Assignment 4: Dictionary

Logistics

- The assignment is meant to be done individually.
- The **deadline** for this assignment is **11:59 PM on Dec 2, 2024, Pacific Time**.
- Academic dishonesty is unacceptable and will not be tolerated in this course.
- **Last Modified**: Nov 15, 2024

General Description

In this assignment, you need to build a **dictionary application using Swing.**

The application has the following features. Users can:

- add new words, new meanings to the words
- replace an old word with a new one, **without** changing its original meaning and its frequency
- remove words from the dictionary and check if a word is in the dictionary
- count the frequency of searches for each word.
- display the most frequent words based on the prefix they provide.
- display the search history
- Import and export words in batch

In detail, your dictionary should have the following functionalities:

- Users should be able to add, remove and modify words.
- Users should be able to check if a certain word is in the dictionary and retrieve its meanings.
- Users should be able to see the three most frequent words that can complete their given keyword.
    - e.g., for `uni`, it should return `universe`, `university`, `universal`.
    - e.g., for `apple`, `mapple`, `napple`, `lapple` are feasible for the search.
- The dictionary will display a search history of up to 10 words, showing the most recently searched words first. Only the matched words (including the three most frequent words) will be added to the search history.
- Users should be able to import/export the dictionary from/into a `txt` file.
    - Input format:

```
word0
word_meaning0

word1
word_meaning1

...

wordn
word_meaningn

```

- Output format:

```
word0
frequency0
word_meaning0

word1
frequency1
word_meaning1

...

wordn
frequencyn
word_meaningn

```

- For export, please export the words in the (descending) order of frequency
    - We ensure that no word will have the same frequency in grading test cases.

Exception Handling

In this assignment, you need handle 4 kinds of exceptions:

- `InvalidWordError`: the word entered to add/find/clear is not a word
    - By mentioning `word`, we define it as a `String` consisting of `a-z` and `A-Z`
- `WordNotFoundError`: the word is not found in our dictionary.
- `WordDuplicatedError`: the word to be added has existed in the dictionary.
- `FileNotFoundError`: the file path provided does not exist when importing or exporting from a `txt` file.

Definition of Exceptions

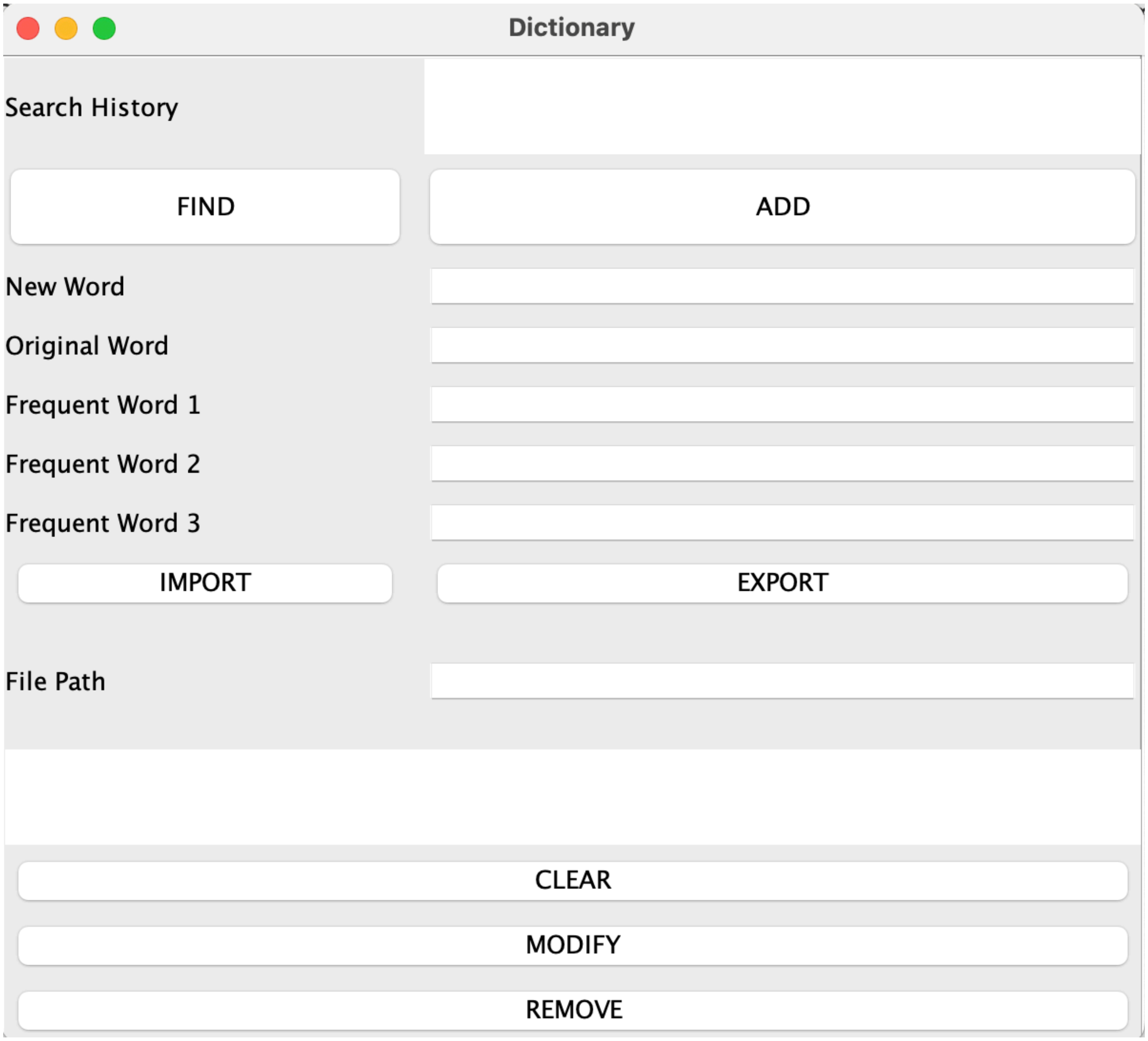
In order to raise those 4 exceptions, please define them by inheriting the `RuntimeException` class.

