```
1   //package assign1;
2
3   /**
4    *
5    * @author ln13ot
6    */
7   import BasicIO.ASCIIDataFile;
8
9   /**
10   * This program search the text file "wordsearch.dat" as a word Search, reads
11   * the file into a char 2D array Display the result when done searching
12   *
13   *
14   * @author Long Nguyen Student# 5427059
15   *
16   * @version 1.0 (Jan. 2014)
17   *
18   * */
19
20  public class Assignment1{
21
22   // Class attributes:
23
24   ASCIIDataFile in = new ASCIIDataFile("wordsearch.dat"); //read to be read in
25
26   public char[][] wordArray = new char[21][]; //A ragged 2d array to sort in the
    words
27   public int wordRowNumber = 0; // This is a counter for wordArray look up
28   public char[][] boardArray = new char[25][25]; // A 2d array for reading in the
    board with the random charters
29   public char[][] outPutDisplayArray = new char[25][25]; // A 2d array for
    outputting the final result when the word is found
30   public int row, column, letter;
31   public String wordToBeSearch;
32   public boolean wordFound = false; // flag if word is found
33   public Letters[][] character = new Letters[25][80]; // Letter Class
34   char firstLetterOfTheWord; // the first letter of the word
35
36   // Class constructor
37   public Assignment1() {
38
39    // These methods are called in the constructor
40
41    ReadFile(); //Method to read in the file
42    SortLettersInBoard(); //Method to sort every character in the boardArray in
    alphabet order, by making a 2d array of the letter class
43
44    wordToBeSearch = getWord(); // Words to be search
45    int wordCounter = 0;  // counter for the while loop
46
47    while(wordCounter < 21){
48    firstLetterOfTheWord = wordToBeSearch.toUpperCase().charAt(0); //getting the
    first character of the string
49    int letterSearch = (int) firstLetterOfTheWord; //casting the first letter to an
    integer
50
51    for (int i = 0; i < character.length; i++) { // looping throught the // character
    Array to // find the letters in // the word Search to
52     // be // search
53
54     for (int j = 0; j < character[i].length; j++) {
55      /*if the character class column index is not out of bounds and the letter
56      is equal to the same letter as the first letter to be search */
57      if (character[i][j] != null && character[i][j].getLetter() == letterSearch) {
58
59          if (wordFound == true  && wordRowNumber < 21) { //If the word is found, get
    a new word
60              wordToBeSearch = getWord();
61           }
62
```

```
63      int rowIndex = character[i][j].getRow(); //getting the character class row
    index
64      int columnIndex = character[i][j].getColumn(); //getting the character class
    column index
65
66      readForWards(wordToBeSearch, rowIndex, columnIndex); //Method looks for the
    word to read for ward, by the index of the rows and columns
67      readBackWards(wordToBeSearch, rowIndex, columnIndex);  //Method looks for the
    word  to read back ward, by the index of the rows and columns
68
69      readColumnDown(wordToBeSearch, rowIndex, columnIndex); //Method looks for the
    word  to read column down, by the index of the rows and columns
70      readColumnUP(wordToBeSearch, rowIndex, columnIndex);  //Method looks for the
    word   to read column up, by the index of the rows and columns
71
72      readRightDownDiagonal(wordToBeSearch, rowIndex, columnIndex); //Method looks
    for the word  to read right down Diagonal, by the index of the rows and columns
73      readRightUpDiagonal(wordToBeSearch, rowIndex, columnIndex);  //Method looks
    for the word  to read right up Diagonal, by the index of the rows and columns
74
75      readLeftDownDiagonal(wordToBeSearch, rowIndex, columnIndex); //Method looks
    for the word  to read left down Diagonal, by the index of the rows and columns
76      readLeftUpDiagonal(wordToBeSearch, rowIndex, columnIndex); //Method looks for
    the word  to read left up Diagonal, by the index of the rows and columns
77      }//end if
78
79     }//end for loop
80    }//end for loop
81   wordCounter ++;
82   }
83
84   PrintDisplay();
85
86  }
87
88  private String getWord() { // getting the word from the word Array and  making
    into a string
89
90   String word = "";// clear for white space
91
92   for (int i = 0; i < wordArray[wordRowNumber].length; i++) {
93    word += Character.toString(wordArray[wordRowNumber][i]);
94   }
95   wordFound = false;
96   word.replaceAll("\\s+", "");// replacing the white spaces
97    wordRowNumber++; // word counter
98         return word;
99  }
100
101 private void SortLettersInBoard() {
