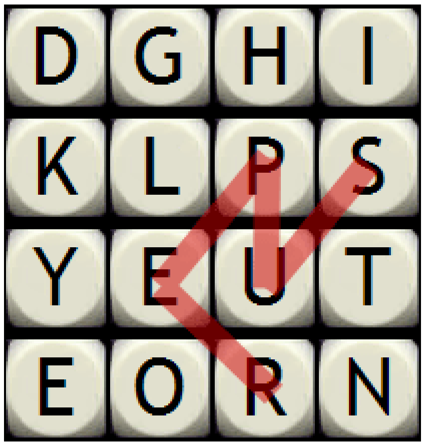
Introduction
This programming problem (a word finding "game") is your opportunity to show us how you write code and tests when given real-world-like constraints: e.g., interfaces to implement and non-optimal data structures. Just as a graphic design firm would ask for a portfolio of work to see, we do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

Since we can't answer questions that you might encounter while writing your code (otherwise we wouldn't get anything else done), 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.

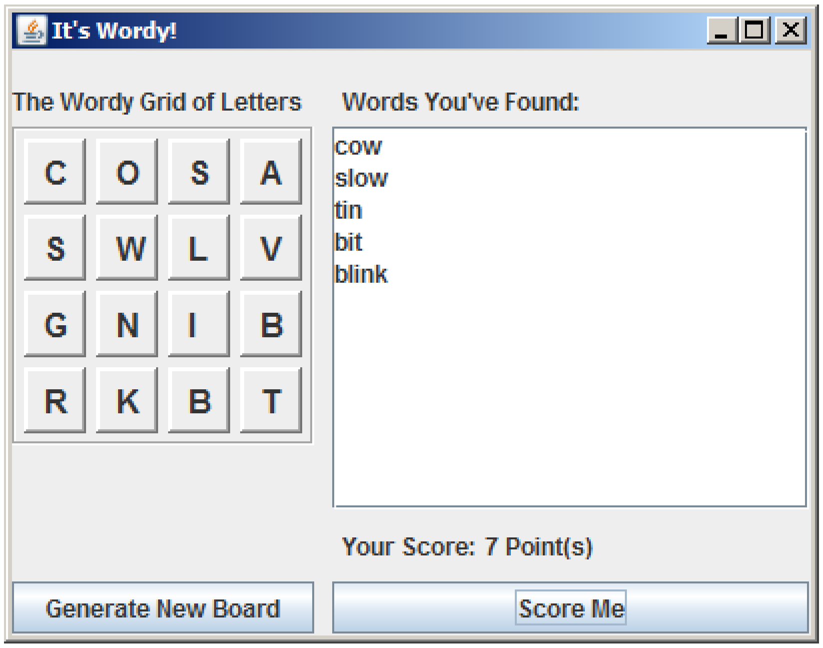
To ensure that we get all of your files, please make sure to ZIP them up and try it out before sending it to us. If you use any third-party libraries, make sure to include those files as well.

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 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:



```
cow
slow
tin
bit
blink
```

Your Score: 7 Point(s)

What You Need To Do

Your task is to write an implementation of the IWordy interface, as well as the IBoardGenerator, IWordValidator, and IWordScorer interfaces, along with tests for these classes. You can not change the interfaces themselves in any way, but you are free to add new classes and interfaces. You should also not modify the WordyFrame class (in fact, you don't even need to look at it).

Also included is a file of words (CROSSWD.TXT) that can be used by your word validator. It's a straight text file consisting of single words. It's up to you how to use this file in your implementation.

Once you have implemented an IWordy, modify the main method in WordyGame to instantiate it and then run the WordyGame via its main() method and the UI should appear.

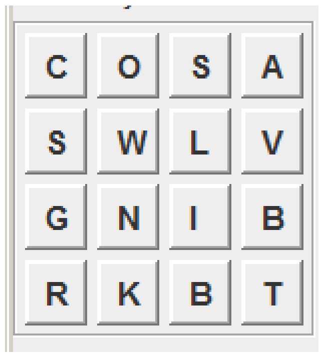
Running the UI

Once you've completed the above tasks, 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. Note that you could type in any word, even if it couldn't be found in the board, and the scorer should still give you credit.

Java Requirements
Java 6 (1.6.0) and JUnit 4.5 (or higher)

Optional/Extra Credit
If you feel like spending the extra time, you can add the following functionality:

Ensure that the words entered in the textbox are not only valid words, but also can be found in the letter board. For example, in this board:



The words "bit" and "scowl" can be found, but the words "kitten" and "tribe" cannot. In the "standard" version, all four words would get scored, but in the extra-credit version, you'd only get points for the first two.