CSC 221 | Final Group Project | Anna Pfaff and Bianna Quiroga | 11/4/2024

Design Document for Magic 8-Ball Project

1. Overview of Software

General Functionality:
The Magic 8-Ball program we will create is a digital version of the fortune-telling toy. When the user asks a yes-or-no question and activates or *shakes* the 8-ball program, it randomly selects and displays one of several possible answers, providing the user with a "prediction."

Problem Addressed:
This software allows users to receive randomized answers to their questions, translating the physical experience of using a Magic 8-Ball with a digital interactive Python program.

2. Analysis of Software

Objective:
The primary objective is to create a functional interactive Python program that asks for user input (a question) and then responds with a randomly selected answer from a set list of possible 8-ball replies.

Requirements:

- The software should prompt the user for a question.
- It should randomly select a response from a predefined list of possible answers.
- The program should print the response to the user.
- The program should allow the user to ask another question.

3. Solution Overview

What the Solution Looks Like:
The solution will be a Python code set that will prompt users' inputs through a command-line interface program that allows inputs of questions, responds with random answers from the set list of possible responses, and then follows up with an en exit or replay prompt.

Potential Solution Information:

- Random response generation will done through using Python's *random* function.
- The program will have the option to loop or exit the program at the end of the code, allowing repeated questions and responses, until the user chooses to exit.
- The potential answers that are randomly generated by the program will come from the official magic 8 ball site, magic-8ball.com which offers 20 possible answers in total, 10

affirmative answers, 5 non-committal answers, and 5 negative answers which are as follows:

Affirmative Answers

- It is certain
- It is decidedly so
- Without a doubt
- Yes definitely
- You may rely on it
- As I see it, yes
- Most likely
- Outlook good
- Yes
- Signs point to yes

Non-Committal Answers

- Reply hazy, try again
- Ask again later
- Better not tell you now
- Cannot predict now
- Concentrate and ask again

Negative Answers

- Don't count on it
- My reply is no
- My sources say no
- Outlook is not so good
- Very doubtful

4. Algorithm

Pseudocode:

```
1. Import random module
2. Define a list titled 'responses' with the 20 possible responses of the Magic 8-Ball
```

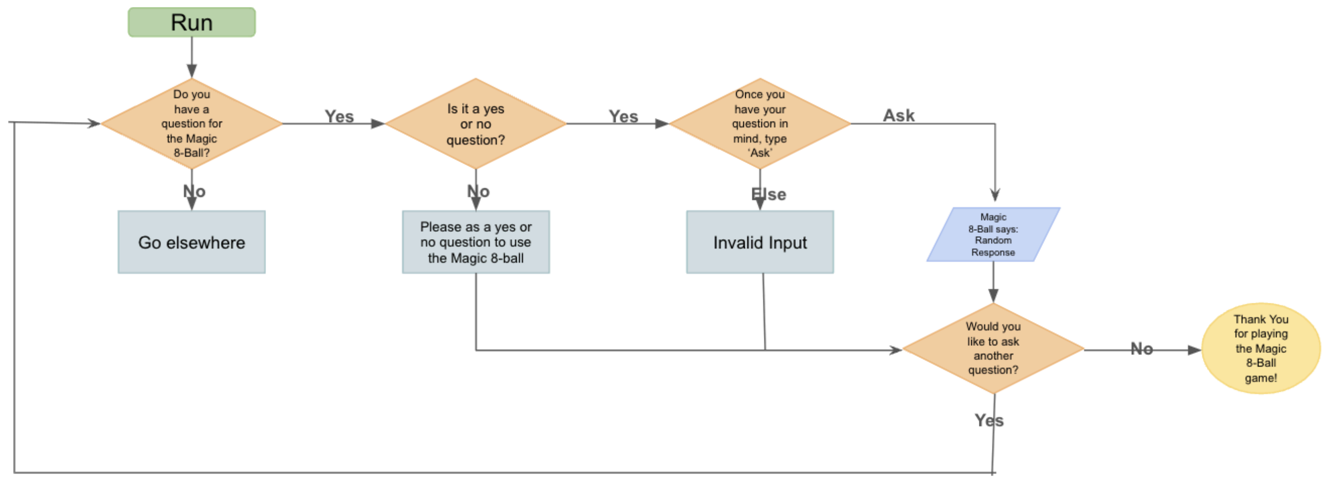
Affirmative Answers	**Non–Committal Answers**	**Negative Answers**
“It is certain”	“Reply hazy, try again”	“Don’t count on it”
“It is decidedly so”	“Ask again later”	“My reply is no”
“Without a doubt”	“Better not tell you now”	“My sources say no”
“Yes definitely”	“Cannot predict now”	“Outlook not so good”
“You may rely on it”	“Concentrate and ask again”	“Very doubtful”
“As I see it, yes”		
“Most likely”		
“Outlook good”		
“Yes”		
“Signs point to yes”		

3. Define function ‘get response () ’ :

   Use random to select and return a random response from ‘self.responses’

