**Project 2 - Making an AI**

*Due Date: 5:00 p.m., October 23, 2015*

*Any changes made to the assignment after posting it will be in red.*

Description

One of the most famous programs in the history of artificial intelligence is Joseph Weizenbaum's program Eliza. Written in the early sixties it emulates, or parodies, a Rogerian psychoanalyst who tries to make the patient talk about his or her deepest feelings by turning everything said into a question. Eliza's fame is due to the remarkable observation that many people found her seem so friendly and understanding. In fact some said she was the first analyst they had met who really listened.

Over the years Eliza has had many successors and the phenomenon she represents is now known as a chatterbot. A possible definition is:

*A chatterbot is a program that attempts to simulate typed conversation, with the aim of at least temporarily fooling a human into thinking they are talking to another person.*

Luckily, you do not need to fool another person, only script.

You are going to build a program that carries on a conversation with a user. Upon creation the chatterbot follows the following steps: say hello, wait for a response, decide to continue conversation or stop, say goodbye. The responses will be selected by using the following methods:

* first attempting a match with a list of responses which are contained in the chatter.txt
  + The file format is as follows: partial matches to user replies/statements are followed by a colon and a list of legitimate responses separated by semicolons
  + Read the chatter.txt file in at the start of your program, and store it in a data structure
* Second, if the user enters a question (anything followed by a question mark) not in chatter.txt, simply respond with, “Interesting, tell me what you think first.”
* If no partial match is found use one of the ‘change the subject’ phrases under the ‘change subject’ label in chatter.txt
* If the response drops down to 1 word, you end the conversation by saying goodbye in as friendly a way as possible.

Alter the chatterbot.txt to give your bot a name. Add to the chatter.txt at least 2 additional partial match phrases with at least 3 additional responses to select from. You may alter additional responses as long as you don’t alter the match phrases. Try to give your chatterbot a unique personality, etc.

Your code should try to read the file script.txt, using each line as a chat line. If it does not find the script, it should ask for input from the user.

Some design guidelines:

* Create a user or Chatter class to represent the user’s current state. Keep track of the number of words in the last response, and what the last response was.
* Whenever the user enters a response, it should trigger an event that requires a response from the chatterbot.
* Each class or module must go into a separate file. The main driver code should also be in a separate file.

In your readme, you should explain how you used the Template Method pattern, the strategy pattern, as well as the Observer pattern in your project.

You may make any changes to the requirements you wish to make the chatterbot more sophisticated or interesting. Just provide an explanation in the Readme. The only rule is that the changes must not simplify the project.

Submission

* Create a readme file following the previous project README format, and add it to your project folder
  + Here is an example submission that your should project should match:  
    [example\_submission.tar.gz](http://#)
* Create a tar archive with the command ”tar -czvf project2.tar.gz \*”, and then upload the archive to Blackboard before the deadline.
* Demo your project by downloading from blackboard and extracting your archive with the command "tar -xvf project2.tar.gz". Then run your code, show your source to to the TA, and answer any questions they may have.

Grading Guidelines

## **Project:**

* + Uses the template method to implement the conversation steps as described (4 points)
  + Uses the strategy pattern for response strategies (4 points)
  + Uses the observer pattern with a user class that triggers an event for the Chatterbot (5 points)
  + Uses the observable module from the Ruby library (2 points)
  + Added additional pattern matches and responses (2 points)
  + All classes in a separate file (2 point)

## **Submission (1 point):**

* + includes readme in plain text format (.txt)
  + submitted tar archive only files (not in a folder) to blackboard
  + Follows example submission