

```
1
2
3 public class WordSearch
4 {
5     public static void main(String[] args)
6     {
7         //---- try to open dictionary file
8
9         //---- try to open grid file
10
11
12         //---- create a list and populate from dictionary file
13
14         //---- create a 2-D array and populate from grid file
15
16
17         //---- check for words horizontally L-R
18
19         //---- check for words horizontally R-L
20
21         //---- check for words vertically T-B
22
23         //---- check for words vertically B-T
24
25     }
26 }
27
28
29
30
31
```

We start with an
outline of what
needs to be done....

```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.Scanner;
4
5 public class WordSearch
6 {
7     public static void main(String[] args) throws FileNotFoundException
8     {
9         //---- try to open dictionary file
10        Scanner dict_file = new Scanner(new File(args[0]));
11
12        while (dict_file.hasNext())
13            System.out.println(dict_file.nextLine());
14
15        //---- try to open grid file
16
17
18        //---- create a list and populate from dictionary file
19
20        //---- create a 2-D array and populate from grid file
21
22
23        //---- check for words horizontally L-R
24
25        //---- check for words horizontally R-L
26
27        //---- check for words vertically T-B
28
29        //---- check for words vertically B-T
30
31    }
32
33 }
34
35

```

We try to open the dictionary file and print its contents...

```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.Scanner;
4
5 public class WordSearch
6 {
7     public static void main(String[] args)
8     {
9         //---- try to open dictionary file
10        Scanner dict_file = null;
11        try {
12            dict_file = new Scanner(new File(args[0]));
13        }
14        catch (FileNotFoundException e) {
15            e.printStackTrace();
16        }
17
18        while (dict_file.hasNext())
19            System.out.println(dict_file.nextLine());
20
21        //---- try to open grid file
22
23
24        //---- create a list and populate from dictionary file
25
26        //---- create a 2-D array and populate from grid file
27
28
29        //---- check for words horizontally L-R
30
31        //---- check for words horizontally R-L
32
33        //---- check for words vertically T-B
34
35        //---- check for words vertically B-T
36
37    }
38
39 }

```

We use a try/catch block
to open the file instead...

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.Scanner;
4
5 public class WordSearch
6 {
7     public static void main(String[] args)
8     {
9         //---- try to open dictionary file
10        Scanner dict_file = null;
11        try {dict_file = new Scanner(new File(args[0]));}
12        catch (FileNotFoundException e) {e.printStackTrace();}
13
14        //---- try to open grid file
15
16
17        //---- create a list and populate from dictionary file
18
19        //---- create a 2-D array and populate from grid file
20
21
22        //---- check for words horizontally L-R
23
24        //---- check for words horizontally R-L
25
26        //---- check for words vertically T-B
27
28        //---- check for words vertically B-T
29
30    }
31
32 }
```

...a bit more compact...

```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.Scanner;
4
5 public class WordSearch
6 {
7     public static void main(String[] args)
8     {
9         //---- try to open dictionary file
10        Scanner dict_file = null;
11        try {dict_file = new Scanner(new File(args[0]));}
12        catch (FileNotFoundException e) {e.printStackTrace();}
13
14        //---- try to open grid file
15        Scanner grid_file = null;
16        try {grid_file = new Scanner(new File(args[1]));}
17        catch (FileNotFoundException e) {e.printStackTrace();}
18
19        while (grid_file.hasNext())
20            System.out.println(grid_file.nextLine());
21
22        //---- create a list and populate from dictionary file
23
24        //---- create a 2-D array and populate from grid file
25
26
27        //---- check for words horizontally L-R
28
29        //---- check for words horizontally R-L
30
31        //---- check for words vertically T-B
32
33        //---- check for words vertically B-T
34
35    }
36
37 }

```

and now the grid file...

