Assignment #4: (Identify Word) and Language Parser

Due: Wednesday, October 9th by 9:00 pm.
Put a block comment at the beginning of the Java file that contains:

```
/*============================================================================
 |   Assignment:  Assignment #[n]:  [Assignment Title]
 |       Author:  [Your Name (Your E-mail Address)]
 |       Grader:  [Your Section Leader's name]
 |
 |       Course:  127A
 |   Instructor:  I. Kishi
 |     Due Date:  [Due Date and Time]
 |
 |  Description:  [Describe the program's goal, IN DETAIL.]
 |
 | Deficiencies:  [If you know of any problems with the code, provide
 |                details here, otherwise clearly state that you know
 |                of no unsatisfied requirements and no logic errors.]
 *============================================================================*/
```

Grading Notes:

Programs should compile without errors or warnings; those that do not compile will not receive a passing grade.

For full credit, programs should produce the output shown below.

Note: For those who already know how to use loops, arrays, and the **`switch`** statement: Do **NOT** use any of these techniques on this assignment. The goal here is to understand how to use **`if`** statements.

Program 1A: Identify Word

Name your program: **`IdentifyWord.java`**.

NOTE: This program will not be turned in for a grade. However, in order to complete LanguageParser.java, you will have to be able to correctly identify when a word is a particular part of speech. So, this program is intended to do just that. You will be able to submit this program to WebGS to get feedback, but you will not be turning this program in for a grade.

A team of archaeologists and linguists have recently discovered what they believe to be the lost relics of an alien language. Coincidentally, this language has a similar alphabet to ours – or, at least, one our team believes can be mapped to ours. Because of this, the linguists have changed all of our data to be represented by English letters – that way they can use our expertise to create an automated system to assist them with their research. They aren't done diagramming the entire language, but so far they have been able to identify four parts of speech:

- A florb is any word that starts and ends with the same character
- A wooble is any word that contains either the character pair "cj" or the character pair "wq"
- A zith is any word that either has an odd number of letters and no 'z', or an even number of letters and no 'k'.
- A zarf is any word that ends with a capital letter.

Here is an example run of the program. For each of these examples, keep in mind that while it would appear that there is an extra newline at the end, there isn't. That was put in manually to make the examples easier to read. Also, user input is underlined:

```
Please enter a word to be analyzed: aldfaioekd
The word 'aldfaioekd' is not one of the four parts of speech
```

Here is another example run of the program:

```
Please enter a word to be analyzed: douksd
The word 'douksd' is a florb
```

Here is another example run of the program:

```
Please enter a word to be analyzed: moucjsz
The word 'moucjsz' is a wooble
```

Here is another example run of the program:

```
Please enter a word to be analyzed: loumsr
The word 'loumsr' is a zith
```

Here is another example run of the program:

```
Please enter a word to be analyzed: sjplkQ
The word 'sjplkQ' is a zarf
```

Here is another example run of the program:

```
Please enter a word to be analyzed: wqlzw
The word 'wqlzw' is a florb
The word 'wqlzw' is a wooble
```

Here is another example run of the program:

```
Please enter a word to be analyzed: dloiujd
The word 'dloiujd' is a florb
The word 'dloiujd' is a zith
```

Here is another example run of the program:

```
Please enter a word to be analyzed: JmskIJ
The word 'JmskIJ' is a florb
The word 'JmskIJ' is a zarf
```

Here is another example run of the program:

```
Please enter a word to be analyzed: wqlsied
The word 'wqlsied' is a wooble
The word 'wqlsied' is a zith
```

Here is another example run of the program:

```
Please enter a word to be analyzed: lmwqkM
The word 'lmwqkM' is a wooble
The word 'lmwqkM' is a zarf
```

Here is another example run of the program:

```
Please enter a word to be analyzed: sldioeL
The word 'sldioeL' is a zith
The word 'sldioeL' is a zarf
```

Here is another example run of the program:

```
Please enter a word to be analyzed: solwqms
The word 'solwqms' is a florb
The word 'solwqms' is a wooble
The word 'solwqms' is a zith
```

Here is another example run of the program:

```
Please enter a word to be analyzed: EikcjE
The word 'EikcjE' is a florb
The word 'EikcjE' is a wooble
The word 'EikcjE' is a zarf
```

Here is another example run of the program:

```
Please enter a word to be analyzed: KlotsK
The word 'KlotsK' is a florb
The word 'KlotsK' is a zith
The word 'KlotsK' is a zarf
```

Here is another example run of the program:

```
Please enter a word to be analyzed: rwqkR
The word 'rwqkR' is a wooble
The word 'rwqkR' is a zith
The word 'rwqkR' is a zarf
```

Here is final example run of the program:

```
Please enter a word to be analyzed: TwqkT
The word 'TwqkT' is a florb
The word 'TwqkT' is a wooble
The word 'TwqkT' is a zith
The word 'TwqkT' is a zarf
```

Program 1: Language Parser

Name your program: **`LanguageParser.java`**.

