Samuel Maschmann, Nathaniel Narunatvanich, Samuel Nicklaus

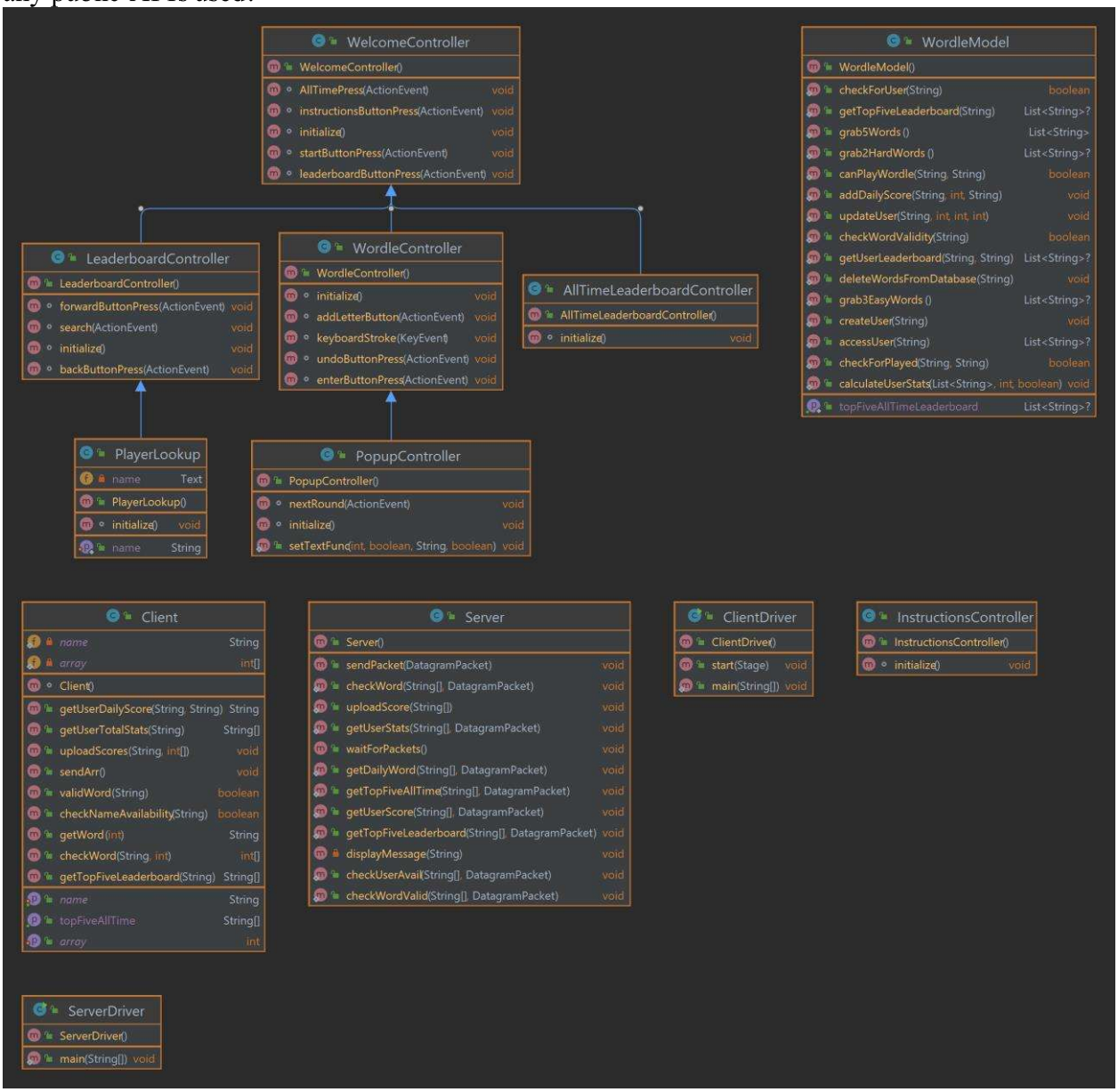
Professor Guadalupe Canahuate

ECE:3330

May 1, 2022

WordleTournament Developer Guide

The developer's guide details how the program works internally, how to run the program, and any public APIs used.



GUI:

This program contains seven separate GUI controllers and fxml files. Class WelcomeController models the first page of the GUI. Through this page, the user has the option to view instructions, view the daily leaderboard, view the all-time leaderboard, or start a game. This class contains the client object that will eventually be passed around to other controllers that contain information such as the player's username and the user's word guesses before being sent to the server. Class InstructionsController is used to display an image in a separate pane to the user that contains the instructions for the game wordle. Class LeaderboardController extends WelcomeController and is used to display the top five scores from any given day. The controller calls a method from the client which pulls information from the server and then the controller displays the results to the user. The user can also look up names in the bottom left corner to find out more detailed information about them (such as win streak, total score, etc). Class player lookup extends LeaderboardController and calls client methods to get user data based on the user input. Class AllTimeLeaderboardController extends WelcomeController and displays the top five total score leaders to the user. It does this by calling a method in the client class that pulls the appropriate data from the server. Class WordleController extends WelcomeController and is used to model a five-round game of wordle. The controller takes in user input from either the buttons at the bottom of the GUI or physical keyboard presses. When the user submits a word for a guess attempt the information is sent to the server for processing. Whatever the server returns tell us information about the guess. First and foremost, if the word the user submitted is not a valid word in the database the user is prompted to try again. If the word is valid the server returns information such as if the word matches is correct, and if not, which letters are in the right location or not in the word at all. This information is used to change the color of the background behind the letters and buttons that contain that letter. Signaling to the user information about each letter they guessed. After the user guesses the word or runs out of attempts Class PopupController is used as a transition window to the next round. This class takes in data about the previous round and presents it to the user (such as correct word, number of attempts, etc). The window then prompts the user to go on to the next round or if they are out of rounds, return them to the homesceen.

