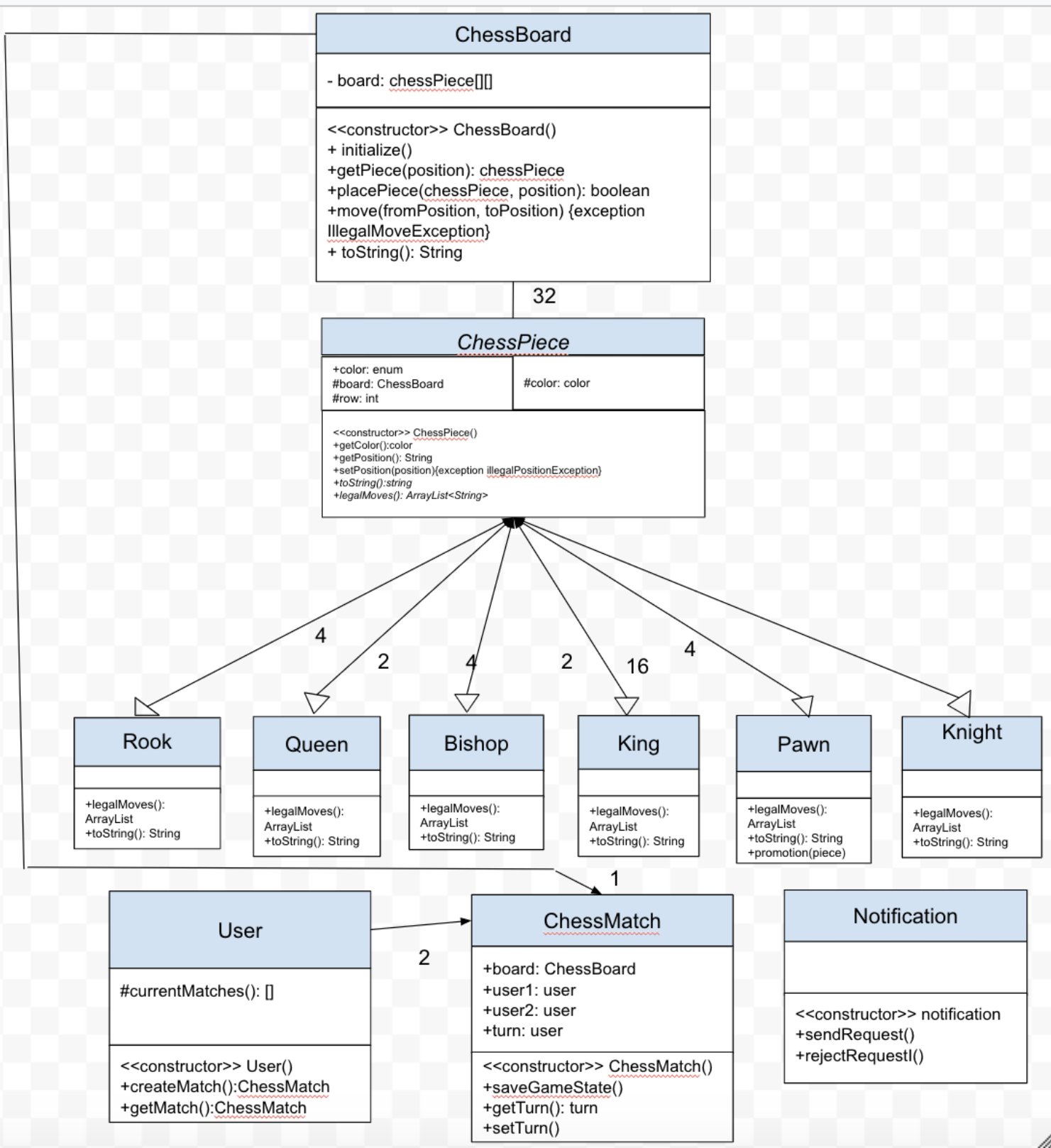
9/25/20

- Summary of Meeting: After a few of us met with Dr. Cubillos today we gained a better understanding of what is expected from us. We are going to learn from this sprint and make sure we talk to the product owner right away instead of waiting till the very end. During our scrum meeting today we went over the user stories that we created and then simplified our tasks down in order to create basic tasks to go along with our new user stories. We then working on finalizing the CRC cards before moving onto working on the class diagram together as a team. We then decided what our next steps will be from here in order to make sure that we are ready to record our presentation.
- Tasks:
    - Maddie - Met with professor tomorrow, record presentation, record quick into video
    - Brent – Met with professor tomorrow, record quick intro video
    - Kai – Met with professor tomorrow, record quick intro video
    - Keabeth - Clean up CRC cards, record quick intro video
    - Adam - record quick intro video

CRC Cards

ChessMatch	
• Saves Game State • Keeps track of whose turn it is	• ChessBoard • User

User	
• Creates matches • Moves pieces	• ChessMatch

ChessPiece	
• Contains Chess Piece type • Contains Color • Contains position • Contains whether it is alive/dead	• ChessBoard • Rook • Bishop • Knight • Queen • King • Pawn

ChessBoard	
• Populates Chess pieces	• Chesspiece and subclasses

King	
• Extends ChessPiece • Checks is movement is legal • Updates position • Keeps track of check • Checks if checkmate	• ChessPiece

Rook	
• Extends ChessPiece • Checks is movement is legal • Updates position	• ChessPiece

Bishop	
• Extends ChessPiece • Checks is movement is legal • Updates position	• ChessPiece

Knight	
• Extends ChessPiece • Checks is movement is legal • Updates position	• ChessPiece

Pawn	
• Extends ChessPiece • Checks is movement is legal • Updates position • Promotion cast pawn to another piece	• ChessPiece • Rook • Bishop • Knight • Queen

Queen	
• Extends ChessPiece • Checks is movement is legal • Updates position	• ChessPiece

Class Diagram

ChessBoard

- board: chessPiece[][]

<<constructor>> ChessBoard()
+ initialize()
+getPiece(position): chessPiece
+placePiece(chessPiece, position): boolean
+move(fromPosition, toPosition) {exception IllegalMoveException}
+ toString(): String

32

ChessPiece

+color: enum	
#board: ChessBoard	#color: color
#row: int	

<<constructor>> ChessPiece()
+getColor():color
+getPosition(): String
+setPosition(position){exception illegalPositionException}
+toString():string
+legalMoves(): ArrayList<String>

4   2   4   2   16   4

Rook

+legalMoves(): ArrayList
+toString(): String

Queen

+legalMoves(): ArrayList
+toString(): String

Bishop

+legalMoves(): ArrayList
+toString(): String

King

+legalMoves(): ArrayList
+toString(): String

Pawn

+legalMoves(): ArrayList
+toString(): String
+promotion(piece)

Knight

+legalMoves(): ArrayList
+toString(): String

User

#currentMatches(): []

<<constructor>> User()
+createMatch():ChessMatch
+getMatch():ChessMatch

2

1

ChessMatch

+board: ChessBoard
+user1: user
+user2: user
+turn: user

<<constructor>> ChessMatch()
+saveGameState()
+getTurn(): turn
+setTurn()

Notification

<<constructor>> notification
+sendRequest()
+rejectRequestI()

Thank you

Colorado State University