Final Project Source Code
Kobe Tran

Code:

```
package Project;

// COEN 160 Final Project Spelling Bee
// Kobe Tran
// Due: Week 10 (first week of December 2021)
// Description: Players create words of different lengths from a given word
// ** Each letter in the given word can be reused any number of times to
create a new word **
// ** The user inputted word must have a length greater than 3 characters**
// ** There will be on letter chosen that will be essential for the user's
inputted words **
// Check if word inputted is correct and exists in the dictionary
// If the word exists, calculate the score based on its length

import COEN160_Lab.Lab4_and_5.Spa;

import java.awt.*;
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class SpellingBee extends JFrame{
    //fields
    private JPanel mainPanel;

    private File mystery_file;
    private String essential_letter; //letter required into word
    private ArrayList<String> jumble_letters; //additional letters to play
    private TreeMap<String, Boolean> mystery_map; //list of words that have
not been found yet
    private int player_points;

    //default constructor
    public SpellingBee(){
        player_points = 0;
    }

    //constructor
    public SpellingBee(File f){
        mystery_file = f;
        player_points = 0;
    }

    /*
     * @params:
     * @returns:
     * Description: testing the code
     */
    public void tester() {
        System.out.println("essential letter: " + essential_letter);

        for(int i = 0; i < jumble_letters.size(); i++) {
            System.out.println("jumble_letters " + i + ": " +
jumble_letters.get(i));
        }

        System.out.println("Return key value pairs of Mystery Map Before: ");
        System.out.println(mystery_map.entrySet());
        String found_word = "boron";
        System.out.println("Is Word Match: " +
mystery_map.containsKey(found_word));
        System.out.println("After: ");
        mystery_map.remove(found_word);
        mystery_map.put(found_word, true);
        System.out.println(mystery_map.entrySet());
    }

    /*
     * @params: none
     * @returns: we find the pathname when we select a file
     * Description: we find the pathname when we select a file from chooser
     * For this exercise, user can choose their own test txt file
     */
    public void selectFile(){
        JFileChooser chooser = new JFileChooser();

        // This file filter allows the user to select JPEG and PNG files only
        FileNameExtensionFilter filter = new FileNameExtensionFilter("txt
files", "txt", "TXT");
        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(mainPanel); //dialog box
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile(); // open a dialog box to
select files
            //create new file with game data
            mystery_file = file;
        }
    }

    /*
     * @params:
     * @returns: none
     * Description: Adds Values to the fields essential_letter,
jumble_letters, and mystery_map
     */
    public void initializeFields() {
        try {
            //allocate memory to fields first
            jumble_letters = new ArrayList<String>();
            mystery_map = new TreeMap<String, Boolean>();

            Scanner file_read = new Scanner(mystery_file); //read mystery
file
            boolean isEssential = false;
            boolean isJumble = false;
            boolean isResults = false;

            //iterate through each line
            while (file_read.hasNextLine()) {
                String data = file_read.nextLine(); //let's read line

                if(data.equals("ESSENTIAL:")){//we need to read essential
letter
                    isEssential = true;
                }else if(data.equals("JUMBLE:")){//we need to read jumble
letters
                    isEssential = false;
                    isJumble = true;
                }else if(data.equals("RESULTS:")){//we need to read result
words
                    isJumble = false;
                    isResults = true;
                }else if(isEssential){
                    essential_letter = data; //record essential letter
                }else if(isJumble){
                    jumble_letters.add(data); //record jumble letters
                }else if(isResults){
                    mystery_map.put(data, false); //add mystery words to list
along with false boolean for not being found yet
                }
            }
            file_read.close(); //close file
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            System.out.println(e.getMessage());
        }
    }

    /*
     * @params:
     * @returns: Boolean
     * Description: A helper function for makeGUI
     * Determine whether player won game or not
     */
    public Boolean isGameWonHelperFunction(){
        Collection<Boolean> isFoundArr = mystery_map.values();
        for(Boolean isFound : isFoundArr){
            if(isFound == false){
                return false;
            }
        }
        return true;
    }

    /*
     * @params:
     * @returns: none
     * Description: Make the GUI to play the Spelling Bee Game
     */
    public void makeGUI() {
        setTitle("Spelling Bee GUI");
        //main Panel
        mainPanel = new JPanel();
        //add yellow color to main panel
        mainPanel.setBackground(Color.yellow);
        //flow layout
        mainPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 50));

        //display instructions on how to play in JTextArea
        JTextArea how_to_play_text_area = new JTextArea(5, 70);
        // make text area non-writable
        how_to_play_text_area.setEditable(false);
        //add text area to main panel
        mainPanel.add(how_to_play_text_area);
