CS 203 HW#2: Mad-Lib Creator

Due: Wednesday, January 29th, at 10:00pm

This assignment is intended to give you practice using and manipulating `String` objects in the Java language. You will implement a program that creates a short Mad Libs[1]-style story. A Mad Lib asks a player to supply a series of words or phrases, each of a specified category, without knowing the context in which the words will be used. These words then become part of a story, often to humorous effect. If you are unfamiliar with Mad Libs, there are example stories at "Wacky Web Tales" (http://www.eduplace.com/tales).

Grading
Your assignment grade will be based on

- Program functionality (80%)
- Code quality (20%)

Program Functionality
Your program should:

- Greet the user with an informative message. (5%)
- Ask the user to enter several words or phrases of certain categories. The following list offers some examples, but you should use categories of your own choosing. (Keep them G-rated, please!) (5%)
  - name
  - movie star
  - color
  - adjective
  - verb ending in -ing
  - adverb
  - famous person
  - song title
  - liquid
  - farm animal
  - singular noun
  - silly sound
  - vehicle
  - article of clothing

After the user has entered a word/phrase for each category, your program should print a Mad-Lib-style story that incorporates the user's words and phrases.

Your program should:

1. Trim all preceding and ending whitespace surrounding the user's input. (5%)
   For example, if the user types:
   `"  green eggs  "` (without the quotes)
   then your program should trim the whitespace to produce the string:
   `"green eggs"` (without the quotes)
2. Include at least one sentence that describes the length of a word or phrase given by the user. (5%) For example, if the user types "zebra" for the category “a type of animal,”

---

1 http://en.wikipedia.org/wiki/Mad_Libs, viewed Sept. 5, 2013.

your story might include a sentence like "A zebra, an animal whose name has 5 letters, is a very interesting beast."

3. Your program should ensure that user responses that should not be capitalized are in lower case letters. (5%)
4. Your madlib should require the user to enter at least one word whose first letter needs to be capitalized either because it is a proper noun or because it appears at the first word in a sentence. Your program should ensure that the user's input has proper capitalization rather than relying upon the user. (10%) For example, if you program asks for a name and the user enters

   ```
   susie
   ```

   then your program should use the name as

   ```
   Susie
   ```
5. Your story should change at least one word or phrase entered by the user to all capital letters. (5%) For example, if the user types "swoosh" for a silly sound, your story could say something like "The airplane made a SWOOSH sound before flying high over the clouds."
6. Convert at least one word to “1337 speak,” wherein some of the letters are replaced with similarly shaped digits or punctuation. (10%) For example, you might convert `“green acres”` to `“&r33n @cr3s”`. Your code for 1337 speak conversion should be able to convert at least 6 different letters.
7. Hyphenate at least one word entered by the user by placing a hyphen roughly in the middle of the string. (10%) The hyphenation need not be correct grammatically.
8. Reverse the order of two words in a two-word phrase. (10%) For example, if the user entered `“bum rush”` your story would use the phrase `“rush bum”`
9. Your story, when printed to the screen, should be formatted as a story. It should have a title, more than one paragraph, and each new paragraph should start with tabbed indentation. (10%)

Note: You may assume users are cooperative, that is, they will always follow directions and give a word or phrase that matches the category that was requested.

Example Interaction

The following is a printout from an example Mad Lib-style program. You should not follow this formatting exactly, of course; you should create your own story.

Welcome. You get to create your own story.
Please enter a girl's name:
susie
Please enter a color:
BLUE
Please enter a children's toy (plural):
rACe CArs
Please enter a noise:
yeoaooow
Please enter a plural noun:
apples
Please enter a thing:
coffee table
Please enter an animal:
giraffe
Please enter an emotional two-word phrase:
holy smokes

Baseball-ville

There once was a girl named Susie who enjoyed playing baseball. Instead of playing baseball with baseballs, though, she played the game with race cars. She liked to hit race cars because they are blue and “blue” is spelled with 4 letters.

One day it was her turn at bat. She stepped up to the plate, saying “Smokes holy!” Suddenly @pp13s started falling from the sky. She yelled “YEOAOOW!” and took refuge under a table coffee. A giraffe rescued her and took her to the land of Baseball-ville, where Susie could play baseball every day.

The End.

Additional Enrichment (optional)
If you have programmed before, or if you like a challenge and have the time, feel free to get a little fancier. If you do, be sure to document the additional features in your code. Here are some ideas:

- Create a repository of more than one story and let the user decide which story to use. This will involve selection.
- Create a "choose your own adventure" story that allows the user to decide what should happen next. This will involve selection and gradually displaying pieces of the story.
- Display the prompts in text fields on a graphical user interface, similar to the ones in "Wacky Web Tales" linked above. This will involve reading about graphics in your textbook.
- Add an error check to ensure the user always types a string with a least one character after each prompt. If they have not, prompt them again for a word or phrase. This will likely involve the use of selection and iteration, which can be found in chapters 3 and 4 of the textbook.

Code Quality (20% total)
A good computer program not only performs correctly, it also is easy to read and understand. As the semester proceeds, we will discuss this issue in more depth. For now, however, your program should exhibit good code quality by meeting the following standard:

- A comment at the top of the program includes the name of the program, a brief statement of its purpose, your name, and the date that you finished the program.
- Variables have names that indicate the meaning of the values they hold.
- Code is indented consistently to show the program's structure. Each new statement should start on a new line. Unless you are tackling the "Additional Enrichment" suggestions, each new statement should start in the same column.

Logistics

Download the starter BlueJ project (madlib.zip) from the course website. The starter file indicates where you should write code for your program. Remember to include comments to remind yourself and your reader of the functionality of the different parts of your program. Please keep the name of the file **Madlib.java**.

2. **What to turn in:** Please submit your .java file to learning.up.edu before the due date and time. Please do not submit Madlib.class, Madlib.ctxt, or any other files. *Do not email your assignment to me.*