**TITLE:** CodTech IT Solutions Internship

**Task Documentation:** “SIMPLE JAVA CHATBOT” Using Java Programing.

**INTERN INFORMATION:**

**Name:** Nikhil Thipparthi

**ID:** ICOD6447

**INTRODUCTION**

A simple chatbot is a computer program designed to simulate conversation with human users, typically through text-based interfaces like messaging platforms or websites. These chatbots are built using various technologies such as natural language processing (NLP), machine learning, and rule-based systems.

At their core, simple chatbots analyze user input, understand the intent behind it, and generate appropriate responses based on predefined rules or learned patterns. While they may lack the sophistication of more advanced AI systems, they are still capable of providing basic assistance, answering common questions, and guiding users through predefined workflows.

Simple chatbots are commonly used in customer service, providing automated responses to frequently asked questions, assisting with basic tasks like appointment scheduling or order tracking, and facilitating engagement on websites or social media platforms. They offer businesses a cost-effective way to provide round-the-clock support and improve user experience without requiring human intervention for every interaction.

**Implementation**

Here's a simple implementation of a chatbot in Java. This example will create a console-based chatbot that interacts with the user by echoing their input.

**JAVA CODE:**

import java.util.Scanner;

import java.io.\*;

public class SampleChatbot {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Chatbot: Hi! How can I assist you today?");

String userInput;

do {

System.out.print("User: ");

userInput = scanner.nextLine();

String botResponse = generateResponse(userInput);

System.out.println("Chatbot: " + botResponse);

} while (!userInput.equalsIgnoreCase("exit"));

System.out.println("Chatbot: Goodbye! Have a great day!");

scanner.close();

}

public static String generateResponse(String userInput) {

String response;

switch (userInput.toLowerCase()) {

case "hi":

case "hello":

response = "Hello there!";

break;

case "What is your name?":

response = "My name is Chatbot!";

break;

case "how are you":

response = "I'm just a simple chatbot.";

break;

case "tell me a joke":

response = "Why did the math book look sad? Because it had too many problems! 😄";

break;

case "wanna be friends":

response = "We are already friends.";

break;

case "open notepad":

try {

Runtime rs = Runtime.getRuntime();

rs.exec("notepad");

} catch (IOException e) {

response = "Error: " + e.getMessage();

}

response = "Notepad opened.";

break;

case "exit":

response = "Sure, exiting now.";

break;

default:

response = "I'm not sure how to respond to that.";

break;

}

return response;

}

}

**CODE EXPLAINATION**

**import java.util.Scanner;**

**import java.io.\*;**

The code starts by importing necessary packages. java.util.Scanner is used to read input from the console, and java.io.\* is imported to handle input/output operations.

**public class SampleChatbot {**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**System.out.println("Chatbot: Hi! How can I assist you today?");**

**String userInput;**

The SampleChatbot class is declared. Inside the main method, a Scanner object is created to read user input from the console. The chatbot starts by greeting the user with "Chatbot: Hi! How can I assist you today?" and declares a String variable userInput to store the user's input.

**do {**

**System.out.print("User: ");**

**userInput = scanner.nextLine();**

**String botResponse = generateResponse(userInput);**

**System.out.println("Chatbot: " + botResponse);**

**} while (!userInput.equalsIgnoreCase("exit"));**

A do-while loop is used to continuously prompt the user for input and provide responses until the user types "exit". Within the loop:

The bot prompts the user with "User: " to indicate where the user should input their message.

The user's input is read using scanner.nextLine() and stored in userInput.

The generateResponse method is called with userInput as an argument to generate the bot's response.

The bot's response is printed to the console along with "Chatbot: ".

The loop continues until the user types "exit", at which point the loop terminates.

**System.out.println("Chatbot: Goodbye! Have a great day!");**

**scanner.close();**

**}**

After the loop exits (i.e., when the user types "exit"), the bot prints "Chatbot: Goodbye! Have a great day!" to bid farewell to the user and closes the Scanner object to release system resources.

**public static String generateResponse(String userInput) {**

**String response;**

**switch (userInput.toLowerCase()) {**

**// Cases for different user inputs and their corresponding responses**

**}**

**return response;**

**}**

The generateResponse method takes the user's input as a parameter and returns a response based on the input. It uses a switch-case statement to determine the response based on the user's input. Each case represents a different input, and the corresponding response is assigned to the response variable. Finally, the method returns the response.

**USAGE**

To use the sample Java chatbot provided, you can follow these steps:

Compile the Code: Save the code provided in a file named SampleChatbot.java. Then, compile the code using the Java compiler. You can do this by opening a terminal or command prompt, navigating to the directory where the file is saved, and running the command:

Copy code

javac SampleChatbot.java

Run the Program: After successfully compiling the code, you can run the program using the Java interpreter. Execute the following command in the terminal or command prompt:

Copy code

java SampleChatbot

Interact with the Chatbot: Once the program starts running, you'll see the chatbot greeting you with a message: "Chatbot: Hi! How can I assist you today?" You can then type your messages as input, and the chatbot will respond accordingly. It will continue this interaction until you type "exit", at which point it will bid you farewell and terminate.

Here's an example of how the interaction might look:

vbnet

Copy code

Chatbot: Hi! How can I assist you today?

User: Hello

Chatbot: Hello there!

User: Tell me a joke

Chatbot: Why did the math book look sad? Because it had too many problems! 😄

User: Open Notepad

Chatbot: Notepad opened.

User: How are you?

Chatbot: I'm just a simple chatbot.

User: Bye

Chatbot: Goodbye! Have a great day!

**CONCLUSION**

The Java sample chatbot provided is a simple yet effective demonstration of basic conversational AI principles. Here's a summary of its key aspects:

1. **Functionality**: The chatbot can engage in conversation with users through the console. It recognizes specific commands and phrases and responds accordingly. Commands include greetings, inquiries about its name and well-being, telling jokes, opening Notepad, and exiting the conversation.
2. **Command Handling**: User input is processed using a switch-case statement in the **generateResponse** method, which determines the appropriate response based on the input. This approach allows for straightforward command recognition and response generation.
3. **Interaction Loop**: The chatbot engages in a continuous conversation with the user until the user types "exit". This loop ensures that the chatbot remains interactive and responsive throughout the conversation session.
4. **Error Handling**: While the chatbot handles expected commands effectively, it lacks robust error handling for unexpected or invalid inputs. Enhancements could be made to provide more informative responses or prompt the user to rephrase their input when necessary.
5. **Extendability**: The code serves as a solid foundation for building more complex chatbots with additional features and capabilities. Developers can extend the functionality by integrating natural language processing (NLP) libraries, implementing external APIs for dynamic responses, or creating a graphical user interface (GUI) for improved user interaction.

Overall, the Java sample chatbot offers a practical starting point for experimenting with conversational AI concepts in Java. With further refinement and enhancement, it could be tailored to suit various use cases, such as customer support, informational assistance, or entertainment.

Top of Form

**OUTPUT**



