Objectives	Use the state machine approach to write a class for counting syllables in words. For this problem you don't need to use objects for states. Test your code by counting syllables in all words in a dictionary.
What to Submit	Commit your code to Bitbucket as Lab9. You should have these classes: WordCounter and Main (count words in a file or URL) Ask TA to check your State Machine diagram on paper before 18:00.

Assignment

1. Draw a State Machine Diagram of an algorithm for counting syllables in a word.

2. Write a class name WordCounter with a method named countSyllables( ) that implements the state machine and counts syllables in a word. If something is not a word, return 0.

4. Count the words and syllables in dictionary.txt located at http://se.cpe.ku.ac.th/dictionary.txt. The file has one "word" per line, but some of them may not be actual words according to our definition. If something is not a word according to the definition below, then don't count it.

How to count syllables?

This assignment uses the same rules as the *Flesch Readability Index* (PA4) to count syllables using vowel sequences.

Count syllables as the number of *vowels sequences* in a word. A *vowel sequence* is one or more vowels that occur together. A *vowel* is a, e, i, o, u, or (sometimes) y. Here are the cases with examples:

1. Sequences of consecutive vowels count as one syllable. vowels are: `a e i o u`. y counts only if it is the **first** vowel in a vowel sequence.

`banana` = 3 vowel sequences `b(a)n(a)n(a)`

`durian` = 2 vowel sequences `d(u)r(ia)n`

`beauty` = 2 vowel sequences `b(eau)t(y)`

`layout` = 2 vowel sequences `l(a)y(ou)t`

2. A final "e" as a single vowel is **not** counted, **unless** it is the only vowel in the word.

`apple` = 1 vowel sequence. Don't count final "e".

`love` = 1 vowel sequence. Don't count final `"e".`

`The, me, he, she, we` = 1 vowel sequence. Count the final "e" because it is the only vowel.

`movie` = 2 vowel sequences. Final "e" is part of a multi-vowel sequence, so count the sequence.

`levee` = 2 vowel sequences. Same reason as above.

3. A dash '-' in the middle of word is like a consonant.

`anti-oxidant` = 5 vowel sequences `(a)nt(i)-(o)x(i)d(a)nt`
`-oxidant` = **not a word**. Dash cannot be at start of word.
`anti-` = **not a word**. Dash cannot be at end of word.

4. **Not a word**. Any string that contains a non-letter or doesn't contain any vowels is not a word. The only exception is "-" in between letters (case 3).

```
mrtg
Java5se
anti-
I.B.M.
```

```
7-Eleven
```

5. "y" is considered a vowel if is the first vowel in a sequence, a consonant otherwise.

```
beyond = 2 vowel sequences: b(e)y(o)nd
yesterday = 3 vowel sequences: (ye)st(e)rd(a)y
Yahoo = 2 vowel sequences (Ya)h(oo)
```

6. Ignore apostrophe (').

```
isn't = isnt
student's = students' = students
```

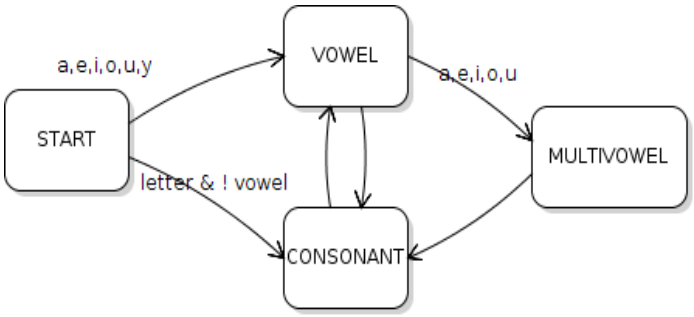
Problem 1. Identify States and Events, Draw a State Machine Diagram

Design a state machine for counting syllables in a sequence of characters without embedded spaces.

Draw a State Machine Diagram with States, Events, and Actions taken during transition or while in a state. *See document Programming a State Machine* in class week9 folder for UML examples.

Show all possible events and transitions, even transitions back to the same state.

States: Some (**not all**) of the *States* are*:*



`START` = start of a string, no characters processed yet. You can use this state to skip leading white space characters.

`CONSONANT` = most recent character is a letter but not a vowel.

`SINGLEVOWEL` = most recent character is a vowel that does *not* follow another vowel.
This is a separate state because we need to differentiate between "e" at the end of a word that is a lone vowel ("move") or part a sequence of vowels ("movie").

`MULTIVOWEL` = most recent char is a vowel that follows another vowel (2 or more vowels together)

`NONWORD` = the character sequence is not a word. Enter this state if you see any character other than letter or hyphen.

Events: The *event* is reading (or processing) a character.

Actions: add 1 to the syllable count. Show this at correct place(s) on state diagram.

Problem 2: Write a class to implement the State Machine

2.1 Write a WordCounter class has a method named **`countSyllables`** to count syllables in a String.

WordCounter
countSyllables(String) : int

`countSyllables` returns the number of syllables in the char sequence. If the parameter is *not a word* then return 0.

2.2 Use your state machine diagram to implement countSyllables.

Requirements:

- use the state machine approach
- only look at one character at a time: don't save the previous character and don't look-ahead at the next character.

