#### CS 203 HW #2Spring 2012

**Due Date:** Wednesday, Feb 1, 11 AM.

**Madlib Story**

You will create a madlib story that allows users to enter words/phrases of their choice which become part of the story. This assignment is intended to give you practice using and manipulating Strings in the Java language. If you are unfamiliar with madlibs, there are example stories at "Wacky Web Tales" at <http://www.eduplace.com/tales/>.

**Specification**1. A user of your program should be greeted with an informative message about your program.

2. The program should then ask the user to enter words and phrases of certain types. The types *could be* from the following list or of your own choosing.

* noun
* famous person
* adjective
* ing verb
* adverb
* famous person
* animal
* food
* plural noun
* silly sound
* color
* city/location

3. Once the user has entered a word/phrase for each type, your program should print the madlib story that includes the user's words and phrases. However, your program must do the following:

* (1 pt) Prompt user and store all words/phrases needed to complete your Madlib.
* (.5 pt) Your program *should trim all preceding and ending whitespace surrounding all user's input*. 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)

* (.5 pt) Your story should *have at least one sentence that describes the length of the word or phrase given by the user*. For example, if the user types "zebra" for an animal, your story could say something like "A zebra, an animal whose name has 5 letters, is a very interesting beast."
* (.5 pt) Your story should *have at least one sentence that describes the last letter/symbol of a word or phrase*. For example, if the user types "swish" for a silly sound, then your story could say something like "Then the car went swish, a sound that ends with the letter 'h'."
* (1 pt) Your story should *have at least one sentence that inserts ### into the middle of the user’s word (if the word has odd length, insert ### such that the first half has one letter less than the second half) .*For example, if the user types "basketball" then your story could say something like "Basketball was her favorite since baske###tball is so much harder to play." (Hint: think about using the % and / operators. You do not need to use a conditional to complete this task.)
* (1 pt) Your story should *alter at least one word or phrase entered by the user by substituting every instance of one letter with \**. For example, if the user types "giraffe" for an animal name, then your story could say something like "The gira\*\*e, a mythical beast related to a giraffe, differs from a giraffe since it has wings." This should work, regardless of the word/phrase entered by the user (Hint: find a character that appears in the word or phrase and then replace that character with a ‘\*’. You could find the last character in the word and then replace all instances of that character). You may assume the user types at least one letter for the word or phrase.
* (.5 pt) Your story should *change at least one word or phrase entered by the user to all lowercase letters.* 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."
* (1 pt) Your story should *prompt the user for a 2-word phrase and then display the two words in the opposite order.* For example, if the user types “freedom ring” then your story should display “ring freedom”.
* (1 pt) Your story, when printed to the screen, *should be formatted as a story*. Your story should have a **title** and start each new paragraph with tabbed indentation. You can print tab using \t in the string.
* You may *assume the user will obey the prompts in your program*, so they will always type a word/phrase when prompted.

**Example Interaction**The following is a printout from an example madlib. You do not need to follow this formatting exactly. In fact, you should create your own story. The following madlib **does not** contain all the requirements listed above, but it should give you some idea about the format of the output.

Welcome. You get to create your own story.
Please enter a girl's name:
Susie
Please enter a sport:
baseball
Please enter a children's toy (plural):
race cars
Please enter a noise:
Yeoaooow
Please enter a plural noun:
apples
Please enter a two-word thing:
coffee table
Please enter an animal:
giraffe
Please enter a 3-word phrase:
Hip Hop Hooray

              Baseball-Ville by Tammy VanDeGrift

     There once was a girl named Susie, who enjoyed playing baseball.
She liked baseball because the word has 8 letters. Instead of playing
baseball with baseballs, she played the game with race cars.
      One day, it was her turn, and she yelled yeoaooow followed by yeoa###ooow

and a ooow. Apples were falling from the sky. As she looked more closely,

she realized they were epples instead of apples. She did not like epples

because they were poisonous. She took refuge under a
table coffee, which starts with the letter 't'. A giraffe rescued her, she said,
"hip hop hooray" and the giraffe took her off to the land of Baseball-Ville,
where Susie could play baseball every day.
               The End.

 **Additional Enrichment**
If you have time you may add the following components to your program. Please document the additional features in your code and in your written summary.

* Add error checking 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.
* 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 extra spaces between the letters of a word/phrase inputted by the user. For example, if the user types "swoosh", then your story could present the word as "s - w - o - o - s - h". This will likely involve iteration, which can be found in chapter 4.
* Create a repository of more than one story and let the user decide which story to use. This will likely involve selection.
* Create a "choose your own adventure" madlib that allows the user to decide what should happen in the story. This will involve selection and gradually displaying pieces of the story.

**Logistics**

1. Download the starter BlueJ project **Madlib.zip**. Remember to include comments in your code to remind yourself and your reader the functionality of the different parts of your program. Please keep the name of the file Madlib.java.
2. Create a new folder called username\_HW2 and put your Madlib.java and MadlibSummary.docx files in the folder. Right-click on the folder to create a zip file. Submit the zip file to Moodle.

**Grading Guidelines [20 points total]**Your program will be graded on a scale of 0 to 7 in two categories:

* Code Quality: Design, Specification, Implementation, Documentation
* Code Operation: Functionality of code and adherence to project specification

Your summary report will be graded on a scale of 0 to 6 based on:

* Correct use of technical vocabulary
* Completeness and clarity
* Formatting
* Evidence of testing

**Report Guidelines and Format [note: these questions are different than for HW1]

1. Introduction and System Use:** List the order of words/phrases the user should type to complete the Madlib story.

**2. System Description:** Describe how the code works for the following tasks:

* Inserting ### in the middle of the word/phrase
* Reversing the two-word phrase
* Replacing a letter with \*

**3. Testing and Evaluation:**

**3.1:** If your program does not meet the specifications, please note these differences. Include a representative example of your test cases, by **including the printout in the window when your program is executed**.

**3.2**: Ask a person who is not enrolled in CS203 to use your program. What did he/she like about using your program and note any improvements that he/she suggests.

**3.3:** Evaluate the quality of your code: is it readable? do the parts make sense? is your code well-commented? Is there sufficient whitespace? Did you use descriptive variable names?

**4. Conclusion:** Describe the most challenging aspect(s) of this program and what you learned from completing it.

1. How long did you spend on this assignment?

2. By typing your name here, you are acknowledging that the code and report you are submitting are your own.

**Appendix:** Copy and paste your code here (use Courier font so the characters line up correctly).

Please keep your reports concise (no more than 2 pages - not including your example printout and code). A high quality computer scientist can explain their code, their process, and their testing to their colleagues and bosses. When writing the report, your audience is someone else taking CS 203. I ask you to do these reports because you will be expected to explain your ideas, whether or not you become a computer scientist.