Criterion A

Word Count: 391

Scenario/Issue

The client is a businessman in China. As when on business trips, he often would visit predominantly English-speaking countries like UK and US, he is required to speak English to a proficient level. Currently the client can speak very simple English and form simple sentences. My client mentioned that he would like to improve his English vocabulary when meeting with clients, as he often reuses the same adjectives when describing objects, and hence we decided to discuss a solution (our conversations are in the Appendix).

Solution

My client discussed with me what he wanted me to achieve: a simple program that can run on his work laptop. In the program, he wanted to be able to input a Chinese word or phrase and wanted to be able to input the relevant English meaning to the side. He wanted to be able to view all the words/phrases he had inputted at any time and be able to save them. He also wanted to be able for the program to create a simple quiz / flashcards for him so he could learn the words at his leisure. Furthermore, he wanted to be able to use the program when not connected to the internet, as he often worked when travelling, and would like to incorporate key words when writing documents. Lastly, he wanted the program to be simple to use and not over-complicated.

Rationale

I decided to use Java to create the solution as Java can be run on both Windows and MacOS, meaning that there would be no restrictions on what type of work laptop my client must use. Furthermore, by styling the program, it would be made more aesthetically pleasing and simple to use for my client. I chose to use JavaFX GUI toolkit in lieu of Java Swing as Swing may be considered outdated and not as aesthetically pleasing.

My client stated that he had used other online quiz programs such as Quizlet and Memrise, however for those apps to download the flashcards offline, he needed to install the mobile apps and upgrade to a paid membership (research in Appendix). Therefore, he wanted to have a suitable alternative that fulfilled his needs and worked offline, creating the necessity for a custom application. The application will help the client to store inputted words, and create a quiz based on stored data.

Success Criteria

1. English words can be inputted in the application.
2. Chinese words can be inputted in the application.
3. Saved words can be viewed in table format.
4. Stored words can be saved and accessed outside the program (in a .csv file).
5. The program can generate quizzes.
6. The program can be used offline.
7. The program can generate flashcards.
8. English words can be searched when viewing the word list.
9. Warnings will be given when an error occurs.

Criterion B

Word Count: 143

Graphical Visualization – Proposed Design

Figure 1:

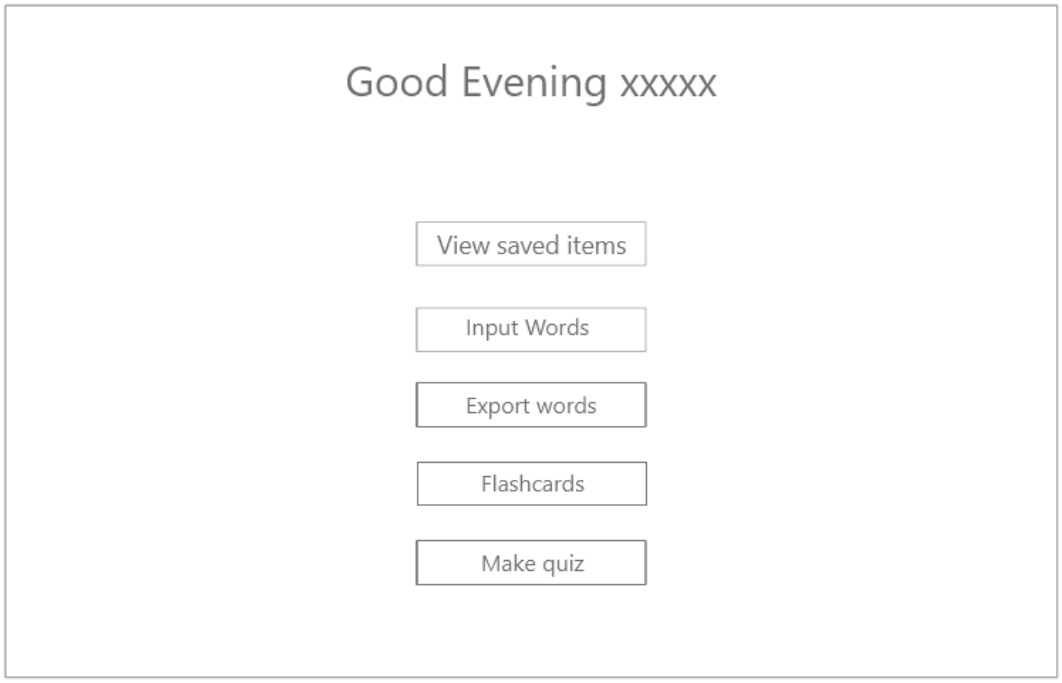


Figure 2:

Figure 3:

Figure 4:

Figure 5:

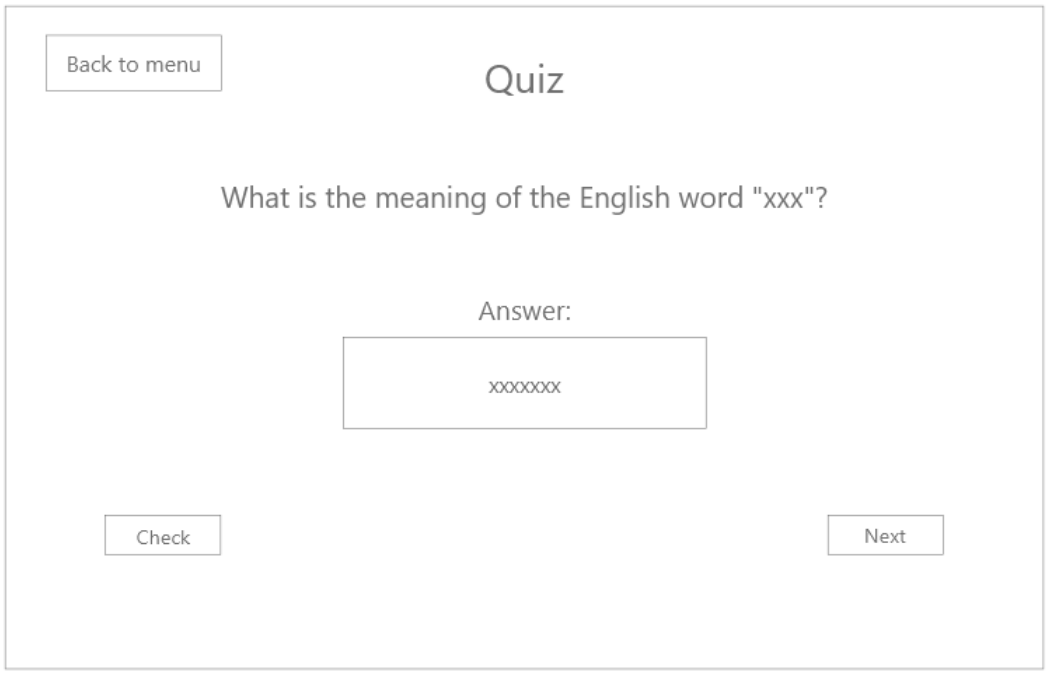

Back to menu
Quiz
What is the meaning of the English word "xxx"?
Answer:
xxxxxxx
Check
Next

Figure 6: **Hierarchal Chart**

This displays how the different windows and interfaces of the proposed program would be linked together.

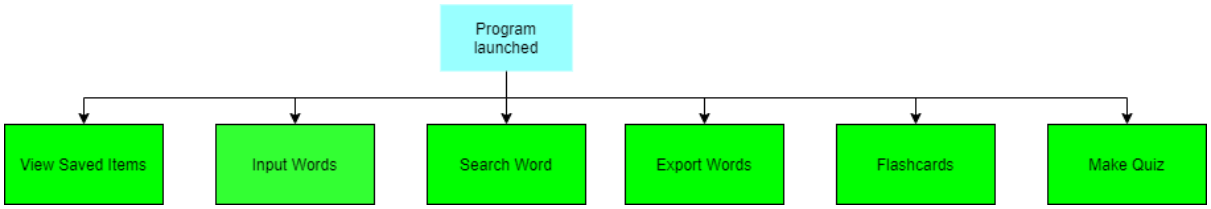


Figure 7: **Data Flow Diagram**

As multiple processes in the program would be writing and reading data, the data flow diagram highlights the process of data flowing through the system when the program is run. I did not show this to my client as this is merely showing the internal workings of the program.

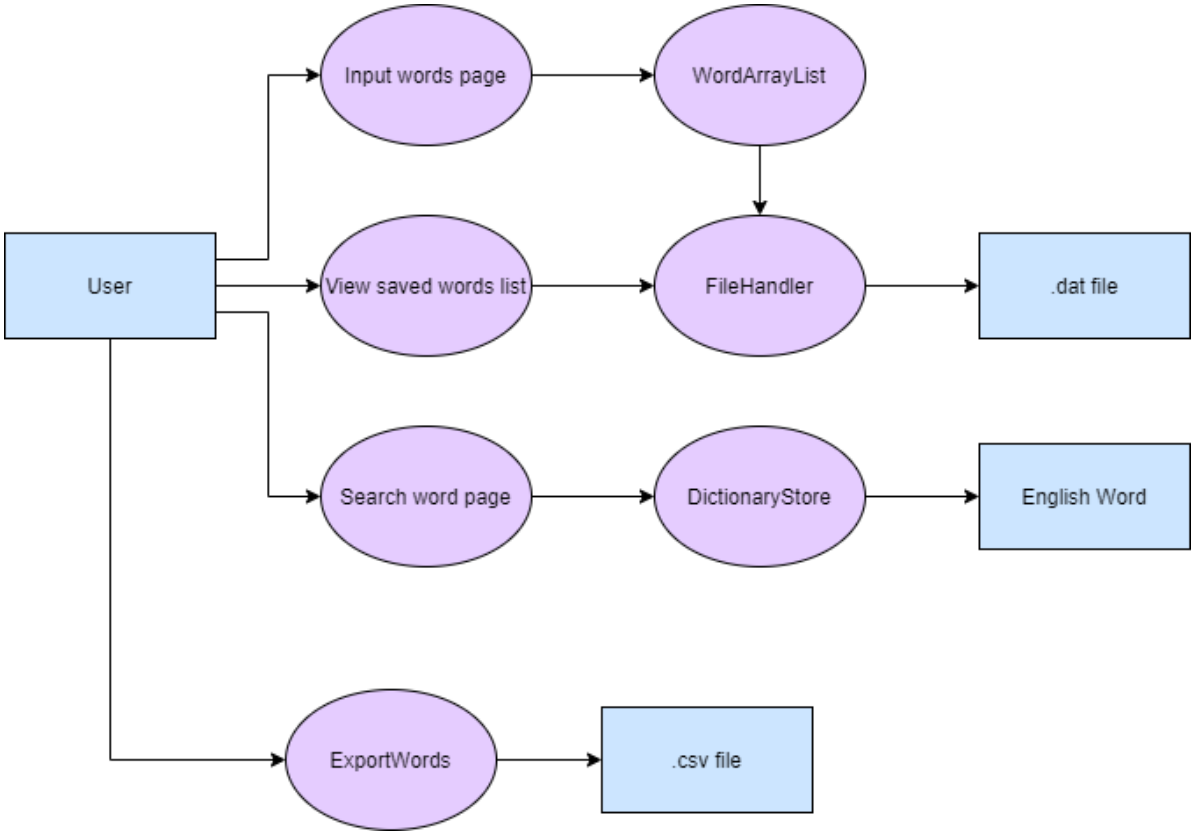


Figure 8: **Flowchart**

This flowchart highlights the processes that occur when a user is interacting with the program. It describes how different elements of the program work in tandem.

