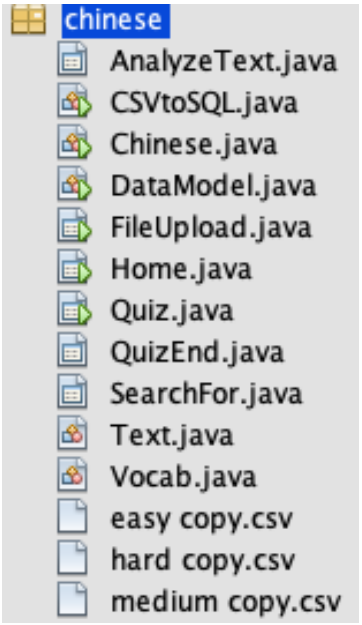
Development

List of Techniques Used:
SQLite Actions: Creating, Selecting, Inserting, Removing, Updating database
Establishing Connection between SQLite Database and Java Classes on Netbeans
Parsing text files
Exception Handling: Try and Catch
Accessors and Mutators (Getters and Setters)
Use of ArrayLists and Parallel ArrayLists
OOP Concepts - Encapsulation and Inheritance
Event Handling
Implementation of GUI using Swing Library

Classes and Imports




```
import java.sql.*;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;
import javax.swing.JFileChooser;
```

SQL

SQLite or Structured Query Language provides a relational database management system that allows users to write queries to perform certain actions such as Inserting, Removing, Updating databases. SQLite is most commonly used for single-user systems.

Most if not all methods pertaining towards SQLite commands were made in the 'DataModel' class. This was done to ensure clarity and neatness within code and for methods which were called whenever they were needed throughout the program. Imports below were used for setting up connection between Application and SQLite, as well as executing sql statements.

```
import java.sql.*;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

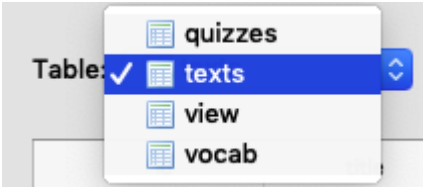
SQLite Database Add and Close Connection

```
//Encapsulation used here and throughout the methods in the DataModel class, where instance variables are declared private and methods are public (highlighted in blue). This is to ensure that DataModel can be reused since it is aggregated to 4 other classes. (seen in Criterion B Front-End UML Diagram)
public class DataModel {
     private Connection connection = null;
     private ResultSet resultSet = null;
     private Statement statement = null;
     private String url = "jdbc:sqlite:database.db";
// SQLite connecter used Java Database Connector (JDBC) API since project was written in Java
// try and catch is used to ensure user is able to still able to run the application in the case where the gaining connection fails
      public void addConnection(){
     try {
         connection = DriverManager.getConnection(url);
         statement = connection.createStatement();
     }
     catch (Exception e)
     {
         e.printStackTrace();  }
 }
      public void closeConnection(){
          try{
              connection.close();
          }
         catch(Exception e){
             e.printStackTrace();
```

```
        }
    }
}
```

SQLite Create and Delete Database Tables

Below is the method to create database tables: 'vocab' to store Chinese vocabulary and its difficulty, 'texts' to store uploaded text's title, passage, and the previous score scored on the quiz based on text, 'quizzes' to store quizzes made from uploaded texts.



View of Database tables on DB Browser.

```
// Use of 'CREATE TABLE' SQL command, having SQL statements in place reduces program execution time as well as allows for efficient updating and refreshing.
public void createTables(){
// try and catch is used to ensure user is able to still able to run the application in the case where the gaining connection fails
        try {
// creates vocab database table which is the primary key, with each word uniquely identified by an id number
         String sql=  "CREATE TABLE IF NOT EXISTS vocab (id INTEGER PRIMARY KEY ,word STRING, difficulty STRING)";
         statement.execute(sql);
// creates texts database table, also a primary key, with each text uniquely identified by an id number
         sql = "CREATE TABLE IF NOT EXISTS texts(id INTEGER PRIMARY KEY,title STRING , passage STRING , prev_score INTEGER) ";
         statement.execute(sql);
// creates quizzes database table
         sql = "CREATE TABLE IF NOT EXISTS quizzes (textid INTEGER, vocabid INTEGER)";
         statement.execute(sql);
         } catch (SQLException ex) {
       Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
         }
    }
    public void dropTables(){
       Try{
// use of 'DROP TABLE' SQL command
           String sql = ("DROP TABLE IF EXISTS vocab; ");
           statement.execute(sql);
           String sql = ("DROP TABLE IF EXISTS texts; ");
           statement.execute(sql);
           String sql = ("DROP TABLE IF EXISTS quizzes; ");
```

```
            statement.execute(sql);

