Hangman Game

Deividas Ovsianikovas, 2nd year BSc Computer Science

19358376

Maynooth University

CS210

Things I came across:

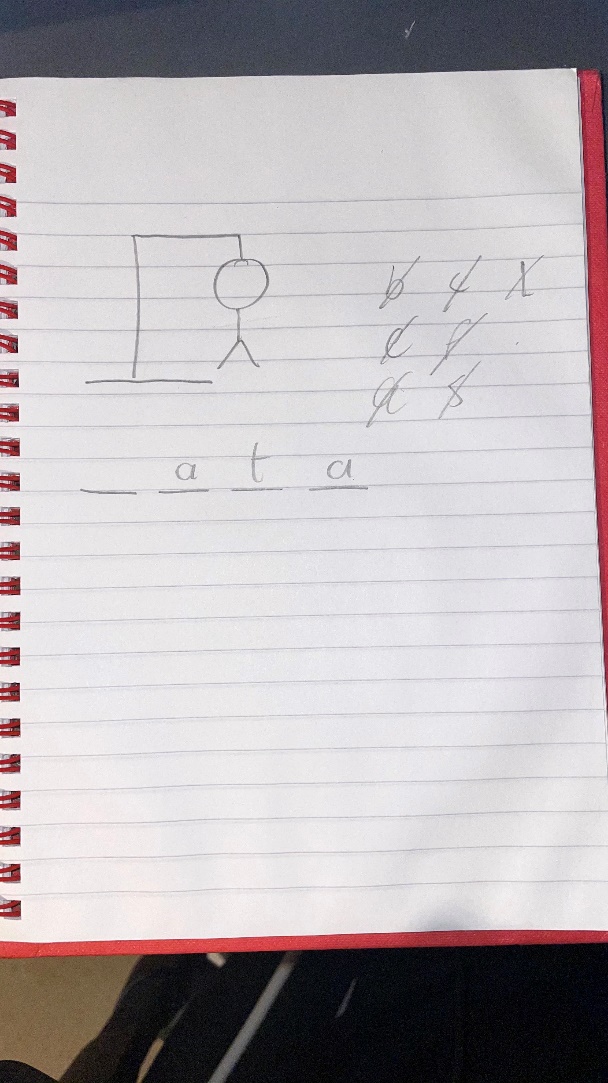
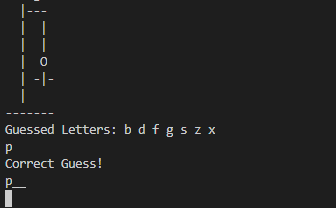
Wanted to display hangman icon but found difficult to find most efficient way of doing it. Decided on making 1 large string containing all of the lives.

Lots of while loops selected Y / N, could this be put into 1 method?

**Overview of game**

Here I wanted to challenge myself and develop Hangman into a version playable from a Java file making it as interactive as possible within a Java Command Prompt.

This is the classic game of Hangman where the player must figure out a word that has been chosen randomly by the computer based on the selected difficulty by guessing it letter by letter. The player has 8 lives and for every incorrect guess they make they lose a life. The player can win by guessing all the correct letters of the word but lose if they run out of lives. The player’s current health is shown by an ascii representation of the Hangman Stick Figure.



**How to play**

1. Run the “Hangman.Java” file in your Integrated Development Environment (IDE).
2. Select whether you would like to see the rules or not by inputting “Y” or “N”.
3. Choose difficulty by inputting the correlating word (Easy, Medium, Hard).
4. AI then chooses a word to guess based on selected difficulty and you will be presented with the length of the word and the state of your current guessed word which will be represented with the “\_” character replacing all characters of the original word.
5. You can now guess the letters contained in the word. If the character is incorrect, then you lose a life and are presented with a Hangman step representing your health. If you guess correctly your current guessed word gets updated and you can proceed.
6. If all characters of the word are guessed correctly you win. If not, you will be presented with the full Hangman and lose.
7. Once complete, you are given the choice to either retry or quit.

**Code structure**

1. **Imports**

Importing Java.utils to use the “Random” and “Scanner” objects, and importing java.io to use the “File”,”StringBuffer”, and “BufferedReader” classes.