A team of archaeologists and linguists have recently discovered what they believe to be the lost relics of an alien language. Coincidentally, this language has a similar alphabet to ours – or, at least, one our team believes can be mapped to ours. Because of this, the linguists have changed all of our data to be represented by English letters – that way they can use our expertise to create an automated system to assist them with their research. They aren't done diagramming the entire language, but so far they have been able to identify four parts of speech:

- A florb is any word that starts and ends with the same character
- A wooble is any word that contains either the character pair "cj" or the character pair "wq"
- A zith is any word that either has an odd number of letters and no 'z', or an even number of letters and no 'k'.
- A zarf is any word that ends with a capital letter.

The researchers need us to write a program that will take any given sentence and tell them if that sentence is part of the language. In order to do that, we need to know what constitutes a sentence in this language. So far our research team has been able to find five different sentence structures

- wooble zith zarf (in other words, rule 1 says that a sentence can have the form of a wooble, followed by a zith, followed by a zarf)
- zarf florb zith (in other words, rule 2 says that a sentence can also have the form of a zarf, followed by a florb, followed by a zith)
- wooble zarf florb (in other words, rule 3 says that a sentence can also have the form of a wooble, followed by a zarf, followed by a florb)
- florb zith zarf (in other words, rule 4 says that a sentence can also have the form of a florb, followed by a zith, followed by a zarf)
- zarf florb wooble (in other words, rule 5 says that a sentence can also have the form of a zarf, followed by a florb, followed by a wooble)

Notice that all five of these sentence structures only have three words in them. That's actually a relief because we don't yet have the technology to deal with sentences of arbitrary length; but we'll get there! The researchers, understanding this, only gave us part of the language definition for now.

HINT: Remember how we've been using a Scanner and attaching it to the keyboard?

```
Scanner keyboard = new Scanner(System.in);

System.out.print("Enter your name please: ");

String name = keyboard.next();
```

Well, it turns out that you can attach Scanners to other sources of input. One such source of input is a String. So, I could create a Scanner that is looking at a String. Because I still have a Scanner, I can still use any Scanner methods that I need or want to use. This will be very helpful on this assignment when it comes to pulling the three words out of the sentence that the user enters.

Write a program that takes in a sentence from the user and lets the user know if the sentence is part of the language, and if so, what rule allows the sentence to be part of the language.

What follows are example runs of the program. The break in the message after the sentence is printed is intentional; that is, you should have "is part of our language" or "is not part of our language" on a second line as is shown in these examples. Again, remember that user input is underlined, and what appears to be an extra space at the end of the example isn't really there as far as the code is concerned; they were manually added to make the examples easier to read:

```
Enter a sentence: cjlokwpq loumsr sjplkQ
According to rule 1, the sentence 'cjlokwpq loumsr sjplkQ'
is part of our language
```

Here is another example run of the program:

```
Enter a sentence: sjplkQ douksd loumsr
According to rule 2, the sentence 'sjplkQ douksd loumsr'
is part of our language
```

Here is another example run of the program:

```
Enter a sentence: cjlokwpq doeftskW folzf
According to rule 3, the sentence 'cjlokwpq doeftskW folzf'
is part of our language
```

Here is another example run of the program:

```
Enter a sentence: douksd loumsr doeftskW
According to rule 4, the sentence 'douksd loumsr doeftskW'
is part of our language
```

Here is another example run of the program:

```
Enter a sentence: sjplkQ douksd moucjsz
According to rule 5, the sentence 'sjplkQ douksd moucjsz'
is part of our language
```

Here is the final example run of the program:

```
Enter a sentence: folzf sjplkQ moucjsz
According to our rules, the setence 'folzf sjplkQ moucjsz'
is not part of our language
```

Turnin:

We are accepting programs via D2L and WebGS only. Please note that you are required to turn your files in to both D2L and WebGS. If you don't turn your submissions into WebGS, you will not receive a grade for that program, since we are now using WebGS to grade the assignments. If you do not turn in your files to D2L, you will receive a 5 point deduction. Also, we are only grading LanguageParser.java. That is, IdentifyWord.java is for you to use – and it should be done first. But it is there for you to use as a tool to make sure you have started on the right track with the program.

- Name your programs **`IdentifyWord.java`** and **`LanguageParser.java`**.
- Log on to D2L, and select CSc 127A.
- Select "Dropbox"
- You will find *Program4* and *Program4-Late*. Only one of these will be active. Click on the active one — will be *Program4* if you are turning in the work on time.

- You should now be at the page: "Submit Files – Program4". Click on the "Add a File" button. A pop-up window will appear. Click on the "Choose File" button. Use the file browser that will appear to select your **`IdentifyWord.java`** or **`LanguageParser.java`** file. Click on the "Upload" button in the lower-right corner of the pop-up window.
- You are now back at "Submit Files – Program4". Click on "Upload" in the lower-right of this window. You should now be at the "File Upload Results" page, and should see the message **`File Submission Successful`**.

D2L will send you a confirmation email after each item is placed in the Dropbox. This email is sent to your UA email (which is <yourUANetId>@email.arizona.edu). Retain these emails at least until you have received a grade for the assignment; it is your confirmation that you did submit the program(s).

You will do this process at least twice, once for **`IdentifyWord.java`** and once for **`LanguageParser.java`**.

You may submit each program more than once. WebGS keeps track of the highest score per file across all submissions.