```java
import BasicIO.*;

/** This program will  solve any word search puzzle given the input is in
 * the form of included wordSearch.dat. The program will find words forwards,
 * backwards, up, down, and on all four diagonals. The private integer
 * variables 'numWords' and 'boardSize' must be changed if the number of
 * words or board size changes.
 *
 * * @author Matt Laidman
 *
 * @version 1.0 (January 2013)

public class wordSearch {

    private int numWords = 21;
    private int boardSize = 25;

    private char[][] words = new char[21][];
    private char[][] board = new char[boardSize][boardSize];
    private char[][] fBoard = new char[boardSize][boardSize];

    private int wordsFound = 0;

    public wordSearch() {
        getData();
        search();
        printSolution();
    }


    private void getData() {
        ASCIIDataFile in = new ASCIIDataFile ("wordSearch.dat");

        for (int i = 0 ; i < numWords ; i++) { // read in words
            words[i] = in.readLine().toCharArray();
        }

        for (int i = 0 ; i < boardSize ; i++) { // read in board
            board[i] = in.readLine().toCharArray();
        }
    }


    private void search() {

        char c;

        for (int wc = 0 ; wc < numWords ; wc++) { // loop through words to
  find
            c = words[wc][0];
            for (int i = 0 ; i < boardSize ; i++) { // loop through board
  looking for first letter
                for (int j = 0 ; j < boardSize ; j++) {
                    if (board[i][j] == Character.toUpperCase(c)) {
                        if (check(wc, i, j)) {
                            i = boardSize; // exit loop
                            j = boardSize;
                        } // check letter found in board for all directions
                    }
                }
            }
```

```
60          }
61      }
62
63
64      private boolean check (int wc, int i, int j) {
65          boolean match;
66
67          // check forward
68          if (j + words[wc].length <= boardSize) {
69              match = true;
70              for (int c = 0 ; c < words[wc].length ; c++) {
71                  if (board[i][j+c] != Character.toUpperCase(words[wc][c])) {
72                      match = false;
73                      break;
74                  }
75              }
76              if (match) {
77                  writeToSol (1, wc, i, j);
78                  return true;
79              }
80          }
81
82          // check backward
83          if (j - words[wc].length >= 0) {
84              match = true;
85              for (int c = 0 ; c < words[wc].length ; c++) {
86                  if (board[i][j-c] != Character.toUpperCase(words[wc][c])) {
87                      match = false;
88                      break;
89                  }
90              }
91              if (match) {
92                  writeToSol (2, wc, i, j);
93                  return true;
94              }
95          }
96
97          // check up
98          if (i - words[wc].length >= 0) {
99              match = true;
100             for (int c = 0 ; c < words[wc].length ; c++) {
101                 if (board[i-c][j] != Character.toUpperCase(words[wc][c])) {
102                     match = false;
103                     break;
104                 }
105             }
106             if (match) {
107                 writeToSol (3, wc, i, j);
108                 return true;
109             }
110         }
111
112         // check down
113         if (i + words[wc].length <= boardSize) {
114             match = true;
115             for (int c = 0 ; c < words[wc].length ; c++) {
116                 if (board[i+c][j] != Character.toUpperCase(words[wc][c])) {
117                     match = false;
118                     break;
119                 }
120             }
```

```
121             if (match) {
122                 writeToSol (4, wc, i, j);
123                 return true;
124             }
125         }
126 
127         //check diagonal up-forward
128         if ((i - words[wc].length >= 0) && (j + words[wc].length <= boardSiz
    e)) {
129             match = true;
130             for (int c = 0 ; c < words[wc].length ; c++) {
131                 if (board[i-c][j+c] != Character.toUpperCase(words[wc][c]))
132                     match = false;
133                     break;
134                 }
135             }
136             if (match) {