1. **Class Hangman**
2. **Static variables**

Private and public variables used throughout the program

1. **void Main()**

The initial method called by the Java program which calls the Start() method.

1. **void Start()**

This method calls the ResetVars() method, introduces the player, allows them to choose a difficulty where GameSetup() is called depending on it, allows the player to guess until the game is over by calling the Guess() method, displays the end game result to the player and also calls the RetrySetup() method.

1. **void GameSetup(int difficulty)**

This sorts the words in the words.txt file based on their length, smallest to largest and sets the position of where the present easy, medium, and hard length words start. It then chooses a word from the sorted array based on the selected difficulty and displays to the user the length of the word and the state of their current guessed word (which will now just be many “\_” characters)

1. **void Guess(String guessedChar)**

This handles the guessing aspect of the game and returns a congratulatory message if the correct character is guessed with the new state of their guessed word, the state of their hangman if an incorrect guess is made or an error message to the player if an invald input is given. Handles if the game is over the players health.

1. **void ShowLives(int health)**

This prints out the current Hangman setup the player is at based on their health.

1. **void RetrySetup()**

This asks the player if they would like to restart the game or quit. If they’d like to restart, the Start() method is called and the process begins again. If they’d like to quit, the method returns and the program ends.

1. **void ResetVars()**

This method resets selected variables to their default state.

1. **boolean YesNoAnswer()**

I wrote this method so I would not have to write the same if statements checking the player player inputted yes or no. It returns true if the player had inputted “yes”, false if “no”, and an error prompt if the input is invalid.

1. **void MergeAlgo(int p, int q, int r)**

This is the Merge Sort algorithm that sorts the words array from smallest to largest.

1. **void Sort(int p, in r)**

This accommodates the MergeAlgo() sorting algorithm splitting the array into 2 and sorting each half until both are sorted.

1. **Class Dictionary**
2. **Public variable**

Declares the input[] array which will hold all of the words in the “words.txt” file.

1. **Dictionary()**

The constructor method where the load() method is called with a link to the “words.txt” directory.

1. **Int getSize()**

This method returns the amount of words contained in the input[] array.

1. **String getWord(int n)**

This returns the word located at position n in the input[] array.

1. **String[] load(String file)**

This method handles extracting all of the data in the “words.txt” file and returning it to the input[] array. It handles any error exceptions that may occur while iterating through the data and also makes sure to only add words that are of only alphabetical characters. (i.e. no numbers, hyphens, dots, etc.)

**Data structures and algorithms used**

* The Merge Sort algorithms is used to sort all the words by length smallest to largest.
* 2D Array for containing the Hangman stick figure.
* 2 1D String arrays to hold all available words before and after sorting.
* Nested Loop to print out the Hangman stick figure.
* For loops, While Loops for various iterative tasks.
* Random Method

**Learning outcomes**

The main learning outcome of this project was to apply the knowledge that I learned during CS210 Semester 1. It allowed me to gain a better grasp of how to use Sorting & Data Structures in real world applications, apply the knowledge I gained from Software Testing to test and debug the program to ensure no presence of bugs, how to work with multiple classes in a Java project, how to import objects and use them in my Java file, Regex, applying Strings and Conditionals.

* Design, develop, test and debug object-oriented programs in Java
* apply data structuring techniques to the design of computer programs

**Appendix**

/\*

--------Make sure to change path of Words.txt to your directory--------------

Hangman game with 1 player playing against computer.

@author Deividas Ovsianikovas

\*/

import java.io.\*;

import java.util.\*;

public class Hangman {

      //1 long string containing all of the lives a player can have split into 7x7 arrays

      public static String[][] bigLivesString =

      {

          {" "," "," "," "," "," "," "},

          {" "," "," "," "," "," "," "},

          {" "," "," "," "," "," "," "},

          {" "," "," "," "," "," "," "},

          {" "," "," "," "," "," "," "},

          {" "," "," "," "," "," "," "},

          {"-","-","-","-","-","-","-"},///Break 1

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {"-","-","-","-","-","-","-"},///Break

