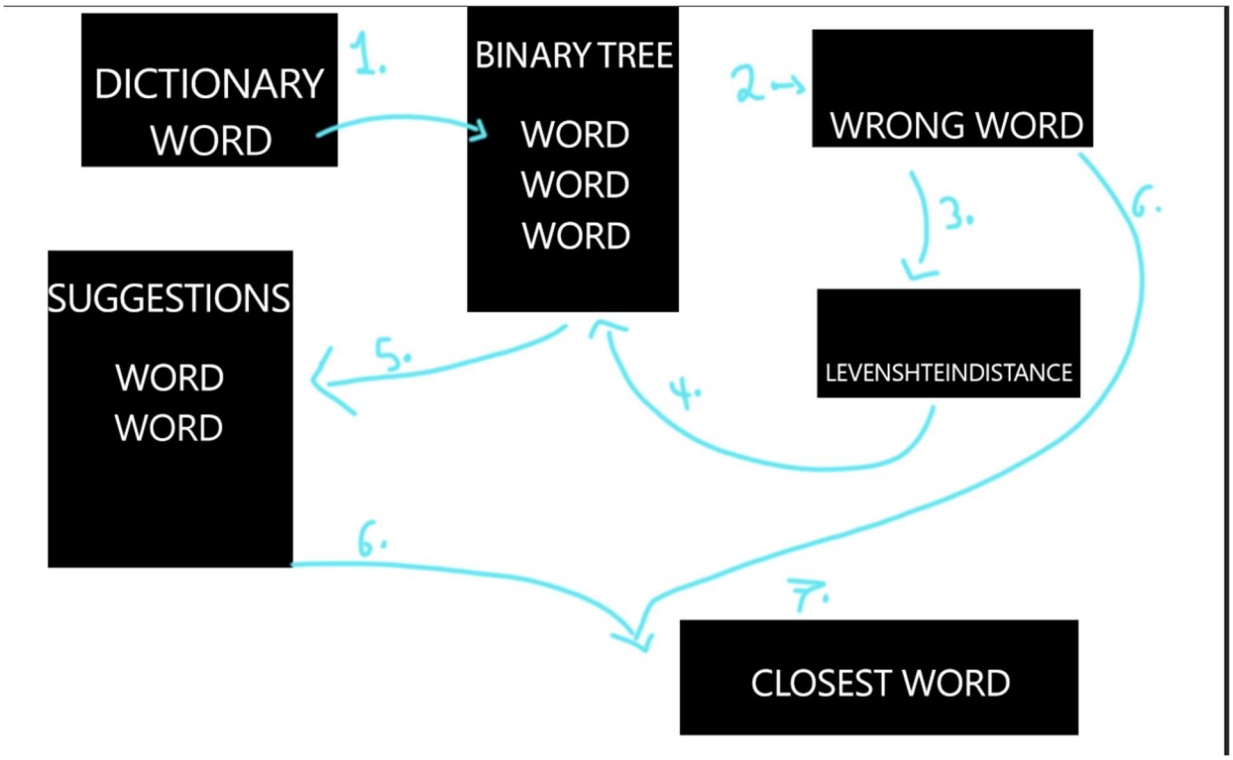
20/04/2023

BLM19202E Data Structures Programming Assignment #2
Spell Checking and Autocomplete with Binary Search Tree

TARIK ALRAYAN–2121221366

1) Design

The plan of the project :



The project should take words and check the if the are in the dictionary or not if the are not found in dictionary it should return the closest word and return it with the wrong word using LevenshteinDistance algorithm.

pseudo-code of LevenshteinDistance algorithm:

```
function getLevenshteinDistance(s, t):
   m = length(s)
   n = length(t)
   dp = new 2D array[m+1][n+1]

   for i from 0 to m:
      dp[i][0] = i

   for j from 0 to n:
      dp[0][j] = j

   for i from 1 to m:
      for j from 1 to n:
         cost = 0
         if s[i-1] != t[j-1]:
            cost = 1
         dp[i][j] = min(dp[i-1][j]+1, dp[i][j-1]+1, dp[i-1][j-1]+cost)

   return dp[m][n]
```

