

Kenshin Drona
Shanni Lian
Le Hoang Anh Nguyen
Nicole Wu
Naga Harini Yadavalli

Report

Class Requirements

LetterTile:

- Variables:
 - Letter
 - Point value of each letter
- Functions:
 - Get letter
 - Get point value of each letter

LetterRack:

- Variables:
 - Array of letter tiles
 - Size of array
 - Number of tiles in the rack
- Functions:
 - Remove a specific letter tile from the rack
 - Fill rack
 - Print rack
 - Exchange tiles
 - Get the number of tiles in the rack
 - Access operator for the tiles in the rack
 - Check if a specific letter is in the rack

LetterBag:

- Variables:
 - Vector of letter tiles
- Functions:
 - Add tiles
 - Check if the bag is empty
 - Draw a random tile

Square:

- Variables:
 - Default empty square

GameBoard:

- Variables:
 - 2D vector of squares
- Functions:
 - Check if the word can be placed at the provided location
 - Prints the current board
 - Places a word on the board if placement was valid
 - Initialize the board
 - Get the tile at a specified position
 - Get all possible words formed from word placement
 - Check all words formed by placement are valid

Player:

- Variables:
 - Name of player
 - Player's points
 - Player's rack
- Functions:
 - Get points
 - Play a word
 - Calculate their score for a word
 - Get their rack
 - Set the player's name
 - Get the player's name
 - Add points to the player's score

Game:

- Variables:
 - A vector of players
 - Count of consecutive turns with no points earned
 - Number of players
 - A game board
 - A tile bag
 - The index of the current player
- Functions:
 - Play game
 - Get count of consecutive turns with no points earned
 - Determine and announce the winner
 - Calculate the number of points remaining on the player's rack
 - Determine the turn order
 - Advance to the next player's turn
 - Get a reference to the current player
 - Check if one of the game end conditions are met

- Player's rack and letter bag are both empty
- Check if the word is in the dictionary
- Print the scores

LetterBag
+ Bag : vector<LetterTile> + addTiles(letter : char, count : int, value : int) : void + is_empty() : bool + draw_tile() : LetterTile

LetterRack
- SIZE : int - rack : LetterTile[SIZE] - tile_count : int
+ remove_letter(letter : char) : LetterTile + fill_rack(bag : LetterBag) : void + exchange_tile(letter : char, bag : LetterBag) : void + print_rack() : void + get_tile_count() : int + operator[](index : int) : LetterTile + has_letter(letter : char) : bool

LetterTile
- letter : char - point_value : int
+ get_letter() : char + get_point_value() : int

Square
+ letter : char

Game
- noPointTurn_count : int - playerNum : int - board : GameBoard - bag: LetterBag - current_player_index : int
+ players : vector<Player> + play_game() : void + get_noPointTurn_count() : int + determine_winner() : void + rack_points(rack : LetterRack) : int + determine_turn_order() : void + next_player() : void + get_current_player() : Player + is_game_over() : bool + dictionaryCheck(word : string) : bool + print_scores() void

GameBoard
- board : vector<vector<Square>>
+ isValidPlacement(word : string, row : int, col : int, direction : char) : bool + printBoard() : void + placeWord(word : string, row : int, col : int, direction : char) : bool + initializeBoard() : vector<vector<Square>> + getTile(row : int, col : int) : char + getAllWordsFormed(word : string, row : int, col : int, horizontal : bool) : vector<string> + allAdjacentWordsValid(word : string, row : int, col : int, horizontal : bool, game : Game) : bool

Player
- name : string - points : int
+ rack : LetterRack + get_points() : int + play_word(board : GameBoard, word : string, row : int, col : int, horizontal : bool) : bool + calculate_score(word : string) : int + get_rack() : LetterRack + set_name(name : string) : void + get_name() : string + add_points(additional_points : int) : void