# 结合大模型的Java知识问答机器人

## 1. 项目概述

本项目旨在开发一个结合大模型的Java知识问答机器人，用户可以通过图形用户界面（GUI）与机器人进行交互，选择不同的大模型（如阿里通义千问、百度文心一言）来获取问题的答案。项目实现了多线程处理，确保用户请求不会阻塞主线程，并且能够记录聊天日志到本地文件。

## 2. 需求分析

人机交互界面：使用图形用户界面（GUI）进行交互。

大模型选择：支持至少两种大模型（阿里通义千问、百度文心一言）。

多线程处理：每个用户请求在新的线程中处理，避免阻塞主线程。

日志记录：记录聊天过程到本地文本文件。

测试案例：编写测试案例以验证程序功能。

## 3. 设计模式

工厂模式：用于创建不同大模型的API实例。

观察者模式：用于监听用户输入和厂商选择的变化。

线程池模式：用于异步处理用户请求。

## 4. 项目结构

JavaKnowledgeBot.java：主类，包含GUI界面和逻辑处理。

ModelAPI.java：接口，定义与大模型交互的方法。

AliyunAPI.java：实现ModelAPI接口，用于阿里通义千问大模型。

BaiduAPI.java：实现ModelAPI接口，用于百度文心一言大模型。

ModelAPIFactory.java：工厂类，根据厂商选择创建相应的ModelAPI实例。

## 5. 代码实现

### 5.1 JavaKnowledgeBot.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.util.Arrays;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class JavaKnowledgeBot extends JFrame {

private JComboBox<String> vendorComboBox;

private JTextArea chatTextArea;

private JTextField inputTextField;

private JButton sendButton;

private ExecutorService threadPool;

// 定义可用的厂商选项

private static final String[] VENDORS = {"阿里-通义千问", "百度-文心一言"};

public JavaKnowledgeBot() {

setTitle("知识问答机器人");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// 初始化线程池用于并行处理请求

threadPool = Executors.newCachedThreadPool();

// 创建布局

setLayout(new BorderLayout());

// 左边面板 - 模型选择和配置

JPanel leftPanel = new JPanel();

leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y\_AXIS));

vendorComboBox = new JComboBox<>(VENDORS);

vendorComboBox.addActionListener(this::onVendorChange); // 添加厂商切换监听器

leftPanel.add(vendorComboBox);

add(leftPanel, BorderLayout.WEST);

// 右边面板 - 聊天显示和输入

JPanel rightPanel = new JPanel();

rightPanel.setLayout(new BorderLayout());

chatTextArea = new JTextArea();

chatTextArea.setEditable(false);

JScrollPane scrollPane = new JScrollPane(chatTextArea);

rightPanel.add(scrollPane, BorderLayout.CENTER);

// 输入和发送按钮

JPanel inputPanel = new JPanel();

inputTextField = new JTextField(40);

sendButton = new JButton("发送");

sendButton.addActionListener(this::onSendMessage);

inputPanel.add(inputTextField);

inputPanel.add(sendButton);

rightPanel.add(inputPanel, BorderLayout.SOUTH);

add(rightPanel, BorderLayout.CENTER);

// 初始加载默认厂商的聊天记录

loadChatRecord();

}

private void onSendMessage(ActionEvent e) {

String userInput = inputTextField.getText().trim();

if (!userInput.isEmpty()) {

// 显示用户消息

chatTextArea.append("User: " + userInput + "\n");

// 保存用户消息到本地

saveChatRecord("User: " + userInput);

// 清空输入框

inputTextField.setText("");

// 在后台线程中处理用户消息并与大模型交互

threadPool.submit(() -> {

try {

// 根据当前选择的厂商创建API实例

ModelAPI api = createAPIInstance((String) vendorComboBox.getSelectedItem());

// 调用API并获取回复

String modelResponse = api.call(userInput);