        }
        catch(SQLException e){
         e.printStackTrace();
        }
    }
```

CSV TO SQL (Parsing text files)

From the HSK Chinese words database, I converted it from excel to CSV as CSV is readily parseable with code. From there, this method was created so that SQLite would be able to read the HSK Chinese word database and store it in SQL Database.

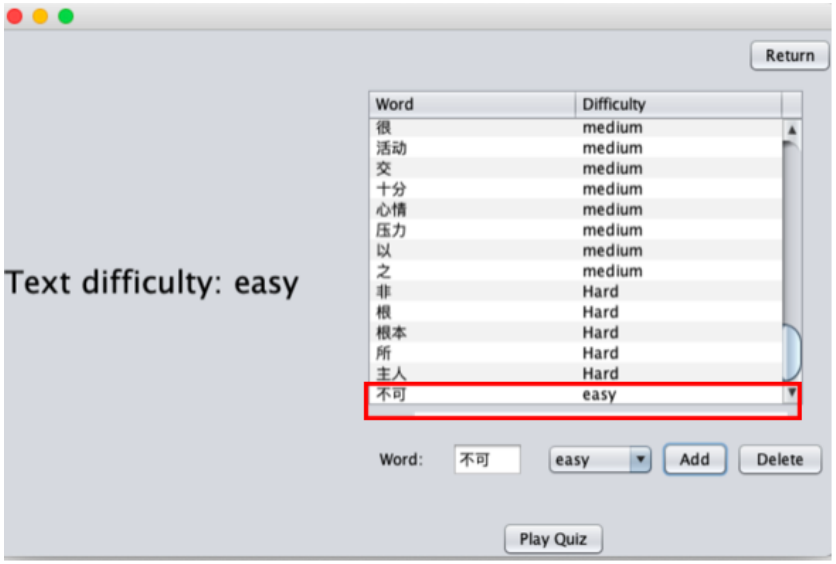
```
public class CSVtoSQL {
    public static void main(String[]args){
        DataModel d = new DataModel();
        d.addConnection();
// reads and parses file that contains the easy words
// try and catch is used to ensure user is able to still able to run the application in the case where the gaining connection fails
        try( Scanner scanner = new Scanner(new File("/Users/michaeltham/NetBeansProjects/Chinese/src/chinese/easy copy.csv"));){
            while(scanner.hasNextLine()){
                String word = scanner.nextLine();
// using addWord() method, adds word from file into the vocab database table with respective difficulty
                d.addWord(word,"easy");
            }
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
// reads and parses file that contains the medium words
        try( Scanner scanner = new Scanner(new File ("/Users/michaeltham/NetBeansProjects/Chinese/src/chinese/medium copy.csv"));){
            while(scanner.hasNextLine()){
                String word = scanner.nextLine();
// using addWord() method, adds word from file into the vocab database table with respective difficulty
                d.addWord(word,"medium");
            }
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
// reads and parses file that contains the hard words
```

```
         try( Scanner scanner = new Scanner(new
File("/Users/michaeltham/NetBeansProjects/Chinese/src/chinese/hard copy.csv"));){
            while(scanner.hasNextLine()){
                String word = scanner.nextLine();
// using addWord() method, adds word from file into the vocab database table with
respective difficulty
                d.addWord(word,"Hard");
            }
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
    d.closeConnection();
    }
}
```

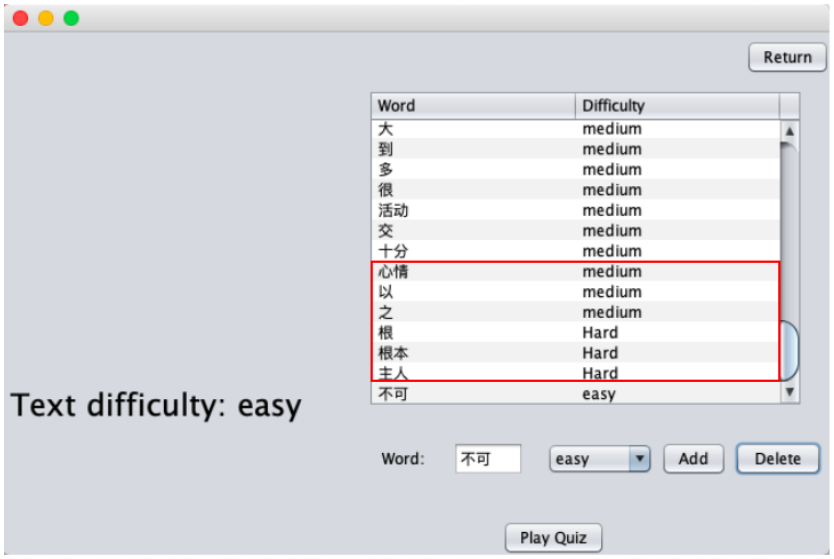
SQL Adding data into 'Vocab' Database table

In the CSVtoSQL() method, I called the addWord() method so that once SQLite reads the word from HSK Chinese word database, it inserts it into the 'vocab' database table.