102
103  char c; // character for checking letters and converting it to ASCII
104
105  int letter = 65; // starting at A which is 65 and increasing to get the
106       // next value
107  int index = 0; // counters for adding new letters to the rows
108  int addedLetter = 0; // counter for adding the same letters
109
110  while (letter <= 90) { // The end of the ASCII character 'Z' which is 90
111   for (int i = 0; i < boardArray.length; i++) { //loop through the boardArray Row
112    for (int j = 0; j < boardArray[i].length; j++) { //loop through the boardArray
    Column
113     c = (char) letter; // casing the letter to the equivalent ASCII character
114     // System.out.print(c);
115     if (c == boardArray[i][j]) { // If the character is the same letter,
116             // as the board Array character
117             // created a new  character class  2d Array of objects class with the
    attributes of the "ROW", "COLUMN" and "CHARTACTER"
118      character[index][addedLetter] = new Letters(i, j,(int) c); // Creating 2D
    array of character object
```

```
119       addedLetter++; // add new columns once added the new Object is created
120      }// End if statement
121
122     }// end forloop
123    }// end while loop
124    addedLetter = 0;//Resetting the column back to zero
125    index++; // Indexing for the next new letter
126    letter++;// counter for added the next ASSCII letter
127   }
128  }
129
130  //This printDisplay Method prints out the 2D array after it finished searching for the words
131  private void PrintDisplay() {
132
133   //prints out how many words it found
134   System.out.println("Found:" + wordRowNumber + "\n");
135
136   //looping through the rowIndex
137   for (int i = 0; i < outPutDisplayArray.length; i++) {
138    //looping through the columnIndex
139    for (int j = 0; j < outPutDisplayArray[i].length; j++) {
140     if (outPutDisplayArray[i][j] == '\0') {
141      System.out.print("  ");
142     } else {
143      System.out.print(outPutDisplayArray[i][j]);
144     }
145    }
146    System.out.println(" ");
147   }
148  }
149
150  //This method reads in the file line by line and converting it to char
151  private void ReadFile() {
152
153   // reading the 21 words into the array list
154   for (int i = 0; i < wordArray.length; i++) {
155    wordArray[i] = in.readLine().toCharArray();
156   }
157   for (int i = 0; i < boardArray.length; i++) {
158    boardArray[i] = in.readLine().toCharArray();
159   }
160   in.close();
161
162  }
163
164  /*This method reads the letter that it is searching for for ward, by taking in the string of the word,
165   * and the row and column of the letter in the boardArray */
166  private void readForWards(String wordToBeSearch, int row, int col) {
167
168   String word = wordToBeSearch;
169   int rowIndex = row;
170   int columnIndex = col;
171
172   String wordMatch = "";
173
174   int wordLength = word.length(); // reinitialize to check every time
175   int outPutDisplayArrayRowIndex = rowIndex; //row index of the letter to be printed
176   int outPutDisplayArrayColumnIndex = columnIndex;  //column index of the letter to be printed
177
178   /*This is the code for searching for ward and making sure the column index is not out of bound*/
179   while (wordLength != 0 && columnIndex < 25) {
180
181    //This is concatenate the char into a string
182    wordMatch += Character.toString(boardArray[rowIndex][columnIndex]);
183
```

```
184    columnIndex++; //increment the column index
185    wordLength--;  //decrement the word length index
186
187   }// /END WHILE LOOP
188
189   //checking if the word matches the in the board array
190   if (word.equalsIgnoreCase(wordMatch)) {
191    wordFound = true; //boolean to set to true
192
193    int outPutDisplayArrayWordLength = word.length(); // reinitialize  to check
    everytime to print the word that's found letter on
194
195    //This code prints out the word in the outPutDisplayArray
196    while (outPutDisplayArrayWordLength != 0
197      && outPutDisplayArrayColumnIndex < 25) {
198
199     outPutDisplayArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex] =
    boardArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex];
200
201     outPutDisplayArrayColumnIndex++; // Increasing the  row by one
202
203     outPutDisplayArrayWordLength--;
204
205    }//end while loop
206
207   }//end if
208  }// end of Method
209
210  /*This method reads the letter that it is searching for back wards, by taking in
    the string of the word,
211   * and the row and column of the letter in the boardArray */
212 private void readBackWards(String wordToBeSearch, int row, int col) {
