CSE 142, Spring 2013
Programming Assignment #1: Song (10 points)
Due Tuesday, April 9th 2013, 11:30 PM

Program Description:

This program tests your understanding of static methods and `println` statements. Write a Java class called `Song` in a file named `Song.java`. (Use exactly this file name, including identical capitalization.)

For this assignment, you will write a program that outputs the cumulative song below, "There Was An Old Lady Who Swallowed A Fly." A *cumulative song* is one where each verse builds upon previous verses. Examples of other well-known cumulative songs are "The House That Jack Built" and "The Twelve Days of Christmas."

```
There was an old woman who swallowed a fly.
I don't know why she swallowed that fly,
Perhaps she'll die.

There was an old woman who swallowed a spider,
That wriggled and iggled and jiggled inside her.
She swallowed the spider to catch the fly,
I don't know why she swallowed that fly,
Perhaps she'll die.

There was an old woman who swallowed a bird,
How absurd to swallow a bird.
She swallowed the bird to catch the spider,
She swallowed the spider to catch the fly,
I don't know why she swallowed that fly,
Perhaps she'll die.

There was an old woman who swallowed a cat,
Imagine that to swallow a cat.
She swallowed the cat to catch the bird,
She swallowed the bird to catch the spider,
She swallowed the spider to catch the fly,
I don't know why she swallowed that fly,
Perhaps she'll die.

There was an old woman who swallowed a dog,
What a hog to swallow a dog.
She swallowed the dog to catch the cat,
She swallowed the cat to catch the bird,
She swallowed the bird to catch the spider,
She swallowed the spider to catch the fly,
I don't know why she swallowed that fly,
Perhaps she'll die.

<< Your custom sixth verse goes here >>

There was an old woman who swallowed a horse,
She died of course.
```

The verses printed by your program must **exactly** reproduce the output at left. This includes identical wording, spelling, spacing, punctuation, and capitalization.

However, the sixth verse of your song (the bold part enclosed in **<< >>**) should print your own original text. Creative verses submitted may be shown in class anonymously at a later date. The only restrictions on your custom verse are the following:

- The verse must be in the same pattern as the first five verses. For example, some versions of the song have a sixth verse for swallowing a goat ("Just opened her throat to swallow a goat"). Notice that the first two lines should either end in the same word (fly/fly, bird/bird, cat/cat, etc) or should end with rhyming words (spider/inside her).
- The verse must not be identical to another verse, consist entirely of text from earlier in the song or be a verse you'll commonly find on the web (e.g., goat and cow).
- The text of the verse should not include hateful, offensive, or otherwise inappropriate speech.
- The code to produce the verse is still subject to the style guidelines on the next page.

(continued on back)

One way to write this program would be to simply write a series of `println` statements that output each line of the song in order. But such a solution would not receive full credit. Part of the challenge of this assignment lies in recognizing the structure and redundancy of the song and improving the code using static methods.

Style Guidelines:

You should not place any `println` statements in your `main` method. (It is okay for `main` to have empty `println` statements to print blank lines.) Instead of printing in `main`, use static methods for two reasons:

1. *Structure*

You should write static methods to capture the structure of the song. You should, for example, have a method for each of the verses of the song (including your custom verse) to print that verse's entire contents.

2. *Eliminating redundancy*

You should use only one `println` statement for each distinct line of the song (other than blank lines). For example, the following line appears several times in the output, but you should have only one `println` statement in your program that prints that line of the song:

```
Perhaps she'll die.
```

A method that prints just one line is not good style. Instead, identify groups of lines that appear in multiple places in the song and create methods to represent those groups. There is a general cumulative structural redundancy to the song that you should eliminate with your methods. Recall that methods can call other methods if necessary (which can themselves call other methods, and so on). The key question to ask is whether you have repeated lines or groups of lines of code that could be eliminated if you structured your methods differently. This includes sequences of `println` statements and also repeated sequences of method calls.

You do *not* have to eliminate partial-line redundancy in lines that are similar but not identical, such as the lines that start with "`There was an old woman who swallowed a.`"

Include a comment at the beginning of your program with some basic information and a description of the program in your own words. For example:

```
// Suzy Student, CSE 142, Autumn 2049, Section XX
// Programming Assignment #1, 06/07/49
//
// This program's behavior is ...
```

For this assignment, you should limit yourself to the Java features covered in Chapter 1 of the textbook. Though we will cover Chapter 2 while you work on this assignment, please do not use Chapter 2 features on this program, such as mathematical expressions, `print` statements (as opposed to `println`), or `for` loops.

Submission and Grading:

Turn in your Java source code file electronically from the Homework link on the course web site.

Part of your program's score will come from its "external correctness." External correctness measures whether the output matches exactly what is expected. We are very picky about the output matching exactly and expect every character and space to match. Use the **output comparison tool** to help you make sure your output is perfect. Programs that do not compile will receive no external correctness points.

The rest of your program's score will come from its "internal correctness." Internal correctness measures whether your source code follows the stylistic guidelines specified in this document. This includes having an adequate comment header and capturing the structure and redundancy of the song as specified previously. You should also limit the lengths of all lines in your program to fewer than 100 characters.