Rocco Piccirillo **The Perfection of Hangman: Project2** Project 2 Writeup

Abstract:

The project that I decided to choose for this class hangman. Hangman is something that has been done countless times and if you think about it, how could someone mess up the perfect formula? Well you can't. If you do not know, hangman is a game that tests the brain to complete a hidden word initially given to you form of _'s in the length of the word. Say your word is dog it would be presented as this "_ _ _". That is until the guesses the rest of the word and the _'s start shifting into actual letters slowly completing the game. To play all you need is a friend, a canvas, and a writing utensil. Though for my project these are not the tools being used to play. My game is being brought to life on the computer screen. What it is, is you, a screen, and your keyboard. All you must do is type in letters accommodating to what your mind thinks is write. Once the stick man's body fully appears you have lost, and the game will lock you out from typing any longer. The game has its very own point system. If you get a letter right, you get 100 points for the first correct letter then a point modifier I implemented will continue to increase your points as your streak grows. If you get a letter right, you do not lose any points, but the stick man slowly is getting "hung". No matter what letter you guess, the letter you typed gets crossed off and changes to red to indicate that it has been used already.

Compared to my last update to the game this one is quite the large one. The game instead of having only 1,500 words the game now reads and randomizes from a text file of 58,110 words. The game starts up in about two seconds and when you click for a new game it restarts instantly. Before all the game could do print a bunch of _'s to indicate the length of the word, let you guess the word by typing a letter into the command prompt, and pressing y or no for a new game. Now the game waits until your guesses are depleted or if you won the game to give you option to click the "New Game" button to start again. Clicking this refreshes the middle blank text boxes resizing it with a certain number of boxes depending on how many letters are in the hidden word. The only downside to the game being updated to GUI, is that the amount of lives you have has been lowered from ten to five. This is because there are only five

limbs on the stick man. Originally the game just printed you have X amount of lives left. Now the game just shows an uncompleted stick man.

Introduction:

The inspiration behind the project at hand is that it will help my mind learn the thought process behind code. I am completely new to computer science. In high school we did not have any computer science courses at all. I could have taken online courses, but I never had a thought in my mind of what I wanted to do in life until senior year when I began applying to colleges and some places asked for a major. I only really chose it because my brother took it and it seemed like the coolest concept at the time to me. As of right now computer science still is the coolest concept to me. Making your own program from scratch, with all these languages the options are limitless with what you can make. But like I said prior there were not any real options for me to start with in high school. The closest we had was a simple website making course, but it was easily forgettable. My first-time coding was last semester in JavaScript. The experience wasn't great though. I did not learn as much as I would have liked and felt like my knowledge is lacking in JavaScript. This semester with Java I wanted to redeem myself and feel good about completing assignments. Last semester I looked up everyone and just took code. That lead to me not improving at all and not feeling satisfied with myself. This semester I am doing the work, learning as I go, and feeling excited whenever something I've been tinkering with for a while comes together. Its an amazing feeling, getting stuck and then overcoming a task you once thought was impossible.

For my final project I wanted to challenge myself the best way I could. I was researching day and night trying to figure out what the perfect project for me would be. I wanted something challenging that I could pick up on work when needed and not go completely insane over. I wanted this project to be my own, something I could be proud of and show off while saying "yeah I did this." My original thoughts led to me do steganography because a friend recommended it saying it would be a clever yet easy idea. After choosing it I eventually realized that it is much more difficult than I once thought. I was doing research and when the time came for me to start working on it myself, I couldn't. I had no idea what I could do or where I

even had to start. After countless nights of me being confused I eventually realized that the only way I would be able to complete this project is if I take all my code from a YouTube tutorial, which is not at all want I wanted to do. Thank to this I changed my mind and decided the best option for me would be to change all together. This is where my brilliant idea of a solo playing hangman game arrived from.

