README

Zachary Maludzinski

HandleInput

String getUserInput()
Calls a scanner to take user input.

String processInput(String input)
Takes in a string, sets it to lowercase and removes punctuation. Returns the edited string.

String checkOccupation(String input)
Takes in a string, and determines what occupation is within by scanning through a text file of occupations.

String checkGender(String input)
Takes in a string, and determines whether the input contains words relating to a male or a female. Returns “man” or “woman”.

String parseInput(String input)
Takes in a string and determines context, using regular expression patterns and matching. Returns a keyword based on that context. Handles limited contexts related to the application.

String parseName(String input)
Takes in a string and determines what word is a name based off of context, capitalization and string structure. Returns the name.

String[] parseQResponse(String input, String qdata, Personality personality)
Takes in an input string to be parsed, a string containing a keyword related to what the asked question was, and a personality. Determines what the response was based upon question context gained from the qdata and outputs a two element string array, with the first element containing the matching likes from the personality and the second element containing the dislikes matching from the personality.

String keywordConvert(String input)
Converts a keyword used by parseQResponse to a keyword used by outputDeterminer.respond(). Takes in a keyword and outputs a keyword.

QuestionAsker

String[] ask()
Outputs a two-element array containing a question as the first element and a keyword that identifies the context of the question as the second. Used to ask the user questions.

String afterAsk(String input[], String qdata)
Outputs a response based upon what the user said following the ask() method. Takes in an array containing likes as the first element and dislikes as the second, directly from handleInput.parseQResponse(), as well as the data that identifies the context of the question asked.

Personality

setArray(ArrayList<String> array, int choose)
setArray(ArrayList<String> textFileArray, ArrayList<String> comparatorArray, int choose)
This method returns an array of randomized strings from the given list, it also avoids adding strings that are already in the comparatorArray.

setString(ArrayList<String> arraylist)
This method chooses a random string out of a given arraylist

txtToArray(String filename)
THis method converts the contents of a given .txt file to an ArrayList<String>

Terminalwgui

actionBtn.setOnAction()
Method to handle button clicked. Checks values and sends input to relevant classes for processing. Outputs bot response to GUI.

start()
A method which sets the stage, scene, buttons and h-boxes for javafx GUI.

DetermineOutput

This class is meant for providing responses dependent upon the data provided by the handleInput class. The HashMap data structure is used to store the responses, and the methods are implemented as follows:

String respond(String data, Personality personality)
This method takes a string and a personality. Responses are stored in a HashMap and provides the output as soon as the key matches to the key in the Hashmap. If a match is not found, the method returns a string containing a default response, otherwise, the string contains a response depending on the context.

String occupation(String response)
Takes a string containing a keyword relating to the occupation from handleInput and returns a string containing a response based upon that keyword.

String returnString()
Takes in an ArrayList and returns a string.

String returnName(Personality p)
Takes an instance of personality and returns bot's name.

The below methods return strings containing random responses(provide output in three different ways).
String returnGreeting()
String returnEndDate()
String returnQdoing()
String returnSwearing()
String returnInsult()
String Howru()

Testing

This class is intended for coverage and unit testing.

Void testParseInput()
Tests if the parseInput method within the handleInput class returns a required keyword.

Void testrespond()
Tests if the respond method within the determineOutput class returns the expected output as per the given input.

Void ArrayListToString()
Checks if it takes an arraylist, and returns a string.

Void testReturnGreeting()
Void testEndDate()
Void testQdoing()
Void testSwearing()
Void testHowru()
Void testInsult()

All the above methods, checks for random responses for a given input.

Scoring

A class for keeping track of score values and conversion between classes. Contains getters and setters. Everything is self-explanatory within the class.

Class Diagrams

Requirement

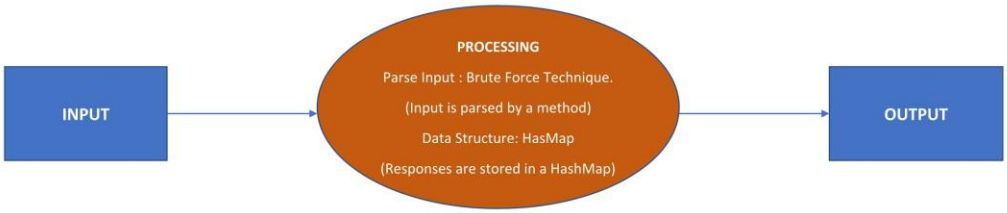
Chatbot for a Blind Date

INPUT : Input will be received.