```
//First takes in a word and difficulty
public void addWord(String word, String difficulty){
// use of 'INSERT INTO' SQL command, which inserts data into database
   String sql = "INSERT INTO vocab (word, difficulty) "
// inserts desired word and its difficulty into vocab database table
        + "VALUES( '"+word + "','" + difficulty + "'); ";
// try and catch is used to ensure user is able to still able to run the application in
the case where the gaining connection fails
         try {
             statement.execute(sql);
         } catch (SQLException ex) {
             Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
         }
    }
// use of 'DELETE FROM' SQL command, which deletes data from database
    public void deleteWord(String word, Text text){
        String sql = "DELETE FROM vocab (word, text) "
// deletes desired word and its difficulty from vocab database table
               + "WHERE( '"+word + "','" + text + "'); ";
        try{
            statement.execute(sql);
        }
           catch (SQLException ex){
             Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
           }
}
```



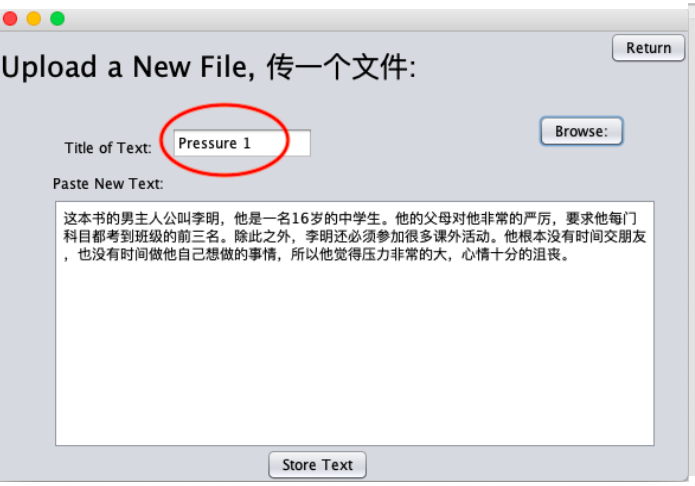
addWord() and deleteWord() methods are also called in the AnalyzeText Class where user can add or delete a word from the vocabulary list generated from an uploaded text. On the left hand side, the red box shows that the new word has been added into the list of words to be tested in the quiz.

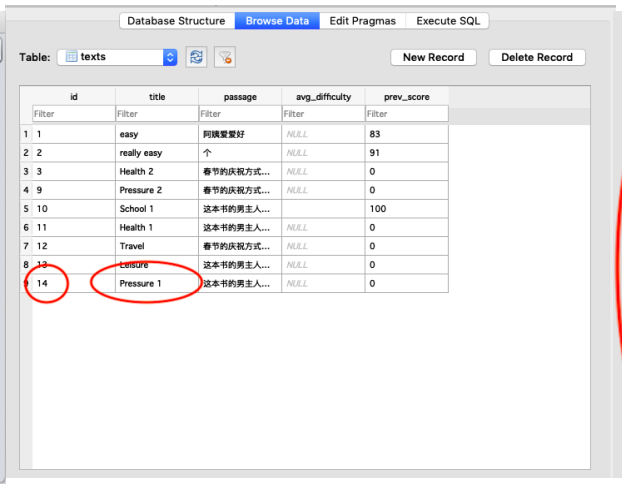


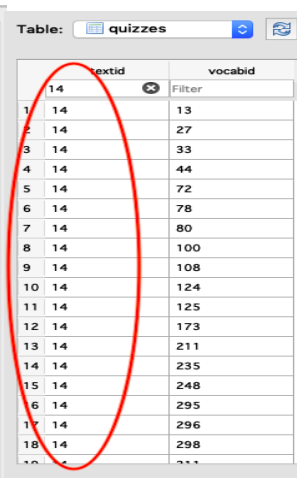
Using the deleteWord() method, I specifically deleted the words '压力' and '非' from the original list of words. These also shows event handling aspect where a behaviour is executed using Swing components. In this case of the addWord() and deleteWord() methods, it was a JButton.

SQL Adding a Text into the 'Texts' Database table

When user clicks 'store text', addText() method will be called, where text gets inserted into 'texts' database table and the program looks for words in a text that matches with the HSK Chinese database and inserts specific wordID into 'Quizzes' table. SQL commands like INSERT and INNER JOIN were used.



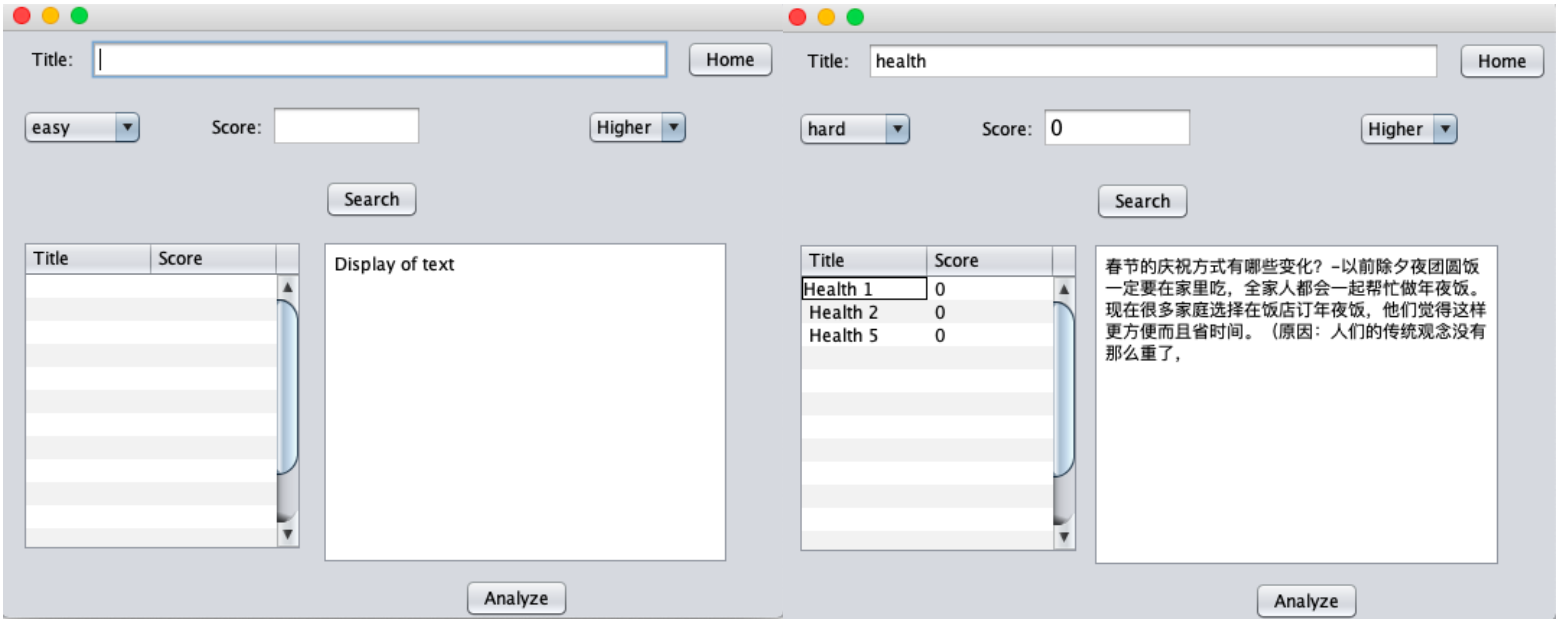