```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;
6
7 public class WordSearch
8 {
9     public static void main(String[] args)
10    {
11        //---- try to open dictionary file
12        Scanner dict_file = null;
13        try {dict_file = new Scanner(new File(args[0]));}
14        catch (FileNotFoundException e) {e.printStackTrace();}
15
16        //---- try to open grid file
17        Scanner grid_file = null;
18        try {grid_file = new Scanner(new File(args[1]));}
19        catch (FileNotFoundException e) {e.printStackTrace();}
20
21        //---- create a list and populate from dictionary file
22        List<String> dict_words = new ArrayList<String>();
23
24        while (dict_file.hasNext())
25            dict_words.add(dict_file.nextLine());
26
27        dict_file.close();
28
29        //---- create a 2-D array and populate from grid file
30
31        //---- check for words horizontally L-R
32
33        //---- check for words horizontally R-L
34
35        //---- check for words vertically T-B
36
37        //---- check for words vertically B-T
38
39    }
40 }

```

We use a List to store the words from the dictionary file...

```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;
6
7 public class WordSearch
8 {
9     public static void main(String[] args)
10    {
11        //---- try to open dictionary file
12        Scanner dict_file = null;
13        try {dict_file = new Scanner(new File(args[0]));}
14        catch (FileNotFoundException e) {e.printStackTrace();}
15
16        //---- try to open grid file
17        Scanner grid_file = null;
18        try {grid_file = new Scanner(new File(args[1]));}
19        catch (FileNotFoundException e) {e.printStackTrace();}
20
21
22        //---- create a list and populate from dictionary file
23        List<String> dict_words = new ArrayList<String>();
24
25        while (dict_file.hasNext())
26            dict_words.add(dict_file.nextLine());
27
28        dict_file.close();
29
30        for (int i = 0; i < dict_words.size(); i++)
31            System.out.println(dict_words.get(i));
32
33        //---- create a 2-D array and populate from grid file
34
35
36        //---- check for words horizontally L-R
37
38        //---- check for words horizontally R-L
39
40        //---- check for words vertically T-B
41
42        //---- check for words vertically B-T
43
44    }
45

```

and do a quick check
to see the contents of
our list...

We notice that some
words have capital
letters and may have
spaces so...

```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;
6
7 public class WordSearch
8 {
9     public static void main(String[] args)
10    {
11        //---- try to open dictionary file
12        Scanner dict_file = null;
13        try {dict_file = new Scanner(new File(args[0]));}
14        catch (FileNotFoundException e) {e.printStackTrace();}
15
16        //---- try to open grid file
17        Scanner grid_file = null;
18        try {grid_file = new Scanner(new File(args[1]));}
19        catch (FileNotFoundException e) {e.printStackTrace();}
20
21
22        //---- create a list and populate from dictionary file
23        List<String> dict_words = new ArrayList<String>();
24
25        while (dict_file.hasNext())
26            dict_words.add(dict_file.nextLine().trim().toLowerCase());
27
28        dict_file.close();
29
30
31        //---- create a 2-D array and populate from grid file
32
33
34        //---- check for words horizontally L-R
35
36        //---- check for words horizontally R-L
37
38        //---- check for words vertically T-B
39
40        //---- check for words vertically B-T
41
42    }
43
44 }

```

We add trim and
toLowerCase...


```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;
6
7 public class WordSearch
8 {
9     public static void main(String[] args)
10    {
11        //---- try to open dictionary file
12        Scanner dict_file = null;
13        try {dict_file = new Scanner(new File(args[0]));}
14        catch (FileNotFoundException e) {e.printStackTrace();}
15
16        //---- try to open grid file
17        Scanner grid_file = null;
18        try {grid_file = new Scanner(new File(args[1]));}
19        catch (FileNotFoundException e) {e.printStackTrace();}
20
21
22        //---- create a list and populate from dictionary file
23        List<String> dict_words = new ArrayList<String>();
24
25        while (dict_file.hasNext())
26            dict_words.add(dict_file.nextLine().trim().toLowerCase());
27
28        dict_file.close();
29
30
31        //---- create a 2-D array and populate from grid file
32        int grid_rows = Integer.valueOf(grid_file.nextLine());
33        int grid_cols = Integer.valueOf(grid_file.nextLine());
34
35        char[][] grid = new char[grid_rows][grid_cols];
36
37        for (int r = 0; r < grid_rows; r++)
38        {
39            String line = grid_file.nextLine().trim().toLowerCase();
40            String[] parts = line.split(" ");
41            for (int c = 0; c < grid_cols; c++)
42                grid[r][c] = parts[c].charAt(0);
43        }
44
45        grid_file.close();
46
47

```

Now we read the
grid file for the
number of rows
and columns...

And we read the rows
of characters into a
2-D char array...