213
214   String word = wordToBeSearch;
215   int rowIndex = row;
216   int columnIndex = col;
217
218   String wordMatch = "";
219
220   int wordLength = word.length(); // reinitialize to check every time
221   int outPutDisplayArrayRowIndex = rowIndex; //row index of the letter to be
    printed
222   int outPutDisplayArrayColumnIndex = columnIndex;   //column index of the letter
    to be printed
223
224   // This code does the searching backward
225   while (wordLength != 0 && columnIndex >= 0) {
226
227    //This is concatenate the char into a string
228    wordMatch += Character.toString(boardArray[rowIndex][columnIndex]); //
229    columnIndex--; //decrement the column index
230    wordLength--; //decrement the word length index
231
232   }// /END WHILE LOOP
233
234   //checking if the word matches the in the board array
235   if (word.equalsIgnoreCase(wordMatch)) {
236    wordFound = true; //boolean to set to true
237
238    int outPutDisplayArrayWordLength = word.length(); // reinitialize  to check
    every time
239
240    //This code prints out the word in the outPutDisplayArray
241    while (outPutDisplayArrayWordLength != 0 && outPutDisplayArrayColumnIndex >= 0)
    {
242
243     outPutDisplayArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex] =
    boardArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex];
244
245     outPutDisplayArrayColumnIndex--; // decreasing the column by one
```

```
246     outPutDisplayArrayWordLength--; // decreasing array word lenght by one
247
248    }//end while loop
249   }//end if
250  }//end method
251
252 /*This method reads the letter that it is searching for column down, by taking in
    the string of the word,
253  * and the row and column of the letter in the boardArray */
254  private void readColumnDown(String wordToBeSearch, int row, int col) {
255
256   String word = wordToBeSearch;
257   int rowIndex = row;
258   int columnIndex = col;
259
260   String wordMatch = "";
261
262   int wordLength = word.length(); // reinitialize to check every time
263   int outPutDisplayArrayRowIndex = rowIndex; //row index of the letter to be
    printed
264   int outPutDisplayArrayColumnIndex = columnIndex; //column index of the letter to
    be printed
265
266   // This code does the searching column down
267   while (wordLength != 0 && rowIndex < 25) {
268
269    //This is concatenate the char into a string
270    wordMatch += Character.toString(boardArray[rowIndex][columnIndex]);
271              rowIndex++; //increment the row index
272    wordLength--; //decrement the word length index
273
274   }// /END WHILE LOOP
275
276   //checking if the word matches the in the board array
277   if (word.equalsIgnoreCase(wordMatch)) {
278    wordFound = true; //boolean to set to true
279
280    int outPutDisplayArrayWordLength = word.length(); // reinitialize  to check
    every time
281
282    //This code prints out the word in the outPutDisplayArray
283    while (outPutDisplayArrayWordLength != 0
284      && outPutDisplayArrayRowIndex < 25) {
285
286     outPutDisplayArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex] =
    boardArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex];
287     outPutDisplayArrayRowIndex++; // Increasing the row
288     outPutDisplayArrayWordLength--; // decrease the the word
    outPutDisplayArrayWordLength
289
290    }//end while
291
292   }//end if
293  }//end method
294
295  /*This method reads the letter that it is searching for column up, by taking in
    the string of the word,
296   * and the row and column of the letter in the boardArray */
297  private void readColumnUP(String wordToBeSearch, int row, int col) {
298
299   String word = wordToBeSearch;
300   int rowIndex = row;
301   int columnIndex = col;
302
303   String wordMatch = "";
304
305   int wordLength = word.length(); // reinitialize to check every time
306   int outPutDisplayArrayRowIndex = rowIndex;  //row index of the letter to be
    printed
307   int outPutDisplayArrayColumnIndex = columnIndex;  //column index of the letter to
```

```
be printed
308
309   // This code does the searching column up
310   while (wordLength != 0 && rowIndex >= 0) {
311
312    //This is concatenate the char into a string
313    wordMatch += Character.toString(boardArray[rowIndex][columnIndex]);
314    rowIndex--; //increment the row index
315    wordLength--; //decrement the word length index
316
317   }// /END WHILE LOOP
318
319   //checking if the word matches the in the board array
320   if (word.equalsIgnoreCase(wordMatch)) {
321    wordFound = true; //boolean to set to true