2) Implementation Details:

Correct implementation of a binary search tree to store a dictionary of words :

```
//a class to handel the dictionary operations and the LevenshteinDistance algorathim
public class Dictionary<T extends Comparable<T>> {

    // this method takes a file and read the words inside it the but them into a binary tree
    public void getDicFromFile(File f, BinaryTree<T> tree) {
        try {
            Scanner scanner = new Scanner(source:f, charsetName: "UTF-8");
            while (scanner.hasNext()) {
                String word = scanner.next();
                tree.insert((T) word);
            }

        } catch (FileNotFoundException e) {
            System.out.println(x: "File not found.");
        }
    }
```

Here I'm using a class Dictionary to handle working with the dictionary file and the algorithms , in the method "getDicFromFile" it takes a binary tree and a file then get all the words inside the file and using the "insert" method from the binarytree class to insert the words into the tree .

1. Correct implementation of a Levenshtein distance algorithm to suggest corrections for misspelled words:

   For this operation I used the method "findClosestWords" with 1 helper recursive method and getLevenshteinDistance to implement the LevenshteinDistance algorithim

```
//this method takes the root of the binary tree of dictionary a word and a k suggest
public BinaryTree<T> findClosestWords(Node<T> root, T targetWord, int k) {

    //an array to hold the distance for the suggestions word
    Object[][] closestWords = new Object[k][2];
    findClosestWords(node: root, targetWord, closestWords, k);

    BinaryTree<T> result = new BinaryTree<>();
    for (int i = 0; i < closestWords.length && closestWords[i][0] != null; i++) {
        result.insert((T) closestWords[i][0]);
    }

    return result;
}
```

This method takes a root of a binary tree that contains the dictionary and a T type target word and k number of suggestions the creates an 2d array with k size to store the suggestions and there distance from the target word and then calls the helper "findClosestWords" to check the distances and fill the array with the suggestions and insert the suggestions to the binary tree according to the distance

Continue with the helper “findClosestWords” :

```
//this method use the levenshteinDistace to compare the distace and find the closest word
private void findClosestWords(Node<T> node, T targetWord, Object[][] closestWords, int k) {
    if (node == null || contains(node, value: targetWord)) {
        return;
    }

    int distance = getLevenshteinDistance(s: node.getData().toString(), t: targetWord.toString());

    for (int i = 0; i < closestWords.length; i++) {
        if (closestWords[i][0] == null) {
            closestWords[i][0] = node.getData();
            closestWords[i][1] = (distance);
            break;
        } else {
            int currDistance = (Integer) closestWords[i][1];

            if (distance < currDistance) {
                for (int j = closestWords.length - 1; j > i; j--) {
                    closestWords[j][0] = closestWords[j - 1][0];
                    closestWords[j][1] = closestWords[j - 1][1];
                }
                closestWords[i][0] = node.getData();
                closestWords[i][1] = (distance);
                break;
            }
        }
    }

    findClosestWords(node: node.getLeft(), targetWord, closestWords, k);
    findClosestWords(node: node.getRight(), targetWord, closestWords, k);
}
```

This recursive method takes a node the same arguments with 2d array and check the word if it null or found in the dictionary it should return else it will take every word in the dictionary and get the distance between them using the levenshtein algorithm then loop the

2d array and store the information like this [distance][word] and check if the array got a distance bigger then the last distance it should replace with it to have an 2d array with the closest word in the first index then keep recursive for right and left node in the tree

In the end I got the “getLevenshteinDistance” method

```
//this method implement the LevenshteinDistance algorithim and return the distance bettwen the givven srting
private int getLevenshteinDistance(String s, String t) {
    int m = s.length();
    int n = t.length();

    int[][] dp = new int[m + 1][n + 1];

    for (int i = 1; i <= m; i++) {
        dp[i][0] = i;
    }

    for (int j = 1; j <= n; j++) {
        dp[0][j] = j;
    }

    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            int cost = s.charAt(i - 1) == t.charAt(j - 1) ? 0 : 1;
            dp[i][j] = Math.min(dp[i - 1][j] + 1, b: Math.min(dp[i][j - 1] + 1, dp[i - 1][j - 1] + cost));
        }
    }

    return dp[m][n];
}
```

the Levenshtein distance algorithm is a method used to measure the similarity between two strings. The algorithm calculates the minimum number of operations (insertions, deletions, or substitutions) required to transform one string into the other.