          {" "," ","|","-","-","-"," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {"-","-","-","-","-","-","-"},///Break

          {" "," ","|","-","-","-"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {"-","-","-","-","-","-","-"},//Break

          {" "," ","|","-","-","-"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","O"," "},

          {" "," ","|"," "," "," "," "},

          {" "," ","|"," "," "," "," "},

          {"-","-","-","-","-","-","-"}, //Break

          {" "," ","|","-","-","-"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","O"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," "," "," "},

          {"-","-","-","-","-","-","-"},  ///Break

          {" "," ","|","-","-","-"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","O"," "},

          {" "," ","|"," ","-","|","-"},

          {" "," ","|"," "," "," "," "},

          {"-","-","-","-","-","-","-"},  ///Break

          {" "," ","|","-","-","-"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","|"," "},

          {" "," ","|"," "," ","O"," "},

          {" "," ","|"," ","-","|","-"},

          {" "," ","|"," ","/"," ","\\"},

          {"-","-","-","-","-","-","-"}   ///Final

      };

    private static String[] allWords;    ///String array containing all of the words from the words.txt document, later sorted

    private static String chosenWord = "";   ///The Computer's random chosen word

    private static String guessedWord = "";     //The player's current guessed word

    private static boolean gameOver = false;    ///States whether the game is over or not

    private static boolean hasRestarted = false;    ///States whether the player has chosen to restart the game

    private static int health = 7;    ///Player's current health

    private static int easyWordsStart = 0, mediumWordsStart = 0, hardWordsStart = 0; ///Variables setting where each difficulties word lengths start

    private static String allGuessedLetters = "";   ///Lists all characters that have been guessed already

    public static void main(String args[]) {

        Start(); ///Start the game session

    }

    /// Method to Start the game

    public static void Start() {

        /// Reset variables to their default state

        ResetVars();

        Scanner sc = new Scanner(System.in);

        boolean difficultySelected = false;

        System.out.println("Welcome to Hangman! Would you like to hear the rules first? Y N");

        if (!hasRestarted) {

            boolean answer = YesNoAnswer();

            if (answer) {

                System.out

                        .println("-----------------------------------------------------------------------------------");

                System.out.println(

                        "This is the classic game of Hangman where you have to figure out a word that has been\nchosen randomly by the computer by guessing it letter by letter.\nYou have 8 lives, for every incorrect guess you make you lose a life.\nYou can win by guessing all of the correct letters of the word, and lose if you run out of lives.\nYour current health is shown by an ascii representation of Hangman.");

                System.out.println();

                System.out.println("Written by Deividas Ovsianikovas for CS210");

                System.out

                        .println("-----------------------------------------------------------------------------------");

            } else if (!answer) // If player does not want to hear rules, continue

            {

                System.out.println();

            } else {

                System.out.println("Please only use inputs as required above");

            }

        }

        System.out.println("Select your difficulty: Easy, Medium, Hard");

        /// Difficulty select -> length of word depends on this

        while (!difficultySelected) {

            String difficultyInput = sc.nextLine();

            difficultyInput = difficultyInput.toLowerCase();

            if (difficultyInput.equals("easy")) {

                difficultySelected = true;

                GameSetup(0);

            } else if (difficultyInput.equals("medium")) {

                difficultySelected = true;

                GameSetup(1);

            } else if (difficultyInput.equals("hard")) {

                difficultySelected = true;

                GameSetup(2);

            } else {

                System.out.println("Please only use inputs as required above");

            }

        }

        // Let the player keep guessing until the game is over

        while (!gameOver) {

            Guess(sc.nextLine());

        }

        /// End game state, if player guessed word equals computer chosen word then player wins, loses otherwise

        if (guessedWord.equals(chosenWord)) {

            System.out.println("...:::: Congrats, you win! ::::...");

        } else {

            System.out.println("...:::: Game Over :( ::::...");

        }

        /// Ask if player wants to play again/quit

        RetrySetup();

    }

    /// Method to initally set up the game based on inputted parameters

    public static void GameSetup(int difficulty) {

        /// Only run this section if the game has been restarted, no need to sort & loop if the game has been restarted

        if (!hasRestarted) {

            Dictionary dictionary = new Dictionary();

            allWords = dictionary.input;

