AP Computer Science A

Period 8, Raihan Nadeem & Jonathan Jiang

Final Project Prototype

RaiJin' Tokens

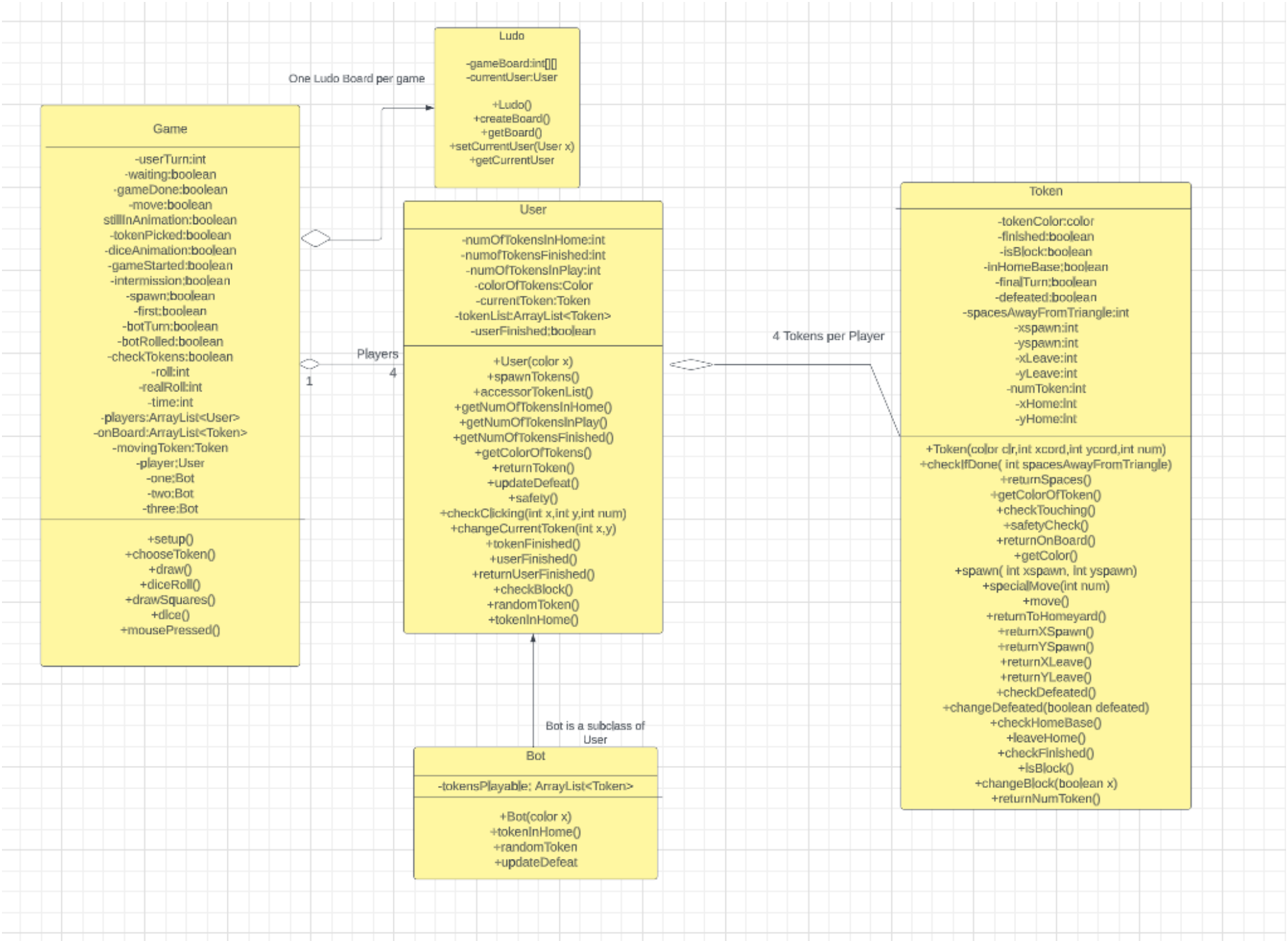
Project Description: The purpose of this project is to create a polished version of the Ludo board game while also adding some neat GUIs and animations of the gameplay. Ludo is played on a square board divided into four colored quadrants, with each quadrant representing a player's home base. The goal is to move all four of your colored game tokens from your home base to the center of the board, known as "home." In order to move the token, players take turns rolling a dice and moving their pieces based on the number rolled. The game incorporates strategic decision-making as players can choose to either move their existing pieces forward or bring new pieces into play. There are also special abilities such as being able to send an enemy token back home given that you land on the same spot as one of their tokens.