PROCESSING: As per the given input, output will be generated.

OUTPUT: Output will be provided to user.

DESIGN

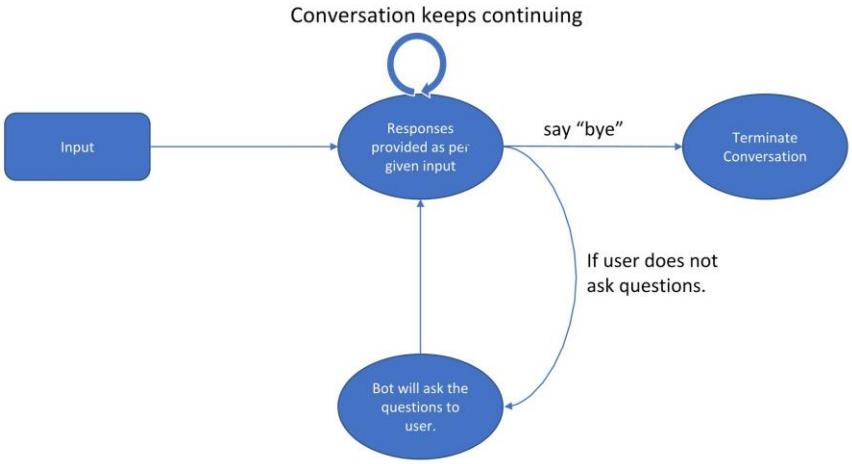


INPUT : Input is received by handleInput class.

PROCESSING: The received input is parsed into a key(keyword) by a method named parseInput(). According to a keyword, the responses are being stored in HashMap.

OUTPUT: As a key matches to a key stored in HashMap, it provides the response.

3-State Diagram



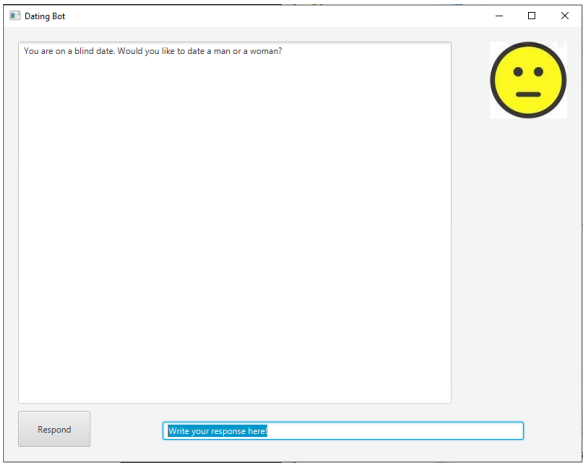
Feature List

Scoring Algorithm w/ GUI Indicator

After each round of conversation, the user's score is updated, and the GUI indicator is updated if applicable. The indicator has 5 different images, for very good, good, neutral, bad and very bad. This gives the user an idea how "well" their date is going. The score is affected by shared likes or dislikes, conflicting likes and dislikes and by insults or vulgar language. This score also impacts whether the bot will ask for another date at the end of the conversation.

Improved GUI

Changed GUI from multiple, buggy windows to one window, with all conversation happening in the one window. A history of the conversation is visible, and the scoring indicator is visible as well.



Added Conversation Topics

Two extra topics were implemented into the chatbot's span of topics. Since the bot covers more of a wide range of topics than a deep dive into the topics, two were implemented rather than one. Food and animals were both implemented.

5 Reasonable Responses Outside of Topics

Swearing, insults and unrecognized questions are all handled already. The user can also ask if the bot is a student, what the bot's job is, the bot's age, what the bot is currently doing and what the bot's zodiac sign is. If a user now asks a question twice, the bot will inform them, and get upset. Most responses choose randomly from three different phrases that get across the same idea to keep the conversation fresh for every try.

Spelling Mistake Handling

Very simple and rudimentary form of handling spelling mistakes using regular expressions.