137                 writeToSol (5, wc, i, j);
138                 return true;
139             }
140         }
141 
142         //check diagonal up-backward
143         if ((i - words[wc].length >= 0) && (j - words[wc].length >= 0)) {
144             match = true;
145             for (int c = 0 ; c < words[wc].length ; c++) {
146                 if (board[i-c][j-c] != Character.toUpperCase(words[wc][c]))
147                     match = false;
148                     break;
149                 }
150             }
151             if (match) {
152                 writeToSol (6, wc, i, j);
153                 return true;
154             }
155         }
156 
157         //check diagonal down-forward
158         if ((i + words[wc].length <= boardSize) && (j + words[wc].length <=
159             match = true;
160             for (int c = 0 ; c < words[wc].length ; c++) {
161                 if (board[i+c][j+c] != Character.toUpperCase(words[wc][c]))
162                     match = false;
163                     break;
164                 }
165             }
166             if (match) {
167                 writeToSol (7, wc, i, j);
168                 return true;
169             }
170         }
171 
172         //check diagonal down-backward
173         if ((i + words[wc].length <= boardSize) && (j - words[wc].length >=
174             match = true;
175             for (int c = 0 ; c < words[wc].length ; c++) {
176                 if (board[i+c][j-c] != Character.toUpperCase(words[wc][c]))
177                     match = false;
178                     break;
179                 }
180             }
```

```
181             if (match) {
182                 writeToSol (8, wc, i, j);
183                 return true;
184             }
185         }
186 
187         return false;
188     }
189 
190     private void writeToSol (int c, int wc, int i, int j) {
191         wordsFound++;
192 
193         switch (c){ // write words to solution matrix
194             case 1:
195                 for (int k = 0 ; k < words[wc].length ; k++) {
196                     fBoard[i][j+k] = Character.toUpperCase(words[wc][k]);
197                 }
198                 break;
199             case 2:
200                 for (int k = 0 ; k < words[wc].length ; k++) {
201                     fBoard[i][j-k] = Character.toUpperCase(words[wc][k]);
202                 }
203                 break;
204             case 3:
205                 for (int k = 0 ; k < words[wc].length ; k++) {
206                     fBoard[i-k][j] = Character.toUpperCase(words[wc][k]);
207                 }
208                 break;
209             case 4:
210                 for (int k = 0 ; k < words[wc].length ; k++) {
211                     fBoard[i+k][j] = Character.toUpperCase(words[wc][k]);
212                 }
213                 break;
214             case 5:
215                 for (int k = 0 ; k < words[wc].length ; k++) {
216                     fBoard[i-k][j+k] = Character.toUpperCase(words[wc][k]);
217                 }
218                 break;
219             case 6:
220                 for (int k = 0 ; k < words[wc].length ; k++) {
221                     fBoard[i-k][j-k] = Character.toUpperCase(words[wc][k]);
222                 }
223                 break;
224             case 7:
225                 for (int k = 0 ; k < words[wc].length ; k++) {
226                     fBoard[i+k][j+k] = Character.toUpperCase(words[wc][k]);
227                 }
228                 break;
229             case 8:
230                 for (int k = 0 ; k < words[wc].length ; k++) {
231                     fBoard[i+k][j-k] = Character.toUpperCase(words[wc][k]);
232                 }
233                 break;
234             default:
235                 break;
236         }
237     }
238 
239     private void printSolution () {
240         ASCIIDisplayer display = new ASCIIDisplayer(boardSize+5, boardSize);
241 
```

```
242         display.show();
243         display.writeLine("Words to find: " + numWords);
244         display.writeLine("Words found: " + wordsFound);
245         display.writeLine("");
246         for (int i = 0 ; i < boardSize ; i++) {
247             for (int j = 0 ; j < boardSize ; j++) {
248                 display.writeChar(fBoard[i][j]);
249             }
250             display.writeLine("");
251         }
252     }
253 
254     public static void main(String[] args) {new wordSearch();}
255 }
```