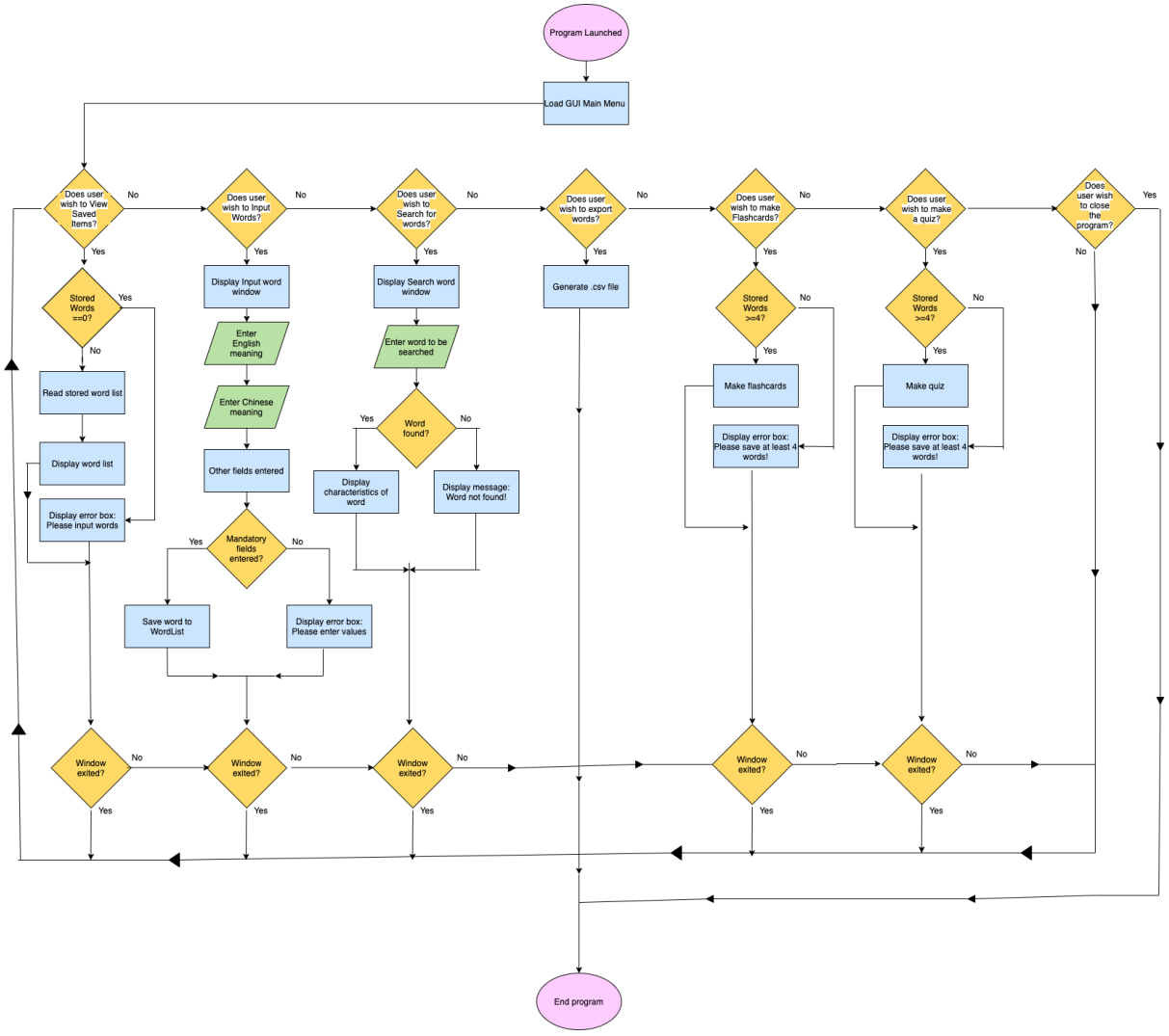
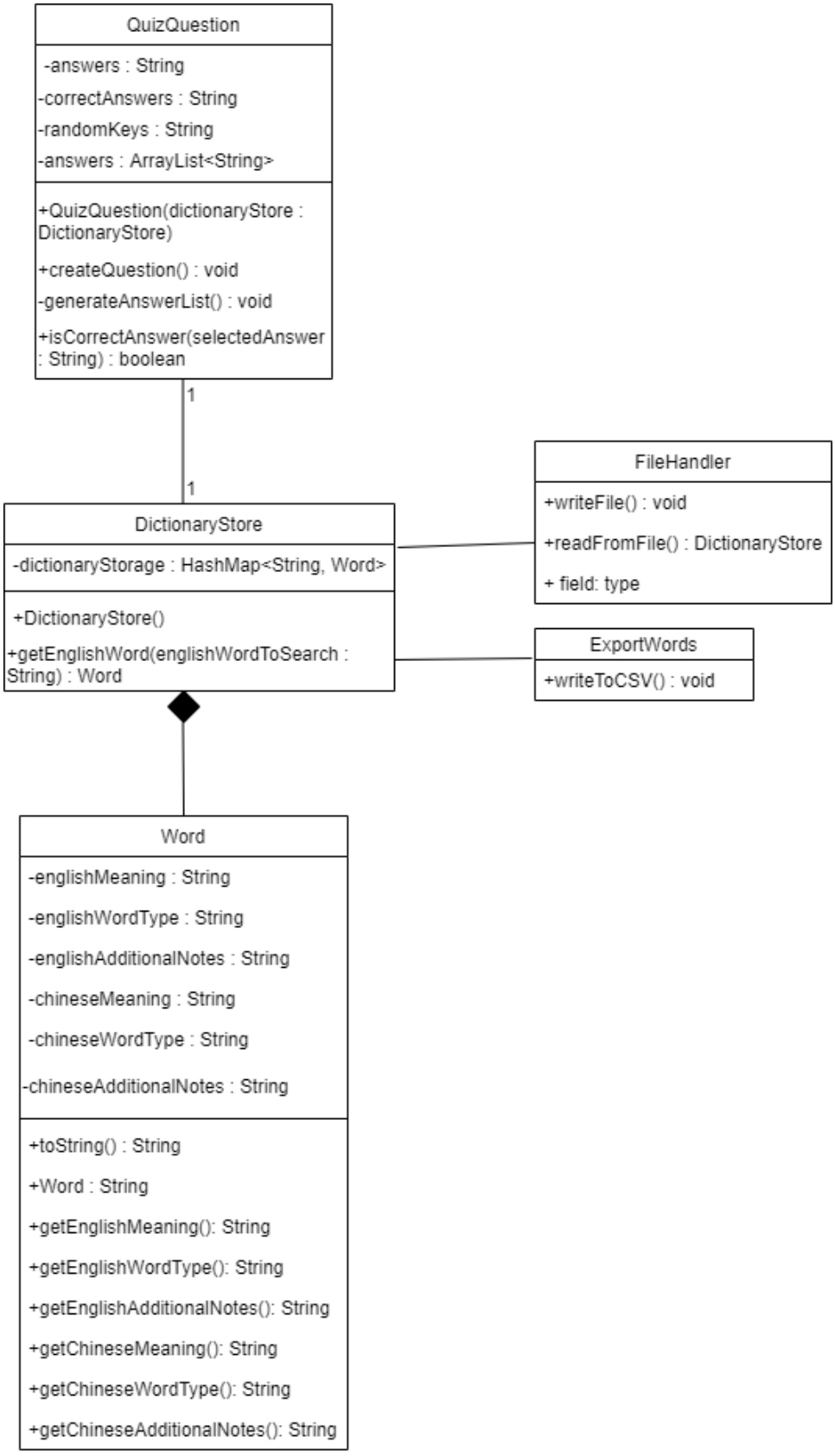


Figure 9: **UML Diagram**



This UML diagram showcases my current thought process regarding the number of classes and their functions in the program at this stage.

Test plan

Success criteria tested	Description of test	Test method	Expected outcome
1	Client should be able to input English words into the application	English word will be inputted into the program	Word can be saved and displayed in table format.
2	Client should be able to input Chinese words into the application	Chinese word will be inputted into the program	Word can be saved and displayed in table format
3	Saved words can be viewed by the client in table format	The saved words button is clicked, and output is compared to expected outcome	A table lists all saved English and Chinese words when the saved words button is clicked
4	Stored words can be exported into .csv format and viewed externally	The word list will be saved and opened in Excel to check if data is intact	The saved word list in .csv format can be opened in other programs
5	A quiz can be generated by the program	The generate quiz button will be clicked and output compared to expected outcome	A quiz with multiple questions is generated for the client to answer
6	The program can be used offline	The computer will be put into airplane mode to restrict online communication	The program works normally when offline
7	The program can generate flashcards	The flashcards button will be clicked and output compared to expected outcome	Flashcards are generated using Chinese-English word pairs
8	Words can be searched	Text will be inputted into the search bar in the word list window	The searched word will appear in the list
9	Warnings will be given when an error occurs	Intentionally enter wrong values (for example integers in lieu of words)	An error box will occur and instruct client to input the right information

Criterion B: Record of Tasks

Task Number	Planned action	Planned outcome	Time estimated	Target completion date	Criterion
1	Initial contact with client	Propose the solution to the client and gain their approval.	30 minutes	02/11/2020	A / PLAN
2	Meeting with the client	Discuss expectations for the product and basic requirements / success criteria.	1 hour	04/11/2020	A / PLAN
3	Discussing idea with supervisor	To gauge if the planned solution is viable.	30 minutes	09/11/2020	A / PLAN
4	Create basic designs for client	Create designs that the client can easily understand.	3 hours	13/11/2020	B / DESIGN
5	Second contact with client	Discuss contrived mock diagrams.	1 hour	14/11/2020	B / DESIGN
6	Make additional diagrams such as flowcharts and show client	To illustrate how the user would experience the program.	4 hours	17/11/2020	B, C / DESIGN
7	Basic GUI design	Complete basic GUI design of intended windows in the program.	4 hours	2/12/2020	B, C / DESIGN, DEVELOP
8	Additional GUI design	Insert additional GUI elements (such as buttonClick events).	4 hours	5/12/2020	B, C / DESIGN, DEVELOP
9	Code basic Controller	Code responses for	7 hours	15/12/2020	C / DEVELOP

	events for GUI	buttonClicks and textInputs.			
10	Code additional Controller events for GUI	Code if, for and while statements.	9 hours	18/12/2020	C / DEVELOP
11	Code FileHandler class to handle file read/write	Code method to read and write files.	5 hours	28/12/2020	C / DEVELOP
12	Test .dat file reading and writing	Test saving and writing process of .dat files for word storage.	3 hours	04/01/2021	C / DEVELOP, TEST
13	Code search algorithm and HashMap function	Code search functionality and implement HashMap dynamic word pairing data structure.	5 hours	06/01/2021	C / DEVELOP
14	Code Flashcards feature	Code Flashcard features, such as retrieving words, revealing words. Test that it works.	5 hours	08/01/2021	C / DEVELOP, TEST
15	Code Quiz feature	Code quiz function, such as multiple choice available and algorithm to compare to correct answer. Test that it works.	6 hours	18/01/2021	C / DEVELOP, TEST
16	Check success criteria and receive feedback from client	Receive any needed changes from the client.	2 hours	19/01/2021	A, C, E / PLAN, DESIGN, IMPLEMENT

17	Additional coding / fixes	Respond to client feedback, test the program.	6 hours	28/01/2021	A, C, E / DEVELOP, TEST
18	Final application given to client for testing	Client uses program to test for any additional errors / problems,	3 hours	30/01/2021	A, C, E / TEST, IMPLEMENT
19	Any additional problems resolved	Any new problems are fixed, and program tested again.	3 hours	04/02/2021	A, C, E / DEVELOP, TEST, IMPLEMENT
20	Evaluate the project	Identify what went well and what did not during the project.	2 hours	05/02/2021	E / PLAN, IMPLEMENT
21	Discuss future development	Discuss future improvements with the client.	30 minutes	05/02/2021	E / PLAN, IMPLEMENT

Criterion C

Word Count: 1107

Contents

1. IDE and Java Libraries used

To create the product, IntelliJ IDE was used.

```
import java.net.URL;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.collections.transformation.FilteredList;
import javafx.collections.transformation.SortedList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
```

Furthermore, various GUI libraries available with JavaFX were imported into the program for use, allowing me to use the resources available to code with. That meant that the program would be simpler to use for the user, fulfilling the goal in Criterion A. These libraries allowed me to design much of the GUI using Scene Builder and provided me with some templates, which immensely helped with designing the program.

