CSE 174 – Fall 2013
PROGRAM: 41 points – Due WEDNESDAY, November 13, by 11:59 p.m.

Outcomes:

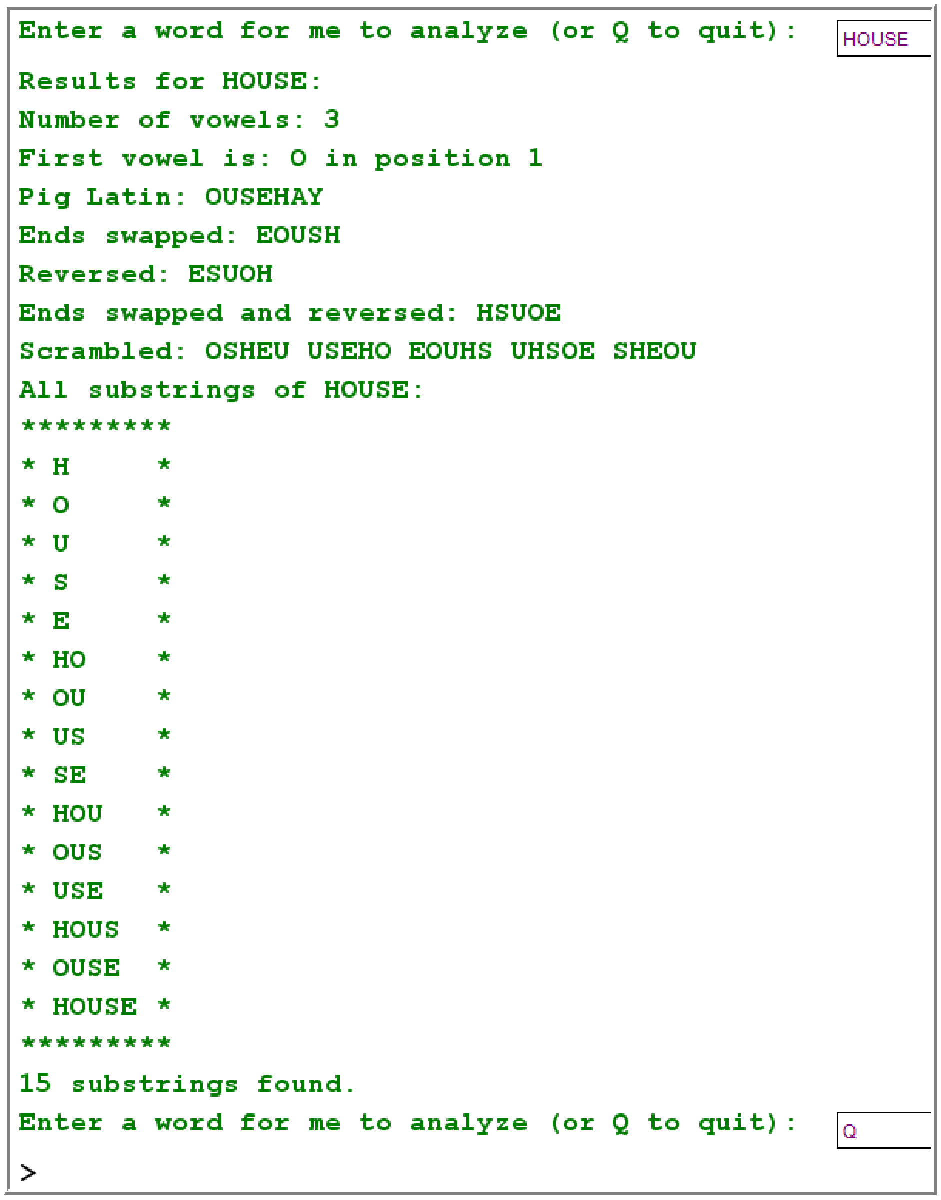
- Write programs that use loops.
- Write programs that process Strings.
- Decompose a larger program into smaller pieces that can be reused.
- Write code that emphasizes readability and understandability.