        //wrap line
        how_to_play_text_area.setLineWrap(true);
        //instructions down below
        how_to_play_text_area.append("How to Play? ");
        how_to_play_text_area.append("(1) Essential Letter Button
(highlighted in orange) requires its letter to be included anywhere in the
word at least once. ");
        how_to_play_text_area.append("(2) All other letter buttons can be
used any number of times or none at all to create a word. ");
        how_to_play_text_area.append("(3) Enter the word to see if it's in
the game's dictionary. ");
        how_to_play_text_area.append("(4) To Win: Find all the words in the
dictionary. ");
        how_to_play_text_area.append("Additional Rules ");
        how_to_play_text_area.append("(a) Word size must be greater than 3
letters. ");
        how_to_play_text_area.append("(b) Again, essential letter is required
at least once to create a word. ");

        //Letter Buttons
        JButton essential_letter_button = new JButton(essential_letter);
        essential_letter_button.setBackground(Color.ORANGE); //make essential
letter button orange
        essential_letter_button.setOpaque(true); //show color change
        mainPanel.add(essential_letter_button);
        JButton jumble_letter_button1 = new JButton(jumble_letters.get(0));
        mainPanel.add(jumble_letter_button1);
        JButton jumble_letter_button2 = new JButton(jumble_letters.get(1));
        mainPanel.add(jumble_letter_button2);
        JButton jumble_letter_button3 = new JButton(jumble_letters.get(2));
        mainPanel.add(jumble_letter_button3);
        JButton jumble_letter_button4 = new JButton(jumble_letters.get(3));
        mainPanel.add(jumble_letter_button4);
        JButton jumble_letter_button5 = new JButton(jumble_letters.get(4));
        mainPanel.add(jumble_letter_button5);
        JButton jumble_letter_button6 = new JButton(jumble_letters.get(5));
        mainPanel.add(jumble_letter_button6);

        //display fields
        JTextField display_field = new JTextField(30);
        // make display field non-writable
        display_field.setEditable(false);
        //add display fields to main panel
        mainPanel.add(display_field);
        //Clear all of player's input
        JButton clear_button = new JButton("CLEAR");
        mainPanel.add(clear_button);
        //Enter Button
        JButton enter_button = new JButton("ENTER");
        mainPanel.add(enter_button);
        JButton exit_button = new JButton("EXIT");

        //display fields
        JTextField message_field = new JTextField(30);
        // make display field non-writable
        message_field.setEditable(false);
        //add message fields to main panel
        mainPanel.add(message_field);

        //display score field
        JTextField score_field = new JTextField(10);
        // make score field non-writable
        score_field.setEditable(false);
        //add score field to main panel
        mainPanel.add(score_field);

        //display found words in JTextArea
        JTextArea found_words_text_area = new JTextArea(3, 4 *
mystery_map.size());
        // make text area non-writable
        found_words_text_area.setEditable(false);
        //add text area to main panel
        mainPanel.add(found_words_text_area);

        //action listener for essential letter button
        essential_letter_button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // get the text in from button
                String essential_letter =
essential_letter_button.getText().toLowerCase();

                //get current characters in display field
                String word = display_field.getText().toLowerCase();
                //append new letter to word
                word += essential_letter;
                //display added letter in the display field
                display_field.setText(word);
            }
        });

        //action listener for jumble letter button1
        jumble_letter_button1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // get the text in from button
                String jumble_letter1 =
jumble_letter_button1.getText().toLowerCase();

                //get current characters in display field
                String word = display_field.getText().toLowerCase();
                //append new letter to word
                word += jumble_letter1;
                //display added letter in the display field
                display_field.setText(word);
            }
        });

        //action listener for jumble letter button2
        jumble_letter_button2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // get the text in from button
                String jumble_letter2 =
jumble_letter_button2.getText().toLowerCase();

                //get current characters in display field
                String word = display_field.getText().toLowerCase();
                //append new letter to word
                word += jumble_letter2;
                //display added letter in the display field
                display_field.setText(word);
            }
        });

        //action listener for jumble letter button3
        jumble_letter_button3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // get the text in from button
                String jumble_letter3 =
jumble_letter_button3.getText().toLowerCase();

                //get current characters in display field
                String word = display_field.getText().toLowerCase();
                //append new letter to word
                word += jumble_letter3;
                //display added letter in the display field
                display_field.setText(word);
            }
        });

        //action listener for jumble letter button4
