Planning For Programming

Website Development

LocalSettings Class

Class Name LocalSettings
Attributes (none)
Methods changeEffects() changeMusic() changeFontSize() colorblindMode()

Method Name changeEffects
Method Description and purpose Change the effect volume with a slider listener.
Parameters sliderValue
Return Values (none)

Method Name changeMusic
Method Description and purpose Change the music volume with a slider listner.
Parameters sliderValue
Return Values (none)

Method Name changeFontSize
Method Description and purpose Change the font size with a slider listener.
Parameters sliderValue
Return Values (none)

Method Name colourblindMode
Method Description and purpose Change between different colourblind modes using separate CSS files.
Parameters colourblindType
Return Values (none)

Server Class

Class Name Server
Attributes (list) ObserverCollection
Methods sendToServer() receiveFromServer()

Method Name constructor()
Method Description and purpose Connect to the server and configure.
Parameters (none)
Return Values (none)

Method Name sendToServer()
Method Description and purpose Takes a header and payload string and forwards it to the server with the correct configuration, formatting, etc.
Parameters (String) headers (string) payload
Return Values (bool) isReceived

Method Name receiveFromServer()
Method Description and purpose Waits for the server and alerts the ObserverCollection once there has been a communication from the server. Uses Sockets.
Parameters (none)
Return Values (none)

Method Name registerObserver()
Method Description and purpose Adds an object to the observer collection
Parameters (none)
Return Values (none)

Observer Abstract Class

Class Name

Observer <<ABSTRACT>>
Attributes
Methods Update()

Method Name update()
Method Description and purpose Called by the server whenever there is an event. Implementation should vary for each child class.
Parameters (none)
Return Values (none)

Lobby Class

Class Name Lobby
Attributes (none)
Methods update() sendRoomCode() joinGame() sendTeam() updateTeam()

Method Name update() <<INHERITED>>
Method Description and purpose Recieves server input and either calls JoinGame() or JoinTeam().
Parameters (none)
Return Values (none)

Method Name sendRoomCode()
Method Description and purpose Send the server the roomcode with the server class.
Parameters roomCode
Return Values (none)

Method Name joinGame()
Method Description and purpose Change the page to the lobby page.
Parameters (none)
Return Values (none)

Method Name sendTeam()
Method Description and purpose Send the server the team pick with the server class.
Parameters (none)
Return Values (none)

Method Name updateTeam()
Method Description and purpose Change the team view when someone joins a team. Includes other players joining.
Parameters (String) playerName (String) team
Return Values (none)

BoardState Class

Class Name BoardState
Attributes (Card) cards[][] (String) clueWord (int) numOfGuesses (int) redScore (int) blueScore (Timer) timer (Structure) player --> (String) team --> (String) role
Methods Onclick() ValidateClick() SendBoardState() Update() SendGameOptions() FinishGame()

Note: the server should be able to receive the boardstate from everyone, but only the one with the correct turn structure will do actions.

Method Name constructor()
Method Description and purpose Create a plain grid ready to be filled by the initial state from the server.
Parameters (none)
Return Values (none)

Method Name onClick

Method Description and purpose
Finds where the click is in locates the matching card. First calls validateClick, then calls the revealCard method.
Parameters (none)
Return Values (none)

Method Name
validateClick
Method Description and purpose • Check whether it is the right turn etc
Parameters cardSelected player
Return Values Bool isValidated

Method Name
sendBoardState
Method Description and purpose • Send boardstate with Server class • (on server-side, send new boardstate back to everyone with recieveBoardState)
Parameters boardState cardSelected
Return Values Nothing (1 way HTTP, data received in *RecieveBoardState*)

Method Name
forwardClue
Method Description and purpose • Send spymaster clue • Send spymaster guess total
Parameters BoardState clue
Return Values Nothing (1 way HTTP, data received in *RecieveBoardState*)

