Introduction

This programming problem (a word finding "game") is your opportunity to show us how you write code and tests when given real-world-like constraints: e.g., interfaces to implement and non-optimal data structures. Just as a graphic design firm would ask for a portfolio of work to see, we do the same thing with code.

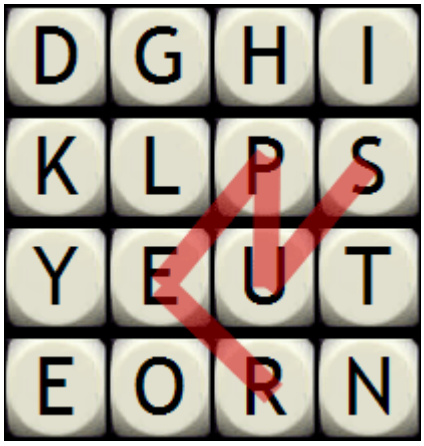
Since we can't answer questions that you might encounter while writing your code, you can make any assumptions you wish, as long as you document what assumptions you're making and why. You can document these in the code, in a text file, or in the email you send along with your coded solution.

Testing is very important to us. Our developers work closely with product management and QA so that we produce well-tested products. Therefore, our developers are expected to write and maintain tests. For this reason, we ask that you thoroughly test the code you submit. If you are not familiar with writing automated tests or have not used JUnit before, please make the attempt at writing the tests anyway. You can use the provided test classes as a template for the tests that you need to write.

To ensure that we get all of your files, please make sure to ZIP them up and try it out before sending it to us. Please do not use any third-party libraries. If you must, please include the reasoning behind your decision with your solution.

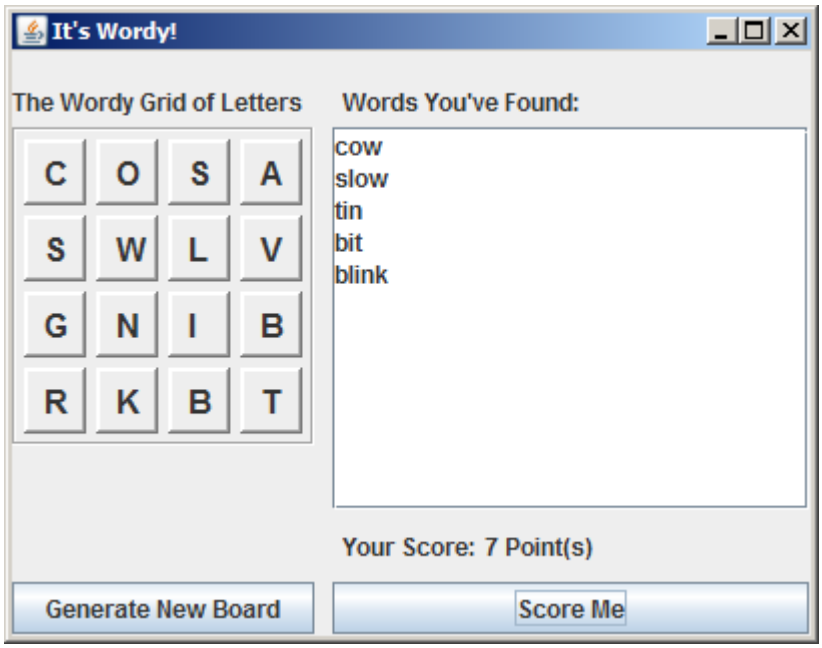
What is Wordy?

Wordy is a game consisting of a 4x4 grid of English letters where the goal is to find as many English words as possible that are 3 or more letters long. Words are formed from adjoining letters. Letters must join in the proper sequence to spell a word. They may join horizontally, vertically, or diagonally, to the left, right, or up-and-down. No letter cell may be used more than once within a single word.



Example Word: SUPER

Running the GUI with a sample implementation looks like this:



What You Need To Do

Your task is to finish the partially completed implementation of Wordy that we provide. You will need to:

- Implement IWordy
- Alter WordyGame to create your new IWordy implementation and pass it to WordyFrame
- Implement IWordInBoardValidator (the current implementation is just a stub)

You must provide tests for your implementations of both IWordy and IWordInBoardValidator. As a guide to what we're expecting, we've provided initial skeletons for some of the IWordy unit tests. You should implement these and add more, plus add a complete set of tests for IWordInBoardValidator.

You should not need to change the interfaces we provide; however, if you do please retain the original interface signatures as we will use them in our automated tests of your solution. You should not need to modify any of the other classes either; if you do please explain your reasoning.

We have provided a few tools to help you along. First, we've provided JUnit as the testing framework. Secondly, we've implemented the rest of Wordy and have provided tests to help you better understand what and how we test. Included are tested implementations of the other Wordy interfaces, plus some utility classes that may be useful in your implementation and a file of words (CROSSWD.TXT) used by the word validator to determine which strings are valid words.

IDE and Java Requirements

We've included project files for IntelliJ, NetBeans, and Eclipse. The project files have the source, test, and libraries setup for Wordy. For Eclipse you will need to import the project file into your current workspace. To run the project you will need Java 6 (1.6.0) and JUnit 4.5 (or higher). We provide a JUnit 4.5 jar with the project.

Running the UI

The provided version of WordyGame will not run because it needs an object that implements the IWordy interface. You need to provide such an object and pass it in to new WordyFrame(…). The UI can be started with just the stub implementation of IWordyInBoardValidator though, of course, it will not score correctly.

Once you have started the UI, you should be able to generate a new board of letters by clicking on the Generate Board button. You should also be able to type in a list of words on the right side and click Score Me to get the score for those words. The score should only include valid words that are actually in the board.

Note that the program assumes that the dictionary file, CROSSWD.TXT, is available in the current directory, so you'll need to run WordyGame from the directory that contains CROSSWD.TXT.

There are not intended to be any tricks to getting Wordy running; we want you to be able to concentrate on implementing the two interfaces described above, plus their tests. If you find our instructions unclear, or find errors in the IDE project files we provide, please let us know so we can fix them for future candidates.