```java
package com.example.wordsearch;

import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.GridPane;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.*;

public class HelloController {

    @FXML
    private ListView wordsView, leaderboardView;
    @FXML
    private Button hardModeButton;
    @FXML
    private Button generalWordsButton;
    @FXML
    private Button spaceWordsButton;
    @FXML
    private Button oceanWordsButton;
    @FXML
    private Button easyModeButton;
    @FXML
    private Button hackButton;
    @FXML
    private Label timeLabel;
    @FXML
    private Label modeLabel;
    @FXML
```

```
42      private Label themeLabel;
43      @FXML
44      private Button mediumModeButton;
45      @FXML
46      private Label nameLabel;
47      @FXML
48      private TextField nameField;
49
50      private Button[][] btn;
51      private String[][] board;
52      private ArrayList<String> words = new ArrayList
    <>();
53      private ArrayList<String> dictionary = new
    ArrayList<>();
54
55      private ArrayList<String> leaderboardArray = new
    ArrayList<>();
56
57      private ArrayList<Integer> directions = new
    ArrayList<>();
58
59      @FXML
60      private AnchorPane apane;
61      @FXML
62      private GridPane searchBoard;
63
64      @FXML
65      private Label wordsLabel, leaderboardLabel;
66
67      private ArrayList<String> searchWords = new
    ArrayList<>();
68
69      int length = 12;
70      int height = 12;
71
72      private int selectedRow;
73      private int selectedColumn;
74
75      int wordStartx = -1;
76      int wordStarty = -1;
77
```

```
 78      int wordEndx = -1;
 79      int wordEndy = -1;
 80
 81      double mode = 1;
 82
 83      boolean hacked = false;
 84
 85      boolean finished = false
 86
 87      int score = 0;
 88      Timer myTimer = new Timer();
 89
 90      TimerTask myTimerTask = new TimerTask(){
 91          @Override
 92          public void run() {
 93              if (!finished) {
 94                  Platform.runLater(new Runnable() {
 95                      @Override public void run() {
 96                          timeLabel.setText("Time: "
     + time + " seconds");
 97                      }
 98                  });
 99                  System.out.println("Time: " + time
     + " seconds");
100                  time++;
101              }
102          }
103      };
104
105      int time = -1;
106
107      int timerTimes = 0;
108
109      private String name = "player";
110
111
112      @FXML
113      protected void handleClickMe(ActionEvent event
     ) {
114          dictionary.clear();
115          try{
```

```
116             FileReader reader = new FileReader("src/
    main/resources/com/example/wordsearch/words_alpha.
    txt");
117             Scanner in = new Scanner(reader);
118 
119             while(in.hasNext()){
120                 dictionary.add(in.next());
121             }
122 
123         } catch (FileNotFoundException ex){
124             System.out.println("Something is very
    wrong");
125         }
126 
127         score = 0;
128 
129         if (time == -1){
130             startTimer();
131         } else{
132             time = 0;
133         }
134 
135         finished = false;
136 
137         searchWords.clear();
138 
139         length = (int) (12 * mode);
140         height = (int) (12 * mode);
141 
142         btn = new Button[length][height];
143         board = new String[length][height];
144 
145         searchBoard.setGridLinesVisible(true);
146         searchBoard.setVisible(true);
147         for (int i = 0; i < length; i++) {
148             for (int j = 0; j < height; j++) {
149                 btn[i][j] = new Button("_");
150                 btn[i][j].setPrefSize(50,50);
151                 searchBoard.add(btn[i][j],j,i);
152             }
153         }
```

```
154
155         EventHandler z = new EventHandler<
    ActionEvent>() {
156             @Override
157             public void handle(ActionEvent event) {
158                 //all code for the buttons goes here
159                 selectedRow = GridPane.getRowIndex
    (((Button) event.getSource()));
160                 selectedColumn = GridPane.
