**COMSATS** University Islamabad, Lahore campus

-------------------------------------------------------------------------------------------------------

Submitted by Areeba Zia

Roll no.: FA23-BSE-039

Session: Fall 2023-2027

Section: B

Assignment no. 1

This code represents a simple messaging application implemented in Java. It consists of three classes: MessaginApp, Message, and MessageDriver. Here's a breakdown of each class and their roles:

### MessaginApp Class

### ****Attributes****:

* Message[] messages: An array to store messages, initialized to a size of 100.
* static int messageCounter: A static counter to keep track of the number of messages sent.

**Constructor**:

* Initializes the messages array and adds a default message.

**Methods**:

* sendMessage(String sender, String receiver, String content): Increments the messageCounter and creates a new Message object, storing it in the messages array.
* displaySenderMessage(String sender): Loops through the messages array and prints messages sent by a specified sender.

public class MessaginApp {

Message messages[];

static int messageCounter = 0;

public MessaginApp()

{

System.out.println("MessaginApp DEFAULT constructor called");

messages = new Message[100];

// for (int i = 0; i < messages.length; i++)

messages[0] = new Message("sender1", "receiver1", "message content", false);

}

public void sendMessage(String sender, String receiver, String content) {

System.out.println("send message called");

messages[++messageCounter] = new Message(sender, receiver, content, false);

}

public void displaySenderMessage(String sender) {

System.out.println("Message counter: " + messageCounter);

for (int i = 0; i <= messageCounter; i++)

if (messages[i].getSender().equals(sender)) {

System.out.println(messages[i].toString());

//System.out.println("Sender: " + messages[i].getSender());

//System.out.println("Receiver: " + messages[i].getReceiver());

//System.out.println("Content: " + messages[i].getContent());

//System.out.println("Status: " + messages[i].isStatus());

//messages[i].setRead();

}

}

}

### Message Class

**Attributes**:

* String sender: The sender of the message.
* String receiver: The receiver of the message.
* String content: The actual message content.
* String messageId: An identifier for the message (not set in the constructor).
* boolean status: Indicates if the message has been read or not.
* LocalDateTime timestamp: The time when the message was created.

**Constructor**:

* Initializes the attributes and prints a message when called. It sets the timestamp to the current time.

**Getters and Setters**:

* Methods to access and modify the attributes.

toString() **Method**:

* Overrides the default toString() to provide a string representation of a Message object.

package messageapp;

import java.time.LocalDateTime;

public class Message {

String sender;

String receiver;

String content;

String messageId;

boolean status;

LocalDateTime timestamp;

public Message(String sender, String receiver, String content, boolean status) {

System.out.println("Constructor called");

this.sender = sender;

this.receiver = receiver;

this.content = content;

this.status = status;

timestamp = LocalDateTime.now();

}

public String getSender() {

return sender;

}

public void setSender(String sender) {

this.sender = sender;

}

public String getReceiver() {

return receiver;

}

public void setReceiver(String receiver) {

this.receiver = receiver;

}

public String getContent() {

return content;

}

public void setContent(String content) {

this.content = content;

}

public String getMessageId() {

return messageId;

}

public void setMessageId(String messageId) {

this.messageId = messageId;

}

public boolean isStatus() {

return status;

}

public void setStatus(boolean status) {

this.status = status;

}

public LocalDateTime getTimestamp() {

return timestamp;

}

public void setTimestamp(LocalDateTime timestamp) {

this.timestamp = timestamp;

}

@Override

public String toString() {

System.out.println("Overide");

return "Message{" +

"sender='" + sender + '\'' +

", receiver='" + receiver + '\'' +

", content='" + content + '\'' +

", messageId='" + messageId + '\'' +

", status=" + status +

", timestamp=" + timestamp +

'}';

}

}

MessageDriver Class

**Main Method**:

* Creates an instance of MessaginApp.
* Sends a few messages using sendMessage.
* Calls displaySenderMessage for different senders to display messages.
* Demonstrates the use of an ArrayList to store and print a list of strings.

### Key Points

**Static Counter**:

* The messageCounter is static, meaning it is shared among all instances of MessaginApp.
* This is used to track the number of messages sent.

**Message Storage**:

* The array can hold a maximum of 100 messages, and there’s no check for overflow, which can lead to an ArrayIndexOutOfBoundsException if more than 100 messages are sent.

**Displaying Messages**:

* The application is designed to display messages from specific senders, which makes it easy to filter the output.

**Java Features**:

* The code uses basic Java features such as classes, objects, arrays, and collections (like ArrayList).

### Example Output

* When you run MessageDriver, you would see:
* Initialization messages from the constructors.
* Calls to sendMessage and displaySenderMessage methods.
* Printed representations of messages according to the sender specified.
* This code serves as a basic framework for a messaging application, illustrating object-oriented programming concepts in Java.

package messageapp;

import java.util.ArrayList;

import java.util.List;

public class MessageDriver {

public static void main(String[] args) {

// Message message = new Message("Hello, World!");

MessaginApp messageApp = new MessaginApp();

messageApp.sendMessage("Areeba", "Usama", "Its too late");

messageApp.sendMessage("Nabeeha", "Arham", "Its not too late");

messageApp.sendMessage("Uzma", "Nosheen", "Its not too lateeeeeeeee");

messageApp.displaySenderMessage("sender1");

messageApp.displaySenderMessage("Areeba");

messageApp.displaySenderMessage("Nabeeha");

messageApp.displaySenderMessage("Uzma");

List<String> list = new ArrayList<String>();

list.add("areeba");

list.add("nabeeha");

list.add("usama");

System.out.println(list.toString());

System.out.println("HERE");

}

}