```
public void addText(Text text){
        try {
// inserts text into 'texts' table, use of SQL command INSERT INTO
// try and catch is used to ensure user is able to still able to run the application in
the case where the gaining connection fails
            String sql = "INSERT INTO texts (title,passage,prev_score) "
                    + " VALUES ( "
                    + "'"+ text.getTitle() + "',"
                    + "'"+ text.getPassage() + "',"
                    + text.getPrevScore() + ");" ;

            statement.execute(sql);

//gets text id
            sql = "SELECT id FROM texts "
                 + "ORDER BY id DESC LIMIT 1;";
            ResultSet results = statement.executeQuery(sql);
            while(results.next()){
                text.setId(results.getInt("id"));
            }
//gets all words that are in the passage, use of SQL command SELECT, INNER JOIN, WHERE
            sql = "SELECT vocab.id, texts.id FROM vocab "
                       + "INNER JOIN texts "
                       + "ON passage LIKE '%' || word || '%' "
                       + "WHERE texts.id = " + text.getId() + ";";
            results = statement.executeQuery(sql);

            Statement statement2 = connection.createStatement();
//adds vocabid and textid into quizzes table
            while (results.next()) {

                String vocabid = results.getString(1);
                String textid = results.getString(2);

                sql = "INSERT INTO quizzes (vocabid, textid) "
                       + "VALUES (" + vocabid + ',' + textid + ");";
                statement2.execute(sql);
            }
        } catch (SQLException ex) {
            Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