322
323    int outPutDisplayArrayWordLength = word.length(); // reinitialize to check
every time
324
325
326    //This code prints out the word in the outPutDisplayArray
327    while (outPutDisplayArrayWordLength != 0
328      && outPutDisplayArrayRowIndex >= 0) {
329
330     outPutDisplayArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex] =
boardArray[outPutDisplayArrayRowIndex][outPutDisplayArrayColumnIndex];
331
332     outPutDisplayArrayRowIndex--; // decreasing the row by one
333     outPutDisplayArrayWordLength--; // decreasing word lenght
334
335    }//while loop
336   }//end if
337  }//end method
338
339  /*This method reads the letter that it is searching right down diagonal, by taking
in the string of the word,
340   * and the row and column of the letter in the boardArray */
341  private void readRightDownDiagonal(String wordToBeSearch, int row, int col) {
342
343   String word = wordToBeSearch;
344   int rowIndex = row;
345   int columnIndex = col;
346
347   String wordMatch = "";
348
349   int wordLength = word.length(); // reinitialize to check  every time
350   int outPutDisplayArrayrowIndex = rowIndex; //row index of the letter to be
printed
351   int outPutDisplayArraycolumnIndex = columnIndex; //column index of the letter to
be printed
352
353
354   // This code does the searching right down diagonal
355   while (wordLength != 0 && rowIndex < 25 && columnIndex < 25) {
356
357    //This is concatenate the char into a string
358    wordMatch += Character.toString(boardArray[rowIndex][columnIndex]);
359    columnIndex++; //increment the column index
360    rowIndex++; //increment the row index
361    wordLength--; //decrement the word length
362
363   }// /END WHILE LOOP
364
365   //checking if the word matches the in the board array
366   if (word.equalsIgnoreCase(wordMatch)) {
367    wordFound = true; //boolean to set to true
368
369    int outPutDisplayArraywordLength = word.length(); // reinitialize to check every
time
370
```

```
371    //This code prints out the word in the outPutDisplayArray
372    while (outPutDisplayArraywordLength != 0
373      && outPutDisplayArrayrowIndex < 25
374      && outPutDisplayArraycolumnIndex < 25) {
375
376     outPutDisplayArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex] =
    boardArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex];
377
378     outPutDisplayArrayrowIndex++; // increasing the row by one
379     outPutDisplayArraycolumnIndex++; // Increasing the column by one
380
381     outPutDisplayArraywordLength--;
382    }//end while loop
383   }//end if
384  }//end method
385
386  /*This method reads the letter that it is searching right up diagonal, by taking
    in the string of the word,
387   * and the row and column of the letter in the boardArray */
388  private void readRightUpDiagonal(String wordToBeSearch, int row, int col) {
389
390   String word = wordToBeSearch;
391   int rowIndex = row;
392   int columnIndex = col;
393
394   String wordMatch = "";
395
396   int wordLength = word.length(); // reinitialize to check every time
397   int outPutDisplayArrayrowIndex = rowIndex;  //row index of the letter to be
    printed
398   int outPutDisplayArraycolumnIndex = columnIndex; //column index of the letter to
    be printed
399
400   // This code does the searching right up diagonal
401   while (wordLength != 0 && rowIndex != 0 && columnIndex < 25) {
402
403    //This is concatenate the char into a string
404    wordMatch += Character.toString(boardArray[rowIndex][columnIndex]);
405    rowIndex--; //row the row index
406    columnIndex++; //increment the column index
407    wordLength--; //decrement the column index
408
409   }// /END WHILE LOOP
410
411   //checking if the word matches the in the board array
412   if (word.equalsIgnoreCase(wordMatch)) {
413    wordFound = true;  //boolean to set to true
414
415
416    int outPutDisplayArraywordLength = word.length(); // reinitialize to check every
    time
417
418    //This code prints out the word in the outPutDisplayArray
419    while (outPutDisplayArraywordLength != 0
420      && outPutDisplayArrayrowIndex != 0
421      && outPutDisplayArraycolumnIndex < 25) {
422
423     outPutDisplayArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex] =
    boardArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex];
424
425     outPutDisplayArrayrowIndex--; // decreasing the row by one
426     outPutDisplayArraycolumnIndex++; // increasing the column by one
427     outPutDisplayArraywordLength--;// decreasing the word lenght by one
428    }//end while
429   }//end if
430  }//end method
431