An example of using a non-OO state machine is:

```
int countSyllables( String word ) {
```

```
    int syllables = 0;
    state = START;
    for(int k=0; k<word.length(); k++) {
        char c = word.charAt(k);
        for( char c = ??? ) {  // process each character in word
            switch(state) {
            // process character c using state machine
            case CONSONANT:
                if ( isVowel(c) ) state = VOWEL;
                else if (Character.isLetter(c)) /* consonant */;
                else if (c == '-') state = ? ;
                else state = ?  ;
                break;
            case SINGLEVOWEL:
                if (isVowel(c)) state = MULTIVOWEL;
                else if (Character.isLetter(c)) state = CONSONANT;
                …
                break;
        } // end of switch
    } // end of loop for chars in word
    // End of word. Correct syllable count for the "final e" rule.
```

The Character class has some useful methods for testing characters:

`Character.isLetter( c )` - true if c is a letter
`Character.isWhitespace( c )` - true if c is whitespace (space, tab, newline)

2.3 Ignore accidental *whitespace* before the beginning of the word. *Whitespace* means a space, tab, or newline character.

2.4 Please **don't write BAD CODE**. Don't do this:

```
int countSyllables(String word) {
    int syllables = 0;
    state = START;
    for( int k=0; k<word.length(); k++ ) {
        switch(state) {
            case VOWEL:
                if (word.charAt(k)=='y' &&
                 (word.charAt(k-1) == 'a' || word.charAt(k-1)=='e' ...
```

1. **Don't** repeatedly call `charAt(k)`. It's *inefficient* and makes the code hard to read.

2. **Don't** look ahead (next char) or look back (previous char). The state should contain all the information you need to decide what action and/or transition to perform for every possible input.

In a state machine, you don't need look-ahead or look-back.

If it appears you *do need* to look-ahead or look-back, then redefine your states or add more states to differentiate the cases.

Alternative Approach: Design an O-O style State Machine

You can use the object-oriented approach to a state machine if you want. Since reading a character is an event, each State object needs a method like `handleChar(char)`. You should also define `enterState()` and use it to increment syllable count.

```
interface State {
     public void handleChar(char c);
```

```
    public void enterState( );
}
```

In the WordCounter class, you need to provide a setState( ) method and countSyllable() method so that the states can change state and count syllables:

```
class WordCounter {
    private final State START = new StartState( );
    private final State SINGLEVOWEL = new SingleVowelState( );
    private State state;  // the current state
    /** change to a new state */
    public void setState( State newstate ) {
        if (newstate != state) newstate.enterState( );
        state = newstate;
    }
```

```
class SingleVowelState implements State {
    public void handleChar( char c ) {
        if ( isVowel(c) ) setState( MULTIVOWEL );
        else if ( isLetter(c) ) setState( CONSONANT );
        else if ( c == DASH ) setState( HYPHEN );
        else setState( NONWORD );
    }
    public void enterState( ) {
        syllableCount++;
    }
}
```

Problem 3: Test the syllableCounter

Create a test class to test syllableCounter for some words with known syllable counts. There is a **`WordCounterTest.java`** class in the same folder as this lab assignment.

Problem 4: Write a Main class to count a dictionary and calculate elapsed time

4.1 Write a Main class with a method (not the "main" method) that reads all the words from a URL or File and calls countSyllables. Output the total number of words, syllables, and the elapsed time in seconds. For example:

```
Reading words from http://se.cpe.ku.ac.th/dictionary.txt
Counted 102,000 syllables in 38,600 words
Elapsed time: 1.220 sec
```

4.2 Use this URL for the dictionary file: `http://se.cpe.ku.ac.th/dictionary.txt`
There is one word for line, but the file may contain blank lines and whitespace chars (check for them).
Example code for opening a URL as input stream is:

```
final String DICT_URL = "http://se.cpe.ku.ac.th/dictionary.txt";
URL url = new URL( DICT_URL );
InputStream input = url.openStream( );
```

For fast reading of input as Strings, use a BufferedReader. Since the file contains only one word per line, parsing it is easy.

```
BufferedReader reader =
                    new BufferedReader( new InputStreamReader( input ) );
```

```
while( true ) {
    String word = reader.readLine();
    // BufferedReader.readLine() returns null at end of the input
    if (word == null) break;
    // analyze the word
```

There is a short, C-style idiom for this (but harder to write try-catch):

```
while( (word = reader.readLine()) != null ) {
```

Programming Hints

2. java.util.Scanner is slow. A faster way to read *lines* of input is BufferedReader.

2. An InputStream contains only bytes. We need characters or Strings. A Reader reads input as *characters*. InputStreamReader and BufferedReader are subclasses of Reader.

```
Reader reader = new InputStreamReader( inputStream ); // read InputStream
Reader reader = new StringReader( string );           // read String
```

3. BufferedReader is a decorator (wrapper) for Reader that can read an entire line at once.

```
// You can create a BufferedReader object from any Reader.
BufferedReader reader = new BufferedReader( new InputStreamReader(in) );
// read the input one line at a time.
// readLine() returns null when there is nothing to read (end of stream)
while( true ) {
    String line = reader.readLine( );
    if (line == null) break;
    // process the line
```

Reference

- *Programming a State Machine* in class week9 folder.
- Wikipedia, *Finite State Machines*.