    getColumnIndex((Button) event.getSource());
161                 checkWord(selectedRow,
    selectedColumn);
162             }
163         };
164
165         for (int i = 0; i < length; i++) {
166             for (int j = 0; j < height; j++) {
167                 btn[i][j].setOnAction(z);
168             }
169         }
170
171         //Example code will only do 1 word in 1
    direction
172         for (int i = 0; i < btn.length; i++) {
173             for (int j = 0; j < btn.length; j++) {
174                 board[i][j] = "_";
175             }
176         }
177
178
179         for (int r = 0; r < 8 * mode; r++) {
180             fillWordSearch();
181         }
182
183         finishWordSearch();
184
185         updateGraphics();
186     }
187
188     private void startTimer() {
189         timerTimes++;
```

```
190         myTimer.scheduleAtFixedRate(myTimerTask, 0,
    1000 * timerTimes);
191     }
192
193     private void checkWord(int row, int col) {
194         timeLabel.setText("Time: " + time + "
    seconds");
195         if (wordStartx == -1) {
196             wordStartx = col;
197             wordStarty = row;
198         } else if (wordEndx == -1) {
199             wordEndx = col;
200             wordEndy = row;
201
202             System.out.println(wordStartx + " " +
    wordEndx);
203             System.out.println(wordStarty + " " +
    wordEndy);
204
205             String word = "";
206
207             int changex = wordEndx - wordStartx;
208             int changey = wordEndy - wordStarty;
209             double eqchangex = 0;
210             double eqchangey = 0;
211             if (changey == 0) {
212                 if (changex > 0) {
213                     eqchangex = 1;
214                 } else {
215                     eqchangex = -1;
216                 }
217                 eqchangey = 0;
218             } else if (changex == 0) {
219                 if (changey > 0) {
220                     eqchangey = 1;
221                 } else {
222                     eqchangey = -1;
223                 }
224                 eqchangex = 0;
225             } else if (!(changex == 0 && changey ==
    0)) {
```

```
226                 eqchangex = changex / Math.abs(
    changey);
227                 eqchangey = changey / Math.abs(
    changex);
228             }
229
230
231             if (!(eqchangex == 1 || eqchangex == -1
     || eqchangex == 0)) {
232                 System.out.println("doesn't work");
233             } else if (!(eqchangey == 1 || eqchangey
     == -1 || eqchangey == 0)) {
234                 System.out.println("doesn't work");
235             } else {
236                 for (int i = 0; i <= Math.max(Math.
    abs(changex), Math.abs(changey)); i++) {
237                         word += board[(int) (
    wordStarty + eqchangey * i)][(int) (wordStartx +
    eqchangex * i)];
238                 }
239             }
240
241             System.out.println(word);
242
243             boolean wordInArray = false;
244
245             for (String testword : searchWords) {
246                 if (testword.equals(word)) {
247                     System.out.println(testword);
248                     wordInArray = true;
249                     for (int i = 0; i <= Math.max(
    Math.abs(changex), Math.abs(changey)); i++) {
250                         btn[(int) (wordStarty +
    eqchangey * i)][(int) (wordStartx + eqchangex * i)].
    setStyle("-fx-background-color: #ffff00; ");
251                     }
252                 }
253             }
254
255             if (wordInArray) {
256                 searchWords.remove(word);
```

```
257                 updateGraphics();
258             } else if (dictionary.contains(word)) {
259                 score += 50;
260                 for (int i = 0; i <= Math.max(Math.
    abs(changex), Math.abs(changey)); i++) {
261                     btn[(int) (wordStarty +
    eqchangey * i)][(int) (wordStartx + eqchangex * i)].
    setStyle("-fx-background-color: #ffff00; ");
262                 }
263             }
264 
265             wordStartx = -1;
266             wordEndx = -1;
267             System.out.println(wordStartx + " " +
    wordEndx);
268         }
269         checkEnd();
270     }
271 
272     private void checkEnd() {
273         if (searchWords.size() == 0){
274             finished = true;
275             SimpleDateFormat formatter = new
    SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
276             Date date = new Date();
277             score += (500 - time) * 2 * mode;
278             String leaderboard = name + "; Date - "
     + formatter.format(date) +  " Score - " + score;
279 
280             System.out.println(leaderboard);
281 
282             try {
283                 FileWriter output = new FileWriter("
    src/main/resources/com/example/wordsearch/output.txt
    ", true);
284 
285                 output.write(leaderboard + "\n");
286                 output.close();
287             }
288             catch(Exception e) {
289                 e.printStackTrace();
```

```
290             }
291         }
292         updateGraphics();
293     }
294 
295 
296     private void finishWordSearch() {
297         for (int i = 0; i < length; i++) {
298             for (int j = 0; j < height; j++) {
299                 if (board[i][j].equals("_")){
300                     int asciiLetter = (int)(Math.