4. In the main program write notes
   - Title the code with #CSC 221 Design Document Magic 8 Ball Project by Anna Pfaff and Bianna Quiroga
   - Directions for how to use the program on the user side
   - “Do you have a question for the Magic 8-Ball? Yes or No?
     - If yes, continue
     - If no, print (“Go elsewhere”)
   - “Is it a yes or no question” Yes or No?
     - If yes, continue
     - If no, print (“Make it into a yes or no question.”) then continue
   - “Would you like to play the Magic 8-Ball game?”
     - If yes, continue
     - If no, print (“Please ask a yes or no question to use the Magic 8-Ball.”)
   - “Once you have your question in mind, type ‘Ask’ to get a response: ”
   - Print a random one of the defined responses
   - Replay: “Would you like to ask another question?”
     - If yes, continue with the restarted prompts
     - If no, then exit the program

Flowchart:


Run
Do you have a question for the Magic 8-Ball?
Yes
No
Go elsewhere
Is it a yes or no question?
Yes
No
Please as a yes or no question to use the Magic 8-ball
Once you have your question in mind, type 'Ask'
Ask
Else
Invalid Input
Magic 8-Ball says: Random Response
Would you like to ask another question?
No
Yes
Thank You for playing the Magic 8-Ball game!

5. Object-Oriented Design (Optional)

For a more advanced version, we could use an OOP approach:

- **Class**: `Magic8Ball`
    - **Attributes**: `responses` (list of possible answers)
    - **Methods**:
        - get_response() – returns a random response from responses
        - ask_question() – prompts the user for input, retrieves a response, and prints it.

6. Algorithm Discovery

Top-Down Approach: The top-down approach was chosen for the development of this program as it starts by defining the main functionality and user flow before diving into smaller, detailed functions. This method allows us to first map out the broader structure of how the user interacts with the program and then break down each part into simpler, more manageable subcomponents.

Reasoning:

- **User Interaction**: The main loop of the program was structured first, ensuring that the user flow from asking a question to receiving a response was clear and logical.

- **Core Functions**: After outlining the main flow, we defined the get_8_ball_response() function to handle the random selection of responses.
- **Modular Design**: By starting with the main framework, we were able to identify areas that could be modularized, such as user prompts and response handling.
- **Scalability**: This approach allows easy expansion or enhancement of the program (e.g., adding more response categories or improving user input handling).

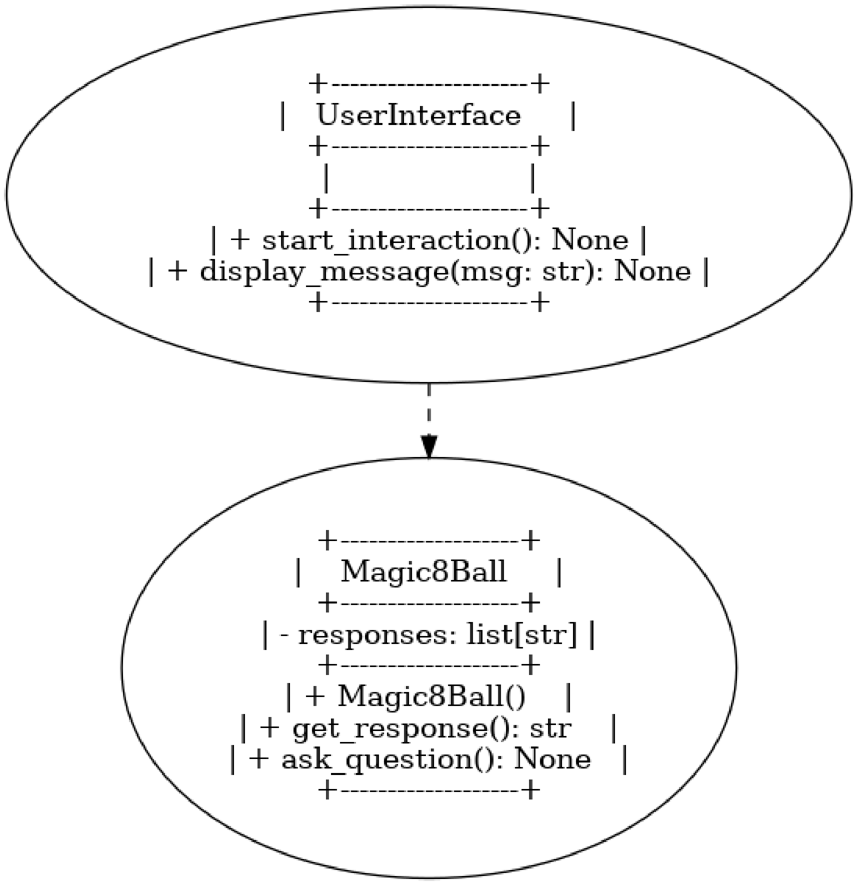
Implementation Details:

1. Start with the main program structure to handle input and the primary user loop.
2. Define auxiliary functions like get_8_ball_response() that are called within the main loop.
3. Refine the process by ensuring the program handles edge cases, such as when a user doesn't input a valid response.
4. Test each function iteratively to confirm they integrate seamlessly into the main flow.

Benefits:

The top-down approach was the most effective for structuring the Magic 8-Ball program, providing a clear roadmap from overall functionality to specific task implementations.

UML Diagram:


UserInterface
+ start_interaction(): None
+ display_message(msg: str): None
Magic8Ball
- responses: list[str]
+ Magic8Ball()
+ get_response(): str
+ ask_question(): None