Final Project

CSCI E-10b

submitted by Anna Ntenta

“ SAGE ”

- Description

SAGE is a tongue-in-cheek program that emulates a jovial concierge. In this particular implementation, the concierge is in Rome, Italy and the user is assumed to be a tourist looking for sights and / or food. SAGE can however be easily altered to communicate about a different topic (more on that below.)
The program is loosely based on the famous ELIZA, the therapist computer from the 1970’s, and Apple’s more recent assistant program SIRI.
SAGE’s main function is to help the user find things in an unfamiliar environment. Pattern matching lies behind most of the SAGE’s supposed knowledge. The user interface consists of a very simple window in which SAGE and the user pose and answer questions. In the end you can choose to print a transcript of your conversation with SAGE. The program clandestinely saves the conversation in a log file when user quits the program.

SAGE uses two different .csv files to simulate its intelligence. The .csv files contain pairs of keywords and replies that SAGE uses when it finds a keyword in a question. One can expand SAGE’s knowledge by adding pairs to these files as one wishes, or, if you want to change the body of knowledge altogether, just replace the keyword-responses pairs altogether with new ones. The best way is to do this in a spreadsheet and export it to a .csv file.

I imagine this program installed on a computer somewhere where people often ask questions that mostly fall within a small field. Instead of waiting in line for a human to help, perhaps some people will choose to ask SAGE? My goal has been to make the program as ‘sage-like’ as possible, but I surmise that it will instead mostly aggravate the user with its nonsense replies.
Below I will explain how one can change SAGE’s knowledge base and

make the program more “intelligent”.

- Usage instructions

To run the program, you need to install three files in a writable directory:
1) *Sage.java* - the main program
2) *rome.csv* - a CSV file containing keyword- and answer pairs on the topic of Rome
3) *pointOfView.csv* - a CSV file containing keyword- and answer pairs that change user’s questions into the computer’s “point of view”.

4) You also need to install openCSV on your computer in order for the program to work. I have included the zip file for openCSV-2.3. On a Mac, put the opencsv-2.3.jar file in Library/Java/Extensions. If you’re using a different system you probably know how to install java libraries on it.

Then, to run the program, do the following while in the said directory:

a) Type “`javac Sage.java`” (no quotes)
b) Type “`java Sage`”

Please note that the directory cannot be write protected because the program will create a .txt file containing a log in the directory.
The only class is the Sage-class, which implements ActionListener and Printable

- Features

SAGE works best when the user asks about touristy things like food, transportation and local sights. It knows very little about anything else. This is not an innate constraint of the program - I just have not expanded SAGE’s body of knowledge as much as could be done in a live situation because it is not the main point of this exercise.

SAGE also has a few answers to questions about him/herself and to words

like “OK”, “hello”, “yes”, “no”, “thanks” and “thank you.”

SAGE randomly stores some of user’s questions and occasionally uses a rephrased version of the question when no key word is found in the latest question. See if you can talk to SAGE long enough for that to happen!

- Expansions

SAGE could be expanded in mainly two ways:
1) One could add more keyword-responses pairs to the .csv files in order to add more “intelligence” to the program.
2) Expand the GUI so that there could be a way for a superuser to log in and adapt SAGE to his/her business entirely.
Once the superuser is logged in, there could be a form where one can:
i) change the “greeting” message,
ii) change the generic replies that SAGE uses when there is no matching keyword, and
iii) add topic-specific intelligence to the program by typing in keyword pairs and save to a .csv file that will be used by the program.

- Reactions

I thought this project was a good choice for me. Its scope was just enough to have time to finish it with many other obligations on one’s plate. It did perhaps not teach me as much as I would have wished about stringent object-oriented principles or best practices - mainly because I think you need to work with someone experienced for good depth on those topics. I wish I could have built something better looking and more expansive in scope, but I am satisfied with my efforts nonetheless. I spent 40+ hours on actual coding (not including planning and designing.)