---Class CalSentences

```
public class CalSentences {

    public static int countSentences(String text){
        int count = 0;
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c == '.' || c == ':' || c == ';' || c == '?'
|| c == '!'){
                count++;
            }

        }
        return count;
    }
}
```

---Class CalWord

```
public class CalWord {

    public static int countWords(String text){
        CalcSyllables calcSyllables = new CalcSyllables();
        int words = 0;
        boolean prevCharWasALetter = false;
        StringBuilder word = new StringBuilder();
        for (int i=0; i<text.length(); i++) {
            char c = text.charAt(i);

            // apostrophes (and other characters?) should be ignored
            if (c=='\'')
                continue;

            // everything else is either a letter, or it isn't
            boolean charIsALetter = ((c>='A' && c<='Z') || (c>='a'
&& c<='z'));

            // and a word is defined as a letter following a non-
letter
            if (charIsALetter) {
                if (!prevCharWasALetter) {
                    words++;

                    if (word.length()>0)

                        System.out.println("'" + word.toString() +
"'" + " has " + calcSyllables.countSyllablesInWord(word.toString())
+ " Syllables");
                    word.setLength(0);
                }

                word.append(c);

            }

            prevCharWasALetter = charIsALetter;
        }

        if (word.length()>0)

            System.out.println("'" + word.toString() + "'" + " has "
+ calcSyllables.countSyllablesInWord(word.toString()) + "
Syllables");

        return words;


    }
}
```

---Class CalcSyllables

```
public static int countSyllablesInWord(String word) {
        int syllablesCount = 0;

        boolean preWasVowel = false;
        for (int i = 0; i < word.length(); i++) {
            char c = word.charAt(i);
            //current char is vowel
            if (isVowel(c)) {
                //current char is vowel and previous char is not vowel
                if (!preWasVowel) {
                    syllablesCount++;
                }
                preWasVowel = true;
            } else {
                preWasVowel = false;
            }

            if (i == word.length() - 1 && isVowel(c) && i > 0
&& !isVowel(word.charAt(i - 1))) {
                syllablesCount--;
            }

        }
        // Ensure each word has at least 1 syllable
        if (syllablesCount < 1) {
            syllablesCount = 1;
        }
        return syllablesCount;
    }

    public static boolean isVowel(char c) {
        char vowel[] = {'a', 'e', 'i', 'o', 'u'};
        c = Character.toLowerCase(c);//Converts characters to lower
case
        for (int i = 0; i < vowel.length; i++) {
            if (c == vowel[i]) {
                return true;
            }
        }
        return false;

    }

    public static int countSyllablesTotal(String text) {
        String[] words = text.split("\\s+");
        int totalSyllables = 0;

        for (int i = 0; i < words.length; i++) {
            totalSyllables += countSyllablesInWord(words[i]);


        }
        return totalSyllables;
    }


}
```

Main

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Please type (or paste) some text:
");
        String userText = scanner.nextLine();

        int words = CalWord.countWords(userText);
        int sentences = CalSentences.countSentences(userText);
        int syllables =
CalcSyllables.countSyllablesTotal(userText);



        System.out.printf("Words:" + words + " Sentences: " +
sentences + " Syllables: " + syllables);


        //index calculation

        double index = 206.835 - 84.6 *
((double)syllables/(double)words) - 1.015 *
((double)words/(double)sentences);
        String result = String.format("%.6f", index);


        System.out.println("\nFlesch Readability Index = " +
result);




    }


}
```

Output:

1.

It doesn't appear to have any effect in terms of functionality, but is recommended as a form of compile-time correctness checking. For example, consider the scenario where you write a method called print() which replaces a superclass method called Print(). If you have used @override then the compiler will give an error, but if you don't use it then you'll simply have created a different method which has nothing to do with the original, since the method names are not the same. So the use of @override could potentially save you hours of debugging work.

```
Flesch Readability Index = 52.931250
```

```
'work' has 1 Syllables
Words:95 Sentences: 4 Syllables: 146
Flesch Readability Index = 52.711908
```

??????

2.

If you have used @override then the compiler will give an error.

```
'If' has 1 syllables
'you' has 1 syllables
'have' has 1 syllables
'used' has 2 syllables
'override' has 3 syllables
'then' has 1 syllables
'the' has 1 syllables
'compiler' has 3 syllables
'will' has 1 syllables
'give' has 1 syllables
'an' has 1 syllables
'error' has 2 syllables

Words=12, Sentences=1, Syllables=18.

Flesch Readability Index = 67.755000
```

```
'If' has 1 Syllables
'you' has 1 Syllables
'have' has 1 Syllables
'used' has 2 Syllables
'override' has 3 Syllables
'then' has 1 Syllables
'the' has 1 Syllables
'compiler' has 3 Syllables
'will' has 1 Syllables
'give' has 1 Syllables
'an' has 1 Syllables
'error' has 2 Syllables
Words:12 Sentences: 1 Syllables: 18
Flesch Readability Index = 67.755000
```

3.

I AM SAM. I AM SAM. SAM I AM. DO YOU LIKE GREEN EGGS AND HAM?

```
'I' has 1 syllables
'AM' has 1 syllables
'SAM' has 1 syllables
'I' has 1 syllables
'AM' has 1 syllables
'SAM' has 1 syllables
'SAM' has 1 syllables
'I' has 1 syllables
'AM' has 1 syllables
'DO' has 1 syllables
'YOU' has 1 syllables
'LIKE' has 1 syllables
'GREEN' has 1 syllables
'EGGS' has 1 syllables
'AND' has 1 syllables
'HAM' has 1 syllables

Words=16, Sentences=4, Syllables=16.

Flesch Readability Index = 118.175000
```

```
'I' has 1 Syllables
'AM' has 1 Syllables
'SAM' has 1 Syllables
'I' has 1 Syllables
'AM' has 1 Syllables
'SAM' has 1 Syllables
'SAM' has 1 Syllables
'I' has 1 Syllables
'AM' has 1 Syllables
'DO' has 1 Syllables
'YOU' has 1 Syllables
'LIKE' has 1 Syllables
'GREEN' has 1 Syllables
'EGGS' has 1 Syllables
'AND' has 1 Syllables
'HAM' has 1 Syllables
Words:16 Sentences: 4 Syllables: 16
Flesch Readability Index = 118.175000
```

4.
A path from a point approximately 330 metres east of the most south westerly corner of 17 Batherton Close, Widnes and approximately 208 metres east-south-east of the most southerly corner of Unit 3 Foundry Industrial Estate, Victoria Street, Widnes, proceeding in a generally east-north-easterly direction for approximately 28 metres to a point approximately 202 metres east-south-east of the most south-easterly corner of Unit 4 Foundry Industrial Estate, Victoria Street, and approximately 347 metres east of the most south-easterly corner of 17 Batherton Close.

Flesch Readability Index = −34.246220

```
'Close' has 1 Syllables
Words:74 Sentences: 1 Syllables: 153
Flesch Readability Index = -43.191216
```