See the Assessment Guide for information on how to interpret this report.

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     PASSED
PMD:          PASSED
Checkstyle:   PASSED

Correctness:  13/13 tests passed
Memory:       3/3 tests passed
Timing:       9/9 tests passed

Aggregate score: 100.00%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 60%, Timing: 10%, Memory: 20% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
5.0K Dec  9 10:54 BoggleSolver.java


================================================================================
*  COMPILING
================================================================================


% javac BoggleSolver.java
*------------------------------------------------------------


================================================================


Checking the APIs of your programs.
*------------------------------------------------------------
BoggleSolver:

================================================================


================================================================================
*  CHECKING STYLE AND COMMON BUG PATTERNS
================================================================================


% spotbugs *.class
*------------------------------------------------------------


================================================================


% pmd .
*------------------------------------------------------------


================================================================


% checkstyle *.java
```

```
*-----------------------------------------------------------


================================================================


********************************************************************************
*  TESTING CORRECTNESS
********************************************************************************

Testing correctness of BoggleSolver
*-----------------------------------------------------------
Tests 1-9 create one BoggleSolver object corresponding to the specified
dictionary and call getAllValidWords() with several different boards
as arguments.

Running 13 total tests.

Test 1: check getAllValidWords() on two fixed 4-by-4 boards given in assignment
  * dictionary = dictionary-algs4.txt; board = board4x4.txt
  * dictionary = dictionary-algs4.txt; board = board-q.txt
==> passed

Test 2: check getAllValidWords() on fixed 4-by-4 boards
  * dictionary = dictionary-yawl.txt; board = board4x4.txt
  * dictionary = dictionary-yawl.txt; board = board-points1.txt
  * dictionary = dictionary-yawl.txt; board = board-points2.txt
  * dictionary = dictionary-yawl.txt; board = board-points3.txt
  * dictionary = dictionary-yawl.txt; board = board-points4.txt
  * dictionary = dictionary-yawl.txt; board = board-points5.txt
==> passed

Test 3: check getAllValidWords() on more fixed 4-by-4 boards
  * dictionary = dictionary-yawl.txt; board = board-points100.txt
  * dictionary = dictionary-yawl.txt; board = board-points200.txt
  * dictionary = dictionary-yawl.txt; board = board-points300.txt
  * dictionary = dictionary-yawl.txt; board = board-points400.txt
  * dictionary = dictionary-yawl.txt; board = board-points500.txt
  * dictionary = dictionary-yawl.txt; board = board-points750.txt
  * dictionary = dictionary-yawl.txt; board = board-points1000.txt
  * dictionary = dictionary-yawl.txt; board = board-points1250.txt
  * dictionary = dictionary-yawl.txt; board = board-points1500.txt
  * dictionary = dictionary-yawl.txt; board = board-points2000.txt
==> passed

Test 4: check getAllValidWords() on random Hasbro boards
  * dictionary = dictionary-yawl.txt; board = 10 random Hasbro boards
  * dictionary = dictionary-yawl.txt; board = 50 random Hasbro boards
  * dictionary = dictionary-yawl.txt; board = 100 random Hasbro boards
==> passed

Test 5: check getAllValidWords() on high-scoring n-by-n boards
  * dictionary = dictionary-yawl.txt; board = board-points4410.txt
  * dictionary = dictionary-yawl.txt; board = board-points4527.txt
  * dictionary = dictionary-yawl.txt; board = board-points13464.txt
  * dictionary = dictionary-yawl.txt; board = board-points26539.txt
==> passed

Test 6: check getAllValidWords() on exotic boards
  * dictionary = dictionary-yawl.txt; board = board-dodo.txt
  * dictionary = dictionary-yawl.txt; board = board-noon.txt
  * dictionary = dictionary-yawl.txt; board = board-couscous.txt
  * dictionary = dictionary-yawl.txt; board = board-rotavator.txt
  * dictionary = dictionary-yawl.txt; board = board-estrangers.txt
  * dictionary = dictionary-yawl.txt; board = board-antidisestablishmentarianisms.txt
  * dictionary = dictionary-yawl.txt; board = board-dichlorodiphenyltrichloroethanes.txt
  * dictionary = dictionary-yawl.txt; board = board-pneumonoultramicroscopicsilicovolcanoconiosis.txt
==> passed

Test 7: check getAllValidWords() on boards with a Q
  * dictionary = dictionary-yawl.txt; board = board-qwerty.txt
  * dictionary = dictionary-yawl.txt; board = board-quinquevalencies.txt
  * dictionary = dictionary-yawl.txt; board = board-inconsequentially.txt
  * dictionary = dictionary-yawl.txt; board = board-qaimaqam.txt
```

```
  * dictionary = dictionary-yawl.txt; board = board-aqua.txt
  * dictionary = dictionary-yawl.txt; board = 100 random Hasbro boards
  * dictionary = dictionary-16q.txt; board = board-9q.txt
  * dictionary = dictionary-16q.txt; board = board-16q.txt
==> passed