    random()*25 + 97);
301                     char letter = (char)asciiLetter;
302                     board[i][j] = String.valueOf(
    letter);
303                 }
304             }
305         }
306     }
307 
308     private void fillWordSearch() {
309         //directions start right at 0 and goes
    counterclockwise
310         boolean boardWorks = false;
311         String currentWord;
312         directions.clear();
313         int tryCount = 0;
314         int rowStart;
315         int colStart;
316 
317         do{
318             //get random start location
319             rowStart = (int)(Math.random ()*btn.
    length);
320             colStart = (int) (Math.random ()*btn.
    length);
321 
322             currentWord = words.get((int) (Math.
    random() * words.size()));
323             while (searchWords.contains(currentWord
    )){
```

```
324                 currentWord = words.get((int) (Math.
    random() * words.size()));
325             }
326             //check if start position is empty or
    equal to the first letter of the word
327             if (board[rowStart][colStart].equals("_"
    ) || currentWord.substring(0,1).equals(board[
    rowStart][colStart])) {
328                 //choose direction right check if
    their are enough buttons in that direction
329                 if (board.length-colStart>=
    currentWord.length()){
330                     var directionWorks = true;
331                     //check if each letter of the
    word con go in the buttons in that direction
332                     for (int i = 0; i < currentWord.
    length(); i++) {
333                         if (!(board[rowStart][
    colStart + i].equals(currentWord.substring(i, i + 1
    )) || board[rowStart][colStart + i].equals("_"))){
334                             directionWorks = false;
335                         }
336                     }
337                     //if it works store that
    direction in an array
338                     if (directionWorks) {
339                         directions.add(0);
340                     }
341                 }
342             }
343             if (board[rowStart][colStart].equals("_"
    ) || currentWord.substring(0,1).equals(board[
    rowStart][colStart])) {
344                 //choose direction down check if
    their are enough buttons in that direction
345                 if (board.length-rowStart>=
    currentWord.length()){
346                     var directionWorks = true;
347                     //check if each letter of the
    word con go in the buttons in that direction
348                     for (int i = 0; i < currentWord.
```

```
348 length(); i++) {
349                             if (!(board[rowStart + i][
    colStart].equals(currentWord.substring(i, i + 1
    )) || board[rowStart + i][colStart].equals("_"))){
350                                 directionWorks = false;
351                             }
352                         }
353                         //if it works store that
    direction in an array
354                         if (directionWorks) {
355                             directions.add(1);
356                         }
357                     }
358                 }
359                 if (board[rowStart][colStart].equals("_"
    ) || currentWord.substring(0,1).equals(board[
    rowStart][colStart])) {
360                     //choose direction left check if
    their are enough buttons in that direction
361                     if (colStart>=currentWord.length()){
362                         var directionWorks = true;
363                         //check if each letter of the
    word con go in the buttons in that direction
364                         for (int i = 0; i < currentWord.
    length(); i++) {
365                             if (!(board[rowStart][
    colStart - i].equals(currentWord.substring(i, i + 1
    )) || board[rowStart][colStart - i].equals("_"))){
366                                 directionWorks = false;
367                             }
368                         }
369                         //if it works store that
    direction in an array
370                         if (directionWorks) {
371                             directions.add(2);
372                         }
373                     }
374                 }
375                 if (board[rowStart][colStart].equals("_"
    ) || currentWord.substring(0,1).equals(board[
    rowStart][colStart])) {
```

```
376                 //choose direction up check if their
     are enough buttons in that direction
377                 if (rowStart>=currentWord.length()){
378                     var directionWorks = true;
379                     //check if each letter of the
    word con go in the buttons in that direction
380                     for (int i = 0; i < currentWord.
    length(); i++) {
381                         if (!(board[rowStart-i][
    colStart].equals(currentWord.substring(i, i + 1
    )) || board[rowStart-i][colStart].equals("_"))){
382                             directionWorks = false;
383                         }
384                     }
385                     //if it works store that
    direction in an array
386                     if (directionWorks) {
387                         directions.add(3);
388                     }
389                 }
390             }
391
392             if (directions.contains(0) && directions
    .contains(1)) {
393                 //choose direction down right check
    if their are enough buttons in that direction
394                 if (board.length-colStart>=
    currentWord.length() && board.length-rowStart>=
    currentWord.length()){
395                     var directionWorks = true;
396                     //check if each letter of the
    word con go in the buttons in that direction
397                     for (int i = 0; i < currentWord.