SwingUtilities.invokeLater(() -> {

// 更新UI，显示大模型回复

chatTextArea.append("Bot: " + modelResponse + "\n");

// 保存大模型回复到本地

saveChatRecord("Bot: " + modelResponse);

});

} catch (Exception ex) {

ex.printStackTrace();

SwingUtilities.invokeLater(() -> {

chatTextArea.append("Error: " + ex.getMessage() + "\n");

// 保存错误信息到本地

saveChatRecord("Error: " + ex.getMessage());

});

}

});

}

}

private synchronized void saveChatRecord(String message) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(getLogFilePath(), true))) {

writer.write(message);

writer.newLine(); // 确保每条记录在新的一行开始

} catch (IOException e) {

e.printStackTrace();

}

}

private void loadChatRecord() {

String logFilePath = getLogFilePath();

try (BufferedReader reader = new BufferedReader(new FileReader(logFilePath))) {

String line;

while ((line = reader.readLine()) != null) {

chatTextArea.append(line + "\n");

}

} catch (IOException e) {

// 文件不存在或读取失败，可能是首次使用或没有之前的记录

// 可以选择忽略或者通知用户

}

}

private void onVendorChange(ActionEvent e) {

// 当厂商选择改变时，重新加载对应厂商的日志记录

chatTextArea.setText(""); // 清空当前聊天记录

loadChatRecord(); // 加载新选厂商的聊天记录

}

private String getLogFilePath() {

// 根据选择的厂商返回不同的日志文件路径

String selectedVendor = (String) vendorComboBox.getSelectedItem();

return "chat\_log\_" + selectedVendor.replace(" ", "\_") + ".txt";

}

private ModelAPI createAPIInstance(String vendorModelPair) {

return ModelAPIFactory.createAPI(vendorModelPair);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JavaKnowledgeBot bot = new JavaKnowledgeBot();

bot.setVisible(true);

});

}

}

### 5.2 ModelAPI.java

public interface ModelAPI {

String call(String userInput) throws Exception;

}

### 5.3 AliyunAPI.java

import com.alibaba.dashscope.aigc.generation.Generation;

import com.alibaba.dashscope.aigc.generation.GenerationParam;

import com.alibaba.dashscope.aigc.generation.GenerationResult;

import com.alibaba.dashscope.common.Message;

import com.alibaba.dashscope.common.Role;

import com.alibaba.dashscope.exception.ApiException;

import com.alibaba.dashscope.exception.InputRequiredException;

import com.alibaba.dashscope.exception.NoApiKeyException;

import java.util.Arrays;

public class AliyunAPI implements ModelAPI {

private final String apiKey;

private final String model;

public AliyunAPI() {

this.apiKey = Config.ALIYUN\_API\_KEY;

this.model = Config.ALIYUN\_MODEL;

}

@Override

public String call(String userInput) throws ApiException {

Generation gen = new Generation();

Message systemMsg = Message.builder()

.role(Role.SYSTEM.getValue())

.content("You are a helpful assistant.")

.build();

Message userMsg = Message.builder()

.role(Role.USER.getValue())

.content(userInput)

.build();

GenerationParam param = GenerationParam.builder()

.apiKey(apiKey)

.model(model)

.messages(Arrays.asList(systemMsg, userMsg))

.resultFormat(GenerationParam.ResultFormat.MESSAGE)

.build();

GenerationResult result = null;

try {

result = gen.call(param);

} catch (NoApiKeyException e) {

e.printStackTrace();

} catch (InputRequiredException e) {

e.printStackTrace();

}

return extractContentFromResult(result);

}

private String extractContentFromResult(GenerationResult result) {

if (result.getOutput() != null && !result.getOutput().getChoices().isEmpty()) {

return result.getOutput().getChoices().get(0).getMessage().getContent();

}

return "Sorry, I couldn't generate a response.";