Method Name
update <<INHERITED>>
Method Description and purpose Takes the data from the Server class update and displays a new board (also works for an initial board). If the game has finished call FinishGame().
Parameters (none)
Return Values (none)

Method Name sendGameOptions
Method Description and purpose Sent when starting the game.
Parameters • (bool) hasBombCard • (int) aiDifficulty • (float) timerLength • (int) numOfAI
Return Values (none)

Method Name finishGame()
Method Description and purpose Says whether you have won or lost and displays the score.
Parameters Boolean hasWon
Return Values (none)

Card Class

Class Name Card
Attributes (string) colour (String) word (bool) isRevealed (String) imageURL
Methods revealCard()

Method Name revealCard
Method Description and purpose When clicked the card is revealed. Changes to the correct colour with image being shown Card.IsRevealed = true;
Parameters cardSelected
Return Values (none)

Timer Class

Class Name Timer
Attributes (float) timerLength
Methods tick()

timerRunout()

Method Name tick()
Method Description and purpose Decrease size of the timer
Parameters (none)
Return Values (none)

Method Name timerRunOut()
Method Description and purpose Probably call sendBoardState with new turn.
Parameters (none)
Return Values (none)

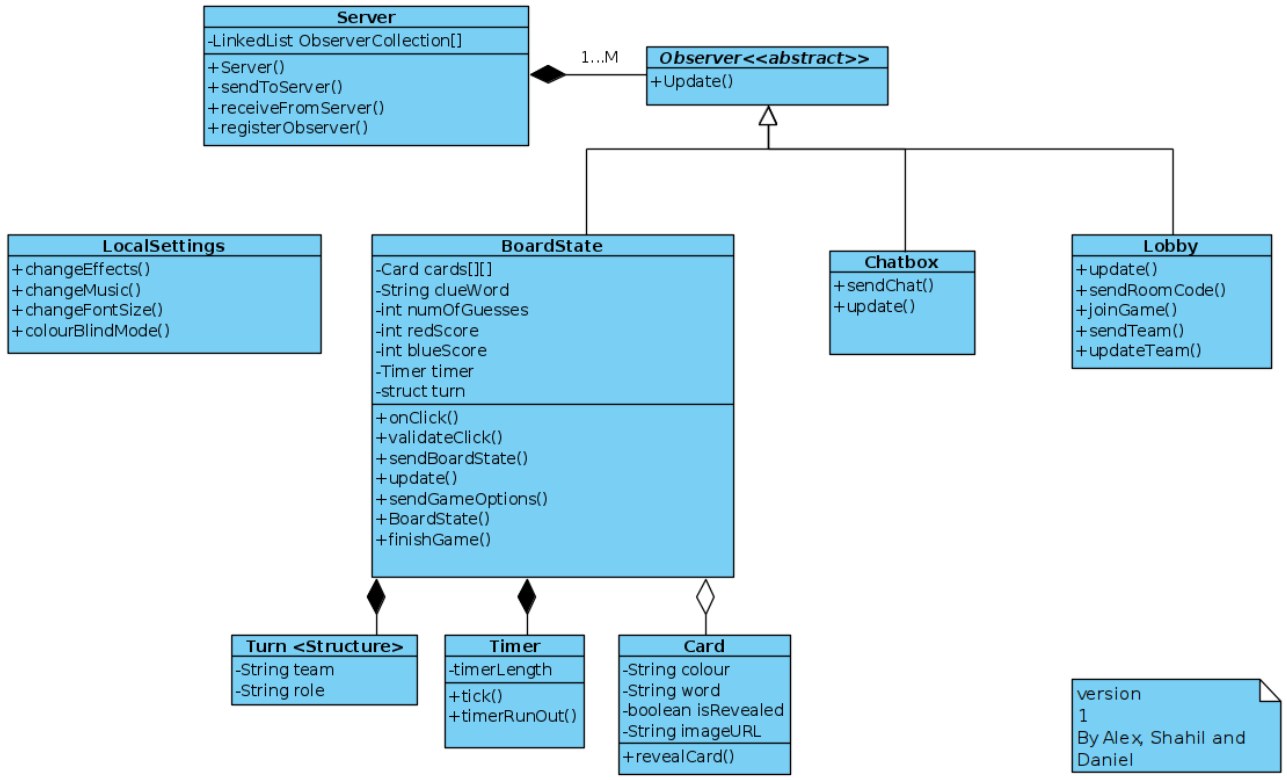
Chatbox Class

Class Name Chatbox
Attributes (none)
Methods sendChat() update()

Method Name sendChat
Method Description and purpose Send chat message to chat using Server class
Parameters (none)
Return Values (none)

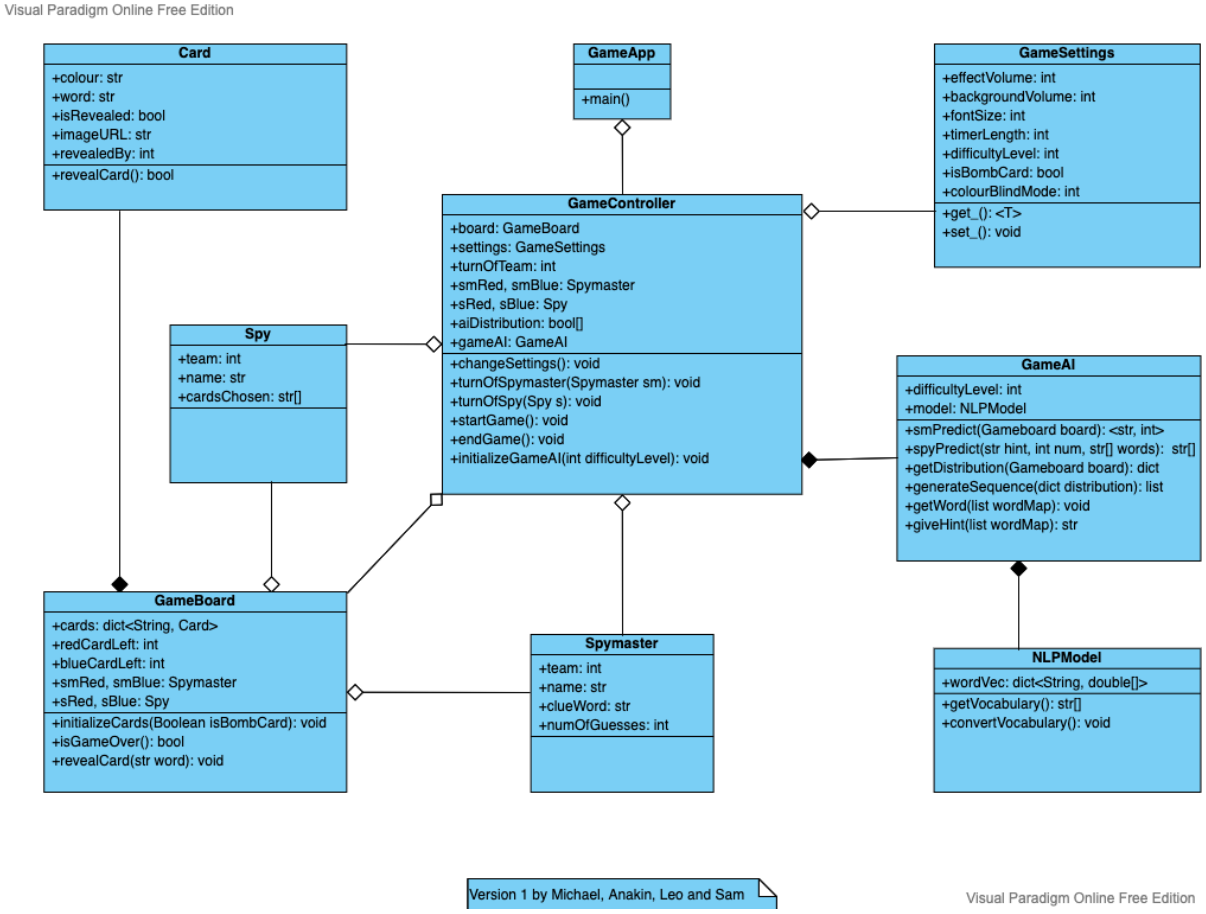
Method Name Update <<INHERITED>>
Method Description and purpose Prints incoming chat messages using Server class
Parameters (none)
Return Values (none)