    length(); i++) {
398                         if (!(board[rowStart+i][
    colStart+i].equals(currentWord.substring(i, i + 1
    )) || board[rowStart+i][colStart+i].equals("_"))){
399                             directionWorks = false;
400                         }
401                     }
402                     //if it works store that
```

```
direction in an array
                    if (directionWorks) {
                        directions.add(4);
                    }
                }
            }

            if (directions.contains(0) && directions
.contains(3)) {
                //choose direction down right check
if their are enough buttons in that direction
                if (board.length-colStart>=
currentWord.length() && rowStart>=currentWord.length
()){
                    var directionWorks = true;
                    //check if each letter of the
word con go in the buttons in that direction
                    for (int i = 0; i < currentWord.
length(); i++) {
                        if (!(board[rowStart-i][
colStart+i].equals(currentWord.substring(i, i + 1
)) || board[rowStart-i][colStart+i].equals("_"))){
                            directionWorks = false;
                        }
                    }
                    //if it works store that
direction in an array
                    if (directionWorks) {
                        directions.add(5);
                    }
                }
            }

            if (directions.contains(2) && directions
.contains(1)) {
                //choose direction down right check
if their are enough buttons in that direction
                if (colStart>=currentWord.length
() && board.length-rowStart>=currentWord.length()){
                    var directionWorks = true;
                    //check if each letter of the
```

```
430 word con go in the buttons in that direction
431                     for (int i = 0; i < currentWord.
    length(); i++) {
432                         if (!(board[rowStart+i][
    colStart-i].equals(currentWord.substring(i, i + 1
    )) || board[rowStart+i][colStart-i].equals("_"))){
433                             directionWorks = false;
434                         }
435                     }
436                     //if it works store that
    direction in an array
437                     if (directionWorks) {
438                         directions.add(6);
439                     }
440                 }
441             }
442 
443             if (directions.contains(2) && directions
    .contains(1)) {
444                 //choose direction down right check
    if their are enough buttons in that direction
445                 if (colStart>=currentWord.length
    () && rowStart>=currentWord.length()){
446                     var directionWorks = true;
447                     //check if each letter of the
    word con go in the buttons in that direction
448                     for (int i = 0; i < currentWord.
    length(); i++) {
449                         if (!(board[rowStart-i][
    colStart-i].equals(currentWord.substring(i, i + 1
    )) || board[rowStart-i][colStart-i].equals("_"))){
450                             directionWorks = false;
451                         }
452                     }
453                     //if it works store that
    direction in an array
454                     if (directionWorks) {
455                         directions.add(7);
456                     }
457                 }
458             }
```

```
459
460             if (directions.size()>0) {
461                 boardWorks = true;
462             }
463             tryCount++;
464         }while (!boardWorks && tryCount<10000);
465
466         if (tryCount >= 10000){
467             System.out.println(currentWord + " didn'
    t work");
468             words.remove(currentWord);
469         } else {
470             words.remove(currentWord);
471             if(directions.size()>0) {
472                 searchWords.add(currentWord);
473                 int index = (int) (Math.random() *
    directions.size());
474                 int directionUsed =  directions.get(
    index);
475                 if (directionUsed == 0) {
476                     for (int i = 0; i < currentWord.
    length(); i++) {
477                         board[rowStart][colStart + i
    ] = currentWord.substring(i, i + 1);
478                         if (hacked) {
479                             btn[rowStart][colStart
     + i].setStyle("-fx-background-color: #00ffff; ");
480                         }
481                     }
482                 }
483                 else if (directionUsed == 1) {
484                     for (int i = 0; i < currentWord.
    length(); i++) {
485                         board[rowStart + i][colStart
    ] = currentWord.substring(i, i + 1);
486                         if (hacked) {
487                             btn[rowStart + i][
    colStart].setStyle("-fx-background-color: #00ffff; "
    );
488                         }
489                     }
```

```
490                 }
491                 else if (directionUsed == 2) {
492                     for (int i = 0; i < currentWord.
    length(); i++) {
493                         board[rowStart][colStart - i
    ] = currentWord.substring(i, i + 1);
494                         if (hacked) {
495                             btn[rowStart][colStart
     - i].setStyle("-fx-background-color: #00ffff; ");
496                         }
497                     }
498                 }
499                 else if (directionUsed == 3) {
500                     for (int i = 0; i < currentWord.
    length(); i++) {
501                         board[rowStart - i][colStart
    ] = currentWord.substring(i, i + 1);
502                         if (hacked) {
503                             btn[rowStart - i][
    colStart].setStyle("-fx-background-color: #00ffff; "
    );
504                         }
505                     }
506                 }
507                 else if (directionUsed == 4) {
508                     for (int i = 0; i < currentWord.
