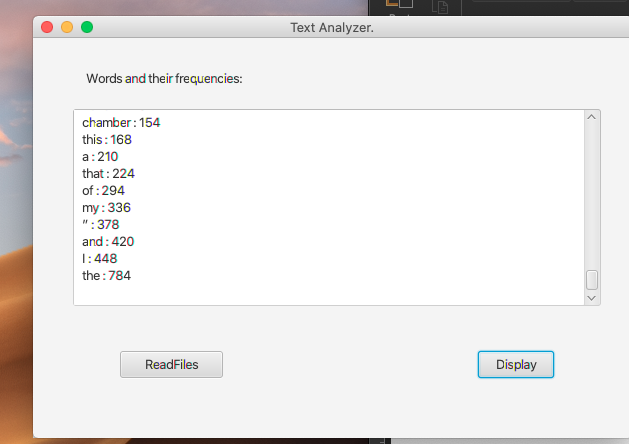
My plan for creating the application is as follows:

* Create the back-end part of the application. This involves all the controller classes, the main classes and the model classes.
* Develop and program the GUI part of the application, this involves the View part of the application.
* Integrate the GUI part of the system with the back-end codes.
* The output are the words pair and their frequencies sorted in descending fashion as shown in the screenshot below.



For the backend, I had the following files:

The main driver class as shown below.

import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class TextAnalyzer extends Application {  
 @Override  
 public void start(Stage primaryStage) throws Exception {  
 Parent root=(Parent) FXMLLoader.*load*(getClass().getResource("/view/Main.fxml"));  
 Scene scene=new Scene(root);  
 primaryStage.setScene(scene);  
 primaryStage.setTitle("Text Analyzer.");  
 primaryStage.centerOnScreen();  
 primaryStage.setResizable(false);  
 primaryStage.show();  
 }  
 public static void main(String [] args)  
 {  
 *launch*(args);  
 }  
  
}

The controller class as shown below.

package controller;  
  
import javafx.event.ActionEvent;  
import javafx.event.EventHandler;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextArea;  
  
import javax.swing.\*;  
import java.awt.event.ActionListener;  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.\*;  
  
public class Controller {  
 @FXML  
 public Button readFiles;  
 @FXML  
 public Button displayResults;  
 @FXML  
 public TextArea displayText;  
 static Map<String, Long> *counts* = new HashMap<>();  
 //method to sort HashMap values  
 private static HashMap sortValues(Map map)  
 {  
 List list = new LinkedList(map.entrySet());  
//Custom Comparator  
 Collections.*sort*(list, new Comparator()  
 {  
 public int compare(Object o1, Object o2)  
 {  
 return ((Comparable) ((Map.Entry) (o1)).getValue()).compareTo(((Map.Entry) (o2)).getValue());  
 }  
 });  
//copying the sorted list in HashMap to preserve the iteration order  
 HashMap sortedHashMap = new LinkedHashMap();  
 for (Iterator it = list.iterator(); it.hasNext();)  
 {  
 Map.Entry entry = (Map.Entry) it.next();  
 sortedHashMap.put(entry.getKey(), entry.getValue());  
 }  
 return sortedHashMap;  
 }  
 public void readFiles() throws IOException  
 {  
 // Replace the path with your own  
 File file = new File("/Users/home/Downloads/Text Analyzer/src/Gutenberg.html");  
 BufferedReader br = new BufferedReader(new FileReader(file));  
 String line;  
 while ((line = br.readLine()) != null) {  
 // Customize the regex according to your needs  
 //remove html tags  
 line = line.replaceAll("\\<.\*?\\>", "");  
 //record strings from spaces, comma, full stop and other listed symbols.  
 //Read the words from the beginning of the Poem till the end.  
 String[] words = line.split("[\\s.;,?:!()\"]+");  
 for (String word : words) {  
 word = word.trim();  
 if (word.length() > 0) {  
 if (*counts*.containsKey(word)) {  
 *counts*.put(word, *counts*.get(word) + 1);  
 } else {  
 *counts*.put(word, 1L);  
 }  
 }  
 }  
 }  
 Map<String , Long> count = *sortValues*(*counts*);  
 br.close();  
 }  
  
 @FXML  
 private void readFilesBtn(ActionEvent action)  
 {  
 readFiles.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent action) {  
 try {  
 readFiles();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
// EventHandler  
// for(int i = 0; i < counts.size(); i++)  
// {  
// readFiles();  
// }  
// }catch (IOException e)  
// {  
// JOptionPane.showMessageDialog(null,"The file couldn't be found");  
// }  
  
 }  
 @FXML  
 private void displayResults(ActionEvent action)  
 {  
 //Set the number of Rows to 10.  
 displayText.setPrefRowCount(20);  
// displayText.setText(String.valueOf(counts.size()));  
// System.out.println(counts.size());  
// System.out.println(counts.toString());  
// displayText.setText(counts.toString());  
// int i = 0;  
 Map<String , Long> count = *sortValues*(*counts*);  
 for (Map.Entry<String, Long> entry : count.entrySet()) {  
 displayText.appendText(entry.getKey() + " : " + entry.getValue() + "\n");  
// displayText.appendText(entry.getKey() + " : " + entry.getValue());  
// System.out.println(i + " " + entry.getKey() + " " + entry.getValue());  
// i++;  
 }  
 }  
  
}

I used Javafx to handle the GUI part and below are the codes for the fxml file:

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextArea?>  
<?import javafx.scene.layout.AnchorPane?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.Controller">  
 <children>  
 <TextArea fx:id="displayText" layoutX="40.0" layoutY="71.0" prefHeight="197.0" prefWidth="528.0" />  
 <Button fx:id="readFiles" layoutX="87.0" layoutY="313.0" mnemonicParsing="false" onAction="#readFilesBtn" prefHeight="25.0" prefWidth="103.0" text="ReadFiles" />  
 <Button fx:id="displayResults" layoutX="445.0" layoutY="313.0" mnemonicParsing="false" onAction="#displayResults" prefHeight="25.0" prefWidth="76.0" text="Display" />  
 <Label layoutX="53.0" layoutY="28.0" prefHeight="25.0" prefWidth="171.0" text="Words and their frequencies:" />  
 </children>  
</AnchorPane>

And this is how the GUI part looks like with two buttons, one for reading the files and the other for displaying the results..

