WordSearch.java

```
1  package Trie;
2
3  import java.util.ArrayList;
4  import java.util.HashSet;
5  import java.util.List;
6  import java.util.Set;
7
8  class WordSearch {
9      Set<String> result = null;
10     char [] [] board  = null;
11     Trie1 trie = null;
12     public List<String> findWords(char[][] board, String[] words) {
13         this.board = board;
14         result = new HashSet<>();
15         // Build Trie1 on words
16         trie = new Trie1();
17 1       for(String word: words) trie.add(word);
18         //////////
19
20         // Start recursion calls on each character in the board which  contains entry in the root of the trie.
21 2       for(int i=0;i<this.board.length;i++) {
22 2           for(int j=0; j<this.board[i].length; j++) {
23 2               if(trie.root.array[this.board[i][j] - 'a'] != null) {
24 2                   findWords(i, j, trie.root.array[this.board[i][j] - 'a'], new HashSet<String>());
25                }
26            }
27        }
28 1       return new ArrayList<>(result);
29    }
30
31
32    void findWords(int i, int j, TrieNode curr, Set<String> visited) {
33 1       if(curr.value != null) {
34            result.add(curr.value);
35        }
36        visited.add(i+"#"+j);
37        TrieNode temp = null;
38        /*
39            * Conditional statements prunes invalid branches.
40        */
41 7       if(i>0 && (temp=curr.array[board[i-1][j] -'a']) != null && !visited.contains((i-1)+"#"+j)) {
42 2           findWords(i-1, j, temp, visited);
43        }
44
45 7       if(j>0 && (temp=curr.array[board[i][j-1] -'a']) != null && !visited.contains(i+"#"+(j-1))) {
46 2           findWords(i, j-1, temp, visited);
47        }
48
49 8       if(i<board.length-1 && (temp=curr.array[board[i+1][j] -'a']) != null && !visited.contains(""+(i+1)+"#"+j)) {
50 2           findWords(i+1, j, temp, visited);
51        }
52
53 8       if(j<board[i].length-1 && (temp=curr.array[board[i][j+1] -'a']) != null && !visited.contains(i+"#"+(j+1))) {
54 2           findWords(i, j+1, temp, visited);
55        }
56        visited.remove(i+"#"+j);
57
58    }
59 }
60
61 class TrieNode {
62    TrieNode [] array;
63    String value;
64    TrieNode() {
65        array = new TrieNode[26];
66    }
67 }
68
69 class Trie1{
70    TrieNode root;
71    Trie1() {
72        root = new TrieNode();
73    }
74
75    void add(String word) {
76        TrieNode temp=root;
77 2       for(int i=0;i<word.length();i++) {
78            char ch=word.charAt(i);
79 2           if(temp.array[ch-'a'] == null) {
80 1               temp.array[ch-'a']= new TrieNode();
81 1               temp = temp.array[ch-'a'];
82            } else {
```

```
83 1                temp=temp.array[ch-'a'];
84                 }
85             }
86             temp.value=word;
87         }
88     }
```

Mutations

17	1. removed call to Trie/Trie1::add → KILLED
21	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
22	1. negated conditional → KILLED 2. changed conditional boundary → KILLED
23	1. negated conditional → KILLED 2. Replaced integer subtraction with addition → KILLED
24	1. removed call to Trie/WordSearch::findWords → KILLED 2. Replaced integer subtraction with addition → KILLED
28	1. replaced return value with Collections.emptyList for Trie/WordSearch::findWords → KILLED
33	1. negated conditional → KILLED
41	1. Replaced integer subtraction with addition → KILLED 2. Replaced integer subtraction with addition → NO_COVERAGE 3. changed conditional boundary → KILLED 4. negated conditional → KILLED 5. Replaced integer subtraction with addition → SURVIVED 6. negated conditional → KILLED 7. negated conditional → NO_COVERAGE
42	1. Replaced integer subtraction with addition → NO_COVERAGE 2. removed call to Trie/WordSearch::findWords → NO_COVERAGE
45	1. negated conditional → KILLED 2. negated conditional → KILLED 3. changed conditional boundary → KILLED 4. negated conditional → KILLED 5. Replaced integer subtraction with addition → KILLED 6. Replaced integer subtraction with addition → KILLED 7. Replaced integer subtraction with addition → KILLED
46	1. Replaced integer subtraction with addition → KILLED 2. removed call to Trie/WordSearch::findWords → KILLED
49	1. Replaced integer addition with subtraction → KILLED 2. negated conditional → KILLED 3. Replaced integer addition with subtraction → KILLED 4. Replaced integer subtraction with addition → SURVIVED 5. Replaced integer subtraction with addition → KILLED 6. changed conditional boundary → SURVIVED 7. negated conditional → KILLED 8. negated conditional → KILLED
50	1. removed call to Trie/WordSearch::findWords → KILLED 2. Replaced integer addition with subtraction → KILLED
53	1. negated conditional → KILLED 2. negated conditional → KILLED 3. negated conditional → KILLED 4. Replaced integer subtraction with addition → KILLED 5. Replaced integer subtraction with addition → KILLED 6. changed conditional boundary → KILLED 7. Replaced integer addition with subtraction → SURVIVED 8. Replaced integer addition with subtraction → KILLED
54	1. Replaced integer addition with subtraction → KILLED 2. removed call to Trie/WordSearch::findWords → KILLED
77	1. negated conditional → KILLED 2. changed conditional boundary → KILLED
79	1. Replaced integer subtraction with addition → KILLED 2. negated conditional → KILLED
80	1. Replaced integer subtraction with addition → KILLED
81	1. Replaced integer subtraction with addition → KILLED
83	1. Replaced integer subtraction with addition → NO_COVERAGE

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- Trie.WordSearchTest.main(Trie.WordSearchTest) (9 ms)

Report generated by PIT 1.15.0