}

}

### 5.4 BaiduAPI.java

import okhttp3.\*;

import org.json.JSONArray;

import org.json.JSONObject;

import java.io.IOException;

public class BaiduAPI implements ModelAPI {

private final String apiKey;

private final String secretKey;

private final String apiUrl;

private final OkHttpClient httpClient = new OkHttpClient();

public BaiduAPI() {

this.apiKey = Config.BAIDU\_API\_KEY;

this.secretKey = Config.BAIDU\_SECRET\_KEY;

this.apiUrl = Config.BAIDU\_API\_URL;

}

@Override

public String call(String userInput) throws Exception {

String accessToken = getAccessToken();

String url = apiUrl + "?access\_token=" + accessToken;

MediaType mediaType = MediaType.parse("application/json");

JSONObject requestBodyJson = new JSONObject();

JSONArray messagesArray = new JSONArray();

JSONObject messageObject = new JSONObject();

messageObject.put("role", "user");

messageObject.put("content", userInput);

messagesArray.put(messageObject);

requestBodyJson.put("messages", messagesArray);

RequestBody body = RequestBody.create(mediaType, requestBodyJson.toString());

Request request = new Request.Builder()

.url(url)

.post(body)

.addHeader("Content-Type", "application/json")

.build();

try (Response response = httpClient.newCall(request).execute()) {

if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

JSONObject jsonResponse = new JSONObject(response.body().string());

return jsonResponse.get("result").toString();

}

}

private String getAccessToken() throws IOException {

MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");

RequestBody body = RequestBody.create(mediaType, "grant\_type=client\_credentials&client\_id=" + apiKey

+ "&client\_secret=" + secretKey);

Request request = new Request.Builder()

.url("https://aip.baidubce.com/oauth/2.0/token")

.post(body)

.addHeader("Content-Type", "application/x-www-form-urlencoded")

.build();

try (Response response = httpClient.newCall(request).execute()) {

if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

JSONObject jsonResponse = new JSONObject(response.body().string());

return jsonResponse.getString("access\_token");

}

}

}

### 5.5 ModelAPIFactory.java

public class ModelAPIFactory {

public static ModelAPI createAPI(String vendor) {

switch (vendor.toLowerCase()) {

case "阿里-通义千问":

return new AliyunAPI();

case "百度-文心一言":

return new BaiduAPI();

default:

throw new IllegalArgumentException("Unknown vendor: " + vendor);

}

}

}

## 6. 设计模式解释

工厂模式 (ModelAPIFactory)：用于创建不同厂商的大模型 API 实例，根据传入的供应商名称（vendor）选择相应的实现类。这种设计使得添加新的大模型支持变得容易，只需要在 ModelAPIFactory 中新增一个 case 即可。