The algorithm uses dynamic programming to efficiently calculate the distance between two strings. It creates a two-dimensional array with dimensions (m+1) x (n+1), where m and n are the lengths of the two strings being compared. The algorithm initializes the first row and column of the array with values representing the cost of transforming an empty string into the corresponding substring of the

other string. It then iterates through the remaining cells of the array, calculating the cost of transforming each substring and storing the minimum cost in each cell.

After the array is filled, the final value in the bottom-right corner represents the minimum cost of transforming the entire strings. This value is returned as the Levenshtein distance between the two strings.

2. Correct implementation of auto-completion

```
private void autoComplete(JTextArea txt) {
    String word = getLastWord(txt);
    String closest = findClosestWord(word);
    if (!closest.isEmpty()) {
        String text = txt_text.getText().trim();
        String[] words = text.split(regex: " ");
        words[words.length - 1] = closest;
        String updatedText = String.join(delimiter:" ", elements: words);
        txt.setText(t: updatedText);
        findClosestWords();
    }

}
```

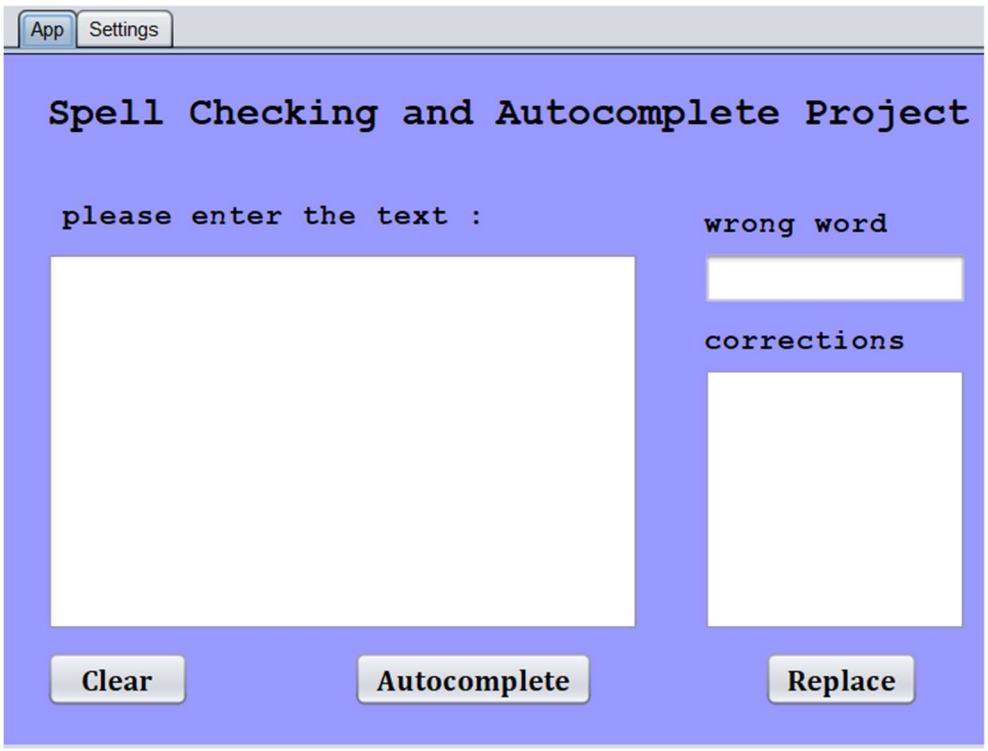
For auto-complete I use the same algorithm that I used in the previous step but I only get the closest element and replace it I implement this method in the Gui