Once needed, you need to:

- First, throw the related exception
- Then, display texts to notify that an error has occurred.

Notice: For grading, we will test only **one error** for each exception-handling test case.

Layout Example for Dictionary



Recommended Procedure for Implementation

1. Create a project named `Dictionary`
2. Create a package under `src` named `Dictionary`

3. Create a form under package `Dictionary` with the name `Dictionary.form`. Check create bound class and it will automatically create `Dictionary.java`
4. On `Dictionary.form`, design the GUI of Dictionary. In particular, you should (at least) have the following components as you can see in the figure we show above:
    1. `FINDButton`

       Note:

        - **only** increase the frequency counts of the **top 3** words.
        - The words feasible to be searched should **include** the keyword.
        - Display the top 3 (or 2 or 1) words in the `TextFreqWord1`, `TextFreqWord2`, `TextFreqWord3` fields in the (descending) order of frequency
            - If there are fewer than 3 words to display, please leave those `TextFreqWord` fields empty.
    2. `ADDButton`
    3. `MODIFYButton`

       Note: replace the old word with the new one, but **keep its original meaning and frequency**.
    4. `REMOVEButton`
    5. `CLEARButton`
    6. `IMPORTButton`
    7. `EXPORTButton`
    8. `TextNewWord`: where you type in the word you want to add/find/remove
    9. `TextOriginalWord`: where you type in the frequency you want to modify
    10. `TextFreqWord1`, `TextFreqWord2`, `TextFreqWord3`: 3 TextFields to display frequent words in finding procedure
    11. `TextArea`: where:
        1. word meanings are displayed
        2. error messages are displayed
    12. `searchHistoryList`: a JList where show the search history
    13. `xTextFilePath`: where you type in the path of import ot export

    Be careful: The names of the components should be the same as showed above. Don't forget to consider and handle 4 self-defined exceptions during implementation!
5. Define 4 self-defined exceptions mentioned above **using inheritance of `RuntimeException`**.
6. Implement the Action Listens for all the buttons.
7. Test your code and see if the GUI works smoothly and has the expected functionality.

SampleTest

Please notice that `SampleTest` is just to help you understand the whole assignment more easily. Sample tests are only for clearer explanation and simple testing during your implementation. **Passing all the sample tests does not mean you will get a full score.** You need to read the description carefully, and think it comprehensively when implementing this assignment.