SQL Get Desired Text from Search Bar

The getTexts() method in the DataModel class uses SQL commands like SELECTING. Arrays, Loops, and If Statements are used as well. Texts Arraylist is created and texts are stored, and getAvgDifficulty() method is called to give a text difficulty. Text is given a difficulty so as to when a user searches for a text with a particular difficulty, the text with other difficulties are removed from ArrayList and only texts of that difficulty are shown. In the example below, texts with only 'hard' difficulty and the title 'health' are displayed after searching.
Search JButton is event handling.




```
public double getAvgDifficulty(Text text){

      double total = 0;
      int count = 0;

      String sql = "SELECT difficulty, COUNT(*) FROM vocab "
                 + "INNER JOIN quizzes "
                 + "ON vocabid = id "
                 + " WHERE textid = " + text.getId()
                 + " GROUP BY difficulty;";
// try and catch is used to ensure user is able to still able to run the application in the case where the gaining connection fails
        try {
            ResultSet results = statement.executeQuery(sql);
            while (results.next()) {
                if (results.getString("difficulty").equals("easy")) {
// if word in text is easy, adds 1 to the total variable
                    total += results.getInt("COUNT(*)");
```

```
                }
                else if (results.getString("difficulty").equals("medium")) {
// if word in text is medium, adds 2 to the total variable
                  total += results.getInt("COUNT(*)") *2;
                }
                else if (results.getString("difficulty").equals("hard")) {
//if word in text is hard, adds 3 to the total variable
                  total += results.getInt("COUNT(*)")*3;
                } // count adds by 1 each iteration of the while loop
                count++;
            }
        } catch (SQLException ex) {
            Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
        }
//gets the average by dividing total/count which is returned as a double
        return total/count;
    }
```

```
public ArrayList<Text> getTexts(String title, String difficulty, int score, String ho){
        String sql = "";
//if statement which allows user to choose between higher than a certain score or lower, use of SQL command SELECT, WHERE, AND
        if (ho.equals("Higher")){
        sql = "SELECT * FROM texts"
                + " WHERE title LIKE '%" + title + "%' "
                + " AND prev_score >= " + score + ";";
       } else if (ho.equals("Lower")){
         sql = "SELECT * FROM texts"
                + " WHERE title LIKE '%" + title + "%' "
                + " AND prev_score <= " + score + ";";
       }
        //creates text arraylist
        ArrayList<Text> texts = new ArrayList<Text>();
// try and catch is used to ensure user is able to still able to run the application in the case where the gaining connection fails
        try {
            ResultSet results = statement.executeQuery(sql);
        // adds text
            while (results.next()){
              Text text = new Text();
              text.setId(results.getInt("id"));
              text.setTitle(results.getString("title"));
              text.setPassage(results.getString("passage"));
              text.setPrevScore(results.getInt("prev_score"));
```

```
                texts.add(text);
              }
              for (int i=texts.size()-1; i >= 0 ;i--){
                double diff = getAvgDifficulty(texts.get(i));
//sets the difficulty for that specific text
                texts.get(i).setDifficulty(diff);
// if the text is not of that difficulty that the user selects to search, it gets removed
from the texts arraylist
                if (!texts.get(i).getDifficulty().equals(difficulty)){
                    texts.remove(i);
                }
              }
          } catch (SQLException ex) {
              Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
          }
// prints the texts that fit the desired search parameters of the user in the Jtable
          System.out.println(texts);
          return texts;
}
```

Getting Quiz Words

AnalyzeText class calls the getQuizWords() method (highlighted in purple) so as to get the quiz words.

```
public AnalyzeText(Text text) {
          initComponents();
          model = (DefaultTableModel) qTable.getModel();
          d.addConnection();
          this.t = text;
//stores vocab ArrayList in the getQuizWords() method into the variable 'vocab'
          vocab = d.getQuizWords(text);
          DefaultTableModel model = (DefaultTableModel) qTable.getModel();
// loop through vocab ArrayList
          for(int i=0; i < vocab.size(); i++) {
 //prints the 'matched' words and its respective difficulty in the Jtable, 'model'
variable
              Vocab vocabobj = vocab.get(i);
              System.out.println(vocabobj.word);
              String[] row = {vocabobj.word, vocabobj.difficulty};
              model.addRow(row);
          }
      difficultyDisplay.setText("Text difficulty: " + t.getDifficulty());
     }
```

```
 public ArrayList<Vocab> getQuizWords(Text text){
        ArrayList<Vocab> vocab = new ArrayList();
//gets textid of the uploaded text and the vocabid of each 'matched' word and puts them
into the 'quizzes' database table
// try and catch is used to ensure user is able to still able to run the application in the
case where the gaining connection fails
        try {
             String sql = "SELECT id, word, difficulty FROM vocab"
                     + " LEFT JOIN quizzes ON id = vocabid"
                     + " WHERE textid =" + text.getId() + ";";
             ResultSet quiz = statement.executeQuery(sql);
//adds 'matched' words into vocab ArrayList of Vocab objects
             while(quiz.next()){
                 Vocab qword = new Vocab();
                 qword.difficulty = quiz.getString("difficulty");
                 qword.id = quiz.getInt("id");
                 qword.word = quiz.getString("word");
                 vocab.add(qword);
             }
         } catch (SQLException ex) {
             Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
         }
        return vocab;
    }
```