```
30 //---- create a 2-D array and populate from grid file
31 int grid_rows = Integer.valueOf(grid_file.nextLine());
32 int grid_cols = Integer.valueOf(grid_file.nextLine());
33
34 char[][] grid = new char[grid_rows][grid_cols];
35
36 for (int r = 0; r < grid_rows; r++)
37 {
38     String line = grid_file.nextLine().trim().toLowerCase();
39     String[] parts = line.split(" ");
40     for (int c = 0; c < grid_cols; c++)
41         grid[r][c] = parts[c].charAt(0);
42 }
43
44 grid_file.close();
45
46 for (int r = 0; r < grid_rows; r++)
47 {
48     for (int c = 0; c < grid_cols; c++)
49         System.out.print(grid[r][c] + " ");
50     System.out.println();
51 }
52
53
54
55
```

Do a quick check to
see that we have a
good array...

```

47 //---- check for words horizontally L-R
48 checkL2R(grid, dict_words);
49
50 //---- check for words horizontally R-L
51
52 //---- check for words vertically T-B
53
54 //---- check for words vertically B-T
55
56
57 }
58
59 // Checks Left to Right.
60 //-----
61 private static void
62 checkL2R( char[][] grid, List<String> dict_words )
63 {
64     System.out.println("inside checkL2R");
65     for (int r = 0; r < grid.length; r++)
66     {
67         for (int c = 0; c < grid[0].length; c++)
68             System.out.print(grid[r][c] + " ");
69         System.out.println();
70     }
71 }
72 }
73
74
75

```

Now we start to
check for words...

The body of the function is
initially just printing the
grid to see that it arrives
correctly...

```

47 //---- create a list of strings for words found in the grid
48 List<String> found_words = new ArrayList<String>();
49
50 //---- check for words horizontally L-R
51 checkL2R(grid, dict_words, found_words);
52
53 //---- check for words horizontally R-L
54
55 //---- check for words vertically T-B
56
57 //---- check for words vertically B-T
58
59 }
60
61 // Checks Left to Right.
62 //-----
63 private static void
64 checkL2R( char[][] grid, List<String> dict_words, List<String> found_words )
65 {
66     for (int r = 0; r < grid.length; r++)
67     {
68         String word = new String(grid[r]);
69         checkForWord(word, dict_words, found_words);
70     }
71 }
72
73 // Checks row for a word
74 //-----
75 private static void
76 checkForWord(String str, List<String> dictionary, List<String> found_words)
77 {
78     System.out.println("checking str " + str);
79 }
80 }
81
82

```

Now we go row by row and treat each row like a String and pass it to a function that will do the hard work...

```

47 //---- create a list of strings for words found in the grid
48 List<String> found_words = new ArrayList<String>();
49
50 //---- check for words horizontally L-R
51 checkL2R(grid, dict_words, found_words);
52
53 //---- check for words horizontally R-L
54
55 //---- check for words vertically T-B
56
57 //---- check for words vertically B-T
58
59 }
60
61 // Checks Left to Right.
62 //-----
63 private static void
64 checkL2R( char[][] grid, List<String> dict_words, List<String> found_words )
65 {
66     for (int r = 0; r < grid.length; r++)
67         checkForWord(new String(grid[r]), dict_words, found_words);
68 }
69
70 // Checks row for a word
71 //-----
72 private static void
73 checkForWord(String str, List<String> dictionary, List<String> found_words)
74 {
75     System.out.println("checking str " + str);
76 }
77
78 }
79
80
81

```

Compacted this...