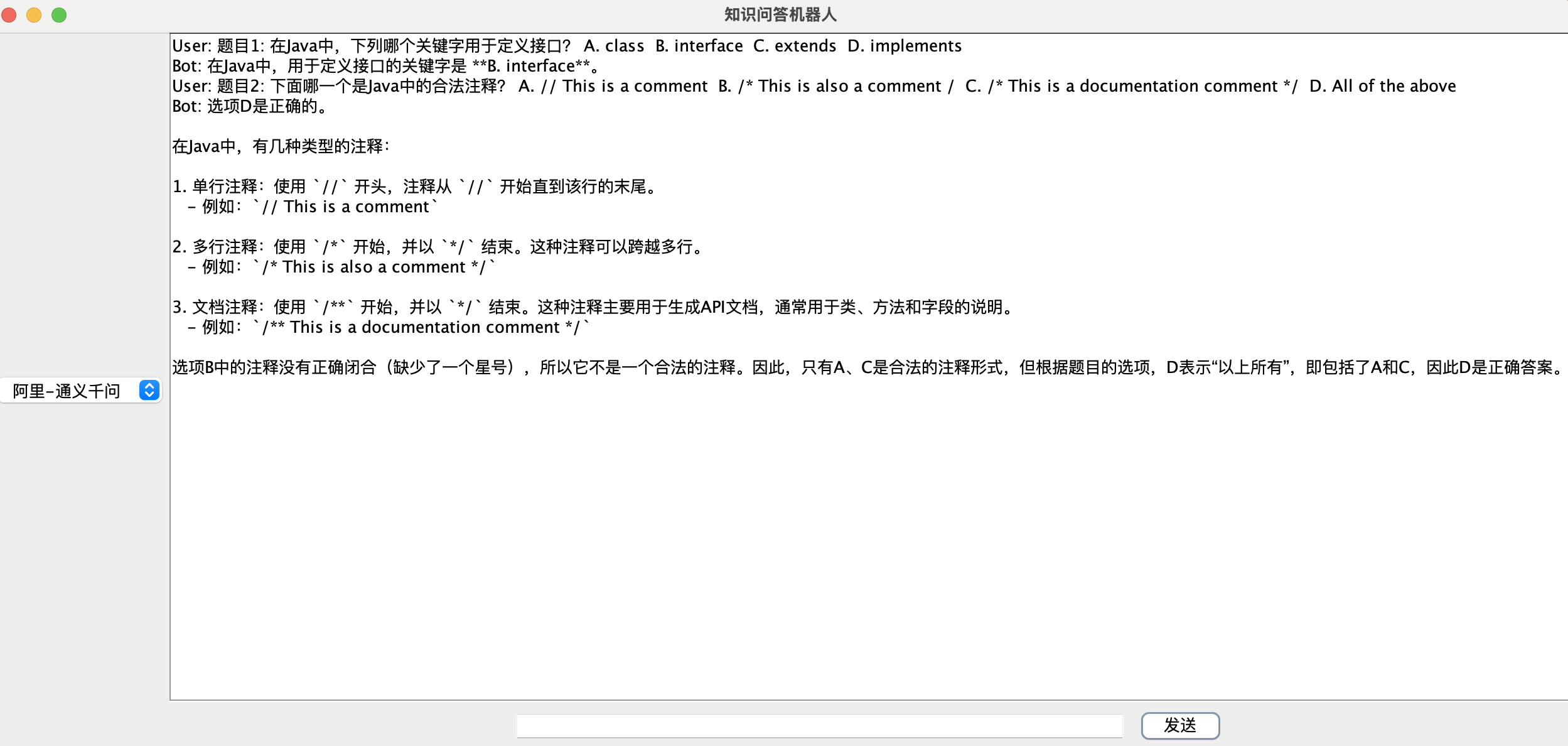
观察者模式 (JavaKnowledgeBot 中的 onVendorChange 方法)：监听用户界面中的事件（如厂商选择的变化），并做出响应。例如，当用户改变厂商选择时，程序会重新加载对应厂商的日志记录。

线程池模式 (ExecutorService threadPool)：使用线程池来管理并发任务。每当有用户发送消息时，都会提交一个新的任务给线程池执行，这确保了即使在等待外部 API 响应时，UI 线程也不会被阻塞，从而保持应用程序的响应性。