        jumble_letter_button4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // get the text in from button
                String jumble_letter4 =
jumble_letter_button4.getText().toLowerCase();

                //get current characters in display field
                String word = display_field.getText().toLowerCase();
                //append new letter to word
                word += jumble_letter4;
                //display added letter in the display field
                display_field.setText(word);
            }
        });

        //action listener for jumble letter button5
        jumble_letter_button5.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // get the text in from button
                String jumble_letter5 =
jumble_letter_button5.getText().toLowerCase();

                //get current characters in display field
                String word = display_field.getText().toLowerCase();
                //append new letter to word
                word += jumble_letter5;
                //display added letter in the display field
                display_field.setText(word);
            }
        });

        //action listener for jumble letter button6
        jumble_letter_button6.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // get the text in from button
                String jumble_letter6 =
jumble_letter_button6.getText().toLowerCase();

                //get current characters in display field
                String word = display_field.getText().toLowerCase();
                //append new letter to word
                word += jumble_letter6;
                //display added letter in the display field
                display_field.setText(word);
            }
        });

        //action listener for clear button
        clear_button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                //display empty string in the display field
                display_field.setText("");
            }
        });

        //action listener for enter button
        enter_button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                //get current characters in display field
                String word = display_field.getText().toLowerCase();

                if(mystery_map.containsKey(word) && mystery_map.get(word) ==
false){
                    //points earned
                    int points_earned = word.length();

                    //accumulate earned points
                    player_points += points_earned;

                    //display updated score field
                    score_field.setText(player_points + " total points");

                    //display success message
                    message_field.setText("SUCCESS! Points Earned: " +
points_earned);
```

```
                    //remove mystery_map key value pair
                    mystery_map.remove(word);

                    //add value pair as true instead
                    mystery_map.put(word, true);

                    //append to text area
                    found_words_text_area.append(word + " ");

                    //display empty string in the display field
                    display_field.setText("");

                    //check if game won
                    if(isGameWonHelperFunction()){
                        //display Won Game Message
                        message_field.setText("CONGRATS. YOU WON THE GAME!");
                        //disable all game buttons to force player to exit
                        essential_letter_button.setEnabled(false);
                        jumble_letter_button1.setEnabled(false);
                        jumble_letter_button2.setEnabled(false);
                        jumble_letter_button3.setEnabled(false);
                        jumble_letter_button4.setEnabled(false);
                        jumble_letter_button5.setEnabled(false);
                        jumble_letter_button6.setEnabled(false);
                        enter_button.setEnabled(false);
                        mainPanel.add(exit_button); //add exit button to end
game
                    }
                }else if(mystery_map.containsKey(word) &&
mystery_map.get(word) == true){
                    //display already found message
                    message_field.setText("WORD ALREADY FOUND. TRY AGAIN.");

                    //display empty string in the display field
                    display_field.setText("");
                }else if(word.length() <= 3){
                    //display word is too small
                    message_field.setText("WORD IS TOO SHORT. TRY AGAIN.");

                    //display empty string in the display field
                    display_field.setText("");
                }else{
                    //display word invalid
                    message_field.setText("INVALID WORD. TRY AGAIN.");

                    //display empty string in the display field
                    display_field.setText("");
                }
            }
        });

        //action listener for exit button
        exit_button.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            System.exit(0);
          }
        });
        //make window visible
        add(mainPanel);
        setSize(900, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    //main method
    public static void main(String[] args){
        SpellingBee bee_game = new SpellingBee();
        bee_game.selectFile(); //let user choose which file to play game
(load game files)
        bee_game.initializeFields(); //data from file is stored in program
        bee_game.makeGUI(); //show game GUI

        //test method
        //bee_game.tester();
    }
}
```