2. Word list inputting and presentation

ArrayLists

```
public void initialize(URL url, ResourceBundle resourceBundle) {

    ArrayList<Word> wordArrayList=updateTable();
    viewAllWordsPane.setStyle("-fx-background-color: #F5F5DC");

    ObservableList<Word> data = FXCollections.observableArrayList(wordArrayList);
    englishMeaningTab.setCellValueFactory(new PropertyValueFactory<Word,String>( s: "englishMeaning"));
    englishTypeTab.setCellValueFactory(new PropertyValueFactory<Word,String>( s: "englishWordType"));
    englishNotesTab.setCellValueFactory(new PropertyValueFactory<Word,String>( s: "englishAdditionalNotes"));
    chineseMeaningTab.setCellValueFactory(new PropertyValueFactory<Word,String>( s: "chineseMeaning"));
    chineseTypeTab.setCellValueFactory(new PropertyValueFactory<Word,String>( s: "chineseWordType"));
    chineseNotesTab.setCellValueFactory(new PropertyValueFactory<Word,String>( s: "chineseAdditionalNotes"));
    tabelView.setItems(data);


}
```

Figure 1: Example of ArrayList used in the program

In the program, I utilised an ArrayList to store the different words and values that the user was presented with when inputting an English word and its corresponding Chinese meaning. ArrayLists were used in lieu of other methods such as LinkedLists as an ArrayList is a dynamically resizing array, and has an efficiency of O(1) when accessing stored data, whilst also using less memory than its counterparts. This means that a the time to access data remains constant regardless of set size,

which ensures efficient access to data. Furthermore, usage of ArrayLists make it easy for the data to be stored in a dedicated setting, making future additions to the list straightforward. In the ArrayList, different values would be stored, for example both the English and Chinese meaning and any additional user notes.

TableView

```
public ArrayList<Word> updateTable(){
    ArrayList<Word> wordArrayListTemp=new ArrayList<~>();
    //get hash map keys arrayList
    ArrayList<String> keysArrayList;
    keysArrayList= QuizQuestion.readAllKeys(MainWindowController.getDictionary());

    //sort by alphabetically
    Collections.sort(keysArrayList);

    this.elementListInListView=new ArrayList<String>();

    //update to list view
    for(int i=0;i<keysArrayList.size();i++){
        Word word=MainWindowController.getDictionary().getDictionaryStorageByEnglishKey().get(keysArrayList.get(i));
        wordArrayListTemp.add(word);

    }

    return wordArrayListTemp;
}
}
```

Figure 2: Showing the process where the tableView is updated

The tableView is updated by using an ArrayList to read all the keys generated by the HashMap function. After that, the keys would then be sorted alphabetically using Java built-in Collections function. Finally, the list view in the tableView would be updated accordingly by referring to objects in other classes. A for loop is used to ensure that every word in the keysArrayList is updated to the list view.

If else statements

```
@FXML
public void search() {
    //we take user input and search for it's Chinese word if exists in our dictionary
    String searchWord=englishMeaningText.getText();

    Word selectedWord=MainWindowController.getDictionary().getByEnglishWord(searchWord);

    if(selectedWord!=null){
        chineseMeaningText1.setText(selectedWord.getChineseMeaning());
        chineseTypeText11.setText(selectedWord.getChineseWordType());
        chineseNotesText11.setText(selectedWord.getChineseAdditionalNotes());
    }
    else {
        chineseMeaningText1.setText("Not in Dictionary! ");
    }

}
```

Figure 3: If else statements

A search algorithm is used to take user input in English and search for its corresponding Chinese definition. The englishMeaningText is an user-input field, and the searchWord string would be derived from that. If the corresponding word is in the saved word list, then the “Meaning”, “Type of word” and “Additional notes” boxes would be filled with the correct information. If not, then an error “Not in Dictionary!” would occur. This is visualized below:

Figure 4: When the searched word is not in the word list

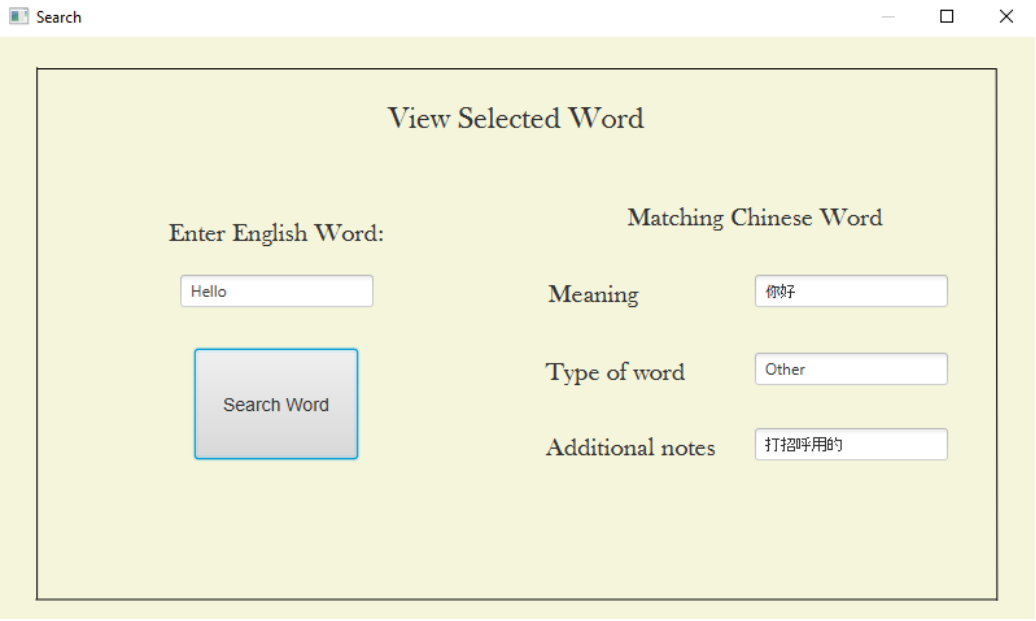


Figure 5: When the searched word is present in the word list

3. GUI Elements

```
public AnchorPane makeQuizGUIPane;
public Button nextQuestion;
public Button previousQuestion;
public RadioButton answerA;
public RadioButton answerB;
public RadioButton answerC;
public RadioButton answerD;
public Button back;
public TextArea questionWord;
public Label marksLabel;
private int questionNumber;
private MakeQuiz makeQuiz;
```

Figure 6: Buttons used in the Quiz window

In the various windows in the program, I implemented different types of buttons and textAreas to facilitate user input in the form of typing text, selecting from multiple choice questions or from drop-down boxes. This is a form of abstraction, as users are presented with buttons to click and textAreas to type in, rather than an algorithm to input values into. Therefore, unwanted details are hidden from users and only essential information is shown, with the processing happening in the background.

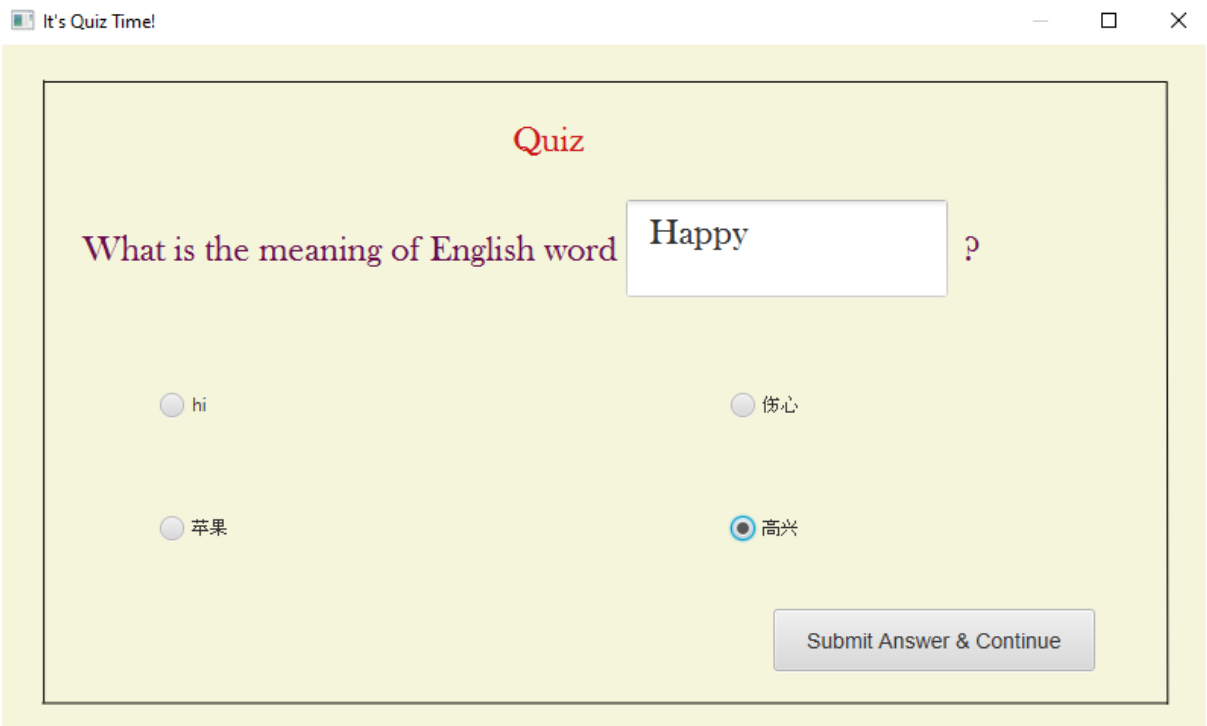


Figure 7: Example of Abstraction

4. Flashcards

```
public class MakeFlashCards implements Serializable {
    private int numberOfQuestions;
    private ArrayList<QuizQuestion> flashCardQuestionArrayList;
    private DictionaryStore dictionaryStore;

    public MakeFlashCards(int numberOfQuestions, DictionaryStore dictionaryStore) {

        if(dictionaryStore.getDictionaryStorageByEnglishKey().size()>=(numberOfQuestions+1)){
            this.numberOfQuestions = numberOfQuestions+1;
        }
        else {
            this.numberOfQuestions=dictionaryStore.getDictionaryStorageByEnglishKey().size();
        }

        this.dictionaryStore = dictionaryStore;
        this.flashCardQuestionArrayList = new ArrayList<QuizQuestion>();

        //make flash cards if there is at least 4 words in dictionary
        if(this.dictionaryStore.getDictionaryStorageByEnglishKey().size()>=4) {
            this.createFlash();
        }
        else {
            System.out.println("Not Enough Words");
        }
    }

    private void createFlash(){
        int i=0;
        while(i<this.numberOfQuestions){
            QuizQuestion quizQuestion=new QuizQuestion(this.dictionaryStore);

            if(isUniqueQuestion(quizQuestion)){
                this.flashCardQuestionArrayList.add(quizQuestion);
                i++;
            }
        }
    }
```

Figure 8: Flashcard-making process

Flash cards are made in the program if there are four or more words present in the word list. An ArrayList is called QuizQuestion is used to store the information regarding words for the program to interpret. An error is coded in the form of a println statement to inform me if there are insufficient words when testing the program. A while loop is used to iterate through the createFlash method to check if there are words in the word list.

For loop

```
//makes sure that flashcards are not duplicated
private boolean isUniqueQuestion(QuizQuestion quizQuestion){
    for(int i = 0; i<this.flashCardQuestionArrayList.size(); i++){
        if(this.flashCardQuestionArrayList.get(i).getQuestionWord().equals(quizQuestion.getQuestionWord())){
            return false;
        }
    }
    return true;
}
```

Figure 9: for loop to determine whether a question is unique or not

A for loop is utilised to ensure that the flashcards given to the user are not duplicated, but rather unique until a certain value.

Quiz question

```
public void createQuestion(){
    //we choose random key no 1 as for our question index
    this.QuestionWord=this.dictionaryStore.getDictionaryStorageByEnglishKey().get(randomKey1).getEnglishMeaning();

    this.questionWordWord=this.dictionaryStore.getDictionaryStorageByEnglishKey().get(randomKey1);
    this.correctAnswer=this.dictionaryStore.getDictionaryStorageByEnglishKey().get(randomKey1).getChineseMeaning();
    this.readAllAnswers();
    this.generateAnswerList(); //assign random answers including the correct answer for choices A,B,C & D
}

//this makes sure that the inputted answer is correct by comparing
public boolean isCorrectAnswer(String selectedAnswer){
    if(selectedAnswer.equals(this.correctAnswer)){
        return true;
    }
    else return false;
}
```

Figure 10: How a quiz question is created

To create a question option for the quiz part of the program, a random key is generated from the keysArrayList, then the inputted option between the multiple choices is compared with the correct answer by using an if statement and a Boolean.