Class Diagram



Game Mechanics & Artificial Intelligence

Class Diagram



Class Descriptions

GameApp

- Attributes
- Methods
    - void main()
        - Initialize <GameBoard> and <GameSettings>, pass them to <GameController>, then begin main loop in startGame()

GameBoard

- Attributes
    - (Card) cards[]
    - (int) redCardLeft, blueCardLeft
- Methods

- void initializeCards(Boolean isBombCard)
    - Initialize a fixed-length list of <Card>, assign attributes to each
- Boolean isGameOver(Card cards[i])
    - Call by <GameController>, check if one team's cards are all revealed or bomb card is revealed

Card

- Attributes
    - (String) colour
    - (String) word
    - (Boolean) isRevealed
    - (String) imageURL
    - (int) revealedBy
    - (Boolean) isBomb
- Methods

Boolean revealCard(Boolean isRevealed, Boolean isBomb)

If boolean isRevealed is false and isBomb is false, show the true colour of the card to all players and return false.

If boolean isBomb is true, show the true colour of all cards to all players

and return true.

GameSettings

- Attributes
    - (int) effectVolume
    - (int) backgroundVolume
    - (int) fontSize
    - (int) timerLength
    - (int) difficultyLevel
    - (Boolean) isBombCard
    - (int) colourblindMode
- Methods

void setSFXVolume(int effectVolume)

Sets game effect volume to the value specified by the player.

void setBGVolume(int backgroundVolume)

Sets volume of background music to the value specified by the player.

void setFontSize(int fontSize)

Sets font size of the game to the value specified by the player.

void timerLength(int timerLength)

Sets the duration of the timer during the game to the value specified by the player.

void difficultyLevel(int difficultyLevel)

Sets the difficulty level of the AI to the value specified by the player.

void bombPresent(Boolean isBombCard)

Sets whether there will be a bomb card present or not in the game.

void colourblindMode(int colourblindMode)

Sets the colour of the game cards to help players with colour blindness,

to the values set by the player.

GameController

- Attributes
    - (GameBoard) board
    - (GameSettings) settings
    - (int) turnOfTeam
    - (Spymaster) smRed, smBlue
    - (Spy) sRed, sBlue
    - **(Boolean) aiDistribution[]**
    - (GameAI) gameAI
- Methods
    - void changeSettings() - implements MouseListener and/or MouseMotionListener
        - Checks for changes to the sliders or buttons in the settings menus and updates settings accordingly e.g. if button clicked to enable the bomb card, settings.bombPresent(true).
    - void turnOfSpymaster(Spymaster spymaster)

        - Get user-input or AI prediction of clueWord and numOfGuesses in settings.timerLength, store them in the class instance
    - void turnOfSpy(Spy spy)
        - Display clueWord and numOfGuesses from teammate spymaster, get user-input list or AI prediction of cardsChosen[] in settings.timerLength, store them in the class instance
    - void startGame()
        - Initialize spymasters and spies and the game depending on settings, then enter main game loop, call turnOfSpymaster() and turnOfSpy(), passing arguments according to turnOfTeam, check board.isGameOver() and call endGame() accordingly
    - void endGame()
        - The end ☺
    - void initializeGameAI(int difficultyLevel)
        - If at least one AI in this game, initialize one instance of GameAI according to difficultyLevel

