

# WES-CS GROUP MEETING #6

## Exercise 1: PigLatin

The rules for translating an English word to Pig-Latin are as follows:

- If the word starts with a vowel, it is unchanged.
- Otherwise, all of the consonants at the beginning of the word (up to the first vowel) are moved to the end of the word, preceded by a dash (for readability), and followed by “ay”.

For example, the sentence “I love Madison in the springtime” would be translated to “I ove-lay adison-May in e-thay ingtime-spray”.

For this exercise, you will complete the *PigLatin* class started on the next page, which will allow users of the class to translate an English word to Pig-Latin.

First, make sure you understand the translation rules by translating the following phrases into Pig-Latin:

Hello world

I love WESCS

the car says ooga ooga

**Part (a):** Take a look at the incomplete *PigLatin* class on the next page. Make sure you understand what each method is supposed to do, and discuss the reasons for making some of the methods private and others public.

```

class PigLatin {
    private String englishWord;

    // constructor
    public PigLatin( String s ) {
        englishWord = s.toLowerCase();
    }

    // translate: translate the English word to Pig-Latin
    public String translate( ) {
        /* part (c) */
    }

    // firstVowelPos: return the position of the first vowel
    //                  in the English word; return -1 if there
    //                  is NO vowel in the word
    //
    private int firstVowelPos( ) {
        /* part (b) */
    }
}

```

**Part (b):** Write the *firstVowelPos* method, which returns the position of the first vowel in the *englishWord* field, or returns -1 if there is no vowel in the word. Here are some examples:

<b>englishWord</b>	<b>Result of calling <i>firstVowelPos</i></b>
hello	1
ice	0
spring	3
qyzzx	-1

To test your code, write a *main* method for the *pigLatin* class that creates several *PigLatin* objects, calling their *firstVowelPos* methods and printing the results.

**Part (c):** Now describe in English how the *translate* method of the *PigLatin* class should work (using the rules for translating from English to Pig-Latin given above, and making use of the *firstVowelPos* method). Then write the actual code. Also write a *TestPigLatin* class with a *main* method that asks the user of the program to type in a word, reads the word, and prints the translation. Compile your code and run it!

## Exercise 2: The Time class

Consider the *Time* class partially defined below. Notice that the *Time* class stores its time in 24-hour format: it has two fields called *hour* and *minute*; *hour* is an integer between 0 and 23 that represents the hours part of the time, and *minute* is an integer between 0 and 59 that represents the minutes part of the time.

```
public class Time {
    private int hour;    // 0 <= hour <= 23
    private int minute; // 0 <= minute <= 59

    // constructor
    public Time(int hr, int min) {
        hour = hr;
        minute = min;
    }

    // printMinutes
    //
    // if minutes < 10, print "0" followed by the value of minutes
    // followed by a newline
    // otherwise, just print the value of minutes followed by a
    // newline
    private void printMinutes() {
        // part (a)
    }

    // print24HourTime
    public void print24HourTime() {
        System.out.print(hour + ":");
        printMinutes();
    }

    // print12HourTime
    public void print12HourTime() {
        // part (b)
    }

    // timeRemaining
    public int timeRemaining( Time otherTime ) {
        // part (c)
    }
}
```

**Part (a):** Write the *printMinutes* method. As specified by the comments for that method, it should print the appropriate two digits followed by a newline.

**Part (b):** Write the *print12HourTime* method, which works as follows:

- If the 24-hour time represents midnight (0 hours and 0 minutes) or noon (12 hours and 0 minutes), just print "midnight" or "noon".
- Otherwise, if the time is before noon (*hours* is less than 12), print it in 12-hour format ending with "am".
- Otherwise, print the time in 12-hour format, ending with "pm".

Here are some examples.

<i>hours</i>	<i>minutes</i>	How to print the 12-hour time
0	24	12:24am
0	4	12:04am
6	15	6:15am
12	15	12:15pm
14	5	2:05pm

**Part (c):** Write the *timeRemaining* method, which returns the number of minutes that would have to be added to the time represented by the *Time* object whose method is called, to get to the time represented by parameter *other*. For example, if you have a *Time* object *now* that represents 11:30, and another *Time* object *appointment* that represents 13:45, then the method call *now.timeRemaining( appointment )* should return 135 because it takes 135 minutes (which is two hours and fifteen minutes) to get from 11:30 to 13:45. The method call *appointment.timeRemaining( now )* should return 1305, because it takes 1305 minutes (which is 21 hours and 45 minutes) to get from 13:45 to 11:30 (the next day).

**Part (d):** Change the constructor to check whether the values passed in for the hours and minutes are valid (are in the correct range). What could the constructor do if the values are *not* valid?