    length(); i++) {
509                         board[rowStart + i][colStart
     + i] = currentWord.substring(i, i + 1);
510                         if (hacked) {
511                             btn[rowStart + i][
    colStart + i].setStyle("-fx-background-color: #
    00ffff; ");
512                         }
513                     }
514                 }
515                 else if (directionUsed == 5) {
516                     for (int i = 0; i < currentWord.
    length(); i++) {
517                         board[rowStart - i][colStart
     + i] = currentWord.substring(i, i + 1);
```

```
518                         if (hacked) {
519                             btn[rowStart - i][
    colStart + i].setStyle("-fx-background-color: #
    00ffff; ");
520                         }
521                     }
522                 }
523                 else if (directionUsed == 6) {
524                     for (int i = 0; i < currentWord.
    length(); i++) {
525                         board[rowStart + i][colStart
     - i] = currentWord.substring(i, i + 1);
526                         if (hacked) {
527                             btn[rowStart + i][
    colStart - i].setStyle("-fx-background-color: #
    00ffff; ");
528                         }
529                     }
530                 }
531                 else if (directionUsed == 7) {
532                     for (int i = 0; i < currentWord.
    length(); i++) {
533                         board[rowStart - i][colStart
     - i] = currentWord.substring(i, i + 1);
534                         if (hacked) {
535                             btn[rowStart - i][
    colStart - i].setStyle("-fx-background-color: #
    00ffff; ");
536                         }
537                     }
538                 }
539             }
540         }
541         updateGraphics();
542     }
543 
544     protected void updateGraphics() {
545         for (int i = 0; i < board.length; i++) {
546             for (int j = 0; j < board.length; j++) {
547                 btn[i][j].setText(board[i][j]);
548             }
```

```
549         }
550         wordsView.getItems().clear();
551         wordsView.getItems().addAll(searchWords);
552 
553         leaderboardArray.clear();
554         try{
555             FileReader reader = new FileReader("src/
    main/resources/com/example/wordsearch/output.txt");
556             Scanner in = new Scanner(reader);
557 
558             while(in.hasNext()){
559                 leaderboardArray.add(in.nextLine());
560             }
561 
562         } catch (FileNotFoundException ex){
563             System.out.println("Something is very
    wrong");
564         }
565 
566         leaderboardView.getItems().clear();
567         leaderboardView.getItems().addAll(
    leaderboardArray);
568     }
569 
570 
571     @FXML
572     protected void handleModeHard() {
573         mode = 1.5;
574     }
575 
576     @FXML
577     protected void handleModeMedium() {
578         mode = 1;
579     }
580 
581     @FXML
582     protected void handleModeEasy() {
583         mode = 0.75;
584     }
585 
586     @FXML
```

```
587     protected void handleSetGeneralWords() {
588         words.clear();
589         try{
590             FileReader reader = new FileReader("src/
    main/resources/com/example/wordsearch/~
    1000CommonWords.txt");
591             Scanner in = new Scanner(reader);
592 
593             while(in.hasNext()){
594                 words.add(in.next());
595             }
596 
597         } catch (FileNotFoundException ex){
598             System.out.println("Something is very
    wrong");
599         }
600     }
601 
602     @FXML
603     protected void handleSetSpaceWords() {
604         words.clear();
605         try{
606             FileReader reader = new FileReader("src/
    main/resources/com/example/wordsearch/SpaceWords.txt
    ");
607             Scanner in = new Scanner(reader);
608 
609             while(in.hasNext()){
610                 words.add(in.next());
611             }
612 
613         } catch (FileNotFoundException ex){
614             System.out.println("Something is very
    wrong");
615         }
616     }
617 
618     @FXML
619     protected void handleSetOceanWords() {
620         words.clear();
621         try{
```

```
622             FileReader reader = new FileReader("src/
    main/resources/com/example/wordsearch/OceanWords.txt
    ");
623             Scanner in = new Scanner(reader);
624 
625             while(in.hasNext()){
626                 words.add(in.next());
627             }
628 
629         } catch (FileNotFoundException ex){
630             System.out.println("Something is very
    wrong");
631         }
632     }
633 
634     @FXML
635     protected void handlehackedMode() {
636         hacked = !hacked;
637     }
638 
639     @FXML
640     protected void handleGetName() {
641         name = nameField.getText();
642     }
643 }
644 
```

```java
package com.example.wordsearch;

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

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 1000, 1000);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

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