Scoring:

At a bare minimum, the source code you submit must compile, run, and use a loop to get user input, and must do at least *some* of the string processing in this assignment. Code that does not meet these minimum standards (no matter how much code was written) will earn a score of zero.

	Full credit	No credit or Partial credit
Use a loop to obtain user input (4 points)	Your program repeatedly prompts the user for a word until the user enters an uppercase Q to quit the program.	Though the program uses a loop for user input, it does not work as expected (for example, not quitting when the user types an uppercase Q).
Solve the vowel-related problems (4 points)	Your program correctly counts vowels and identifies the location of the first vowel.	There are some errors in the vowel results.
Pig latinize words (3 points)	Your program correctly pig latinizes all words as specified in the assignment.	The program does not always produce correct results.
Reverse words (2 points)	Your program correctly reverses all words as specified in the assignment.	The program does not always produce correct results.
Swap ends of words (2 points)	Your program correctly swaps the first and last letter of given words.	The program does not always produce correct results.
Scramble words (3 points)	Your program randomly scrambles given words.	The program does not always produce correct results.
Listing substrings (3 points)	Your program lists and counts all the substrings of a given word.	The program does not always produce correct results.
Formatting substrings (3 points)	Your program formats the output of the substrings as specified.	The output of the substrings is not formatted as specified.
Format output as specified (4 points)	The output of your program is formatted as shown in the examples, including any output not mentioned in other parts of the rubric.	Some output is missing or is formatted incorrectly, or there is extra output that should not have been included.
Class and method names are as specified. (5 points)	Your class and method names match what is specified in this assignment, including method return types and parameter types.	Though you have methods that perform the required functionality, the names and/or return types and/or parameter types do not follow specifications.
Write additional methods of your own to help improve code reuse and understandability. (5 points)	You wrote other methods in addition to those specified in order to make the specified methods easier to understand.	You did not write any additional methods of your own, putting all your code only in the specified methods.
Source code emphasizes readability (3 points)	You used comments, variable names, and code organization/indentation to make your code readable to others	You did not follow the guidelines we have been emphasizing in class for code readability

In this assignment, you will use loops to process a user-entered word in a variety of ways. Here is sample run #1 of your program: You should match the format of this output as closely as possible.



```
Enter a word for me to analyze (or Q to quit):   HOUSE
Results for HOUSE:
Number of vowels: 3
First vowel is: O in position 1
Pig Latin: OUSEHAY
Ends swapped: EOUSH
Reversed: ESUOH
Ends swapped and reversed: HSUOE
Scrambled: OSHEU USEHO EOUHS UHSOE SHEOU
All substrings of HOUSE:
*********
* H     *
* O     *
* U     *
* S     *
* E     *
* HO    *
* OU    *
* US    *
* SE    *
* HOU   *
* OUS   *
* USE   *
* HOUS  *
* OUSE  *
* HOUSE *
*********
15 substrings found.
Enter a word for me to analyze (or Q to quit):   Q
>
```

Name your class **WordPlay**. This name is important because it allows your instructor to write test code that calls the methods in your class. Your instructor's source code won't compile unless you name your class and methods as specified.

Methods your program has to have (the names, parameters, and return types should be exactly as described below). You may have other methods in addition to these:

Method	**What it does**
`main(String[] args)`	Uses a loop to prompt the user for a word, and then sends that word to showStats()
`showStats(String word)`	Calls each of the other methods below to process an uppercase version of the given word
`vowelCount(String word)`	Returns the number of vowels in the word. Vowels are A, E, I, O, and U.
`indexOfFirstVowel(String word)`	Returns the position of the first vowel in the word. If the word does not contain a vowel, then -1 should be returned. For example, for "APPLE", the method would return 0. For "HMM", the method would return -1.
`pigLatinize(String word)`	Returns the "pig latin" version of the word. To convert a word to pig latin: If the word begins with a vowel, simply add "WAY" to the end of the word. If the word begins with one or more consonants, move all the consonants *up to the first vowel*, move those consonants to the end of the word, and then add "AY" to the end. If a word contains only consonants, simply add "AY" to the end of the word. As examples: • The pig latin for "HAPPY" is "APPYHAY" (move all the consonants up to the first vowel to the end of the word, and add "AY") • The pig latin for "STRENGTH" is "ENGTHSTRAY" (move all the consonants up to the first vowel to the end of the word, and add "AY") • The pig latin for "RHYTHM" is "RHYTHMAY" (no vowels, so add "AY" to the end) • The pig latin for "APPLE" is "APPLEWAY"
`swapEnds(String word)`	Returns a string with the first and last letters swapped. If the word only contains one letter, then that word is returned unchanged.
`reverse(String word)`	Returns a string that is the reverse of the given string.
`scramble(String word)`	Returns a string that randomly scrambles the given word.
`printAllSubstrings(String word)`	Does not *return* anything. Instead, this method prints all the substrings of the given word, from shortest to longest, and left to right. It prints a border around the output. The border should look like the one shown in the example, but should be based on the length of the word. Notice that "HOUSE" contains 5 letters, but the border is 9 characters across, and the final word is centered in the border. Match this formatting. The method also prints a count of the number of substrings that were found.

Breaking your problem into methods:

- **You may (in fact, you should) consider writing other "helper" methods besides the ones shown above.** For example, notice that you have to print a row of asterisks twice. Why not write a method that helps you do just that?
- **Most methods should be very short.** My showStats() and printAllSubstrings() methods were each just under 20 lines of code (including blank lines). I also wrote one "helper" method of my own that was 18 lines of code including blank lines. My main() method was 11 lines long. The rest of my methods were no more than 10 lines of code, and several were only around 6 lines of code. **Your goal is not to match my numbers. Your goal is to keep each method "simple".**
- **Don't try to do too much in one method.** The primary job of main() is to get user input and call showStats(). The primary job of showStats() is to call the various other methods you wrote.
- **Methods can (and should) call on each other for help.** For example, pigLatinize() sometimes needs to know where the first vowel is located. So, you should call the firstVowel() method from within the pigLatinize() method.
- **Try using "stepwise refinement".** Not sure how to print all the substrings and the border? Then start with a part of the problem that you do know how to do: Why not just print all the substrings first? Get that working, and *then* figure out a way to add a border. Not sure how to print all the substrings? Then step back and see if you can just print all the 2-letter substrings? Got that figured out? Can you do it with a loop? OK, then can you modify it to print all the 3-letter substrings?

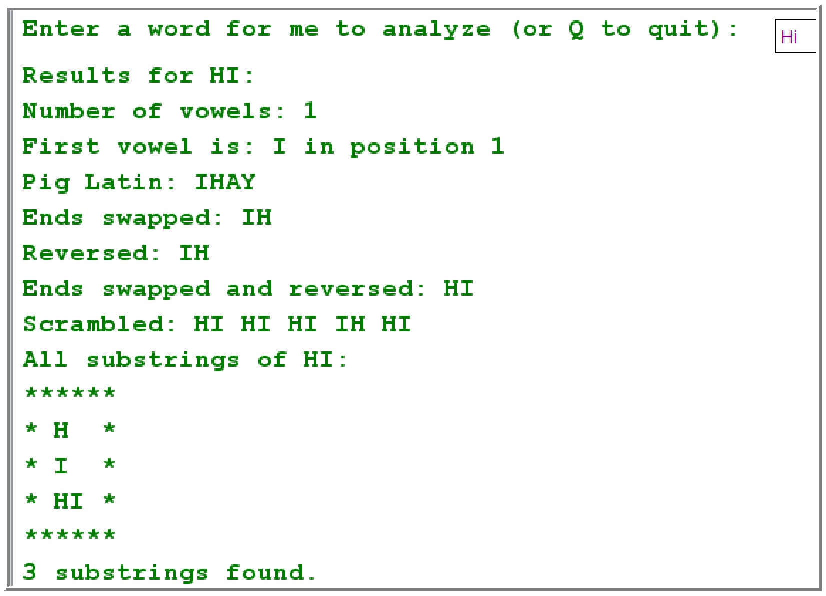
TEST ALWAYS: Don't write a new method if you haven't thoroughly tested the last method you wrote.