SQL Updating quiz score

After the quiz ends, updateScore() method is called and using SQL commands, updates the prev_Score in the texts database table.

```
  public void updateScore(Text text){
//use of SQL command UPDATE, SET, WHERE
        String sql = "UPDATE texts"
                +" SET prev_score = "
                + text.getPrevScore()
                + " WHERE id =" + text.getId()+ ";";
// try and catch is used to ensure user is able to still able to run the application in the
case where the gaining connection fails
         try {
             statement.execute(sql);
         } catch (SQLException ex) {
             Logger.getLogger(DataModel.class.getName()).log(Level.SEVERE, null, ex);
         }
    }
```

Notice how some texts have a score of 0 and some of more than 0, which is due to the default score being 0. If the quiz is not played for that specific uploaded text, it will be 0.

10	I	这本书的男主人...	NULL	100
11	Health 1	这本书的男主人...	NULL	0
12	Travel	春节的庆祝方式...	NULL	83
13	Leisure	这本书的男主人...	NULL	91
14	pressure	这本书的男主人...	NULL	0

EVENT HANDLING

Certain components on the JPanel (such as the buttons used in the application like Search, Add, Delete) listen to events generated by the user (e.g. mouse clicks). So when an event is called or fired, the method attached to the event handler is called, thus allowing me to create an interactive application. Event handling also makes it easier for debugging in the future as it the location of the error is easier to be identified.

Method when User presses button 'Search'

Search button in the SearchFor page

```
private void SearchActionPerformed(java.awt.event.ActionEvent evt) {
//gets whether the user has chosen higher or lower in the JcomboBox when searching for a text
      String higherOrLower = (String)ho.getSelectedItem();
//calls getTexts() method so program searches for the desired texts based on the search parameters entered by user and returns an ArrayList of texts that is stored in 'texts' variable
      texts = d.getTexts(title.getText(), (String)difficulty.getSelectedItem(), Integer.parseInt(score.getText()), higherOrLower);
//loops through ArrayList of texts and prints the texts in the Jtable
      for(int i=0; i < texts.size();i++){
         Text t = texts.get(i);
         table.getModel().setValueAt(t.getTitle(), i, 0);
         table.getModel().setValueAt(t.getPrevScore(), i, 1);
      }
```

Correct button

'Correct' button during quiz

```
private void correctWordActionPerformed(java.awt.event.ActionEvent evt) {
        Vocab word = allWords.get(i);
// words that are 'wrong' are stored in the wrongWords Array List. The If statement below checks if the current word in the quiz is in the wrongWords array list, and if it isn't, score would increase by 1.
        if (!wrongWords.contains(word)){
            score++;
```

```
        } // then calls nextWord() method to move on to the next word in the quiz
         nextWord();
         progressBar.setValue(progressBar.getValue()+1);
}
```

Skip button

'Skip' button during quiz

```
 private void skipActionPerformed(java.awt.event.ActionEvent evt) {
//calls nextWord() method to move on to the next word in the quiz
        nextWord();
//adds 1 to the progress bar, one more value closer to the maximum value
        progressBar.setValue(progressBar.getValue()+1);

    }
```

Browse button

'Browse' button in FileUpload

```
private void BrowseActionPerformed(java.awt.event.ActionEvent evt) {

        JFileChooser chooser = new JFileChooser();
        chooser.showOpenDialog(null);
        File f = chooser.getSelectedFile();
        String filename = f.getAbsolutePath();

        //reads text from file chosen
        try (Scanner scanner = new Scanner(new FileReader(filename));) {
        StringBuilder sb = new StringBuilder();
            while(scanner.hasNext()){
                sb.append(scanner.next());
            }
            scanner.close();
            String out = sb.toString();
         // prints into text area
            pasteText.setText(out);

            } catch (FileNotFoundException ex) {
         Logger.getLogger(FileUpload.class.getName()).log(Level.SEVERE, null, ex);
        }

     }
```

Play quiz button

'Play Quiz' in AnalyzeText

```
 private void playQuizActionPerformed(java.awt.event.ActionEvent evt) {
//moves from AnalyzeText page to Quiz page
        this.dispose();
        Quiz obj = new Quiz(vocab,t);
        obj.setVisible(true);

    }
```

Search button

'Search' in SearchFor

```
private void SearchActionPerformed(java.awt.event.ActionEvent evt) {
      String higherOrLower = (String)ho.getSelectedItem();
//calls getTexts() method which returns an arraylist of text objects and is stored in 'texts' variable
      texts = d.getTexts(title.getText(), (String)difficulty.getSelectedItem(), Integer.parseInt(score.getText()), higherOrLower);
//In the Jtextfield where the user enters the score, it is stored as a String, so need to convert the String into an integer so as to search through the prev_Score column in the texts database table
//loop through texts arraylist
      for(int i=0; i < texts.size();i++){
         Text t = texts.get(i);
//prints the texts that meet user search parameters in the Jtable
         table.getModel().setValueAt(t.getTitle(), i, 0);
         table.getModel().setValueAt(t.getPrevScore(), i, 1);
      }
    }
```