After you complete the assignment following the recommended procedures listed above, you could the following `SampleTest` code to test if your implementation works smoothly:

```java
package Dictionary;

import java.awt.*;

public class SampleTest {
    public static void main(String[] args) {
        Dictionary myDictionary = new Dictionary();

        // Test for ADD
        // InvalidWordError
        myDictionary.TextNewWord.setText("s1mple");
        myDictionary.TextArea.setText("Best AWPer");
        try {
            myDictionary.ADDButton.doClick();
        } catch (InvalidWordError ex) {
            System.out.println("InvalidWordError passed");
        }

        // WordDuplicatedError
        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextArea.setText("Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextArea.setText("Rifler");
        try {
            myDictionary.ADDButton.doClick();
        } catch (WordDuplicatedError ex) {
            System.out.println("WordDuplicatedError passed");
        }

        // Valid ADD
        myDictionary = new Dictionary();

        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextArea.setText("Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        System.out.println("Word niko ADD successfully.");

        // Test for CLEAR button
        myDictionary = new Dictionary();

        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextOriginalWord.setText("nikoo");
        myDictionary.TextFreqWord1.setText("aaa");
        myDictionary.TextFreqWord2.setText("bbb");
        myDictionary.TextFreqWord3.setText("ccc");
        myDictionary.TextArea.setText("Fortunate entry fragger");

        myDictionary.CLEARButton.doClick();
        String tmp0 = myDictionary.TextNewWord.getText();
        String tmp1 = myDictionary.TextOriginalWord.getText();
        String tmp2 = myDictionary.TextFreqWord1.getText();
        String tmp3 = myDictionary.TextFreqWord2.getText();
        String tmp4 = myDictionary.TextFreqWord3.getText();
        String tmp5 = myDictionary.TextArea.getText();

        if (tmp0.equals("") && tmp1.equals("") && tmp2.equals("") && tmp3.equals("") && 
tmp4.equals("") && tmp5.equals("")){
            System.out.println("CLEAR button test passed");
        }

        // Test for FIND button
        myDictionary = new Dictionary();
        // "No Word Matched."
        myDictionary.TextNewWord.setText("NIKO");
        myDictionary.FINDButton.doClick();
        if (myDictionary.TextArea.getText().equals("No Word Matched.")) {
            System.out.println("No Word Matched test passed");
        }

        // Valid FIND Test 1
        myDictionary = new Dictionary();

        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextArea.setText("Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        myDictionary.CLEARButton.doClick();

        myDictionary.TextNewWord.setText("nIko");
        myDictionary.TextArea.setText("Least Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        myDictionary.CLEARButton.doClick();

        myDictionary.TextNewWord.setText("nIKO");
        myDictionary.TextArea.setText("So-so Fortunate entry fragger");
        myDictionary.ADDButton.doClick();
```

```java

        myDictionary.CLEARButton.doClick();

        myDictionary.TextNewWord.setText("ni"); // return niko
        myDictionary.FINDButton.doClick();

        if (myDictionary.TextFreqWord1.getText().equals("niko")) {
            System.out.println("FIND Test 1 passed");
        }

        // Valid FIND Test 2
        myDictionary = new Dictionary();

        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextArea.setText("Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        myDictionary.CLEARButton.doClick();

        myDictionary.TextNewWord.setText("nIko");
        myDictionary.TextArea.setText("Least Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        myDictionary.CLEARButton.doClick();

        myDictionary.TextNewWord.setText("nIKO");
        myDictionary.TextArea.setText("So-so Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        myDictionary.CLEARButton.doClick();

        for (int numi = 0; numi < 3; numi++) {
            // freq of "niko" is 3
            myDictionary.TextNewWord.setText("niko");
            myDictionary.FINDButton.doClick();
        }

        for (int numi = 0; numi < 4; numi++) {
            // freq of "nIko" is 4
            myDictionary.TextNewWord.setText("nIko");
            myDictionary.FINDButton.doClick();
        }

        for (int numi = 0; numi < 5; numi++) {
            // freq of "nIKO" is 5
            myDictionary.TextNewWord.setText("nIKO");
            myDictionary.FINDButton.doClick();
        }

        myDictionary.CLEARButton.doClick();
        myDictionary.CLEARButton.doClick();
        if(myDictionary.searchHistoryList.getModel().getElementAt(0).equals("nIKO")&&
myDictionary.searchHistoryList.getModel().getElementAt(1).equals("nIko")&&
myDictionary.searchHistoryList.getModel().getElementAt(2).equals("niko")){
            System.out.println("Search History Test passed");
        }
        else{
            System.out.println("Search History Test failed");
        }

        myDictionary.TextNewWord.setText("n");
        myDictionary.FINDButton.doClick();

        if (myDictionary.TextFreqWord1.getText().equals("nIKO")
                && myDictionary.TextFreqWord2.getText().equals("nIko")
                && myDictionary.TextFreqWord3.getText().equals("niko")) {
            System.out.println("FIND Test 2 passed");
        }

        // Test for REMOVE button
        // test for WordNotFoundError
        myDictionary = new Dictionary();

        myDictionary.TextNewWord.setText("NIKO");
        try {
            myDictionary.REMOVEButton.doClick();
        } catch (WordNotFoundError ex) {
            System.out.println("WordNotFoundError passed");
        }

        // Valid REMOVE
        myDictionary = new Dictionary();

        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextArea.setText("Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        System.out.println("Word niko added.");

        myDictionary.CLEARButton.doClick();
        myDictionary.TextNewWord.setText("niko");
        myDictionary.REMOVEButton.doClick();

        System.out.println("REMOVE Test passed");

        // Test for MODIFY Button
        myDictionary = new Dictionary();

        myDictionary.TextNewWord.setText("niko");
        myDictionary.TextArea.setText("Fortunate entry fragger");
        myDictionary.ADDButton.doClick();

        myDictionary.CLEARButton.doClick();

        myDictionary.TextOriginalWord.setText("niko");
        myDictionary.TextNewWord.setText("NIKO");
        myDictionary.MODIFYButton.doClick();

        myDictionary.CLEARButton.doClick();

        myDictionary.TextNewWord.setText("NIKO");
        myDictionary.FINDButton.doClick();

        if (myDictionary.TextFreqWord1.getText().equals("NIKO")) {
            System.out.println("MODIFY Button Test passed");
        }

        // Test for IMPORT and EXPORT Button
        myDictionary = new Dictionary();
        myDictionary.TextFilePath.setText("./src/input.txt"); // change your path to input.txt
        myDictionary.IMPORTButton.doClick();

        for (int numi = 0; numi < 3; numi++) {
            // freq of "niko" is 3
            myDictionary.TextNewWord.setText("niko");
            myDictionary.FINDButton.doClick();
        }

        for (int numi = 0; numi < 4; numi++) {
            // freq of "nIko" is 4
            myDictionary.TextNewWord.setText("nIko");
            myDictionary.FINDButton.doClick();
        }

        for (int numi = 0; numi < 5; numi++) {
            // freq of "nIKO" is 5
            myDictionary.TextNewWord.setText("nIKO");
            myDictionary.FINDButton.doClick();
        }

        myDictionary.TextFilePath.setText("./src/output.txt"); // change your path to output.txt
        myDictionary.EXPORTButton.doClick();
        // compare your output.txt with output_ref.txt
    }
}
```

```
}