3. Proper handling of capitalization and punctuation in input text:

```
private String getClearText(JTextArea txt) {
    String text = txt.getText();
    String cleaned = text.trim();
    // Replace all multiple spaces with a single space
    cleaned = cleaned.replaceAll(regex: "\\s+", replacement: " ");
    // Remove all punctuation except for periods, question marks, and exclamation points
    cleaned = cleaned.replaceAll(regex: "[^\\w\\s.?!]", replacement: "");
    cleaned = cleaned.toLowerCase();

    return cleaned;
}
```

To handle the capitalization and punctuation in input text I used the getClearText it gives a clear text from the text area

4. GUI implementation :



I used a text area and a j list to present the suggestions to the user , so when the user enter a word in the text area

1) If the user pressed space or enter that means the user finish so I added an event to check if the space or enter keys pressed then it will go and check the last word and if it wrong its going to highlight the word and get the suggestions to the j list if the user select a suggestion and pressed the replace button the text will update and replace the word .

   And here how it works in code :

```
private void txt_textKeyPressed(java.awt.event.KeyEvent evt) {
    //event to check if the pressed key is equal to space or enter and call the findCloses
    if (evt.getKeyCode() == KeyEvent.VK_SPACE || evt.getKeyCode() == KeyEvent.VK_ENTER) {
        findClosestWords();
    }

}
```

An event to handle if the pressed key equal to space or enter then calls the findclossestword method.

```
private void findClosestWords() {

    //clear the old highlighter on the text
    Highlighter highlighter = txt_text.getHighlighter();
    highlighter.removeAllHighlights();
    String text = getClearText(txt:txt_text);
    String[] words = text.split(regex: "\\s+"); // Split by whitespace characters
    //loop all the word in the text area
    for (String word : words) {
        if (!word.isEmpty()) {

            //check if the
            if (!dictionaryOp.contains(node: dictionary.getRoot(), value: word)) {
                DefaultListModel<String> listModel = new DefaultListModel<>();
                corrections = dictionaryOp.findClosestWords(root: dictionary.getRoot(), targetWord: word, k);
                txt_word.setText(t: word);
                try {
                    highlightWord(textArea: txt_text, wordToHighlight: word, highlighter);
                } catch (BadLocationException ex) {
                    Logger.getLogger(name: ProjectFrame.class.getName()).log(level: Level.SEVERE, msg:null, thrown:ex);
                }
                setListFromTree(listModel, node: corrections.getRoot());
                list_corrections.setModel(model: listModel);

            } else {
                DefaultListModel<String> EmptylistModel = new DefaultListModel<>();
                txt_word.setText(t: "");
                list_corrections.setModel(model: EmptylistModel);
            }
        }
```

First of all delete all the previous highlighted words and get the clear text from the text area and put the text into array and start checking all the words if they are in dictionary or not if they not found the word should highlight in red color and it's going to call the find Closest Words from the dictionary object to get the closest words to this word then put the suggestions into the list using the set list from tree method then the user can select the suggestion .

```
private void btn_replaceActionPerformed(java.awt.event.ActionEvent evt) {
    // event to replace the wrong word with the sellected word in the list

    //check if th eword is found in the dictionary and then aplly to replace the word with the correction
    if (!dictionaryOp.contains(node: dictionary.getRoot(), value: txt_word.getText())) {
        String word = list_corrections.getSelectedValue();
        String wrongWord = txt_word.getText();
        String text = getClearText(txt:txt_text);

        int index = text.indexOf(str:wrongWord);

        while (index != -1) {
            // Replace the found word with the replacement word
            text = text.substring(beginIndex: 0, endIndex: index) + word + text.substring(index + wrongWord.length());

            // Find the next occurrence of the word to find
            index = text.indexOf(str:wrongWord, index + word.length());
        }

        // Update the text area with the replaced text
        txt_text.setText(t: text);
    }

    findClosestWords();

}
```

An event for the replace button here I check if the word not found in the dictionary it goes to the list and get the selected item the replace it with the wrong word the check again if the any wrong words calling the find closest words method again.