Unique Numbers

```
//to get unique random numbers in a range we define
public ArrayList<Integer> UniqueRandomNumbers(int from,int to,int noOfNumbers) {
    ArrayList<Integer> uniqueRandomNumbers=new ArrayList<~>();

        ArrayList<Integer> list = new ArrayList<~>();
        for (int i=from; i<=to; i++) {
            list.add(i);
        }
        Collections.shuffle(list);
        for (int i=0; i<noOfNumbers; i++) {
            uniqueRandomNumbers.add(list.get(i));
        }
    return uniqueRandomNumbers;
}
```

Figure 11: How the unique numbers are generated

This shows how the numbers in the defined range are shuffled using the Java built-in Collections function. The UniqueRandomNumbers ArrayList would then be referenced later in the code to ensure that no questions are repeated.

5. File Handling

File reading

```
public class FileHandler {

    public static void writeFile(){
        Object obj= MainWindowController.getDictionary();
        FileOutputStream f = null;
        try {
            f = new FileOutputStream(new File( pathname: "dictionary.dat"));

        ObjectOutputStream o = new ObjectOutputStream(f);

        // Write objects to file
        o.writeObject(obj);

        o.close();
        f.close();
        } catch (FileNotFoundException e) {
            System.out.println("");
        } catch (IOException e) {
            System.out.println("");
        }
    }
```

Figure 12: Process of reading files

To utilise the HashMap that stores the pairs of English and Chinese words, the program must be able to write to an local file to save their values. Therefore, the FileHandler class creates a new file in the local folder named “dictionary.dat” and proceeds to insert values. When reading the data the word pairs would be converted into keys and stored. The ability to access and alter the .dat save file is paramount for the application, as it allows for words entered to be saved across multiple launches of the program without data loss. Therefore, the data contained within the file is able to be updated and edited.

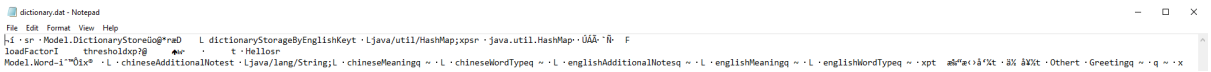


Figure 13: How the dictionary.dat files looks when storing the word pairs of “Hello” and “你好“

Exporting to .csv

```
public class ExportWords implements Serializable {



    public static void writeToCSV() {

        //this sorts keyWordsArrayList alphabetically
        ArrayList<String> keyWordsArrayList=QuizQuestion.readAllKeys(MainWindowController.getDictionary());
        Collections.sort(keyWordsArrayList);



            PrintWriter pw = null;
            try {
                pw = new PrintWriter(new File( pathname: "AllWords.csv"));
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            StringBuilder builder = new StringBuilder();
            String columnNamesList = "English Meaning,English Word Type,English Notes,Chinese Meaning,Chinese Word Type,Chinese Notes";

            builder.append(columnNamesList +"\n");

            viewAllWordsController viewAllWordsController1=new viewAllWordsController();

        for ( int i=0;i<keyWordsArrayList.size();i++) {
            Word currentWord=MainWindowController.getDictionary().getDictionaryStorageByEnglishKey().get(keyWordsArrayList.get(i));


                builder.append(currentWord.getEnglishMeaning()+",");
                builder.append(currentWord.getEnglishWordType()+",");
                builder.append(currentWord.getEnglishAdditionalNotes()+",");
                builder.append(currentWord.getChineseMeaning()+",");
                builder.append(currentWord.getChineseWordType()+",");
                builder.append(currentWord.getChineseAdditionalNotes());
                 builder.append('\n');

        }

                pw.write(builder.toString());




            pw.close();
            System.out.println("done!");
        }


}
```

Figure 14: How words are exported into a .csv file

When trying to export the word list, the user can choose the export all words to a .csv file, which would be called “AllWords.csv”. The keyWordsArrayList would be sorted using the Java Collections function.

6. Word List Data Structure

```
public class DictionaryStore implements Serializable {
    private HashMap<String,Word> dictionaryStorageByEnglishKey;

    public DictionaryStore() { this.dictionaryStorageByEnglishKey = new HashMap<String,Word>(); }

    public HashMap<String, Word> getDictionaryStorageByEnglishKey() { return dictionaryStorageByEnglishKey; }

    //this pairs the English word to its meaning
    public Word getByEnglishWord(String englishWordToSearch){
        return this.dictionaryStorageByEnglishKey.get(englishWordToSearch);
    }


}
```

Figure 15: HashMap creation

A HashMap is created to store the String and Word pairs, effectively pairing the English word to its meaning. Therefore, the searching algorithm would be able to search for an English word and its matching pair. A HashMap would convert each pair into a key, the String of the word, and the object of the word is stored in the dictionary entry of the HashMap.

7. Encapsulation (OOP)

```
public class Word implements Serializable {
    private String englishMeaning;
    private String englishWordType;
    private String englishAdditionalNotes;
    private String chineseMeaning;
    private String chineseWordType;
    private String chineseAdditionalNotes;

    public Word(String englishWord, String englishWordType, String englishAdditionalNotes, String chineseMeaning, String chineseWordType, String chineseAdditionalNotes) {
        this.englishMeaning = englishWord;
        this.englishWordType = englishWordType;
        this.englishAdditionalNotes = englishAdditionalNotes;
        this.chineseMeaning = chineseMeaning;
        this.chineseWordType = chineseWordType;
        this.chineseAdditionalNotes = chineseAdditionalNotes;
    }

    public String getEnglishMeaning() { return englishMeaning; }

    public String getEnglishWordType() { return englishWordType; }

    public String getEnglishAdditionalNotes() { return englishAdditionalNotes; }

    public String getChineseMeaning() { return chineseMeaning; }

    public String getChineseWordType() { return chineseWordType; }

    public String getChineseAdditionalNotes() { return chineseAdditionalNotes; }
```

Figure 16: Word class utilising encapsulation (OOP)

Throughout the program, encapsulation is used to ensure that variables created cannot be inadvertently access directly from another class. This is achieved by utilising getters and setters for variables, increasing usability and rendering the program extensible. Furthermore, due to encapsulation, the internal data of objects would be protected. Other objects would be unable to inadvertently change the internal state of encapsulated objects.

8. Error Handling

```
@FXML
public void inputWord(){
    boolean flag=true;
    if(!textAreaNotEmpty(englishMeaningText)){
        Alert errorAlert = new Alert(Alert.AlertType.ERROR);
        errorAlert.setHeaderText("Error Input");
        errorAlert.setContentText("Please Enter English Meaning!");
        errorAlert.showAndWait();
        flag=false;
    }
    else if(englishTypeSelection.getSelectionModel().isEmpty()){
        Alert errorAlert = new Alert(Alert.AlertType.ERROR);
        errorAlert.setHeaderText("Error Input");
        errorAlert.setContentText("Please Enter English Word Type!");
        errorAlert.showAndWait();
        flag=false;
    }
    else if(!textAreaNotEmpty(chineseMeaningText1)){
        Alert errorAlert = new Alert(Alert.AlertType.ERROR);
        errorAlert.setHeaderText("Error Input");
        errorAlert.setContentText("Please Enter Chinese Word Meaning!");
        errorAlert.showAndWait();
        flag=false;
    }
    else if(chineseTypeSelection.getSelectionModel().isEmpty()){
        Alert errorAlert = new Alert(Alert.AlertType.ERROR);
        errorAlert.setHeaderText("Error Input");
        errorAlert.setContentText("Please Enter Chinese Word Type!");
        errorAlert.showAndWait();
        flag=false;
    }
```

Figure 16: Error handling

To inform the user if an error has occurred due to missing input, I used JavaFX to include error boxes that would occur to inform the user how to fix the error.

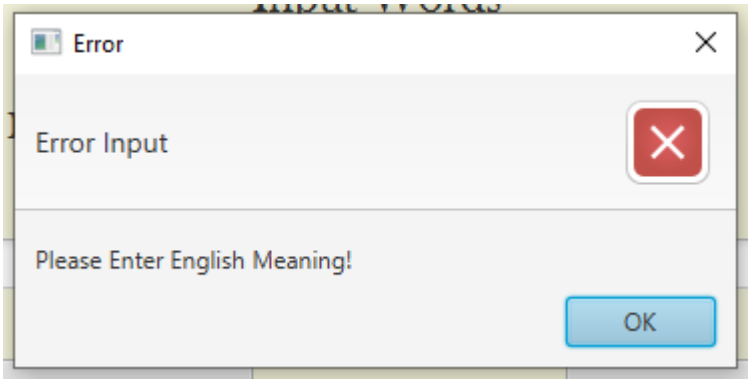


Figure 17: Error box example

9. References

Baimagambetov, Almas. “Java Serialization (Save/Load data).” *YouTube*, 7 Jan. 2015, www.youtube.com/watch?v=-xW0pBZqpjU.

Begin Coding Fast. “Create a multiple-choice test - Java Example.” *YouTube*, 16 Oct. 2019, www.youtube.com/watch?v=28kAWFjaugY.

Cool IT Help. “JavaFX - Adding Data in TableView (Using FXML) | Populating Rows in TableView.” *YouTube*, 9 Oct. 2017, www.youtube.com/watch?v=4RNhPZJ84P0

Feel Free To Code. “#59. Showing Random Questions | Javafx Quiz Applications.” *YouTube*, 27 Apr. 2020, www.youtube.com/watch?v=aQYfWVuWA4Q.

Java HashMap. javatpoint.com, www.javatpoint.com/java-hashmap.

Jenkov, Jakob. “JavaFX TableView.” jenkov.com, 7 Jan. 2021, tutorials.jenkov.com/javafx/tableview.html.

Lee, Alex. “HashMap Java Tutorial.” *YouTube*, 24 May 2019, www.youtube.com/watch?v=70qy6_gw1Hc

O’Didily, Max. “Writing to a CSV file in Java.” *YouTube*, 2 May 2015, www.youtube.com/watch?v=lp0xQXUEw-k.

thenewboston. “JavaFX GUI Tutorials.” *YouTube*, 4 Mar. 2015, www.youtube.com/watch?v=FLkOX4Eez6o&list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG.

Criterion E

Word Count: 350

Evaluation of the program

Success Criteria	Fulfilled?	Feedback from client
English words can be inputted in the application.	Yes	English words can be entered into the program without any problem
Chinese words can be inputted in the application.	Yes	Chinese words can be entered into the program without any problem
Saved words can be viewed in table format.	Yes	There is a dedicated table list page for the client to read
Stored words can be saved and accessed outside the program (in a .csv file).	Yes	The word list can be exported as a .csv file
The program can generate quizzes.	Yes	The program can generate quizzes
The program can be used offline.	Yes	Works
The program can generate flashcards.	Yes	The program can generate flashcards
English words can be searched when viewing the word list.	Yes	Different English words can be searched for

Extending the Application

The client discussed with me how I could further improve the product (our conversation is in the Appendix).

1. Using a database

The client indicated his wish to be able to utilise a database to access the word list whilst he had internet access. This meant that the word list could be continually updated, and he would have quick access to information, further negating the risk that locally stored .csv file would be inadvertently deleted or corrupted. The locally stored .csv file would be updated against the online database if an internet connection is present.

2. Ability to filter tables

Currently in the table view, the client is only able to view all saved words in a pre-selected sorted fashion and use the search bar to search for individual words. However, in the future he has expressed that he would like to be able to filter the word lists by other avenues, for example sorting by word type instead of alphabetically. This would allow the client to further manipulate the data to his advantage and preference, adding to convenience.

3. Ability to incorporate multiple users

The client has expressed that he would like to be able to share the program with different colleagues for their use if necessary. Therefore, he would like the ability to have multiple user profiles, each with their own .csv file able to be saved offline, and in a database. This would allow multiple people to use the program at the same time and have their word list individually stored.