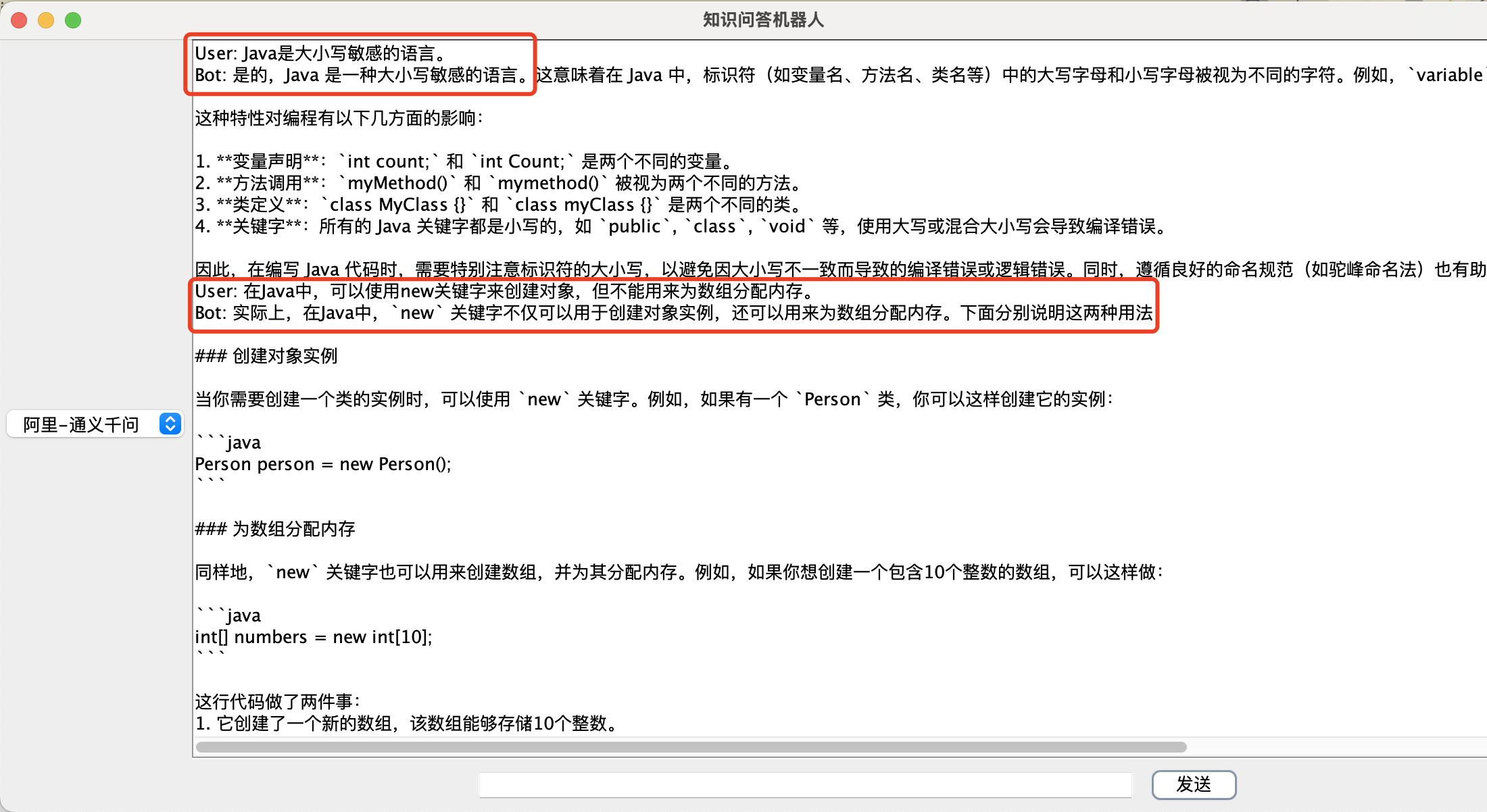
## 7. 测试案例

为了验证项目功能，可以编写以下测试案例：

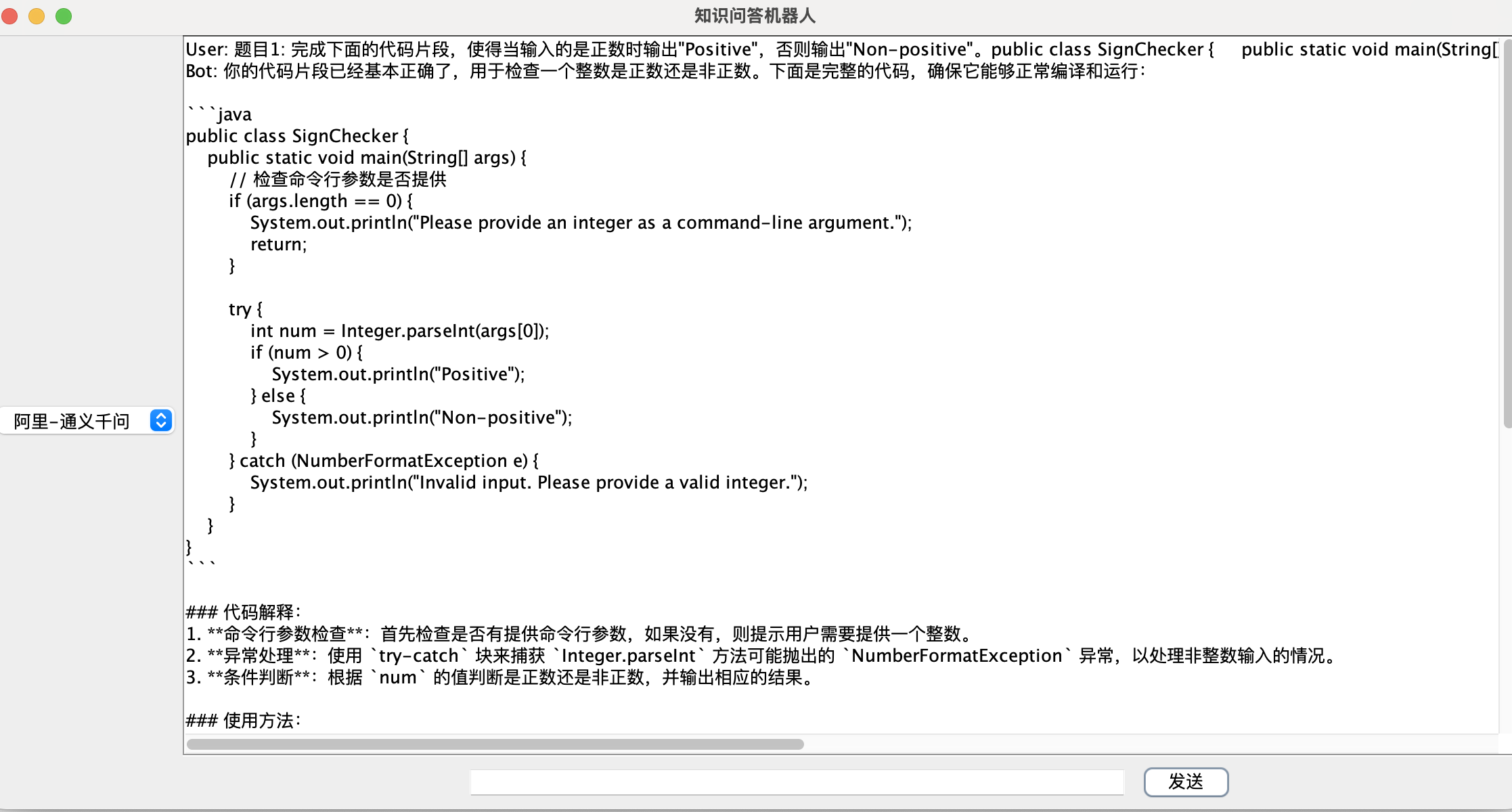