Expanded Description:

- Critical features (Minimum Viable Product) - What you want to have after 1.5 weeks.
    - A game board in which tokens can move spaces while they're in play
    - A home yard that holds the tokens when they're not in play, preventing players from accessing them while they're in that area.
    - A randomized dice functions with specific scenarios, depending on the outcome (If you roll 6's, you get to take a token out of your home yard, but if there are no more tokens in your home yard, you get to move your token and then roll one more time)
    - Automatically letting one token out of the home base to prevent a long wait time (you typically need to roll a six in order to get a token out of your home base)
    - Basic movement for the tokens (instantaneous)
    - Interactions between tokens when they land on the same space, depending on whether it's a friendly or enemy token
    - A feature that allows the player to pick the token they want to move with their mouse
    - An endpoint (home triangle) for all the tokens to go once they have all moved around the entire board

- Nice to have features - What you would want to have by the end
    - A more complicated end goal of reaching the home triangle with the EXACT number of spaces moved (If the home triangle is 3 spaces away and you roll a 4, you don't get to put your token inside the home triangle)
    - An animation depicting the movement of the tokens across the board instead of instantaneously moving them to the correct spot
    - Add extra "special" spots on the board that give you boosts (e.g: roll twice/can't get killed by an enemy on certain spots)
    - Designing a Bot class that allows players to play with enemies to make the game more engaging
    - Create GUIs that allow the user to customize their game (e.g: pick color, pick # of bots matched against)
    - When a user rolls a six, allow them to pick between getting a token out of their home base *or* move one of their current tokens.

Page 2: UML Diagram



Outline for Methods

1) Board is created through the createBoard() method and actually drawn with the drawSquares() and spawnTokens() method that's called in the setup()
    a) It creates all the tiles that the tokens can move on, including important landmarks like where the token spawns when it leaves the home yard and where it starts moving towards the end point.
    b) It also creates the users to play the game, setting numOfTokensInHome at 4 and numOfTokensFinished at 0, with colorOfTokens set to the color the player selected before the game started.
2) At the beginning of each player's turn, they can roll the dice by clicking on the dice outside of the gameboard.
    a) The diceRoll() method is subsequently called, returning the user a number between 1 and 6.
    b) The user would then have the opportunity to click on one of their tokens, if the token is outside of the homebase or if the user rolled a 6, they can bring one of their tokens out.
        i) If the user doesn't have a token out of their homebase and they don't roll a 6, their turn will be skipped and they will have to wait for another chance to roll a 6.
    c) Once the mouse is pressed, the token that was on the mouse's cursor would be selected as the current token that will be moved with the move() method.
        i) Once the token is moved onto a tile, the result from diceRoll is subtracted from the spacesAwayFromTriangle variable.
        ii) It then performs the methods checkTouching(), checkBlock() and checkFinished().
        iii) If checkTouching returns true, it causes the other Token that's a different color to use the method returnToHomeYard(), where the other token is forced to move back into their homeyard, and removed from play, so the other user can't select that token to move until they roll another 6.
        iv) However, if the other token is in a block or in other words the same tile with another token of the same color (checked with the isBlock() method), the user's token that landed on the tile with the block is returned back to the homeyard and taken out of play.
        v) If checkFinished() returns true (which is when spacesAwayFromTriangle=0), it puts the token out of play, adding and subtracting 1 to numOfTokensFinished and numOfTokensInPlay respectively, and prevents users from moving it anymore, as it has reached its destination.
    d) Their turn ends after everything has been resolved and all variables have been updated accordingly.
        i) The bots make their moves, as they follow the same steps the user makes, with the exception of picking a random available token. If they roll a 6, they will always move a token out of their homeyard.