4. Ability to delete words from the list

The client has expressed that he would like to be able to delete individual words from the saved word list, as currently there is no way to do so. Therefore, the client has to be extra careful when saving words into the program, as a typo would necessitate complete deletion of the dictionary.dat file, which contains all the saved word pairs. This would mean that deleting the .dat file would mean that all saved words are wiped from the system, which poses a major inconvenience.

Appendix

The client and I conversed mainly in Chinese, hence our conversations are translated.

First meeting with the client

Me (in Chinese): Hi {Client}, how are you? You mentioned to me recently that you go on a lot of business trips to the UK, and would like to have a way to store English keywords?

Client (translated from Chinese): Yes {Me}. I would like to have a way to note down English key words when I am travelling, I don't want to pay for programs though… it would be great if you could help me create an application! I know you have experience already from before. Could you make it so I can have quizzes and flashcards in the program, and also want to view all the saved words. Thank you {Me}!

Me (in Chinese): Yes sure, I can help you create a program. Could I contact you a few days later after I think about how to implement what you have asked?

Client (translated from Chinese): Yes sure, I will wait for you to contact me then.

Second meeting with client

Me (in Chinese): Hi {Client}, how are you? I have thought about our program and have come up with some criteria for it. However it's all in English so I can translate it into Chinese for you if you want.

Client: Yes sure. Please translate it for me so we don't take too long.

Me (in Chinese): Here are the success criteria

1. English words can be inputted in the application.
2. Chinese words can be inputted in the application.
3. Saved words can be viewed in table format.
4. Stored words can be saved and accessed outside the program (in a .csv file).
5. The program can generate quizzes.
6. The program can be used offline.
7. The program can generate flashcards.
8. English words can be searched when viewing the word list.
9. Warnings will be given when an error occurs.

Does that sound ok to you, anything else you'd like to add?

Client (in Chinese): That's fine, please make the program look nice too. But not too complicated. I don't want to get bored when using it! Also thank you for offering to develop this, I didn't want to pay for apps like Memrise or Quizlet.

Alternative Research

As my client mentioned, online flashcard/quiz applications such as Memrise and Quizlet require a paid subscription to be able to be used offline. Therefore, the proposed solution's ability to be used offline would make the use case scenarios of the program broader.

Memrise: Learn a language. Meet the world. | Memrise

Quizlet: Upgrade your account | Quizlet

Third meeting with client

Client (in Chinese): Hi {Me}, because of COVID-19 situation you can take your time with developing the app, since I cannot travel anyways. How is the progress so far?

Me (in Chinese): It is almost done. I have created a prototype. Would you like to test it?

Client (in Chinese): Yes sure. Send the program over to me now please.

Me (in Chinese): There are options in the main menu where you can click. I did not put a lot of graphics in the program as you said you want it to be simple, so I only changed the colours of the background.

Client (in Chinese): Ok, demonstrate the other functions then. It looks fine.

(I open the input words window)

Me (in Chinese): In here you can input different English words and Chinese words. Most of the text is in English, but you can understand simple words right?

Client (in Chinese): Yes I can. That's fine.

(Demonstrating entering words and saving)

Me (in Chinese): Ok, that's how you enter it. Now you try.

(Client tries)

Me (in Chinese): Ok, so you can now cross off the window and go back to the main menu. To view the saved words click the View Saved items list. This screen is temporary, I plan to change the display to a table later. Like in Excel.

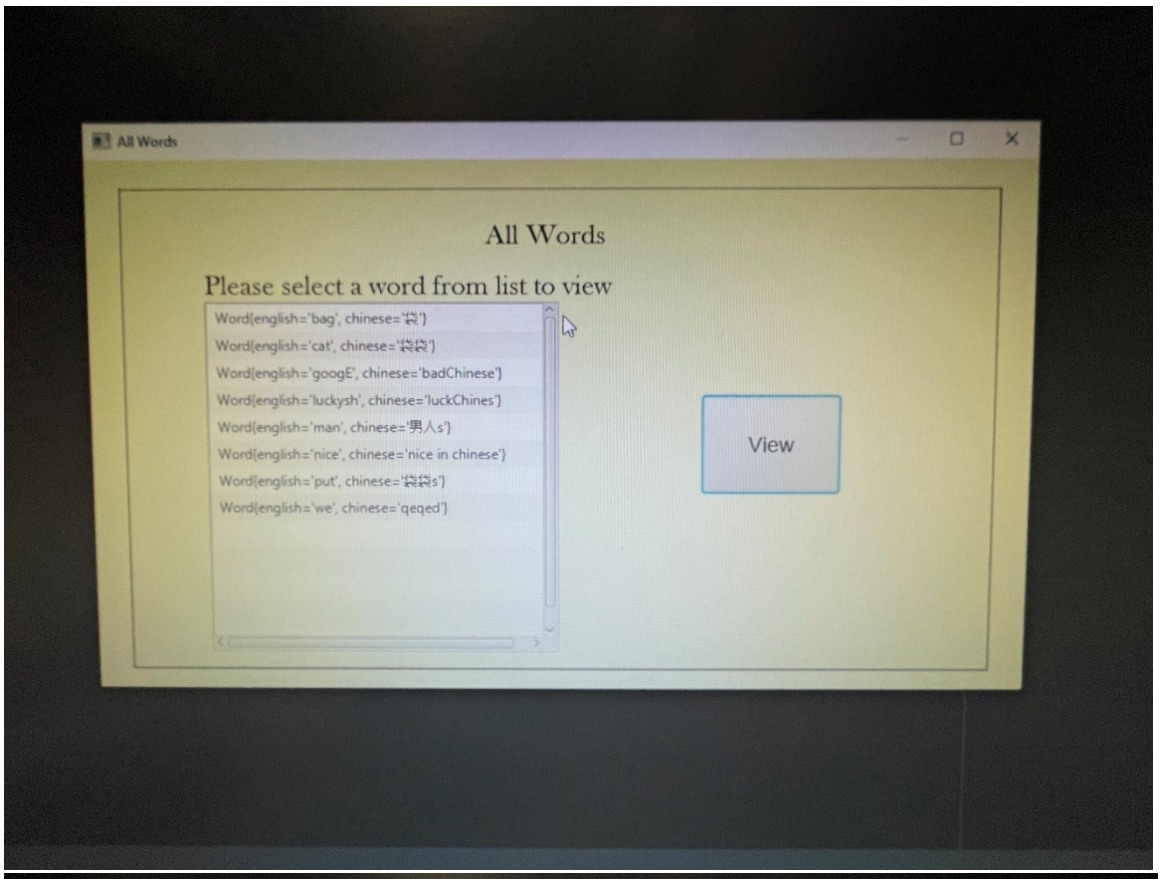
Client (in Chinese): Ok, I understand. Yes, you should change it. This seems very complicated to use. Need to click each word to see the word type.

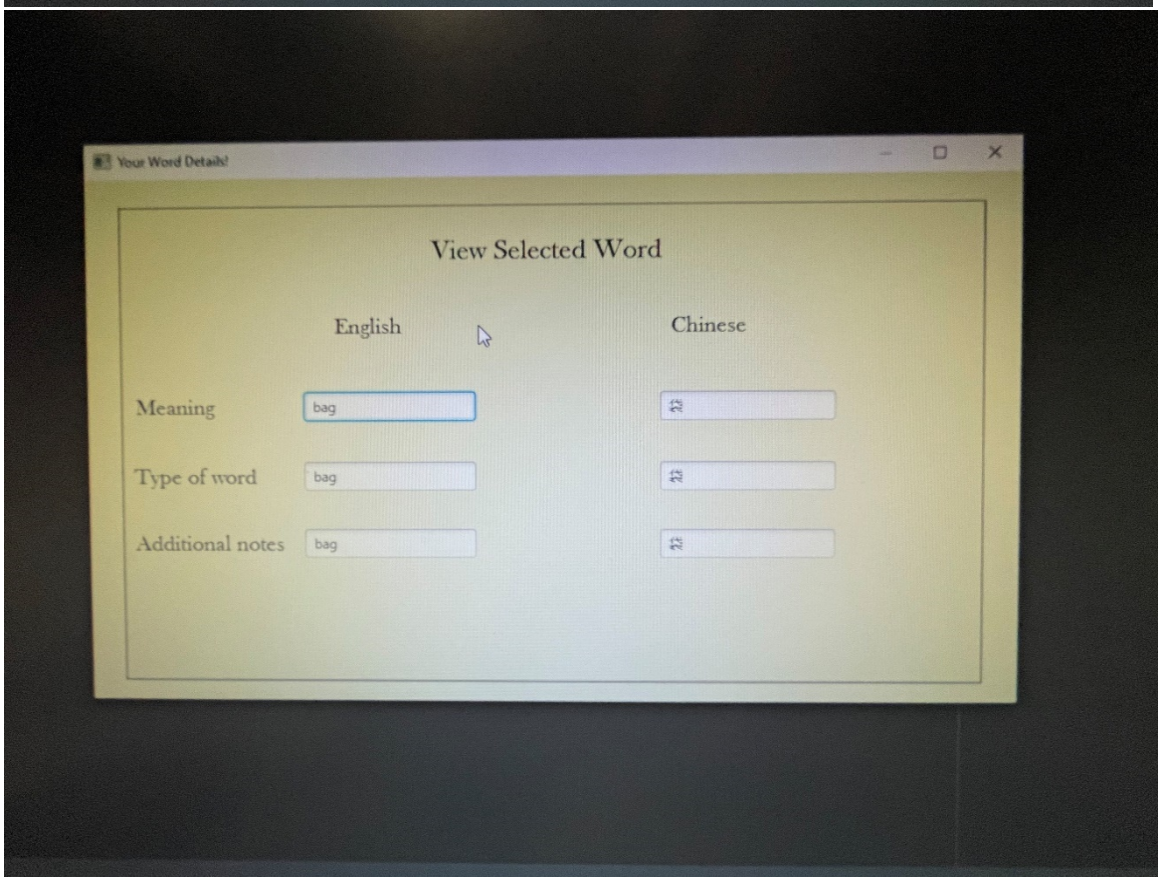
Me (in Chinese): Yes I know, I still need to find out how to make Table view in the programming code. I will fix it though, don't worry.

Client (in Chinese): Ok, I trust you. When it is almost done contact me again. Thanks.

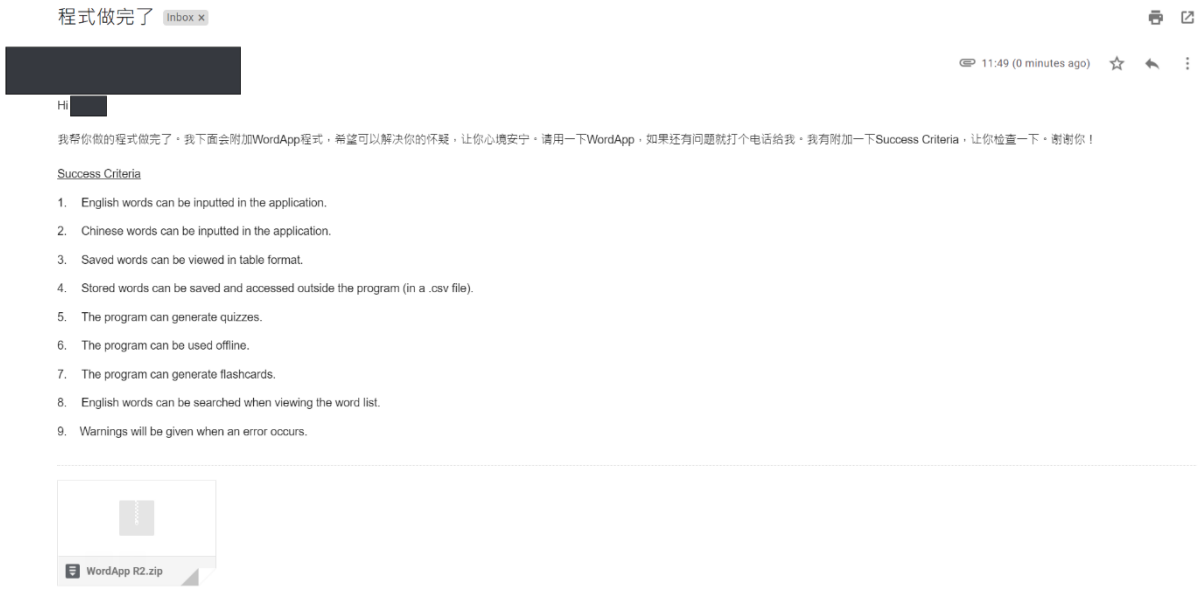
Client testing program

The client testing the program, specifically the first version of the view saved words window.