测试用例1：通过问答测试选择题如下。



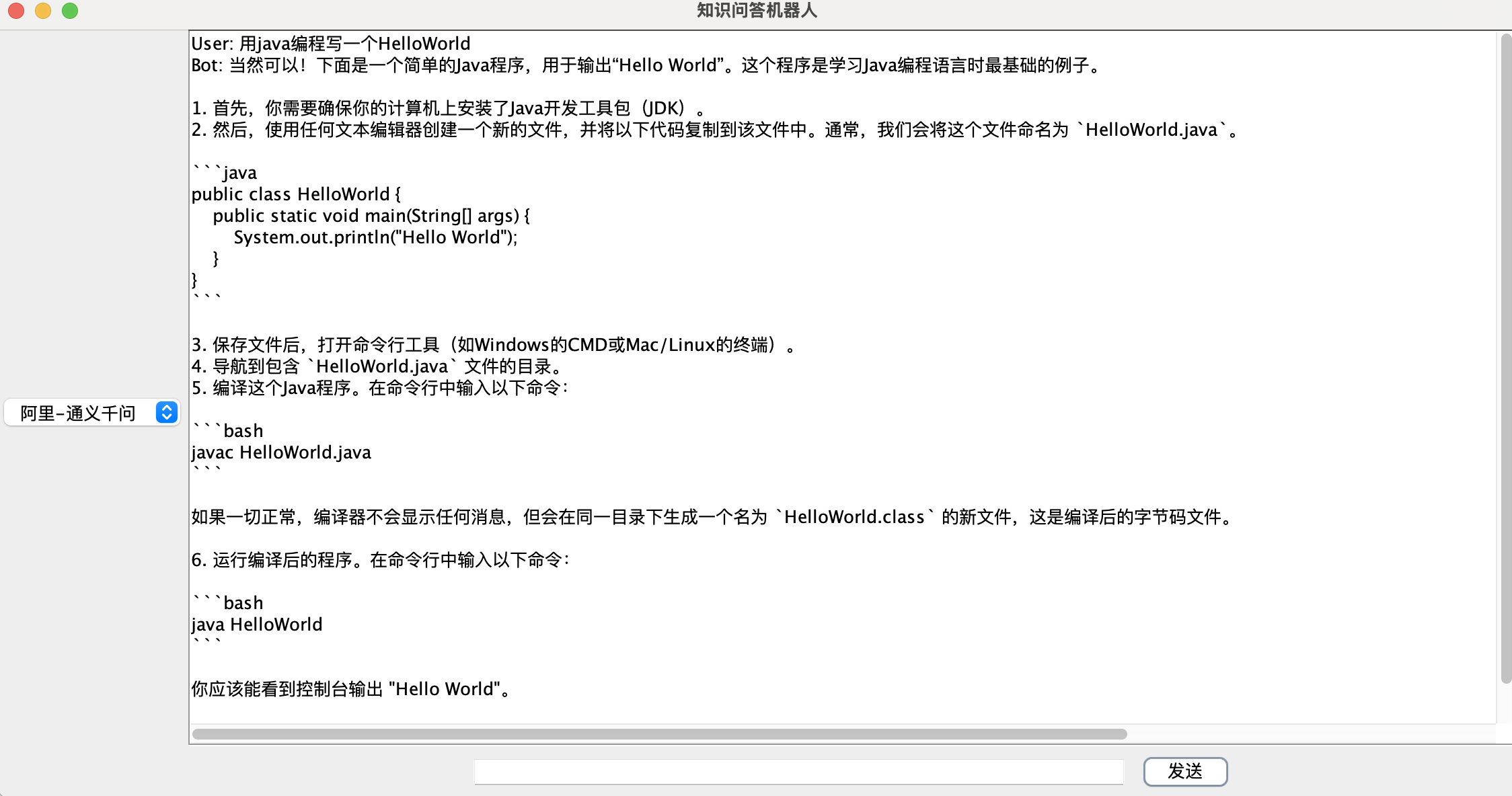
测试用例2：通过问答测试判断题如下。