            Sort(0, allWords.length - 1);/// Sort words smallest to largest

            /// Loop to set the positions of where each difficulties word length starts in array

            for (int i = 0; i < allWords.length; i++) {

                if (allWords[i].length() == 4 && Hangman.easyWordsStart == 0){ /// If the words are at a minimum length of 3 and var has not been assigned yet

                    easyWordsStart = i;

                } else if (allWords[i].length() == 6 && Hangman.mediumWordsStart == 0) {

                    mediumWordsStart = i;

                } else if (allWords[i].length() == 9 && Hangman.hardWordsStart == 0) {

                    hardWordsStart = i;

                }

            }

        }

        /// Used to generate random word based on given range

        Random r = new Random();

        /// Pick word based on selected difficulty above

        switch (difficulty) {

            case 0:/// Easy

                   // (Max Value) + Min Value

                chosenWord = allWords[r.nextInt(mediumWordsStart - 1) + easyWordsStart]; /// Chose random word in given range

                break;

            case 1:/// Medium

                chosenWord = allWords[r.nextInt(hardWordsStart - 1) + mediumWordsStart];

                break;

            case 2:/// Hard

                chosenWord = allWords[r.nextInt(allWords.length - 1) + hardWordsStart];

                break;

        }

        chosenWord = chosenWord.toLowerCase();

        // System.out.println("The word is " + chosenWord); /// For debugging

        /// Change guessedword to only contain "\_" characters

        guessedWord = chosenWord.replaceAll("[A-Za-z]", "\_");

        System.out.println(guessedWord);

        System.out.println("Word Length: " + (guessedWord.length() - 1)); /// Show the player how many characters are in the word

    }

    /// Method to get guessed characters from player

    public static void Guess(String guessedChar) {

        guessedChar = guessedChar.toLowerCase();

        /// -----Error states-----

        if (guessedChar.length() > 1)

        {

            System.out.println("Only guess 1 character at a time!");

            return;

        }

        if (!guessedChar.matches("[A-Za-z]+")) {

            System.out.println("Please only use alphabetical characters to guess");

            return;

        }

        if (guessedWord.contains(guessedChar) || allGuessedLetters.contains(guessedChar)) {

            System.out.println("You have already guessed this letter! Try something else");

            return;

        }

        /// -----End of Error states-----

        /// If guessed character equals a character in the word

        if (chosenWord.contains(guessedChar)) {

            System.out.println("Correct Guess!");

            /// Add guessed character to correct positions in the guessedWord variable

            for (int i = 0; i < chosenWord.length(); i++) {

                if (guessedChar.charAt(0) == chosenWord.charAt(i)) {

                    guessedWord = guessedWord.substring(0, i) + chosenWord.charAt(i) + guessedWord.substring(i + 1);

                }

            }

            System.out.println(guessedWord);

        } else/// Else if an incorrect character was guessed

        {

            allGuessedLetters += guessedChar + " ";

            health -= 1;

            System.out.println("Wrong! Lives Left: " + health);

            ShowLives(8 - health); /// 8 - health as 8 is the starting health

            System.out.println("Guessed Letters: " + allGuessedLetters);

        }

        ///End the game if the player guessed word equals the computer chosen word or if health <= 0

        if (guessedWord.equals(chosenWord) || health <= 0)

            gameOver = true;

        return;

    }

    /// Method to show the different life step the player is currently on and print out the corresponding Hangman state

    public static void ShowLives(int health) {

        String[][] livesString = bigLivesString;

        /// Formulas to get current step based on health

        int start = (6 \* health) - 6 + health;

        int end = (6 \* health) + health;

        //Nested loop to print out the required step from livesString 2D array

        for (int i = start - 1; i < end; i++) {

            for (int j = 0; j < livesString[i].length; j++) {

                System.out.print(livesString[i][j]);

            }

            System.out.println();

        }

    }

    // Method to handle if user wants to retry the game or not

    public static void RetrySetup() {

        Scanner sc = new Scanner(System.in);