Networking:

This program relies on a packet-based networking system to communicate between the server and clients. Each client instance contains several methods that send a packet to the server. Each packet is in the form of a signifier integer and the data that needs to be sent to the server separated by commas. The signifier ranges from 1 to 9, and each number corresponds to a method to be called in the server. The server receives the packet, reads the signifier, and calls the necessary method to which the signifier corresponds. The signifiers call the following methods: 1 calls getUserAvail, 2 calls checkWord, 3 calls getDailyWord, 4 calls checkWordValid, 5 calls uploadScore, 6 calls getTopFiveLeaderboard, 7 calls getUserScore, 8 calls getUserStats, and 9

calls getTopFiveAllTime. Each of the server methods interacts with the SQL Database by either sending or retrieving data. If required, the server then puts the retrieved data into a packet and sends it to the client from where the packet came. Finally, the client uses the data to update the GUI.

SQL Database:

This program utilizes a mySQL database server to store and create user information as well as the words used in the WordleTournament game. This database *swd_db005* has four tables: *dailyScores*, *users*, *validWords*, and *wordBank*. The *wordBank* table has the columns word, a varchar variable that contains a 5-letter word and serves as a primary key for each entry, and difficulty, a decimal variable that determines how difficult the word is to guess in the Wordle game based on the frequency and usage of the word and its component letters. The *validWords* table consists of one column to store any potential word the user could input into the Wordle game as a guess. The *users* table is where users can register to play the WordleTournament; each entry contains a varchar userID (primary key) to represent the username of the player, their totalScore and totalRoundsPlayed since the user was created, and the user's current winStreak. The *dailyScores* table is where each user's score for each day of playing the WordleTournament is stored; each entry contains a varchar userID (foreign key), an int dailyScore, and the date the score was recorded.

The WordleModel class holds all of the methods related to communicating with the SQL database, and these methods are called within the Server class methods to send and receive information via networking. Once the Client and Server objects are running, the Server calls the grab5Words method to return a list of 5 words (3 easy and 2 difficult) from the wordBank SQL table to be used for the target words of the WordleTournament for the day. While the 5 words for the day have been randomly selected, another method calls a query to delete the 5 entries from the database to prevent reuse of the words on different days. When a user first starts the WordleTournament program, the first SQL-related method called is the canPlayWordle method, which determines whether the inputted username already exists within the SQL database as well as if the user (if it already exists) has played the WordleTournament for the current day. If the userID does not exist, a method is called to insert a new row into the *users* SQL table; if the userID does exist, this method guarantees that each userID can only play the WordleTournament once per day. After passing through the canPlayWordle checker method, the user is able to play the Wordle game. Whenever a user attempts to submit a guessing word, the checkWordValidity method is called using the guessed word; this method sends a query to the SQL table *validWords* to determine if the guessed word is valid/real. If the query does not return a match for the guessed word, the GUI will prompt the user to submit a new guess as their previous one was invalid. Once the user has completed the 5 rounds of WordleTournament, the addDailyScore and calculateUserStats methods are called to add a new entry in the *dailyScores* table and update the user's lifetime statistics, respectively.

How to Run (Client + Server, etc.):

Note to run with IntelliJ: Make sure JavaFx is enabled on your system and that SQL is added to dependencies of the program through Maven.

First, run ServerDriver. Once the window has appeared and it says, "Server is running...", run ClientDriver. Then a window will appear, and it will be interactable. If you try running the client without running the server, the client window will appear, but it will not be interactable. You can open multiple clients on your machine to play multiple games with different users.

Additional Note: make sure to be on University Wi-Fi as it uses open ports and an SQL database connection that is only accessible through the University Wi-Fi.

Public Database:

We decided to use a word list that was created by three individuals at Jadavpur University. This modified dataset of five-letter words also contains a "difficulty number" that was calculated using machine learning (read the paper here: https://github.com/garain/Word-Difficulty-Prediction/blob/master/WORD_DIFFICULTY.pdf). The number is calculated by checking the complexity of the word, how often that word is used in common media, the number of vowels, ETC. The number returned can range from anywhere from –1 to around 2 with –1 representing a simple five-letter word and words closer to 2 representing more complex five-letter words. This number is used as a multiplier in our score calculation to determine the daily total score. (Link to the actual database here: https://www.kaggle.com/datasets/kkhandekar/word-difficulty?resource=download).