Add Word Button

'Add' in AnalyzeText

```
private void addWordActionPerformed(java.awt.event.ActionEvent evt) {

         String word = wordWanted.getText();
         String diff = (String) difficulty.getSelectedItem();

        // adds word to Jtable
         String[] row = {word, diff};
         model.addRow(row);

         // adds word to vocab ArrayList of vocab objects
```

```
        Vocab v = new Vocab();
        v.word = word;
        v.difficulty = diff;
        vocab.add(v);
    }
```

Delete Word Button

'Delete' in AnalyzeText

```
private void deleteWordActionPerformed(java.awt.event.ActionEvent evt) {
        //deletes word from Jtable
        int i = qTable.getSelectedRow();
        DefaultTableModel dtm = (DefaultTableModel) qTable.getModel();
        dtm.removeRow(i);
        //deletes word from vocab arraylist and therefore will not be quizzed
        vocab.remove(i);
    }
```

Encapsulation

There are 2 custom-defined data types, the classes Text and Vocab, which help transfer data between the DataModel and other components by grouping information together into a single object.

Text Class

Encapsulation was used in the text class, where variables were set to private and methods made public. This was done to ensure the reusability of the class. Accessor and mutator methods such as setTitle() and getTitle(), can be updated at any time without the class causing other components that depend on it to malfunction.

```
public class Text {
    private int id = -1;
    private String passage = "";
    private String title = "";
    private int prevScore = 0;
    private double difficulty;
// private instance variables highlighted in blue, public methods in red, and getters + setters in purple

    public void setDifficulty(double difficulty){
        this.difficulty = difficulty;
    }

    public String getDifficulty(){
//depending on the value returned on the getAvgDifficulty(), if statement bases
```

```
        if (difficulty < 1.5){
            return "easy";
        } else if (difficulty < 2.5){
            return "medium";
        } else {
            return "hard";
        }
    }
    public void setTitle(String title){
        this.title = title;
    }
    public String getTitle(){
       return title;
    }
    public void setPassage(String passage){
        this.passage = passage;
    }
    public String getPassage(){
        return passage;
    }
    public void setId(int id){
        this.id = id;
    }
    public int getId(){
        return id;
    }
    public void setPrevScore(int prevScore){
        this.prevScore = prevScore;
    }
    public int getPrevScore(){
       return prevScore;
    }
}
```

Inheritance

Classes listed below extends `javax.swing.JFrame`, (using swing library) and hence inherit all of its attributes and methods.

```
public class AnalyzeText extends javax.swing.JFrame {
                                                .
                                                .
}public class FileUpload extends javax.swing.JFrame {
```

```
                                                .
                                                .
}public class Home extends javax.swing.JFrame {
                                                .
                                                .
}
public class Quiz extends javax.swing.JFrame {
                                                .
                                                .
}
public class QuizEnd extends javax.swing.JFrame {
                                                .
                                                .
}
public class SearchFor extends javax.swing.JFrame {
                                                .
                                                .
}
```

Use of Parallel ArrayLists

Quiz Class

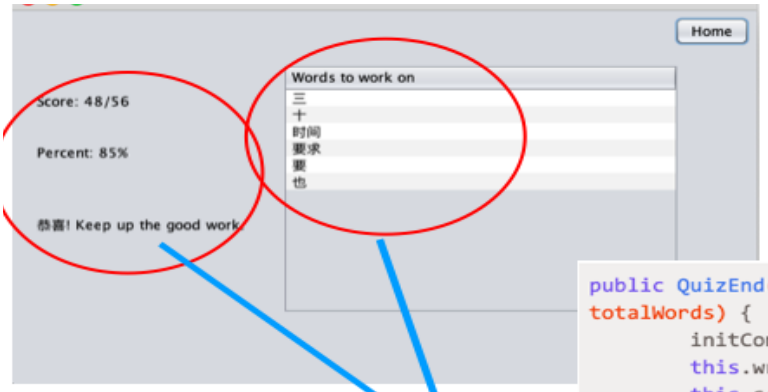
If user were to click 'wrong' button(Event Handler) , word would get put into wrongWords ArrayList and simultaneously adds count of 1 in the timesWrong ArrayList.

```
private void wrongWordActionPerformed(java.awt.event.ActionEvent evt) {

        Vocab word = allWords.get(i);

        if(wrongWords.contains(word)){
// gets index of wrongWord from wrongWord arraylist then in timesWrong ArrayList,
increases the timesWrong by 1
            int index = wrongWords.indexOf(word);
            timesWrong.set(index, timesWrong.get(index)+1);
        } else{
            wrongWords.add(word);
            timesWrong.add(1);
        }
//if word is wrong and only have 3 or less words left to test
        if(allWords.size()-i <= 3){

            allWords.add(word);

        } else{
            allWords.add(i+3,word);
        }
        nextWord();
```

```
        progressBar.setMaximum(allWords.size());
        progressBar.setValue(progressBar.getValue()+1);

    }
```

