CSE 142, Winter 2008
Programming Assignment #1: Song (10 points)
Due Tuesday, January 15, 2008, 4:00 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.) Your program should produce the following song, a modified version of a classic holiday song, as output:

```
On the 1st day of "Xmas", my true love sent to me
a partridge in a pear tree.

On the 2nd day of "Xmas", my true love sent to me
two turtle doves, and
a partridge in a pear tree.

On the 3rd day of "Xmas", my true love sent to me
three French hens,
two turtle doves, and
a partridge in a pear tree.

On the 4th day of "Xmas", my true love sent to me
four calling birds,
three French hens,
two turtle doves, and
a partridge in a pear tree.

On the 5th day of "Xmas", my true love sent to me
five golden rings,
four calling birds,
three French hens,
two turtle doves, and
a partridge in a pear tree.

On the 6th day of "Xmas", my true love sent to me
six geese a-laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves, and
a partridge in a pear tree.

<< your seventh verse goes here >>
```

The first six verses printed by your program (all except the final bold line in **`<< >>`** above) must **exactly** reproduce the output above. This includes identical wording, spelling, spacing, punctuation, and capitalization.

To encourage creativity, the last verse of your song may print any text you like. For example, you could search Google for this song and include the text of its actual next verse, or include another custom verse in the pattern of the rest of the song, or any messages of your own choosing. 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 not be identical to another verse or consist entirely of text from earlier in the song.
- The number of lines in the verse should be at least three (3) but no more than fifty (50).
- 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.

Stylistic Guidelines:

One way to write this program would be to simply write a `println` statement that outputs each line of the song in order. However, 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.

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. To capture the *structure* of the song's verses.

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. To avoid simple *redundancy* in the output.

You should use only one `println` statement for each distinct non-blank line of the song. 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:

```
a partridge in a pear tree.
```

However, a method that prints just one line is not very useful. Instead, identify groups of lines that appear in multiple places in the song and create methods that capture those groups and are called multiple times. 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 of code that could be eliminated if you structured your methods differently. This includes sequences of `println` statements and repeated sequences of method calls.

You do NOT have to eliminate redundancy in lines that are similar but not identical, such as these:

```
On the 1st day of "Xmas", my true love sent to me
On the 2nd day of "Xmas", my true love sent to me
```

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 such as mathematical expressions, `print` statements (as opposed to `println`), or `for` loops on this program.

As a point of reference, our solution to this program has 13 methods other than `main` and occupies 82 lines including comments and blank lines. But this is just a rough guideline; you do not have to match this exactly.

Submission and Grading:

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

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. Every character and space must match.) 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.