JavaFX Turkish Spelling Bee

Your job is to implement an interactive Turkish JavaFX Spelling Bee game.

The purpose of this project is threefold:

1. to let you create a JavaFX GUI Model-View-Controller design pattern,
2. to let you practice working with strings, array lists, and text files in Java,
3. to give you a chance to implement an exciting puzzle that emphasizes interactivity.

The Spelling Bee Puzzle

Spelling Bee is one of the most popular word games in the *New York Times*, which appears daily on the web at https://www.nytimes.com/puzzles/spelling-bee.

Each Spelling Bee puzzle consists of seven hexagons arranged in a small beehive-like shape. Figure 1 shows a screen image of a sample Spelling Bee puzzle.

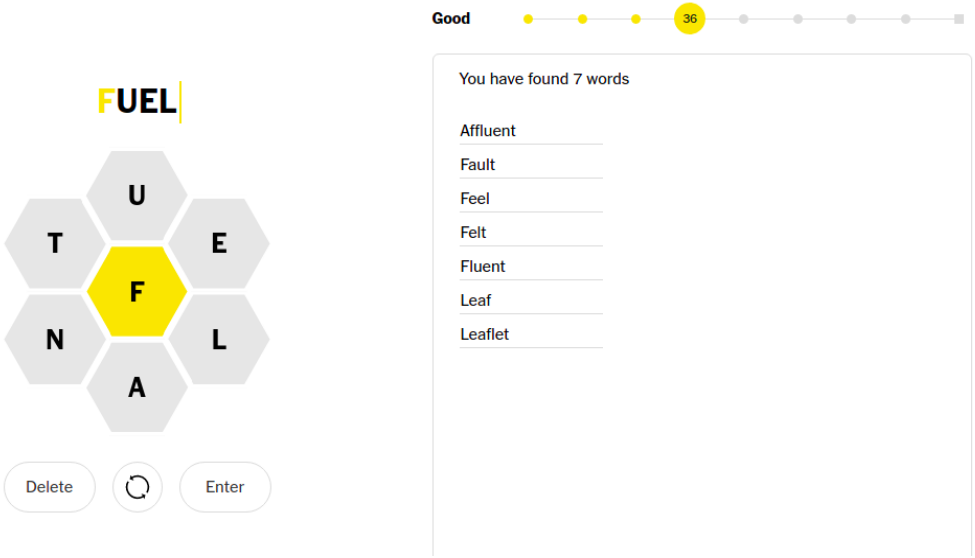

Figure 1: Screenshot of a Spelling Bee game

The aim of the puzzle is to find as many words in this layout as you can, subject to the following rules:

- Each word must be at least four letters long (e.g., *fan* is not a valid word, it is too short).
- Each word must contain the center letter at least once. (e.g., *lean* is not a valid word since it does not contain the center letter *F*.)
- Each word must not contain any letters other than the seven letters in the layout (e.g., *fail* is not a valid word since it contains the letter *I*).
- Each word can contain the same letter more than once (e.g., *feel* is a valid word).
- In the original game, each word must be a legal English word. However, in your assignment, each word must be a legal Turkish word. You can use a Turkish word list TurkishWords.txt which you can find in ERUDM.

JavaFX Turkish Spelling Bee Project

Create the JavaFX graphical display: First, a JavaFX GUI must be implemented using Model-View-Controller design pattern.

GUI should be similar to original one which can be seen in Fig. 1. However, you are free to design your own GUI as long as you stick to the basic principles. Your GUI should contain a clickable beehive that shows letters, a delete button, a shuffle button, an enter button, columns listing the words that the user has found, a label or any other component that shows the user's current score. The user should also be able to write letters using the keyboard and delete the last letter by pressing the return key.

There will be two different ways of generating the puzzle word, as given in the next section. Therefore, your GUI should be such that it allows the game to be started in either of these two different ways.

Read word list from the text file: Your program should read the given Turkish Words text file and then obtain a Turkish word list.

Generate the puzzle word: In the project, the puzzle word should be generated in two different ways. In the first one, the user is responsible for entering the seven-letter puzzle string. So, there should be a text field in your GUI where the user can enter the letters and a button that the user can click to generate the puzzle. When the user writes a seven-letter puzzle string in the text field and clicks generate button, your program must check if the entered letters meet the following conditions:

- The puzzle must contain exactly seven characters.
- Every character must be one of the 29 Turkish letters.
- No letter may appear more than once in the puzzle.

In a second way, the puzzle world should be generated automatically. Puzzles must include at least one **pangram** word. The words that use all seven letters in the puzzle are called pangram and score extra points. For example, the pangram in the puzzle shown in Fig. 1 is the word *affluent*. Also, your puzzle generator should probably not produce word lists that are too large or too short. According to the website **https://nytbee.com**, the number of words in the *New York Times* puzzles has varied between 21 and 81, and the total number of points has ranged from 50 to 444. Your puzzle should contain words between 20 and 80, and the total number of points should be ranged from 100 to 400. Consequently, the automatically generated puzzle must meet the following conditions:

- The puzzle must contain exactly seven characters.
- Every character must be one of the 29 Turkish letters.
- No letter may appear more than once in the puzzle.
- The puzzle must contain at least one pangram word.
- The puzzle should contain words between 20 and 80.
- The total number of points in the puzzle should be ranged from 100 to 400.

Display the Puzzle: Once the puzzle is created, the letters must appear on the screen. Suppose the user's entered string does not meet the above conditions. In that case, your code should display a message telling the user why the puzzle is not properly formed.

Find the all the possible answers: Your program should go through the dictionary and check each word to see whether it appears in the puzzle according to the following rules:

- The word is at least four letters long.
- The word does not contain any letters other than the seven letters in the puzzle.
- The word contains the center letter.

You must use an appropriate data structure to store the possible answers. Your program must add each word that fits these rules to the data structure (list of true answers).

Start the game and let the user try to find the words: Your program must let the user try to find the words using your GUI. Then it must check if the word that the user guess is in the list of true answers. If so, the *SpellingBee* application should add it to the list of found words, along with its score. If not, the application should display a message to tell the user what is wrong with the word. The possible reasons for rejecting a word are:

- The word is not in the dictionary.
- The word includes letters not in the beehive.
- The word does not include at least four letters.
- The word does not include the center letter.
- The user has already found the word and is not allowed to score it twice.

Update the word list and score: When the user finds an acceptable word, the program should display the word in the list of found words and update the score. Players receive one point per letter after three letters (a word with four letters is one point, a word with five letters is two points etc.). If the user finds a pangram, receive an extra seven points

Documentation

At the end of your project, you must prepare a detailed report that describes your work. You must give details about your project, provide the class, and use case diagrams. You can find a template for your project report on the ERUDM.

Academic Honesty

Honesty and integrity are very important. Never submit the work of others as your own. A student that copies the same code from an outside source or another one or allows someone to copy from his/her project will receive 0 points.

Submission

You should upload your project in **zip** format to ERUDM. The name of your submission must be your group number (e.g., group1.zip). You will not get any marks if you forget to name your submission as your group number. Each submission should consist of all java files required to compile and run your program plus a project report. We cannot grade something that does not run.

Assignment Defense

Groups will defend their project in the computer lab at a date to be announced later. All the group members must be present in the lab to show their work and explain their contributions.

Note: No third-party toolkits or libraries are allowed. However, you are free to use any of the sample code provided in this course.

Acknowledgments: Special thanks to Prof. Jed Rembold.