Jasque Saydyk

Professor Elwakil, Melton, and McCarty

CS 136L Section 3801

27 January 2017

Lab 01 - MadLib

1. Problem Statement

The important features of this problem is the program needs to be able to ask for input, then accept the given input, then use the given input to insert it in a predefined paragraph of text that makes some amount of sense.

Requirements

- Paragraph with needed components and words noted
- Variables to store string inputs
- Asking the user for input
- Display paragraph with noted components and words replaced with input

2. Planning

While we can get fancy with this project and implement the input and output as separate methods in a separate class, then just run them in the main method, we decided not to do this. This is because the main advantage this would bring is helping us narrow down the source of an issue if we were to have one, and this problem is simple enough to where we don't expect we be of any use at all. Therefore we will implement this program in a procedural style in the main method of the program.

We could also make extensive use of arrays and loops to potentially cut down on the amount of code that needs to be written, however the problem isn't complex enough to warrant this treatment and may make the output code less readable.

As for error checking on the input, we decided that it wasn't necessary as the worst the user will be able to do is make their paragraph of text make no sense, and the simple act of ensuring the user inputs a word that exists increases the scope of the project to an unreasonable amount.

For gathering the input from the user, we will use the Scanner class, primarily the nextLine() method as it allows for the user to use spaces in their answer without bugging the program. The only other issue that need to be addressed is the usage of + to concatenate the paragraph with the input. To simply put it, the performance lost by using + to concatenate is not great enough to

justify replacing it with concat or StringBuilder. However, if this program were to be scaled up, this would have to be changed.

3. Implementation and Testing

Implementation of this plan was very straight-forward. After creating a list of String variables, we then made an input statement for each variable. After giving each String variable a value, we then insert it into the paragraph, concatenating it all together. As for testing the program, we just did a visual confirmation that the resulting paragraph was correct and that the program didn't crash with a variety of common keyboard strokes.

```
Input a Inanimate Object: Rock
Input a Monster: Unicorn
Input a Liquid: Water
Input a Verb: Sing
Input a Distance: Inch
Input a Size: Small
Input a Attack Name: Falcon Punch

 My First MadLib Story
========================
Once upon a time there was a Tree who was a(n) Blue
Plumber. One day, the Tree received a letter instructing It
to Attack the dragon protecting the Rock. This was quite strange, as Tree
has never Attack or even seen a dragon before. Despite that, Tree went on It's,
merry way to the dragon, but was accosted by a Unicorn. Tree flung his Water at the Unicorn
making the Unicorn five times larger and hungry. Tree tried to run away, only to be Sing
by the Unicorn, which Tree enjoyed. Tree then used his skills as a(n) Blue
Plumber to Sing the Monster to put it to sleep. Tree then traveled Inch
to encounter the dragon, who turned out to be a Small dragon. Tree
then to use his most powerful attack, Falcon Punch, but was burnt to a crisp by the dragon.
Tree's crispy body then fell into a lake of Water, and here Tree's journey ends.
```

4. Reflection

For the problem we were given and the intended size of this project, we believe we arrived at the best solution. We consider other possible ways we could implement this same program, but those methods had caveats that outweighed the benefits they would bring to the project. That being said, this program is not scalable in anyway, and if this program were to be built upon, then pursuing the other possible ways of implementing this program would be worth pursuing, specifically using a separate class for the input and output functionality of the program, replacing + with StringBuilder for faster processing, making extensive use of arrays and loops, and potentially outputting to a file.