432  /*This method reads the letter that it is searching left down diagonal, by taking
    in the string of the word,
433   * and the row and column of the letter in the boardArray */
```

```
 private void readLeftDownDiagonal(String wordToBeSearch, int row, int col) {

  String word = wordToBeSearch;
  int rowIndex = row;
  int columnIndex = col;

  String wordMatch = "";

  int wordLength = word.length(); // reinitialize to check every time
  int outPutDisplayArrayrowIndex = rowIndex;  //row index of the letter to be printed
  int outPutDisplayArraycolumnIndex = columnIndex; //column index of the letter to be printed

  // This code does the searching right down diagonal
  while (wordLength != 0 && rowIndex < 25 && columnIndex != 0) {

   //This is concatenate the char into a string
   wordMatch += Character.toString(boardArray[rowIndex][columnIndex]);
   rowIndex++; //increment the row index
   columnIndex--; //decrement the column index
   wordLength--; //decrement the word index

  }// /END WHILE LOOP

  //checking if the word matches the in the board array
  if (word.equalsIgnoreCase(wordMatch)) {
   wordFound = true; //boolean to set to true

   int outPutDisplayArraywordLength = word.length(); // reinitialize to check every time

   //This code prints out the word in the outPutDisplayArray
   while (outPutDisplayArraywordLength != 0
     && outPutDisplayArrayrowIndex < 25
     && outPutDisplayArraycolumnIndex != 0) {

    outPutDisplayArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex] = boardArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex];

    outPutDisplayArrayrowIndex++; // Increasing the row  by one
    outPutDisplayArraycolumnIndex--; // Deceasing the column by one
    outPutDisplayArraywordLength--; // Deceasing the word by one
   }//end while
  }//end if
 }//end method

 /*This method reads the letter that it is searching right up diagonal, by taking in the string of the word,
  * and the row and column of the letter in the boardArray */
   private void readLeftUpDiagonal(String wordToBeSearch, int row, int col) {

  String word = wordToBeSearch;
  int rowIndex = row;
  int columnIndex = col;

  String wordMatch = "";

  int wordLength = word.length(); // reinitialize to check everytime
  int outPutDisplayArrayrowIndex = rowIndex;
  int outPutDisplayArraycolumnIndex = columnIndex;

  // This code does the searching left up diagonal
  while (wordLength != 0 && rowIndex != 0 && columnIndex != 0) {
   //This is concatenate the char into a string
   wordMatch += Character.toString(boardArray[rowIndex][columnIndex]);
   rowIndex--;  //decrement the row index
   columnIndex--; //decrement the column index
   wordLength--; //decrement the word lenght

```

```
499   }// /END WHILE LOOP
500
501   //checking if the word matches the in the board array
502   if (word.equalsIgnoreCase(wordMatch)) {
503    wordFound = true; //boolean to set to true
504
505    int outPutDisplayArraywordLength = word.length(); // reinitialize  to check
    every time
506
507    //This code prints out the word in the outPutDisplayArray
508    while (outPutDisplayArraywordLength != 0
509      && outPutDisplayArrayrowIndex != 0
510      && outPutDisplayArraycolumnIndex != 0) {
511
512     outPutDisplayArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex] =
    boardArray[outPutDisplayArrayrowIndex][outPutDisplayArraycolumnIndex];
513
514     outPutDisplayArrayrowIndex--; // decreasing the row  by one
515     outPutDisplayArraycolumnIndex--; // decreasing the column by one
516     outPutDisplayArraywordLength--;
517    }//end while loop
518   }//end if
519  }//end method
520
521    /*This is the class letters, the purpose of this class is to help sort All the
    letters in order by A,B,C,D.....etc..
522     *And add in the location of the rows and columns of where the letters are in
    boardArray on the word search for a easier way to search*/
523
524    public class Letters {
525
526       // Class attributes:
527   private int rowLocation;
528   private int columnLocation;
529   private int letterInASCII;
530
531   // Class constructor
532      public Letters(int row, int column, int letter) {
533       rowLocation = row;
534       columnLocation = column;
535       letterInASCII = letter;
536      }
537
538      //method of the Letter class
539      public int getRow() {
540       return rowLocation;
541      }
542
543      public int getColumn() {
544       return columnLocation;
545      }
546
547      public int getLetter() {
548       return  letterInASCII;
549      }
550
551  }
552
553  public static void main(String[] args) {
554
555   new Assignment1();
556
557  }
558
559 }
```