TURN IN your WordPlay.java source code to the course website.

On the next page are two additional sample runs that should help you understand how the program runs.

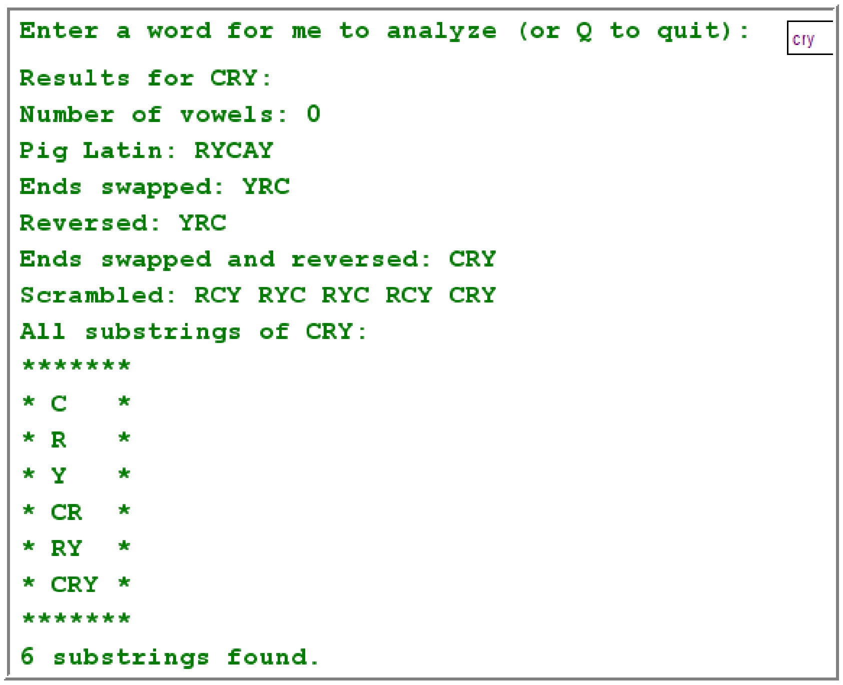
Match the formatting and order of the samples in this assignment. Here are a few more to help:

Sample run #2: Notice that the word was entered in mixed lower/uppercase, but the output is all in uppercase. Notice that there are always 5 scrambled versions of the word.



```
Enter a word for me to analyze (or Q to quit): Hi
Results for HI:
Number of vowels: 1
First vowel is: I in position 1
Pig Latin: IHAY
Ends swapped: IH
Reversed: IH
Ends swapped and reversed: HI
Scrambled: HI HI HI IH HI
All substrings of HI:
******
* H  *
* I  *
* HI *
******
3 substrings found.
```

Sample run #3: Notice that the word was no vowels (we aren't counting Y as a vowel). So notice that this skips the line that shows where the first vowel is located.



```
Enter a word for me to analyze (or Q to quit): cry
Results for CRY:
Number of vowels: 0
Pig Latin: RYCAY
Ends swapped: YRC
Reversed: YRC
Ends swapped and reversed: CRY
Scrambled: RCY RYC RYC RCY CRY
All substrings of CRY:
*******
* C   *
* R   *
* Y   *
* CR  *
* RY  *
* CRY *
*******
6 substrings found.
```