Test 8: check getAllValidWords() on random m-by-n boards
  * dictionary = dictionary-common.txt; board = 100 random 3-by-3 boards
  * dictionary = dictionary-common.txt; board = 100 random 4-by-4 boards
  * dictionary = dictionary-common.txt; board = 100 random 5-by-5 boards
  * dictionary = dictionary-common.txt; board = 20 random 5-by-10 boards
  * dictionary = dictionary-common.txt; board = 20 random 10-by-5 boards
==> passed

Test 9: check getAllValidWords() on random m-by-n boards
  * dictionary = dictionary-common.txt; board = 10 random 2-by-2 boards
  * dictionary = dictionary-common.txt; board = 10 random 1-by-10 boards
  * dictionary = dictionary-common.txt; board = 10 random 10-by-1 boards
  * dictionary = dictionary-common.txt; board = 10 random 1-by-1 boards
  * dictionary = dictionary-common.txt; board = 10 random 1-by-2 boards
  * dictionary = dictionary-common.txt; board = 10 random 2-by-1 boards
==> passed

Test 10: check getAllValidWords() on boards with no valid words
  * dictionary = dictionary-nursery.txt; board = board-points0.txt
  * dictionary = dictionary-2letters.txt; board = board-points4410.txt
==> passed

Test 11: mutating dictionary[] after passing to BoggleSolver constructor
  * dictionary = dictionary-algs4.txt
  * dictionary = dictionary-algs4.txt; board = 10 random Hasbro boards
==> passed

Test 12: create more than one BoggleSolver object at a time
         [ BoggleSolver object 1 uses dictionary-algs4.txt   ]
         [ BoggleSolver object 2 uses dictionary-nursery.txt ]
  * dictionary = dictionary-algs4.txt; board = 10 random Hasbro boards
  * dictionary = dictionary-nursery.txt; board = 10 random Hasbro boards
  * dictionary = dictionary-algs4.txt; board = 10 random Hasbro boards
==> passed

Test 13: check scoreOf() on various dictionaries
  * dictionary = dictionary-algs4.txt
  * dictionary = dictionary-common.txt
  * dictionary = dictionary-shakespeare.txt
  * dictionary = dictionary-nursery.txt
  * dictionary = dictionary-yawl.txt
==> passed


Total: 13/13 tests passed!


================================================================
********************************************************************************
*  MEMORY
********************************************************************************

Analyzing memory of BoggleSolver
*-----------------------------------------------------------
Running 3 total tests.

Test 1: memory with dictionary-algs4.txt (must be <= 2x reference solution)
  * memory of dictionary[]           = 450264 bytes
  * memory of student   BoggleSolver = 4564448 bytes
  * memory of reference BoggleSolver = 5091200 bytes
  * student / reference              = 0.90
==> passed

Test 2: memory with dictionary-shakespeare.txt (must be <= 2x reference solution)
  * memory of dictionary[]           = 1754288 bytes
  * memory of student   BoggleSolver = 15323280 bytes
  * memory of reference BoggleSolver = 17306368 bytes
  * student / reference              = 0.89
```

```
==> passed

Test 3: memory with dictionary-yawl.txt (must be <= 2x reference solution)
  * memory of dictionary[]           = 20259320 bytes
  * memory of student   BoggleSolver = 154401856 bytes
  * memory of reference BoggleSolver = 176424232 bytes
  * student / reference              = 0.88
==> passed


Total: 3/3 tests passed!


================================================================



********************************************************************************
*  TIMING
********************************************************************************

Timing BoggleSolver
*-----------------------------------------------------------
All timing tests are for random 4-by-4 boards (using the Hasbro dice).
The dictionary is specified with each test.

Running 9 total tests.

Test 1: timing constructor (must be <= 5x reference solution)
 *  dictionary-algs4.txt
    - student   solution time (in seconds): 0.01
    - reference solution time (in seconds): 0.00
    - ratio:                                1.47

==> passed

 *  dictionary-enable2k.txt
    - student   solution time (in seconds): 0.02
    - reference solution time (in seconds): 0.02
    - ratio:                                0.95

==> passed

 *  dictionary-yawl.txt
    - student   solution time (in seconds): 0.02
    - reference solution time (in seconds): 0.03
    - ratio:                                0.91

==> passed

 *  dictionary-zingarelli2005.txt
    - student   solution time (in seconds): 0.05
    - reference solution time (in seconds): 0.06
    - ratio:                                0.81

==> passed

Test 2: timing getAllValidWords() for 5.0 seconds using dictionary-yawl.txt
        (must be <= 2x reference solution)
    - reference solution calls per second: 7476.57
    - student   solution calls per second: 7225.56
    - reference / student ratio:           1.03

=> passed    student <= 10000x reference
=> passed    student <=    25x reference
=> passed    student <=    10x reference
=> passed    student <=     5x reference
=> passed    student <=     2x reference


Total: 9/9 tests passed!


================================================================
```