Email exchange with client



程式做完了 Inbox ×

[redacted]

11:49 (0 minutes ago)

Hi [redacted]

我帮你做的程式做完了。我下面会附加WordApp程式，希望可以解决你的怀疑，让你心境安宁。请用一下WordApp，如果还有问题就打个电话给我。我有附加一下Success Criteria，让你检查一下。谢谢你！

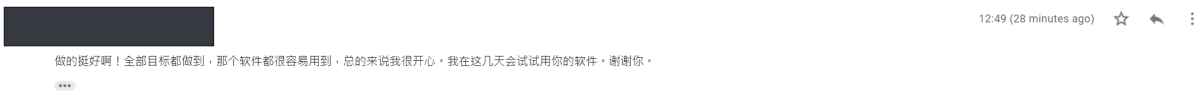
Success Criteria

1. English words can be inputted in the application.
2. Chinese words can be inputted in the application.
3. Saved words can be viewed in table format.
4. Stored words can be saved and accessed outside the program (in a .csv file).
5. The program can generate quizzes.
6. The program can be used offline.
7. The program can generate flashcards.
8. English words can be searched when viewing the word list.
9. Warnings will be given when an error occurs.

WordApp R2.zip

Translation:

Me: I have finished the program and will attach the WordApp program files below, hopefully I could fix the problems from last time. Please use the WordApp program, and if you have any questions call me. I have attached the Success Criteria below again, please examine it. Thank you!



[redacted]

12:49 (28 minutes ago)

做的挺好啊！全部目标都做到，那个软件都很容易用到，总的来说我很开心。我在这几天会试试用你的软件。谢谢你。

Client reply: You did a good job! All the success criteria are met, and the software is easy to use. Overall I am very happy. I will test your program over the next couple of days. Thank you.

[redacted]

19:24 (4 minutes ago)

我已经把你给我的软件用了几天，我有些提议给你，请你考虑一下。

1. 现在我只能在公司提供电脑去用WordApp，有没有办法在云储存存储我打的字？
2. 观看打的字，我不能筛选词语，比如形容子，副词等。请把软件实行这个功能。
3. 现在我不可以和同事平行用软件，觉得有点麻烦。请实行这个功能。
4. 显示表格时，我不能删除不要的字，觉得好麻烦。如果无意输入讹字，就不能删除！

请解决这些问题！谢谢你。

Client suggestions: I have tested the program for a few days, and I have some suggestions for you. Please consider them.

1. I can only use the company computer to use WordApp, is there any way to save the words I inputted into cloud storage?
2. When viewing the inputted words, I cannot filter the words, for example adjective or adverb. Please implement this function into the program.

3. Right now, I cannot use this program simultaneously with my colleague. Please implement this function.
4. When viewing the table, I cannot delete the words that I do not want, I feel this is very annoying. If I accidentally enter the wrong word, I cannot delete it!

Please solve these problems! Thank you.

Code Appendix

App.java

```
package Controller;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class App extends Application {
    private static Scene scene;

// launches the main GUI window
    @Override
    public void start(Stage stage) throws IOException {
        scene = new Scene(loadFXML("MainWindowGUI"));
        stage.setScene(scene);
        stage.show();
    }

    private static Parent loadFXML(String fxml) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(MainWindowController.class.getResource(fxml + ".fxml"));
        return fxmlLoader.load();
    }

    public static void launch(String[] args) {
        launch();
    }

}
```

FlashCardsWindowController.java

```
package Controller;

import Model.QuizQuestion;
import Model.Word;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;

import java.net.URL;
import java.util.ResourceBundle;

public class FlashCardsWindowController implements Initializable {
    public Button nextButton;
    public TextField chineseText;
    public TextField englishText;
    public Button revealButton;
    private Word currentWord;
    MakeFlashCards makeFlashCards;

    @FXML
    AnchorPane flashCardWindowPane;

//multiple ActionEvents to make the flashcards
    public void reveal(ActionEvent actionEvent) {
        englishText.setText(currentWord.getEnglishMeaning());
    }

    public void next(ActionEvent actionEvent) {
        int count=0;

if(count<MainWindowController.getDictionary().getDictionaryStorageByEnglish
Key().size()) {
            clearFields();
            QuizQuestion flashQuestion = new
QuizQuestion(MainWindowController.getDictionary());
            currentWord = flashQuestion.getQuestionWordWord();
            chineseText.setText(currentWord.getChineseMeaning());
            count++;
        }
    }

    public void clearFields(){
        englishText.setText("");
        chineseText.setText("");
    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        flashCardWindowPane.setStyle("-fx-background-color: #F5F5DC");

        QuizQuestion flashQuestion = new
QuizQuestion(MainWindowController.getDictionary());
        currentWord = flashQuestion.getQuestionWordWord();
        chineseText.setText(currentWord.getChineseMeaning());
    }
}
```

InputWordsWindowController.java

```
package Controller;


import Model.Word;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.layout.AnchorPane;

import java.net.URL;
import java.util.Optional;
import java.util.ResourceBundle;

public class InputWordsWindowController implements Initializable {
    public TextField englishMeaningText;
    public TextField englishTypeText1;
    public TextField englishNotesText1;
    public TextField chineseMeaningText1;
    public TextField chineseTypeText11;
    public TextField chineseNotesText11;
    public ComboBox englishTypeSelection;
    public ComboBox chineseTypeSelection;


    @FXML
    AnchorPane inputWordsWindowPane;
    //implementing the error boxes that show up if mandatory fields have
not be inputted
    @FXML
    public void inputWord(){
        boolean flag=true;
        if(!textAreaNotEmpty(englishMeaningText)){
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("Error Input");
            errorAlert.setContentText("Please Enter English Meaning!");
            errorAlert.showAndWait();
            flag=false;
        }
        else if(englishTypeSelection.getSelectionModel().isEmpty()){
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("Error Input");
            errorAlert.setContentText("Please Enter English Word Type!");
            errorAlert.showAndWait();
            flag=false;
        }
        else if(!textAreaNotEmpty(chineseMeaningText1)){
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("Error Input");
            errorAlert.setContentText("Please Enter Chinese Word
Meaning!");
            errorAlert.showAndWait();
            flag=false;
        }
        else if(chineseTypeSelection.getSelectionModel().isEmpty()){
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("Error Input");
            errorAlert.setContentText("Please Enter Chinese Word Type!");
            errorAlert.showAndWait();
            flag=false;
```

```
        }
        if(flag) {
    //confirmation boxes to add the words to wordList
            Alert confirmDialog = new Alert(Alert.AlertType.CONFIRMATION);
            confirmDialog.setHeaderText("Add To Dictionary");
            confirmDialog.setContentText("Do you want to add this word to
dictionary?");
            Optional<ButtonType> result = confirmDialog.showAndWait();

            if(result.get() == ButtonType.OK) {
                String englishMeaning = englishMeaningText.getText();
                String englishWordType =
englishTypeSelection.getSelectionModel().getSelectedItem().toString();
                String englishAdditionalNotes =
englishNotesText1.getText();
                String chineseMeaning = chineseMeaningText1.getText();
                String chineseWordType =
chineseTypeSelection.getSelectionModel().getSelectedItem().toString();
                String chineseAdditionalNotes =
chineseNotesText11.getText();

                Word word = new Word(englishMeaning, englishWordType,
englishAdditionalNotes, chineseMeaning, chineseWordType,
chineseAdditionalNotes);
                //adds inputted word to dictionary

MainWindowController.getDictionary().getDictionaryStorageByEnglishKey().put
(englishMeaning, word);

                Alert informationAlert = new
Alert(Alert.AlertType.INFORMATION);
                informationAlert.setHeaderText("Input Words");
                informationAlert.setContentText("Word input successful!");
                informationAlert.showAndWait();

                //clear fields
                englishMeaningText.clear();
                englishTypeSelection.getSelectionModel().clearSelection();
                englishNotesText1.clear();
                chineseMeaningText1.clear();
                chineseTypeSelection.getSelectionModel().clearSelection();
                chineseNotesText11.clear();
            }
        }
    }


    //javaFX combo box
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        inputWordsWindowPane.setStyle("-fx-background-color: #F5F5DC");


englishTypeSelection.getItems().setAll("Noun","Adjective","Adverb","Verb","
Other");

chineseTypeSelection.getItems().setAll("Noun","Adjective","Adverb","Verb","
Other");
    }

    public boolean textAreaNotEmpty(TextField textField){
```

```
        if(textField.getText().trim().isEmpty()){
            return false;
        }
        else {
            return true;
        }
    }
}
```

Main.java

```
package Controller;

public class Main {

    public static void main(String[] args) {
        App.launch();
    }

}
```

MainWindowController.java

```
package Controller;


import Model.DictionaryStore;
import Model.ExportWords;
import Model.FileHandler;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;

import java.io.IOException;
import java.net.URL;
import java.util.Dictionary;
import java.util.Enumeration;
import java.util.Optional;
import java.util.ResourceBundle;

public class MainWindowController implements Initializable{
    private static DictionaryStore dictionary;
    public Button closeButton;
    public Button inputWordsButton1;

    @FXML
    Button viewSavedItemsButton;

    @FXML
    Button inputWordsButton;

    @FXML
    Button exportWordsButton;

    @FXML

    Button flashCardsButton;

    @FXML
    Button makeQuizButton;

    @FXML
    javafx.scene.control.TextField userName;

    @FXML
    AnchorPane mainWindowPane;

    @FXML
    /*
    main window when launching gui, multiple buttons are shown
    error popups and confirmation boxes are also implemented
```

```
     */
    public void inputWords(){

        FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("inputWordsGUI.fxml"));
        Parent root1 = null;
        try {
            root1 = (Parent) fxmlLoader.load();
        } catch (IOException e) {
            e.printStackTrace();
        }
        Stage stage = new Stage();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("Input Words");
        stage.setScene(new Scene(root1));
        stage.show();

    }

    @FXML
    public void flashCards(){
        if(getDictionary().getDictionaryStorageByEnglishKey().size()==0){
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("No Words!");
            errorAlert.setContentText("Please input and save at least 4
Words!");
            errorAlert.showAndWait();
        }

        else if (getDictionary().getDictionaryStorageByEnglishKey().size()
< 4) {
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("No Enough Words!");
            errorAlert.setContentText("Please input and save at least 4
Words!");
            errorAlert.showAndWait();
        }
        else {

            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("flashCardsGUI.fxml"));
            Parent root1 = null;
            try {
                root1 = (Parent) fxmlLoader.load();
            } catch (IOException e) {
                e.printStackTrace();
            }
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.setTitle("Flashcards");
            stage.setScene(new Scene(root1));
            stage.show();

        }

    }


    @FXML
    public void viewAllSavedWords(){
        if(getDictionary().getDictionaryStorageByEnglishKey().size()==0){
```

```
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("No Words!");
            errorAlert.setContentText("Please input and save some words
first!");
            errorAlert.showAndWait();
        }
        else {

            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("viewAllWordsGUI.fxml"));
            Parent root1 = null;
            try {
                root1 = (Parent) fxmlLoader.load();
            } catch (IOException e) {
                e.printStackTrace();
            }
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.setTitle("All Words");
            stage.setScene(new Scene(root1));
            stage.show();
        }

    }


    @FXML
    public void makeQuiz() {

        if(getDictionary().getDictionaryStorageByEnglishKey().size()==0){
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("No Words!");
            errorAlert.setContentText("Please input and save at least 4
Words!"); //pre-requisite of 4 words for quiz and flashcard functions
            errorAlert.showAndWait();
        }

        else if (getDictionary().getDictionaryStorageByEnglishKey().size()
< 4) {
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("No Enough Words!");
            errorAlert.setContentText("Please input and save at least 4
Words!");
            errorAlert.showAndWait();
        } else {

            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("makeQuizGUI.fxml"));
            Parent root1 = null;
            try {
                root1 = (Parent) fxmlLoader.load();
            } catch (IOException e) {
                e.printStackTrace();
            }
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.setTitle("It's Quiz Time!");
            stage.setScene(new Scene(root1));
            stage.show();

        }
```

```
    }


    @FXML
    public void exportWords() throws IOException {

        Alert confirmDialog = new Alert(Alert.AlertType.CONFIRMATION);
        confirmDialog.setHeaderText("Save All Words");
        confirmDialog.setContentText("Do you want to export all words to
CSV?");
        Optional<ButtonType> result = confirmDialog.showAndWait();

        if(result.get() == ButtonType.OK) {

            if (getDictionary().getDictionaryStorageByEnglishKey().size()
== 0) {
                Alert errorAlert2 = new Alert(Alert.AlertType.ERROR);
                errorAlert2.setHeaderText("No Words!");
                errorAlert2.setContentText("Please input and save some
words first!");
                errorAlert2.showAndWait();
            } else {

                ExportWords.writeToCSV();
            }
        }

    }


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        mainWindowPane.setStyle("-fx-background-color: #F5F5DC");


        //read saved file at program start, if there's no saved file a new
DictionaryStore object will be created

        dictionary= FileHandler.readFromFile();

        if(dictionary==null) {
            dictionary = new DictionaryStore();

        }


    }

    public static DictionaryStore getDictionary() {
        return dictionary;
    }

    @FXML
    public void handleCloseButtonAction(ActionEvent event) {

        System.out.println("Closed");
        FileHandler.writeFile();

((Stage)(((Button)event.getSource()).getScene().getWindow())).close();
    }
```

```
    public void search(ActionEvent actionEvent) {
        if(getDictionary().getDictionaryStorageByEnglishKey().size()==0){
            Alert errorAlert = new Alert(Alert.AlertType.ERROR);
            errorAlert.setHeaderText("No Words!");
            errorAlert.setContentText("Please input and save some words
first!");
            errorAlert.showAndWait();
        }

        else {
            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("searchWordGUI.fxml"));
            Parent root1 = null;
            try {
                root1 = (Parent) fxmlLoader.load();
            } catch (IOException e) {
                e.printStackTrace();
            }
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.setTitle("Search");
            stage.setScene(new Scene(root1));
            stage.show();
        }
    }
}
```

