```java


public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file

        //---- try to open grid file


        //---- create a list and populate from dictionary file

        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

}
```

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

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args) throws FileNotFoundException
    {
        //---- try to open dictionary file
        Scanner dict_file = new Scanner(new File(args[0]));

        while (dict_file.hasNext())
            System.out.println(dict_file.nextLine());

        //---- try to open grid file


        //---- create a list and populate from dictionary file

        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

}

```

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

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file
        Scanner dict_file = null;
        try {
            dict_file = new Scanner(new File(args[0]));
        }
        catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        while (dict_file.hasNext())
            System.out.println(dict_file.nextLine());

        //---- try to open grid file


        //---- create a list and populate from dictionary file

        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

}

```

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

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file
        Scanner dict_file = null;
        try {dict_file = new Scanner(new File(args[0]));}
        catch (FileNotFoundException e) {e.printStackTrace();}

        //---- try to open grid file


        //---- create a list and populate from dictionary file

        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

}
```

...a bit more compact...

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file
        Scanner dict_file = null;
        try {dict_file = new Scanner(new File(args[0]));}
        catch (FileNotFoundException e) {e.printStackTrace();}

        //---- try to open grid file
        Scanner grid_file = null;
        try {grid_file = new Scanner(new File(args[1]));}
        catch (FileNotFoundException e) {e.printStackTrace();}

        while (grid_file.hasNext())
            System.out.println(grid_file.nextLine());

        //---- create a list and populate from dictionary file

        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

}
```

and now the grid file...

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file
        Scanner dict_file = null;
        try {dict_file = new Scanner(new File(args[0]));}
        catch (FileNotFoundException e) {e.printStackTrace();}

        //---- try to open grid file
        Scanner grid_file = null;
        try {grid_file = new Scanner(new File(args[1]));}
        catch (FileNotFoundException e) {e.printStackTrace();}


        //---- create a list and populate from dictionary file
        List<String> dict_words = new ArrayList<String>();

        while (dict_file.hasNext())
            dict_words.add(dict_file.nextLine());

        dict_file.close();


        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

}
```

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

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file
        Scanner dict_file = null;
        try {dict_file = new Scanner(new File(args[0]));}
        catch (FileNotFoundException e) {e.printStackTrace();}

        //---- try to open grid file
        Scanner grid_file = null;
        try {grid_file = new Scanner(new File(args[1]));}
        catch (FileNotFoundException e) {e.printStackTrace();}


        //---- create a list and populate from dictionary file
        List<String> dict_words = new ArrayList<String>();

        while (dict_file.hasNext())
            dict_words.add(dict_file.nextLine());

        dict_file.close();

        for (int i = 0; i < dict_words.size(); i++)
            System.out.println(dict_words.get(i));

        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

```

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…

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file
        Scanner dict_file = null;
        try {dict_file = new Scanner(new File(args[0]));}
        catch (FileNotFoundException e) {e.printStackTrace();}

        //---- try to open grid file
        Scanner grid_file = null;
        try {grid_file = new Scanner(new File(args[1]));}
        catch (FileNotFoundException e) {e.printStackTrace();}


        //---- create a list and populate from dictionary file
        List<String> dict_words = new ArrayList<String>();

        while (dict_file.hasNext())
            dict_words.add(dict_file.nextLine().trim().toLowerCase());

        dict_file.close();


        //---- create a 2-D array and populate from grid file


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

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

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

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

    }

}
```

We add trim and toLowerCase...

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class WordSearch
{
    public static void main(String[] args)
    {
        //---- try to open dictionary file
        Scanner dict_file = null;
        try {dict_file = new Scanner(new File(args[0]));}
        catch (FileNotFoundException e) {e.printStackTrace();}

        //---- try to open grid file
        Scanner grid_file = null;
        try {grid_file = new Scanner(new File(args[1]));}
        catch (FileNotFoundException e) {e.printStackTrace();}


        //---- create a list and populate from dictionary file
        List<String> dict_words = new ArrayList<String>();

        while (dict_file.hasNext())
            dict_words.add(dict_file.nextLine().trim().toLowerCase());

        dict_file.close();


        //---- create a 2-D array and populate from grid file
        int grid_rows = Integer.valueOf(grid_file.nextLine());
        int grid_cols = Integer.valueOf(grid_file.nextLine());

        char[][] grid = new char[grid_rows][grid_cols];

        for (int r = 0; r < grid_rows; r++)
        {
            String line = grid_file.nextLine().trim().toLowerCase();
            String[] parts = line.split(" ");
            for (int c = 0; c < grid_cols; c++)
                grid[r][c] = parts[c].charAt(0);
        }

        grid_file.close();
```

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
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          for (int r = 0; r < grid_rows; r++)
48          {
49              for (int c = 0; c < grid_cols; c++)
50                  System.out.print(grid[r][c] + "  ");
51              System.out.println();
52          }
53
54
```

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

```
46
47
48          //---- check for words horizontally L-R
49          checkL2R(grid, dict_words);
50
51          //---- check for words horizontally R-L
52
53          //---- check for words vertically T-B
54
55          //---- check for words vertically B-T
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
```

Now we start to check for words…

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

```
46
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
```

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  }
85
86
```

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;
6
7  public class WordSearch
8  {
9      private static final int MIN_WORD_LENGTH = 3;
0
1      public static void main(String[] args)
2      {
3          //---- try to open dictionary file
4          Scanner dict_file = null;
5          try {dict_file = new Scanner(new File(args[0]));}
6          catch (FileNotFoundException e) {e.printStackTrace();}
7
8          //---- try to open grid file
```

Declared up here...

```
        //---- 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…