Page 3: Development Phases/Stages

Phases:

- ☑ Phase 1: Class Setup, Game and Data Structures
  - ☑ createBoard()
    - ☑ ~~Creates tiles on a processing sketch for the tokens to move on, setting the tiles' xcors and ycors.~~
  - ☑ checkWinner()
    - ☑ ~~Ran after each move to see if a user got all 4 tokens into the home base.~~
  - ☑ userTurn/numPlayers - setup to keep track of who's turn it is (to support bots) and keep track of the # of players

- ☑ Phase 2: User & Tile Class Methods
  - ☑ diceRoll()
    - ☑ ~~Picks a random number between 1 and 6 during a player's turn.~~
  - ☑ mouseClicked()
    - ☑ ~~Checks if the mouse's cursor is on a token~~
    - ☑ ~~If it is, then it moves the token a number of spaces based on the diceRoll() results.~~
  - ☑ tokenOccupied- this boolean is what the token is using to check whether the incoming token needs to run the checkTouching method since if there's a token on the tile already, this boolean will be true.

- ☑ Phase 3: Token Class, subclassess
  - ☑ checkIfDone()
    - ☑ ~~It checks if the spacesAwayFromTriangle is equal to 0 and returns a boolean.~~
  - ☑ checkTouching()
    - ☑ ~~It checks if the space the token is landing on contains another token, which is done by using an ArrayList that is holding the locations of all the other tokens in play via the tiles they are on.~~
  - ☑ spawn()
    - ☑ ~~It takes a token from the homeyard and puts it into play on the board at a specific x-coordinate and y-coordinate, depending on the user and the color they choose.~~
  - ☑ move()

- ☑ ~~It takes a token and moves it counter-clockwise around the board based on the diceRoll.~~
- ☑ returnToHomeyard()
  - ☑ ~~It removes a token from the board and returns it to the owner's homeyard.~~

- ☑ Phase 4: User-Facing Class Methods and Features
  - ☑ Create a conditional for if a dice roll gives you a 6, two buttons will pop up and allow you to select between taking a token out from your home yard or moving forward 6 tiles.
    - ☑ ~~Make another conditional if the numOfTokensInHome is equal to 0 and dice roll gives you a 6, the user gets to move an existing token 6 tiles and then roll the dice one more time, essentially getting another turn.~~
  - ☑ Have a draw method that updates the board every single time a token is moved so the movement is instantaneous.
  - ☐ Incorporate the block mechanic into the game. If two tokens of the same color are in the same tile, which is when the isBlock boolean is true, and another token of a different color lands on that tile, the token with a different color is send back to the homeyard.
  - ☑ Stopping the game when someone gets all their tokens inside the triangle, and presenting a victory screen with their color as the winner.

- ☑ Phase 5 & 6: (POST-MVP) Additional Features, Bot Class, Customization, GUIs
  - ☑ Create a requirement that a token needs to move in the end point with the exact roll and prevent users from selecting that token from moving if the roll is not equal to the number of spaces away from the end point.
  - ☑ Reorganize the entire Game Class so there would be token animations, creating new boolean variables and conditionals as required.
    - ☑ ~~Put the token movement inside of draw~~
    - ☑ ~~Move the user methods inside of draw and create conditionals so they would only be called once, at the start of their turn.~~
  - ☑ Create a bot class that will consist of two phases
    - ☑ ~~Use the superclass constructor to create the Bots, and then override the User methods with new implementations for the Bot class.~~
    - ☑ ~~Put the methods inside of draw and make it work automatically after the user's turn, automating the moves of the other users for a PVE experience.~~
  - ☑ Create an updated scoreboard that will detail who's move it is at a given time and show who's next.

    - ☑ ~~Use colors in the text to represent who goes next.~~
    - ☑ ~~A counter of the tokens in home, in play, and finished.~~
    - ☐ Showing who's next, even for bots
- ☑ Utilize sound to allow for a more immersive experience
    - ☑ ~~Sound for starting the game~~
    - ☑ ~~Sound for rolling the dice~~
- ☑ Create a GUI that allows players to pick which color they want to play as, and set them as the first player, making the other colors bots.

Ideas to break up work:

- We can split methods for each phase and aim to get them done by a certain day. After that, we will merge our code and push it to main and complete a phase, then move onto the next phase. The phases are ordered in terms of what needs to be done for the next class to function properly.
- Or, we can each work on individual classes and merge them together by a certain day. This may result in some functionality not working since some classes rely on others, so the first option would be better.