MakeFlashCards.java

```
package Controller;

import Model.DictionaryStore;
import Model.QuizQuestion;

import java.io.Serializable;
import java.util.ArrayList;

public class MakeFlashCards implements Serializable {
    private int numberOfQuestions;
    private ArrayList<QuizQuestion> flashCardQuestionArrayList;
    private DictionaryStore dictionaryStore;

    public MakeFlashCards(int numberOfQuestions, DictionaryStore
dictionaryStore) {


if(dictionaryStore.getDictionaryStorageByEnglishKey().size()>=(numberOfQues
tions+1)){
            this.numberOfQuestions = numberOfQuestions+1;
        }
        else {

this.numberOfQuestions=dictionaryStore.getDictionaryStorageByEnglishKey().s
ize();
        }

        this.dictionaryStore = dictionaryStore;
        this.flashCardQuestionArrayList = new ArrayList<QuizQuestion>();

        //make flash cards if there are at least 4 words in dictionary

if(this.dictionaryStore.getDictionaryStorageByEnglishKey().size()>=4) {
            this.createFlash();
        }
        else {
            System.out.println("Not Enough Words");
        }
    }

    private void createFlash(){
        int i=0;
        while(i<this.numberOfQuestions){
            QuizQuestion quizQuestion=new
QuizQuestion(this.dictionaryStore);

            if(isUniqueQuestion(quizQuestion)){
                this.flashCardQuestionArrayList.add(quizQuestion);
                i++;
            }
        }
    }

    //makes sure all questions are unique
    private boolean isUniqueQuestion(QuizQuestion quizQuestion){
        for(int i = 0; i<this.flashCardQuestionArrayList.size(); i++){

if(this.flashCardQuestionArrayList.get(i).getQuestionWord().equals(quizQues
tion.getQuestionWord())){
```

```
                return false;
            }
        }
        return true;
    }


    public int getNumberOfQuestions() {
        return numberOfQuestions;
    }


    public ArrayList<QuizQuestion> getFlashCardQuestionArrayList() {
        return flashCardQuestionArrayList;
    }
}
```

MakeQuiz.java

```
package Controller;

import Model.DictionaryStore;
import Model.QuizQuestion;

import java.util.ArrayList;

public class MakeQuiz {
    private int numberOfQuestions;
    private int marks;
    private ArrayList<QuizQuestion> quizQuestionArrayList;
    private ArrayList<String> userAnswersArrayList;
    private DictionaryStore dictionaryStore;

    public MakeQuiz(int numberOfQuestions, DictionaryStore dictionaryStore)
{


if(dictionaryStore.getDictionaryStorageByEnglishKey().size()>=(numberOfQues
tions+1)){
            this.numberOfQuestions = numberOfQuestions+1;
        }
        else {

this.numberOfQuestions=dictionaryStore.getDictionaryStorageByEnglishKey().s
ize();
        }

        this.dictionaryStore = dictionaryStore;
        this.quizQuestionArrayList = new ArrayList<QuizQuestion>();
        this.userAnswersArrayList=new ArrayList<String>();
        this.marks=0;

        //make quiz if there are at least 4 words in dictionary

if(this.dictionaryStore.getDictionaryStorageByEnglishKey().size()>=4) {
            this.createQuiz();
        }
        else {
            System.out.println("Not Enough Words");
        }
    }

    private void createQuiz(){
        int i=0;
        while(i<this.numberOfQuestions){
            QuizQuestion quizQuestion=new
QuizQuestion(this.dictionaryStore);

            if(isUniqueQuestion(quizQuestion)){
                this.quizQuestionArrayList.add(quizQuestion);
                i++;
            }
        }
    }

    //makes sure all questions are unique
    private boolean isUniqueQuestion(QuizQuestion quizQuestion){
        for(int i=0;i<this.quizQuestionArrayList.size();i++){
```

```
if(this.quizQuestionArrayList.get(i).getQuestionWord().equals(quizQuestion.
getQuestionWord())){
                return false;
            }
        }
        return true;
    }

    //add user answers to arrayList to evaluate for marks
    public void addAnswer(String userAnswer){
        this.userAnswersArrayList.add(userAnswer);
    }

    //calculate marks
    public int calculateMarks(){
        for(int i=0;i<this.quizQuestionArrayList.size();i++){
            String userAnswer=this.userAnswersArrayList.get(i);


if(this.quizQuestionArrayList.get(i).isCorrectAnswer(userAnswer)){
                this.marks+=10;
            }
        }
        return this.marks;
    }

    public int getNumberOfQuestions() {
        return numberOfQuestions;
    }

    public int getMarks() {
        return this.marks;
    }

    public ArrayList<QuizQuestion> getQuizQuestionArrayList() {
        return quizQuestionArrayList;
    }
}
```

MakeQuizWindowController.java

```
package Controller;

import Model.QuizQuestion;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.layout.AnchorPane;

import java.net.URL;
import java.util.ResourceBundle;

public class MakeQuizWindowController implements Initializable {


    public AnchorPane makeQuizGUIPane;
    public Button nextQuestion;
    public Button previousQuestion;
    public RadioButton answerA;
    public RadioButton answerB;
    public RadioButton answerC;
    public RadioButton answerD;
    public Button back;
    public TextArea questionWord;
    public Label marksLabel;
    private int questionNumber;
    private MakeQuiz makeQuiz;
    int clickCount=0;


    @FXML
    public void previous(){



    }

    public void clearFields(){
        questionWord.setText("");
        answerA.isSelected();
        answerB.setText("");
        answerC.setText("");
        answerD.setText("");

        answerA.setSelected(false);
        answerB.setSelected(false);
        answerC.setSelected(false);
        answerD.setSelected(false);
    }

    @FXML
    public void next(){

        if (clickCount<makeQuiz.getNumberOfQuestions()) {
            clickCount++;

            if (answerA.isSelected() && !answerB.isSelected()
&& !answerC.isSelected() && !answerD.isSelected()) {
                makeQuiz.addAnswer(answerA.getText());
            } else if (!answerA.isSelected() && answerB.isSelected()
```

```
&& !answerC.isSelected() && !answerD.isSelected()) {
                makeQuiz.addAnswer(answerB.getText());
            } else if (!answerA.isSelected() && !answerB.isSelected() &&
answerC.isSelected() && !answerD.isSelected()) {
                makeQuiz.addAnswer(answerC.getText());
            } else if (!answerA.isSelected() && !answerB.isSelected()
&& !answerC.isSelected() && answerD.isSelected()) {
                makeQuiz.addAnswer(answerD.getText());
            } else {
                makeQuiz.addAnswer("N/A");
            }
            clearFields(); //clear all fields

            if (makeQuiz.getNumberOfQuestions() > this.questionNumber) {

                QuizQuestion quizQuestion =
makeQuiz.getQuizQuestionArrayList().get(questionNumber);
                questionWord.setText(quizQuestion.getQuestionWord());
                answerA.setText(quizQuestion.getAnswerA());
                answerB.setText(quizQuestion.getAnswerB());
                answerC.setText(quizQuestion.getAnswerC());
                answerD.setText(quizQuestion.getAnswerD());

                this.questionNumber++;


                if (makeQuiz.getNumberOfQuestions() == questionNumber) {
                    nextQuestion.setText("Complete & Evaluate");

                }


            }

            if (clickCount == questionNumber) {
                nextQuestion.setText("Thank You!");
                makeQuiz.calculateMarks();
                marksLabel.setText("Your Marks: " + makeQuiz.getMarks() +"
/ " +questionNumber*10);
            }
        }


    }



    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        makeQuizGUIPane.setStyle("-fx-background-color: #F5F5DC");
        nextQuestion.setText("Submit Answer & Continue");

        makeQuiz=new MakeQuiz(6,MainWindowController.getDictionary());
        this.questionNumber=0;

        QuizQuestion quizQuestion =
makeQuiz.getQuizQuestionArrayList().get(questionNumber);
        questionWord.setText(quizQuestion.getQuestionWord());
        answerA.setText(quizQuestion.getAnswerA());
        answerB.setText(quizQuestion.getAnswerB());
```

```
            answerC.setText(quizQuestion.getAnswerC());
            answerD.setText(quizQuestion.getAnswerD());

            this.questionNumber++;


        }

    }
```