2) If the user continues adding wrong words and not replacing them from the list the words will be highlighted in red so when the user clicks in the word it will take the word and return the suggestions for this word and can replace it

```
private void txt textMouseClicked(java.awt.event.MouseEvent evt) {
    // event to get the word thet the muse click on and if the word wrong it returns th
    JTextComponent textComp = (JTextComponent) evt.getSource();
    Point pt = new Point(x: evt.getX(), y: evt.getY());
    int pos = textComp.viewToModel(pt);

    // Get the word containing the clicked position
    String text = textComp.getText();
    int start = text.lastIndexOf(str: " ", fromIndex:pos) + 1;
    int end = text.indexOf(str: " ", fromIndex:pos);
    if (end == -1) {
        end = text.length();
    }
    String clickedWord = "";
    if (end >= start) {
        clickedWord = text.substring(beginIndex: start, endIndex: end);
    }
    //check if the word is not equal to empty then call the findClosestWords function
    if (!clickedWord.trim().isEmpty()) {
        findClosestWords(word: clickedWord);
    } else {
        findClosestWords();
    }

}
```

An event to get the word on the clicked place in the text area then calls the method find closest word with taking a selected word.

```
// this method takes a sellected word and check if it found in dictionary  else it will returns the li
private void findClosestWords(String word) {
    {
        //check if the word found in the dictionary
        if (!dictionaryOp.contains(node: dictionary.getRoot(), value: word)) {
            DefaultListModel<String> listModel = new DefaultListModel<>();
            corrections = dictionaryOp.findClosestWords(root: dictionary.getRoot(), targetWord: word, k);
            txt_word.setText(t: word);
            setListFromTree(listModel, node: corrections.getRoot());
            list_corrections.setModel(model: listModel);

        } else {
            DefaultListModel<String> EmptylistModel = new DefaultListModel<>();
            txt_word.setText(t: "");
            list_corrections.setModel(model: EmptylistModel);
        }
    }

}
```

It does the same job as the find closest words method but without loop a text areas words just takes one word and check it then the user can select a suggestion and replace it

3)if the user prese the autocomplete button it will takes the last word and find the closest word to it and replace it