endQuiz() method is called when quiz ends.

```
 public void endQuiz(){
//loops through wrongWords array list and checks with timesWrong array list if word has been incorrect 3 or less times.
        for(int i = wrongWords.size()-1;i >= 0;i--){
            if (timesWrong.get(i) <= 3){
                wrongWords.remove(i);
            }
        }
//updates prevScore for text object
        text.setPrevScore((score / totalWords) * 100);
        // updates prevScore in database
        d.updateScore(text);
// removes word from arrayList then prints it on QuizEnd page
        this.dispose();
        QuizEnd obj = new QuizEnd(wrongWords,score,totalWords);
        obj.setVisible(true);
        }
```



'quizScore' variable that is storing the parentage scored in the quiz. It is calculated by getting the score variable divided by the size of the totalWords Array List times 100. This can be seen on the program at the left hand side of the program as 'Percent: 85%".

```
public QuizEnd(ArrayList<Vocab> wrongWords, int score, int totalWords) {
        initComponents();
        this.wrongWords = wrongWords;
        this.score.setText("Score: " + Integer.toString(score) + "/" + Integer.toString(totalWords));

        int quizScore = (int) ((1.0*score / totalWords) * 100);
        this.percent.setText("Percent: " + quizScore + "%");

        if (quizScore < 60){
            this.message.setText("加油! Learn the words in the table and continue working hard!");
        } else {
            this.message.setText("恭喜! Keep up the good work.");
        }

        DefaultTableModel model = (DefaultTableModel) wTable.getModel();
        for(int i=0; i < wrongWords.size(); i++){
            Vocab ww = wrongWords.get(i);
            String [] row = {ww.word};
             model.addRow(row);
        }
```

If-else statement that determines the motivational statement that would be printed. This is based on the score of the variable 'quizScore'.

Loops through the wrongWords ArrayList and gets each word from the wrongWords ArrayList and then prints it onto the 'Words to work on' table.

Running Dynamic Clock (Infinite loop, Getters, and Setters)

Had to import java.util.Calendar and java.util.GregorianCalendar to create a running dynamic clock as seen on the top right of the Home Page that had an infinite for loop.


```
public void CurrentDate(){

        Thread clock = new Thread(){
            public void run(){
// try and catch is used to ensure user is able to still able to run the
application in the case where the gaining connection fails
                try {
                    for(;;){
                    Calendar cal = new GregorianCalendar();
                    int month = cal.get(Calendar.MONTH);
                    int year = cal.get(Calendar.YEAR);
                    int day = cal.get(Calendar.DAY_OF_MONTH);
                    date_txt.
setText("Date:"+day+"/"+(month+1)+"/"+year);

                    int second = cal.get(Calendar.SECOND);
                    int minute = cal.get(Calendar.MINUTE);
                    int hour = cal.get(Calendar.HOUR);
                    time_txt.
setText("Time:"+hour+":"+(minute)+":"+second);
                    sleep(1000);
                    }
                } catch (InterruptedException ex) {
```

```
Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        };
        clock.start();
    }
```

Aiding Extensibility

- Comments were added throughout program to explain what methods were entailing
- Variable and Method Names given were appropriate and simple for ease of referencing

References:

1. StackOverflow Thread aided programming of dynamic running clock:. “Dynamic Clock in Java.” *Stack Overflow*, 1 Oct. 1960, stackoverflow.com/questions/2959718/dynamic-clock-in-java.
2. “An Easy Way to Master SQLite Fast.” *SQLite Tutorial*, www.sqlitetutorial.net/.
3. Rajput-JiStrategy Path planning and Destination matters in success No need to worry about in between temporary failures, et al. “Initialize an ArrayList in Java.” *GeeksforGeeks*, 11 Dec. 2018, www.geeksforgeeks.org/initialize-an-arraylist-in-java/.
4. Bhati, Satendra Singh. “Accessors And Mutators In Java.” *C# Corner*, www.c-sharpcorner.com/UploadFile/3614a6/accessors-and-mutators-in-java/.
5. About StackifyStackify provides developer teams with unparalleled visibility and insight into application health and behavior. “What Are OOP Concepts in Java? 4 Primary Concepts.” *Stackify*, 30 Apr. 2019, stackify.com/oops-concepts-in-java/.’

Word Count: 859