Detailed System Description:

This game has been a work in progress for myself for months in the making, and I can honestly say I am proud with how it came out. The project starts off with key component that holds my whole project together. I am talking about implementing my words list from a text file an ArrayList<String>. How I did this was by using FileReader and opened up WordList which includes over 58,000 words. All FileReader really does for me is read and store the text file. The BufferedReader is more efficient becauses it buffers for the file. I use it pass the file from FileReader into BufferedReader. I then transfer all the words into a string that reads each line. Then run into a while loop that empties this string into an array list until the specified string is detailed. This is all done in its own class called WordBank. The class word bank is also used to randomize words as well. It includes my method known as wordRandomizer which checks initially checks if the word list is empty, if it is it returns the word “DEFAULT” but if it is not it randomizes the list. The way I randomized everything was by creating an int named random that equaled (int)(Math.random() * 58110);. What this does is search through every single word in the list and picks one. I then return wordlist.get(random). This sends back one random word from my text file. I then implement these into my main hangman class. The class starts off and imports just about everything from javafx (not actually but a lot of components) and IO Exception and Hashmap. The first real variable that gets initialized is my SimpleStringProperty for my word which stores the word taken from the class WordBank I created, and the others are lettersToGuess which is how many letters you have left to guess. These are both SimpleIntegerProperties that I declared. I made a SimpleBooleanProperty named playable to determine is the game can be played or not. I then make wordbank variable to use my wordbank from my separate class. Then a method named Parent that sets up the game or the orientation of the board. A quick summary of what it does is, make the game playable, aligns

the alphabet, gives text and properties to my new game button, overall it takes all the layers of my game and orders them. On top is the row of letters, then the row of alphabets, and lastly the hangman images. Then is the startGame void method, basically runs whenever a game begins and just refreshes the page as if nothing happened, except the points don't change, and word becomes something new. And then endgame for when the game ends, it shows all the letters. Next, I created the images for the hangman and set coordinates for each body part, so everything prints in the right location. Your number of guesses equals the number of limbs not yet printed. Once a letter is guessed incorrectly my method lifeLost runs, which means a limb gets printed, and life is lost. There is another void method called restart which sets the hangman image to invisible. I then have a private static class which customizes my letters and the rectangles the hidden word is placed in. Below this is a void named show which is for when a letter is guessed correctly. Say the word is dog and you guess d the box where d would be rotating and prints d after the animation is finished. This also goes hand and hand with my Boolean isEqual which checks if the letter matches a letter in the hidden word. Then here is where the game beings in my void start(Stage gameWindow) which runs the game and calls all my methods that were created prior. Key clicks are checked, if you game is playable you can play it, if a letter was already entered or not in the alphabet, nothing happens, and if the letter was used mark it and set it to be red. Also, basically just checks if the letter is in the hidden word if it is get the letter to show. Just like I explained before, if you guess incorrectly you lose a life, and part of the hangman prints. Finally, I set everything to my gui, you can't resize, the width and height are set, there is a title, and it follows a certain scene. Thanks to this everything runs as it should and the game is playable for the user.

Hangman.java
-DEFAULT_FONT: Font -POINTS_PER_LETTER: int -BONUS_MODIFIER:float -word: SimpleStringProperty -lettersToGuess: SimpleIntegerProperty -playable: SimpleBooleanProperty

-letters:ObersvableList<Node> -alphabet: HashMap<Character, Text> -hangman: HangmanImage -wordBank: WordBank //- guesses: SimpleIntegerProperty// ask if I do this b/c it is in another class
+Hangman() +createContent(): Parent +startGame(): void +endgame(): void -HangmanImage:class +HangmanImage() +restart: void +lifeLost: void -Letter:class + Letter(letter: char) +show(): void +start(gameWindow: Stage): void

WordBank.java
-wordList: ArrayList<String>
+WordBank() +wordRandomizer(): String

Requirements and Literature Survey

The requirements section for my writeup will be rather short, thanks to this I am once again blending my requirements and literature survey. During my work there are no legitimate problems that have gathered that I was trying to solve. My game isn't about solving problems.

It is about the luxurious game known as hangman. Something that just about everyone in the world has most likely played at least once. The only real requirements that I can conceive to solve the mystery of the hidden word is, a word that the hidden one is deriving from, the user having the ability to input, and the ability for the hidden word to convert to the actual word once the user guesses a correct letter. Like I said, this game is a test of mind, making you think your way through it. The games purpose is to help the users guessing capabilities. For all you know, the game could benefit your mental state on something like a multiple-choice test, where sometimes guessing is your best friend.