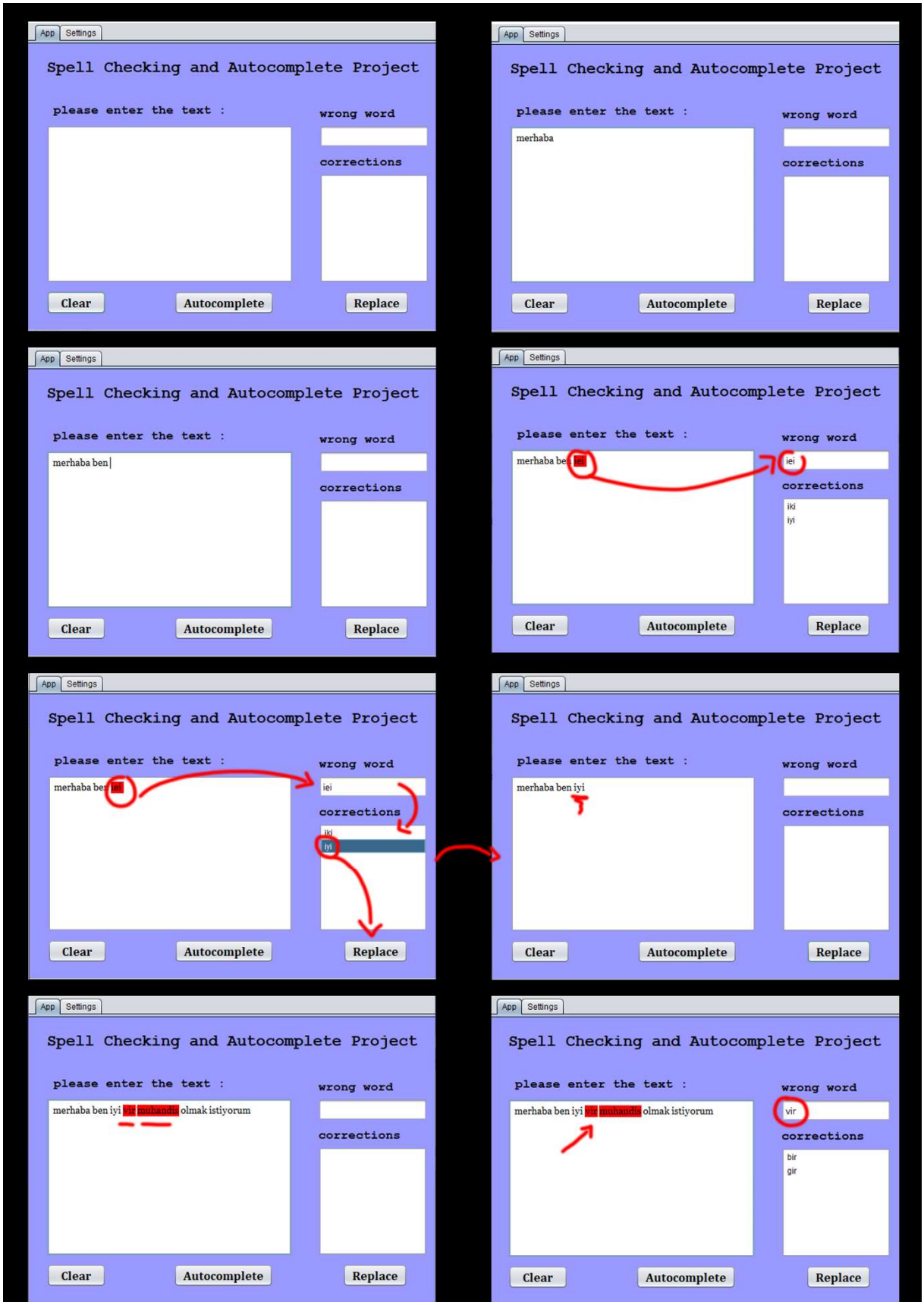
```
private void autoComplete(JTextArea txt) {
    String word = getLastWord(txt);
    String closest = findClosestWord(word);
    if (!closest.isEmpty()) {
        String text = txt_text.getText().trim();
        String[] words = text.split(regex: " ");
        words[words.length - 1] = closest;
        String updatedText = String.join(delimiter:" ", elements: words);
        txt.setText(t: updatedText);
        findClosestWords();
    }

}
```