Spymaster

- Attributes
    - (int) team
    - (String) name
    - (String) clueWord
    - (int) numOfGuesses
- Methods

Spy

- Attributes
    - (int) team
    - (String) name
    - (String) cardsChosen[]
- Methods

GameAI

- Attributes

    - (int) difficultyLevel
    - (NLPModel) model
- Methods
    - <String, int> getSpymasterPrediction(GameBoard board)
    - String[] getSpyPrediction(String hint, String[] words)

NLPModel

- Attributes
    - (dict)<String, double[]> wordVec
- Methods
    - String[] getVocabulary()
        - Get vocabulary from local file
    - void convertVocabulary()
        - Convert word list got from getVocabulary() to word vectors and store it

AI Planned Details

- Two kinds of AI
    - Spymaster
    - Spy
- Team allocation (AI could take either role)
    - 2 Teams with 1 person and 1 AI in one team, 2 AIs (take different roles) in the other team **(1 player)**
    - 2 Teams with 2 people in one team and 2 AIs (take different roles) in the other team **(2 players)**
    - 2 Teams with 1 person and 1 AI in each team **(2 players)**
    - 2 Teams with 1 person and 1 AI in one team, 2 people in the other team **(3 players)**
    - 2 Teams with 2 people and 1 AI in each team **(4 and more players)**
- Methods to get word similarity (use cosine similarity on word vectors)
    - GloVe
    - Word2vec
    - Fasttext (not suitable)
- General pipeline (AI for spymaster)
    - Build vocabulary of the game
    - Get pre-trained word embeddings for the chosen method (training by ourselves is okay, but costs too much :-(
    - Retrieve the set of relevant guesses for the board by finding words in the vocabulary with higher similarity between board words (could be one guess per board word)
    - Loop through relevant guesses and assign a score to each. Firstly, calculate the cosine similarity between the guesses and all board words. Then, the score of each guess could be expressed as

$$score(u) = \sum_{i=0}^{n} cos(u, v_i) - \sum_{j=0}^{m} cos(u, v_j) - k \cdot cos(u, v_a)$$

where vi and vj are vectors for words in AI's team and the other team respectively, and va is for the "bomb" card.

- Take the guess with the maximum score as a hint to teammates
- The max number of guesses n for the spy teammate in one round is expressed as

$$n = f(score(u), c)$$

where c is the number of words left in your team, and f is to be determined.

- General pipeline (AI for spy)
  - Build vocabulary of the game
  - Get pre-trained word embeddings for the chosen method (training by ourselves is okay, but costs too much :-(
  - Calculate cosine similarity between the hint and all board words, chose n words with higher cosine value as guesses. (n is specified by the spymaster)
- Difficulty configuration
  - Vocabulary size
  - AI Accuracy
    - Random noise
    - Quality of word embeddings
    - Training steps (Reinforcement Learning)

Class Name
SpyAImove()
Attributes
aiLevel wordMap
Methods
getType() getDistribution() generateSequence()

Method Name
getType() --- remain discussion
Method Description and purpose
Ai extracts the types from the cards
Parameters
(list)blue (list)red (list)neutral (list)card
Return Values
Blue, red, neutral

Method Name
getDistribution()
Method Description and purpose

Get the distribution over the classes depending on the difficulty
Parameters (dict)distribution (list)board
Return Values distribution

Method Name generateSequence()
Method Description and purpose Generate the sequence for the AI
Parameters (list)sequence (dict)distribution (int)cardId
Return Values sequence

Class Name MasterAImove()
Attributes aiLevel wordMap
Methods getWord() giveHint()

Method Name getWord()
Method Description and purpose Read the wordMap to know the color(type) and the word at every position
Parameters (list)wordMap
Return Values void

Method Name giveHint()
Method Description and purpose Give the hint to human player or AI player
Parameters (list)wordMap
Return Values (String)hint