Hangman has been around for plenty of years by now. The game has been done repeatedly, each time with a new little spin on it. My spin is minor, but it is there. My spin on the genre is that I have over 58,000 words. So, the user must brainstorm up literally any word they know to figure out the hidden one. Whilst playing you only have a limited number of guesses, which makes the game much more challenging. Options range from (but are not limited to) a three-letter word that can prove extremely difficult blending random letters together, to an eight-letter word with repeated words which could take very little time for the user to win. The only real problem in hangman is that the user must find out what the hidden word could be. Like I said earlier, the word could be anything and I mean anything. Other work that has tried to solve this "problem" in the past are other hangman games out there. To find them all you must do is google "Hangman game" and thousands of searches will pop up.

Since there are not any real problems that my code is fixing, I thought why not discuss the problems that I have faced throughout my time working. Making the base game was a cinch, the real difficulty came from the graphical user interface that I wanted to design. This is where most of the time was spent on my project. I had so many different designs I wanted to add to it but with how little I know of JavaFX or JFrame I could not achieve my personal goal. Everything eventually came together after outside help from my brother and the great state of YouTube. I found a simple tutorial on adding GUI to a hangman game. I did borrow code from this, but I have mixed and matched my original code, blending the two together. This helped make my game work to my liking. In my opinion, graphical user interfaces are very interesting

and appealing to the eye, but for all the struggling and late nights were worth it. I hope to work with GUI again in my future college career, but I hope for it to go more smoothly than this time.

User Manual:

The game has reached its full potential, it is completed and fully usable. The game presents you with a box. The box contains about five components in total. First off, the main eye catcher are the rectangles that are printed depending on how many letters are in the hidden word. Below this is the whole alphabet, so basically a letter bank that you can check with to see if the letter you inputted has been used if you have forgotten. Then the last two is a "New Game" button to reset the game for you, and the hangman images that print when you guess your letters incorrectly. The game is straight forward to play all you need is a keyboard and your mind. You type in any letter on your keyboard to continue guessing the word. You have five lives in total and you will notice they run out when the whole stick man is printed. Another easy way of knowing is when you lose, you can no longer type anymore and the whole word gets printed. If you win, you will no longer be able to type in and the word will be fully printed out. The stickman will also not fully be shown as well. A feature of both winning and losing is being able to click on the button to start a new game. A quick heads up is that this can only be used once you win or lose.

Conclusion:

The goals accomplished by the system include but are not limited to:

Being able to read a text file in java and storing the words in an ArrayList<String>

Randomizing over 58,000 words and only selecting one word at a time

Being able to use multiple classes. By this I mean having a second class which gets called in the main class.

Getting down the basics and more of JavaFX

Implementing GUI in general

Making my GUI look flashy as in colors, fonts, pictures, animations, the rectangular shape

Finishing my first project on my own

References/Bibliography:

I have referred to multiple YouTube videos but there was only one that I took straight code from for this because I had no idea. I will cite that below. Everything else I did I just went to stackflow for a quick assist I will try to link some of my assistances that most heavily improved my code. The last video is the GUI tutorial that helped me actually make my GUI workable. I tried for countless hours and I just could not get it. I used this code as reference and did borrow some, but mainly tried to change it as much as possible and implemented my original ideas into it as well. I hope this is ok, all I wanted was to make my GUI to work but it proved as an extremely difficult challenge to complete, and just about everyone I asked had no clue as well.

BrandonioProductions. “Learning Java: Part 15: Reading Text from Files.” *YouTube*, YouTube, 27 Feb. 2012, www.youtube.com/watch?v=VwV1VKvtVtI.

“Retrieving a Random Item from ArrayList.” *Java - Retrieving a Random Item from ArrayList - Stack Overflow*, stackoverflow.com/questions/5034370/retrieving-a-random-item-from-arraylist.

(2015, January 19). Retrieved May 02, 2018, from https://www.youtube.com/watch?v=iphlcKmnHjI&t=1508s