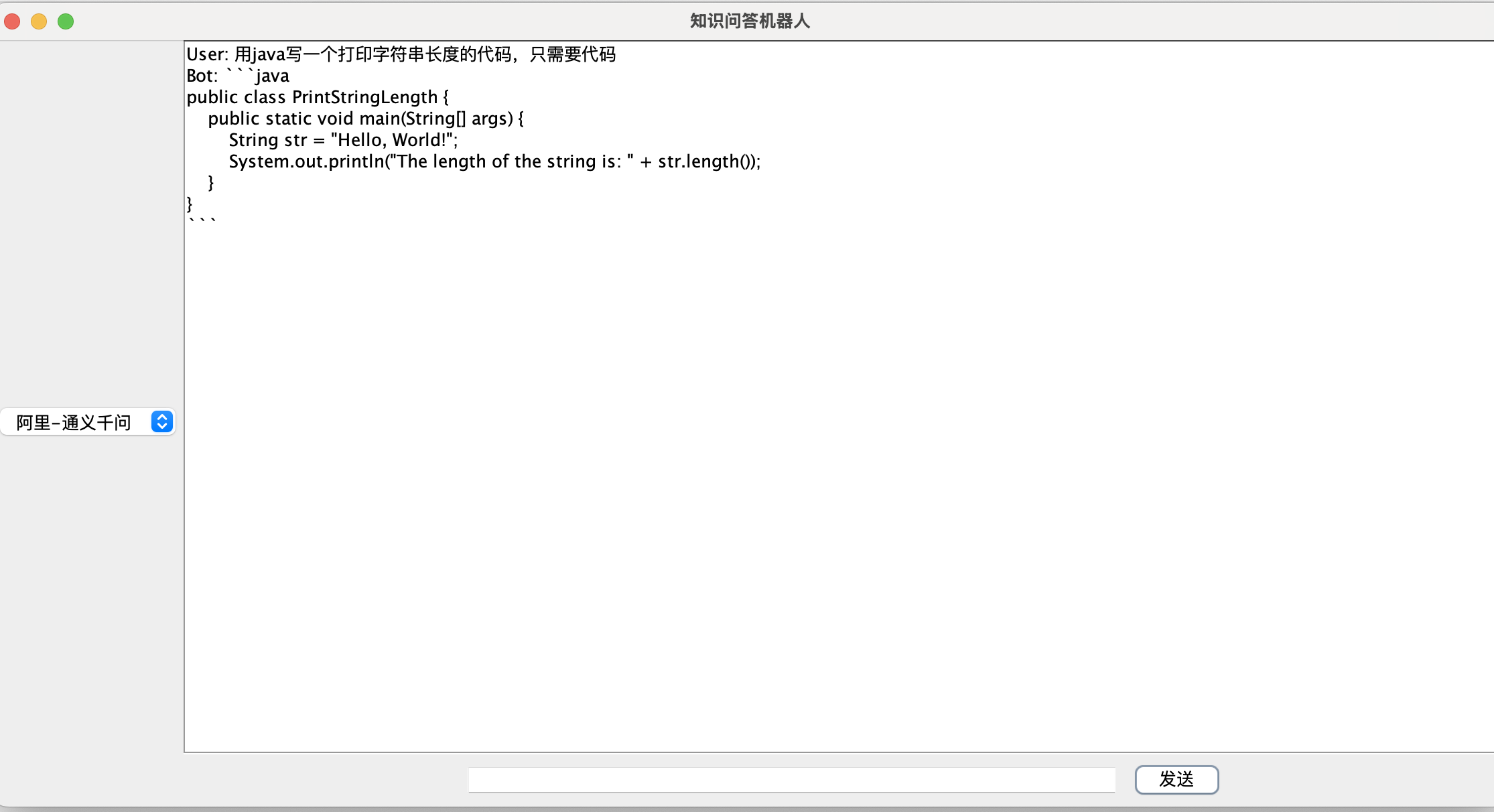


测试用例3：通过问答测试程序填空题如下。



测试用例4：测试编程题目。





8. 结论

本项目通过结合阿里云和百度提供的大模型服务，实现了一个人机交互的知识问答机器人。它不仅展示了如何与第三方 API 进行集成，同时也提供了一个良好的多线程编程示例，保证了应用的高效性和用户体验。此外，项目的模块化设计也便于未来的扩展和维护。