        System.out.println("Play again? Y N");

        boolean answer = YesNoAnswer();

        if (answer) {

            hasRestarted = true;

            Start();/// Start game again

        } else if (!answer) {

            System.out.println("Goodbye");

            sc.close();

            return; /// Terminate program if user no longer wants to play

        } else {

            System.out.println("Please only use inputs as required above");

        }

    }

    /// Method resetting variables to their original state

    public static void ResetVars() {

        chosenWord = "";

        guessedWord = "";

        gameOver = false;

        health = 8;

        allGuessedLetters = "";

    }

    // Method so we would not have to keep writing if("yes") clauses, takes up less memory & time

    public static boolean YesNoAnswer() {

        Scanner sc = new Scanner(System.in);

        boolean answerGiven = false;

        while (!answerGiven) {

            String input = sc.nextLine();

            input = input.toLowerCase();

            if (input.equals("y") || input.equals("yes")) {

                answerGiven = true;

                return true;

            } else if (input.equals("n") || input.equals("no")) {

                answerGiven = true;

                return false;

            } else {

                System.out.println("Please only use inputs as required above");

            }

        }

        return false;

    }

    /// Sort all Words in array by length, shortest -> longest

    /// Merge Sort, Big o = nlogn

    public static void MergeAlgo(int p, int q, int r) {

        int n1 = q - p + 1; /// Length of subarray 1 (mid - start +1)

        int n2 = r - q; /// Length of subarray 2 (end - mid)

        /// Left and Right sides of array

        /// Temp arrays

        String[] L = new String[n1];

        String[] R = new String[n2];

        for (int i = 0; i < n1; i++) /// Copy subarray A[p..q] to L[0..n1]

        {

            L[i] = allWords[p + i];

        }

        for (int j = 0; j < n2; j++) {

            R[j] = allWords[q + j + 1];

        }

        int i = 0;

        int j = 0;

        int k = p;

        /// Merge arrays

        while (i < n1 && j < n2) {

            if (L[i].length() <= R[j].length()) {

                allWords[k] = L[i];

                i++;

            } else {

                allWords[k] = R[j];

                j++;

            }

            k++;

        }

        // Copy remaining elements if any

        while (i < n1) {

            allWords[k] = L[i];

            i++;

            k++;

        }

        while (j < n2) {

            allWords[k] = R[j];

            j++;

            k++;

        }

    }

    public static void Sort(int p, int r) {

        if (p < r) {

            int q = (p + r) / 2;

            // Sort 1st & second half

            Sort(p, q);

            Sort(q + 1, r);

            // Merge the sorted halves

            MergeAlgo(p, q, r);

        }

    }

}

class Dictionary

{

    public String input[];

    public Dictionary()

    {

        input = load("C:\\Users\\Deivid\\Documents\\AnotherHangmanProjectCS210\\words.txt");  ///<--------------Insert path to words.txt doc-----------

    }

    public int getSize(){

        return input.length;

    }

    public String getWord(int n){

        return input[n];

    }

    private String[] load(String file)

    {

        File aFile = new File(file);

        StringBuffer contents = new StringBuffer();

        BufferedReader input = null;

        try {

            input = new BufferedReader( new FileReader(aFile) );

            String line = null;

            input.skip(120); ///First Few lines seem like nonsense, do not use

            while ((line = input.readLine()) != null)

            {

                if(line.matches("[A-Za-z]+")) ////Only include english language characters, no symbols

                {

                    contents.append(line);

                    contents.append(System.getProperty("line.separator"));

                }

            }

        }catch (FileNotFoundException ex){

            System.out.println("Can't find the file - are you sure the file is in this location: "+file);

            ex.printStackTrace();

        }catch (IOException ex){

            System.out.println("Input output exception while processing file");

            ex.printStackTrace();

        }finally{

            try {

                if (input!= null) {

                    input.close();

                }

            }catch (IOException ex){

                System.out.println("Input output exception while processing file");

                ex.printStackTrace();

            }

        }

        String[] array = contents.toString().split("\n");

        for(String s: array){

            s.trim();

        }

        return array;

    }

}

**Github: https://github.com/DE0S**