```

Note:

- You could find `input.txt` and `output_ref.txt` mentioned in `SampleTest` on Canvas.
- To run the `SampleTest`, you may need to change all the attributes into `public` instead of `private`.

Submission

- During implementing this assignment:
  - **Please follow the steps in general description**

  A simple way to check if you are following the steps is to run the sample tests. If every sample test runs smoothly and gets passed, then you are fine.

  Fail to obey this rule may lead to reduction of your final score!

- You need to submit your hw on both Gradescope and Canvas.
  - You need to submit a JAR on Gradescope file for autograding.
  - After you complete this assignment, zip up this project folder into `Dictionary.zip`. After you generate the JavaDoc, also put a screenshot of the JavaDoc in the zip file. **Please do not submit `.rar` file**. Then upload it to Canvas under Assignment 4 submission link
- To build the JAR file, please follow:
  - Create an artifact configuration for the JAR
    - From the main menu, select **File | Project Structure** and click **Artifacts**.
    - Click +, point to **JAR** and select **From modules with dependencies**.
    - To the right of the Main Class field, click and select **Dictionary** in the dialog that opens. IntelliJ IDEA creates the artifact configuration and shows its settings in the right-hand part of the Project Structure dialog.
    - Apply the changes and close the dialog.
  - Build the JAR artifact
    - From the main menu, select **Build | Build Artifacts**.
    - Point to **Dictionary:jar** and select **Build**.
    - If you now look at the **out/artifacts** folder, you'll find your JAR there

Rubrics

This assignment will be graded from two aspects:

- Program logic tests (~70%)
  - test case based
  - Test cases will be in the similar format with `SampleTest`, but will be tested comprehensively. Please

    - Test cases will be in the similar format with `SampleTest`, but will be tested comprehensively. Please do not fully depend on `SampleTest`. It's your responsibility to understand the assignment description in detail, and implement the program with care consideration.
- GUI interaction tests (~30%)
    - We will manually run the GUIs and make manual tests.
    - This part will include:
        - GUI design(10%)
        - Exceptions(10%)
        - JavaDoc(5%)
        - Coding style(5%)