```

47 //---- create a list of strings for words found in the grid
48 List<String> found_words = new ArrayList<String>();
49
50 //---- check for words horizontally L-R
51 checkL2R(grid, dict_words, found_words);
52
53 //---- check for words horizontally R-L
54
55 //---- check for words vertically T-B
56
57 //---- check for words vertically B-T
58
59 }
60
61 // Checks Left to Right.
62 //-----
63 private static void
64 checkL2R( char[][] grid, List<String> dict_words, List<String> found_words )
65 {
66     for (int r = 0; r < grid.length; r++)
67         checkForWord(new String(grid[r]), dict_words, found_words);
68 }
69
70 // Checks row for a word
71 //-----
72 private static void
73 checkForWord(String str, List<String> dictionary, List<String> found_words)
74 {
75     System.out.println("checking str " + str);
76     for (int i = 0; i <= str.length() - 3; i++)
77     {
78         String sub_str = str.substring(i, i + 3);
79         System.out.println(sub_str);
80     }
81 }
82 }
83

```

The min word
length is 3 so let's
look at substrings
of length 3...

```

49 //---- create a list of strings for words found in the grid
50 List<String> found_words = new ArrayList<String>();
51
52 //---- check for words horizontally L-R
53 checkL2R(grid, dict_words, found_words);
54
55 //---- check for words horizontally R-L
56
57 //---- check for words vertically T-B
58
59 //---- check for words vertically B-T
60
61 }
62
63 // Checks Left to Right.
64 //-----
65 private static void
66 checkL2R( char[][] grid, List<String> dict_words, List<String> found_words )
67 {
68     for (int r = 0; r < grid.length; r++)
69         checkForWord(new String(grid[r]), dict_words, found_words);
70 }
71
72 // Checks row for a word
73 //-----
74 private static void
75 checkForWord(String str, List<String> dictionary, List<String> found_words)
76 {
77     System.out.println("checking str " + str);
78     for (int i = 0; i <= str.length() - MIN_WORD_LENGTH; i++)
79     {
80         String sub_str = str.substring(i, i + MIN_WORD_LENGTH);
81         System.out.println(sub_str);
82     }
83 }
84 }

```

We can introduce
a class variable for
that...

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;
```

Declared up here...

```
7 public class WordSearch
8 {
9     private static final int MIN_WORD_LENGTH = 3;
10
11     public static void main(String[] args)
12     {
13         //---- try to open dictionary file
14         Scanner dict_file = null;
15         try {dict_file = new Scanner(new File(args[0]));}
16         catch (FileNotFoundException e) {e.printStackTrace();}
17
18         //---- try to open grid file
```



```

//---- create a list of strings for words found in the grid
List<String> found_words = new ArrayList<String>();

//---- check for words horizontally L-R
checkL2R(grid, dict_words, found_words);

//---- check for words horizontally R-L

//---- check for words vertically T-B

//---- check for words vertically B-T

//---- print the words
for (int i = 0; i < found_words.size(); i++)
    System.out.println(found_words.get(i));
}

// Checks Left to Right.
//-----
private static void
checkL2R( char[][] grid, List<String> dict_words, List<String> found_words )
{
    for (int r = 0; r < grid.length; r++)
        checkForWord(new String(grid[r]), dict_words, found_words);
}

// Checks str for a word in dict_words
//-----
private static void
checkForWord(String str, List<String> dict_words, List<String> found_words)
{
    for (int i = 0; i <= str.length() - MIN_WORD_LENGTH; i++)
        for (int l = MIN_WORD_LENGTH; i + l <= str.length(); l++)
        {
            String sub_str = str.substring(i, i + l);
            if (dict_words.contains(sub_str))
                found_words.add(sub_str);
        }
}
}

```

...prints the words we found...

Now we check substrings
of lengths from 3 up to
the row length...