**RENEILWE NAKANA**

**ST10491568**

**PROG5121 POE PART 2**

package quickchatapp;

import javax.swing.\*;

import java.util.\*;

import java.io.FileWriter;

import java.io.IOException;

public class QuickChatApp {

// User Database

static String registeredUsername;

static String registeredPassword;

static String registeredCellPhone;

static String registeredFirstName;

static String registeredLastName;

// Messaging Static Data

private static int totalMessagesSent = 0;

private static int messageCounter = 0;

private static final List<String> sentMessages = new ArrayList<>();

private static final List<String> jsonMessages = new ArrayList<>();

// Registration Methods

public static boolean checkUsername(String username) {

return username.contains("\_") && username.length() <= 5;

}

public static boolean checkPasswordComplexity(String password) {

boolean hasUpper = false, hasDigit = false, hasSpecial = false;

if (password.length() < 8) return false;

for (char ch : password.toCharArray()) {

if (Character.isUpperCase(ch)) hasUpper = true;

else if (Character.isDigit(ch)) hasDigit = true;

else if (!Character.isLetterOrDigit(ch)) hasSpecial = true;

}

return hasUpper && hasDigit && hasSpecial;

}

public static boolean checkCellPhoneNumber(String cellphone) {

return cellphone.startsWith("+27") && cellphone.length() <= 13;

}

public static boolean loginUser(String username, String password) {

return username.equals(registeredUsername) && password.equals(registeredPassword);

}

public static String returnLoginStatus(boolean loginSuccess) {

if (loginSuccess) {

return "Welcome " + registeredFirstName + ", " + registeredLastName + " it is great to see you again.";

} else {

return "Username or password incorrect, please try again.";

}

}

// Message Class (Inner)

public static class Message {

private final String messageID;

private final String recipient;

private final String message;

private final String messageHash;

public Message(String recipient, String message) {

this.messageID = generateMessageID();

this.recipient = recipient;

this.message = message;

this.messageHash = createMessageHash();

}

private String generateMessageID() {

Random rand = new Random();

StringBuilder id = new StringBuilder();

for (int i = 0; i < 10; i++) {

id.append(rand.nextInt(10));

}

return id.toString();

}

private String createMessageHash() {

String[] words = message.trim().split("\\s+");

return messageID.substring(0, 2) + ":" + messageCounter + ":" + words[0].toUpperCase() + words[words.length - 1].toUpperCase();

}

public String sentMessage() {

String[] options = {"Send Message", "Disregard Message", "Store Message to send later"};

int choice = JOptionPane.showOptionDialog(null, "Choose an option for the message", "Message Options",

JOptionPane.DEFAULT\_OPTION, JOptionPane.INFORMATION\_MESSAGE, null, options, options[0]);

switch (choice) {

case 0 -> {

if (message.length() > 250) {

JOptionPane.showMessageDialog(null, "Please enter a message of less than 250 characters.");

return "Invalid";

} else {

JOptionPane.showMessageDialog(null, "Message sent");

totalMessagesSent++;

sentMessages.add(printMessages());

return "Sent";

}

}

case 1 -> {

return "Discarded";

}

case 2 -> {

storeMessage();

return "Stored";

}

default -> {

return "No Action";

}

}

}

public String printMessages() {

return "MessageID: " + messageID +

"\nMessage Hash: " + messageHash +

"\nRecipient: " + recipient +

"\nMessage: " + message;

}

public void storeMessage() {

String messageObj = "{\n" +

" \"MessageID\": \"" + messageID + "\",\n" +

" \"MessageHash\": \"" + messageHash + "\",\n" +

" \"Recipient\": \"" + recipient + "\",\n" +

" \"Message\": \"" + message + "\"\n" +

"}";

jsonMessages.add(messageObj);

try (FileWriter file = new FileWriter("messages.json")) {

file.write("[\n" + String.join(",\n", jsonMessages) + "\n]");

file.flush();

} catch (IOException e) {

}

}

}

//Main

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Registration

System.out.println("\*\*\*User Registration Page\*\*\*");

System.out.print("Enter First Name: ");

registeredFirstName = input.nextLine();

System.out.print("Enter Last Name: ");

registeredLastName = input.nextLine();

System.out.print("Create username: ");

registeredUsername = input.nextLine();

System.out.println(checkUsername(registeredUsername) ? "Username successfully captured." :

"Username is not correctly formatted, please ensure that your username contains an underscore and is no more than five characters in length.");

System.out.print("Create Password: ");

registeredPassword = input.nextLine();

System.out.println(checkPasswordComplexity(registeredPassword) ? "Password successfully captured." :

"Password is not correctly formatted; please ensure it contains at least eight characters, a capital letter, a number, and a special character.");

System.out.print("Enter Cellphone Number: ");

registeredCellPhone = input.nextLine();

System.out.println(checkCellPhoneNumber(registeredCellPhone) ? "Cell phone number successfully added." :

"Cell phone number incorrectly formatted or does not contain international code.");

// Login

System.out.println("\n\*\*\* Please log in to gain access \*\*\*");

System.out.print("Enter username: ");

String loginUser = input.nextLine();

System.out.print("Enter Password: ");

String loginPassword = input.nextLine();

boolean loggedIn = loginUser(loginUser, loginPassword);

System.out.println(returnLoginStatus(loggedIn));

// Messaging Loop

if (loggedIn) {

while (true) {

String menu = JOptionPane.showInputDialog("Choose an option:\n1. Send Message\n2. View Recent Messages\n3. Quit");

switch (menu) {

case "1" -> {

String num = JOptionPane.showInputDialog("How many messages do you want to send?");

int count = Integer.parseInt(num);

for (int i = 0; i < count; i++) {

messageCounter = i;

String recipient = JOptionPane.showInputDialog("Enter recipient number:");

String msg = JOptionPane.showInputDialog("Enter message:");

Message m = new Message(recipient, msg);

String result = m.sentMessage();

if (result.equals("Sent")) {

JOptionPane.showMessageDialog(null, m.printMessages());

}

}

JOptionPane.showMessageDialog(null, "Total messages sent: " + totalMessagesSent);

}

case "2" -> JOptionPane.showMessageDialog(null, "Feature coming soon.");

case "3" -> System.exit(0);

default -> JOptionPane.showMessageDialog(null, "Invalid choice");

}

}

}

}

}

//Reference: Java Programming. Tenth Edition. 2023. Farrell.J. AND ChatGTP (OpenAI's AI assistant)

[ST10491568-POE-PART-2/src/quickchatapp at master · Reneilwe192/ST10491568-POE-PART-2](https://github.com/Reneilwe192/ST10491568-POE-PART-2/tree/master/src/quickchatapp)

[Reneilwe192/ST10491568-POE-PART-2](https://github.com/Reneilwe192/ST10491568-POE-PART-2)



