SearchWordController.java

```
package Controller;

import Model.Word;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;

import java.net.URL;
import java.util.ResourceBundle;

public class SearchWordController implements Initializable {
    public TextField englishMeaningText;
    public TextField chineseMeaningText1;
    public TextField chineseTypeText11;
    public TextField chineseNotesText11;


    @FXML
    AnchorPane searchWordPane;

    @FXML
    public void search() {
        //take user input and search for its Chinese word in saved word
list
        String searchWord=englishMeaningText.getText();

        Word
selectedWord=MainWindowController.getDictionary().getByEnglishWord(searchWo
rd);

        if(selectedWord!=null){
            chineseMeaningText1.setText(selectedWord.getChineseMeaning());
            chineseTypeText11.setText(selectedWord.getChineseWordType());

chineseNotesText11.setText(selectedWord.getChineseAdditionalNotes());
        }
        else {
            chineseMeaningText1.setText("Not in Dictionary! ");
        }

    }


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        searchWordPane.setStyle("-fx-background-color: #F5F5DC");
    }
}
```

viewAllWordsController.java

```
package Controller;
import Model.QuizQuestion;
import Model.Word;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Collections;
import java.util.ResourceBundle;

public class viewAllWordsController implements Initializable {
    public ScrollPane scrollPane;
    public TableColumn englishMeaningTab;
    public TableColumn englishTypeTab;
    public TableColumn englishNotesTab;
    public TableColumn chineseMeaningTab;
    public TableColumn chineseTypeTab;
    public TableColumn chineseNotesTab;
    public TableView tabelView; //this is spelt wrong on purpose (as naming
it tableView will return an error)
    private ArrayList<String> elementListInListView;


    @FXML
    Button viewSelected;


    @FXML
    AnchorPane viewAllWordsPane;




    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {

        ArrayList<Word> wordArrayList=updateTable();
        viewAllWordsPane.setStyle("-fx-background-color: #F5F5DC");

        ObservableList<Word> data =
FXCollections.observableArrayList(wordArrayList);
        englishMeaningTab.setCellValueFactory(new
PropertyValueFactory<Word,String>("englishMeaning"));
        englishTypeTab.setCellValueFactory(new
PropertyValueFactory<Word,String>("englishWordType"));
```

```
        englishNotesTab.setCellValueFactory(new
PropertyValueFactory<Word,String>("englishAdditionalNotes"));
        chineseMeaningTab.setCellValueFactory(new
PropertyValueFactory<Word,String>("chineseMeaning"));
        chineseTypeTab.setCellValueFactory(new
PropertyValueFactory<Word,String>("chineseWordType"));
        chineseNotesTab.setCellValueFactory(new
PropertyValueFactory<Word,String>("chineseAdditionalNotes"));
        tabelView.setItems(data);


    }

    public ArrayList<Word> updateTable(){
        ArrayList<Word> wordArrayListTemp=new ArrayList<Word>();
        //get hash map keys arrayList
        ArrayList<String> keysArrayList;
        keysArrayList=
QuizQuestion.readAllKeys(MainWindowController.getDictionary());

        //sort by alphabetically
        Collections.sort(keysArrayList);

        this.elementListInListView=new ArrayList<String>();

        //update to list view
        for(int i=0;i<keysArrayList.size();i++){
            Word
word=MainWindowController.getDictionary().getDictionaryStorageByEnglishKey(
).get(keysArrayList.get(i));
            wordArrayListTemp.add(word);

        }

        return wordArrayListTemp;
    }
}
```

DictionaryStore.java

```
package Model;

import java.io.Serializable;
import java.util.HashMap;

public class DictionaryStore implements Serializable {
    private HashMap<String,Word> dictionaryStorageByEnglishKey;

    public DictionaryStore() {
        this.dictionaryStorageByEnglishKey = new HashMap<String,Word>();
    }

    public HashMap<String, Word> getDictionaryStorageByEnglishKey() {
        return dictionaryStorageByEnglishKey;
    }


    //to obtain relevant word object for a entered english word

    public Word getByEnglishWord(String englishWordToSearch){
        return this.dictionaryStorageByEnglishKey.get(englishWordToSearch);
    }



}
```

ExportWords.java

```
package Model;

import Controller.MainWindowController;
import Controller.viewAllWordsController;

import java.awt.image.DataBufferInt;
import java.io.*;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Collections;

public class ExportWords implements Serializable {
    static OutputStream os;



    public static void writeToCSV() throws IOException {

        //get key words arrayList and sort alphabetically using java
collection
        ArrayList<String>
keyWordsArrayList=QuizQuestion.readAllKeys(MainWindowController.getDictiona
ry());
        Collections.sort(keyWordsArrayList);



            PrintWriter pw = null;
            try {
                os = new FileOutputStream("AllWords.csv");



            } catch (IOException e) {
                e.printStackTrace();
            }
        StringBuilder builder = new StringBuilder();
            String columnNamesList = "English Meaning,English Word
Type,English Notes,Chinese Meaning,Chinese Word Type,Chinese Notes";

            builder.append(columnNamesList +"\n");

            viewAllWordsController viewAllWordsController1=new
viewAllWordsController();

        for ( int i=0;i<keyWordsArrayList.size();i++) {
            Word
currentWord=MainWindowController.getDictionary().getDictionaryStorageByEngl
ishKey().get(keyWordsArrayList.get(i));


                builder.append(currentWord.getEnglishMeaning()+",");
                builder.append(currentWord.getEnglishWordType()+",");

builder.append(currentWord.getEnglishAdditionalNotes()+",");
                builder.append(currentWord.getChineseMeaning()+",");
                builder.append(currentWord.getChineseWordType()+",");
                builder.append(currentWord.getChineseAdditionalNotes());
```

```
                builder.append('\n');

        }

//                pw.write(builder.toString());

os.write(builder.toString().getBytes(StandardCharsets.UTF_8));



//            pw.close();
            System.out.println("done!");
        }



}
```

FileHandler.java

```
package Model;

import Controller.MainWindowController;

import java.io.*;

public class FileHandler {

    public static void writeFile(){
        Object obj= MainWindowController.getDictionary();
        FileOutputStream f = null;
        try {
            f = new FileOutputStream(new File("dictionary.dat")); //creates
new .dat save file if none detected

        ObjectOutputStream o = new ObjectOutputStream(f);

        // write objects to file
        o.writeObject(obj);

        o.close();
        f.close();
        } catch (FileNotFoundException e) {
            System.out.println("");
        } catch (IOException e) {
           System.out.println("");
        }
    }

    public static DictionaryStore readFromFile(){
        FileInputStream fi = null;
        try {
            fi = new FileInputStream(new File("dictionary.dat"));

        ObjectInputStream oi = new ObjectInputStream(fi);

        // reads objects
        DictionaryStore savedStore = (DictionaryStore) oi.readObject();

        oi.close();
        fi.close();
        return savedStore;

        } catch (FileNotFoundException e) {
           System.out.println("");
        } catch (IOException e) {
           System.out.println("");
        } catch (ClassNotFoundException e) {
           System.out.println("");
        }

        return null;
    }
}
```

QuizQuestion.java

```
package Model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;

public class QuizQuestion implements Serializable {
    private String QuestionWord;
    private String answerA;
    private String answerB;
    private String answerC;
    private String answerD;
    private String correctAnswer;
    private DictionaryStore dictionaryStore;
    private int noOfAllWordsAvailable;
    private int randomNo1,randomNo2,randomNo3,randomNo4;
    private String randomKey1,randomKey2,randomKey3,randomKey4;
    private Random rand=new Random();
    private ArrayList<String> keysArrayList;
    private ArrayList<String> answersArrayList;
    private Word questionWordWord;

    public QuizQuestion(DictionaryStore dictionaryStore) {
        this.dictionaryStore = dictionaryStore;
        this.noOfAllWordsAvailable=
dictionaryStore.getDictionaryStorageByEnglishKey().size();
        this.keysArrayList=new ArrayList<String>();
        this.answersArrayList=new ArrayList<String>();
        this.generateRandomNumbers();
        this.keysArrayList=readAllKeys(this.dictionaryStore);
        this.assignRandomKeys();
        this.createQuestion();
    }
    //generates random numbers for the quiz questions
    private void generateRandomNumbers(){
            ArrayList<Integer>
uniqueRandomNumbers=this.UniqueRandomNumbers(0,noOfAllWordsAvailable-1,4);
            randomNo1 = uniqueRandomNumbers.get(0);
            randomNo2 = uniqueRandomNumbers.get(1);
            randomNo3 = uniqueRandomNumbers.get(2);
            randomNo4 = uniqueRandomNumbers.get(3);
    }

    public static ArrayList<String> readAllKeys(DictionaryStore
dictionaryStore){
        ArrayList<String> keyWordsArrayList=new ArrayList<String>();
        for ( String key :
dictionaryStore.getDictionaryStorageByEnglishKey().keySet() ) {
            keyWordsArrayList.add(key);
        }
        return keyWordsArrayList;
    }

    //adds all answer choices to an arrayList so they can be randomly
arranged
    private void readAllAnswers(){

this.answersArrayList.add(this.dictionaryStore.getDictionaryStorageByEnglis
```

```
hKey().get(randomKey1).getChineseMeaning());

this.answersArrayList.add(this.dictionaryStore.getDictionaryStorageByEnglis
hKey().get(randomKey2).getChineseMeaning());

this.answersArrayList.add(this.dictionaryStore.getDictionaryStorageByEnglis
hKey().get(randomKey3).getChineseMeaning());

this.answersArrayList.add(this.dictionaryStore.getDictionaryStorageByEnglis
hKey().get(randomKey4).getChineseMeaning());
    }

    private void assignRandomKeys(){
        this.randomKey1=this.keysArrayList.get(randomNo1);
        this.randomKey2=this.keysArrayList.get(randomNo2);
        this.randomKey3=this.keysArrayList.get(randomNo3);
        this.randomKey4=this.keysArrayList.get(randomNo4);

    }

    private void generateAnswerList(){
        boolean run=true;

        int min =0;
        int max=3;

        ArrayList<Integer>
uniqueRandomNumbers=this.UniqueRandomNumbers(0,3,4);
         int num1= uniqueRandomNumbers.get(0);
        int num2 = uniqueRandomNumbers.get(1);
        int num3 = uniqueRandomNumbers.get(2);
        int num4 = uniqueRandomNumbers.get(3);


                this.answerA=this.answersArrayList.get(num1);
                this.answerB=this.answersArrayList.get(num2);
                this.answerC=this.answersArrayList.get(num3);
                this.answerD=this.answersArrayList.get(num4);




    }

    public void createQuestion(){
        //randomKey1 = question index

this.QuestionWord=this.dictionaryStore.getDictionaryStorageByEnglishKey().g
et(randomKey1).getEnglishMeaning();


this.questionWordWord=this.dictionaryStore.getDictionaryStorageByEnglishKey
().get(randomKey1);

this.correctAnswer=this.dictionaryStore.getDictionaryStorageByEnglishKey().
get(randomKey1).getChineseMeaning();
        this.readAllAnswers();
        this.generateAnswerList(); //assign random answers including the
correct answer for choices A,B,C & D
    }
```

```
    //to check the input answer is correct
    public boolean isCorrectAnswer(String selectedAnswer){
        if(selectedAnswer.equals(this.correctAnswer)){
            return true;
        }
        else return false;
    }

    public String getQuestionWord() {
        return QuestionWord;
    }

    public String getAnswerA() {
        return answerA;
    }

    public String getAnswerB() {
        return answerB;
    }

    public String getAnswerC() {
        return answerC;
    }

    public String getAnswerD() {
        return answerD;
    }

    public String getCorrectAnswer() {
        return correctAnswer;
    }

    public Word getQuestionWordWord() {
        return this.questionWordWord;
    }


    //to get random numbers in a range
    public ArrayList<Integer> UniqueRandomNumbers(int from,int to,int
noOfNumbers) {
        ArrayList<Integer> uniqueRandomNumbers=new ArrayList<Integer>();

            ArrayList<Integer> list = new ArrayList<Integer>();
            for (int i=from; i<=to; i++) {
                list.add(i);
            }
            Collections.shuffle(list);
            for (int i=0; i<noOfNumbers; i++) {
                uniqueRandomNumbers.add(list.get(i));
            }
        return uniqueRandomNumbers;
    }



}
```

Word.java

```
package Model;

import java.io.Serializable;

public class Word implements Serializable {
    private String englishMeaning;
    private String englishWordType;
    private String englishAdditionalNotes;
    private String chineseMeaning;
    private String chineseWordType;
    private String chineseAdditionalNotes;
    //saves the words
    public Word(String englishWord, String englishWordType, String
englishAdditionalNotes, String chineseMeaning, String chineseWordType,
String chineseAdditionalNotes) {
        this.englishMeaning = englishWord;
        this.englishWordType = englishWordType;
        this.englishAdditionalNotes = englishAdditionalNotes;
        this.chineseMeaning = chineseMeaning;
        this.chineseWordType = chineseWordType;
        this.chineseAdditionalNotes = chineseAdditionalNotes;
    }

    public String getEnglishMeaning() {
        return englishMeaning;
    }

    public String getEnglishWordType() {
        return englishWordType;
    }

    public String getEnglishAdditionalNotes() {
        return englishAdditionalNotes;
    }

    public String getChineseMeaning() {
        return chineseMeaning;
    }

    public String getChineseWordType() {
        return chineseWordType;
    }

    public String getChineseAdditionalNotes() {
        return chineseAdditionalNotes;
    }

    @Override
    public String toString() {
        return "Word{" +
                "english='" + englishMeaning + '\'' +
                ", chinese='" + chineseMeaning + '\'' +
                '}';
    }
}
```