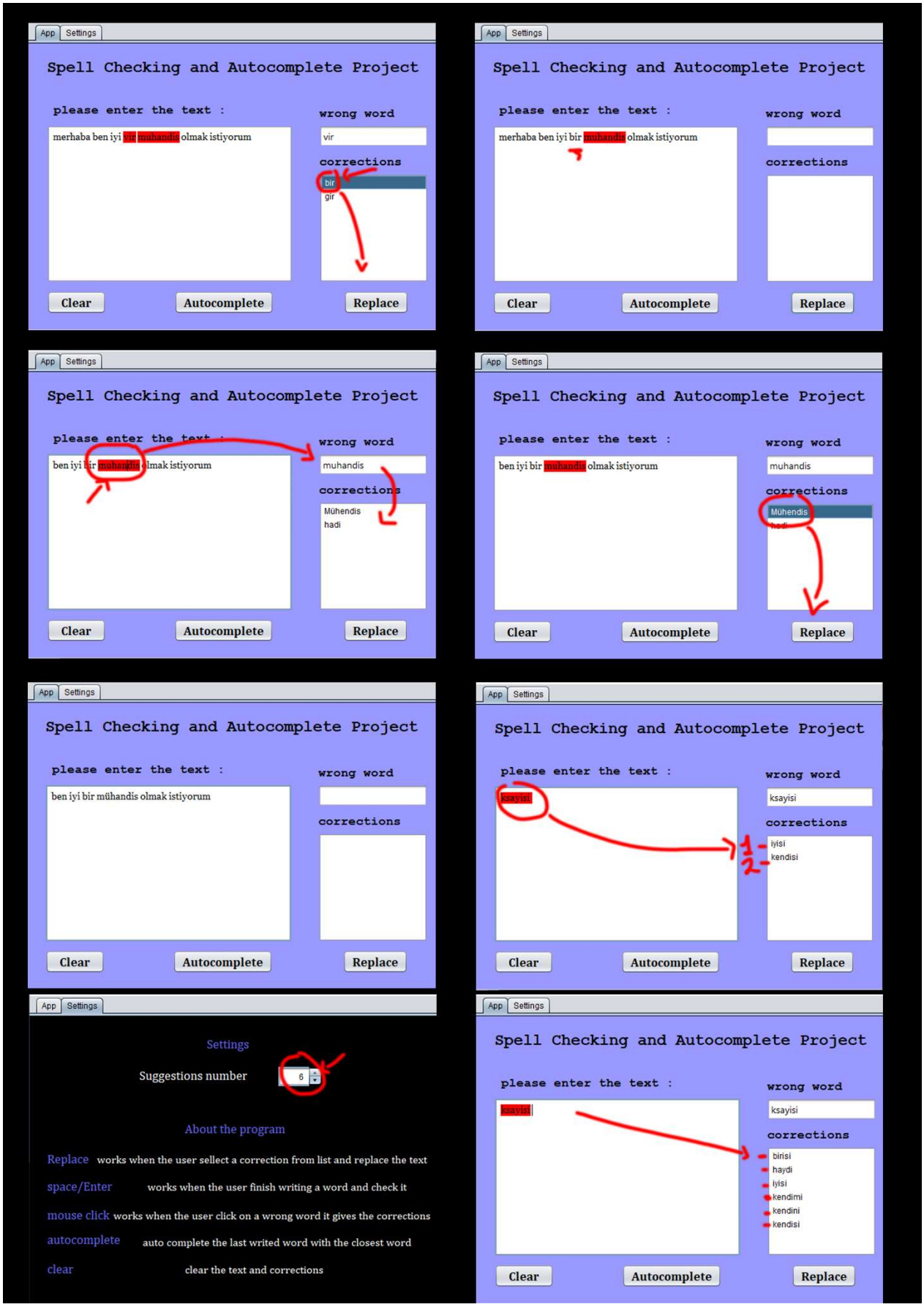
This function uses a find closest word to get the closest word.

```
//this method takes a word and check if it found in dictionary and if it not found resturns the c
private String findClosestWord(String word) {

    String closest = "";
    if (!dictionaryOp.contains(node: dictionary.getRoot(), value: word)) {
        DefaultListModel<String> listModel = new DefaultListModel<>();
        corrections = dictionaryOp.findClosestWords(root: dictionary.getRoot(), targetWord: word, k);
        txt_word.setText(t: word);
        setListFromTree(listModel, node: corrections.getRoot());
        list_corrections.setModel(model: listModel);
        closest = listModel.getElementAt(index: 0);

    } else {
        DefaultListModel<String> EmptylistModel = new DefaultListModel<>();
        txt_word.setText(t: "");
        list_corrections.setModel(model: EmptylistModel);
    }

    return closest;

}
```

It’s the same with the others but the only difference is that it returns only one suggestion that is the closest one to the wanted word.

3) Output:


App
Settings
Spell Checking and Autocomplete Project
please enter the text :
wrong word
corrections
Clear
Autocomplete
Replace
merhaba
merhaba ben
merhaba ben iei
iei
iki
iyi
merhaba ben iyi
merhaba ben iyi vir muhandis olmak istiyorum
vir
bir
gir


App
Settings
Spell Checking and Autocomplete Project
please enter the text :
wrong word
corrections
merhaba ben iyi vir muhandis olmak istiyorum
vir
bir
gir
Clear
Autocomplete
Replace
merhaba ben iyi bir muhandis olmak istiyorum
ben iyi bir muhandis olmak istiyorum
muhandis
Mühendis
hadi
ben iyi bir mühandis olmak istiyorum
ksayisi
iyisi
kendisi
Settings
Suggestions number
6
About the program
Replace works when the user sellect a correction from list and replace the text
space/Enter works when the user finish writing a word and check it
mouse click works when the user click on a wrong word it gives the corrections
autocomplete auto complete the last writed word with the closest word
clear clear the text and corrections
birisi
haydi
iyisi
kendimi
kendini
kendisi