

**Assignment No 4**

|  |  |
| --- | --- |
| **Course Title** | **Object Oriented Programming** |

**Submitted By**

|  |  |
| --- | --- |
| **Sami Ur Rehman** | **SP24-BSE-086** |
| **Muhammad Eman** | **SP24-BSE-071** |

**Submitted To**

|  |  |
| --- | --- |
| **Moderator** | **Muhammad Shahid Bhatti** |

**Date: 19/12/2024**

**Classes**

**1. MainWindow**

*package* com.example.chattingappelements;  
  
*import* java.io.\*;  
*import* java.net.\*;  
  
*import* javafx.application.Application;  
*import* javafx.application.Platform;  
*import* javafx.geometry.Insets;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Node;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.image.Image;  
*import* javafx.scene.input.KeyCode;  
*import* javafx.scene.layout.BorderPane;  
*import* javafx.scene.layout.HBox;  
*import* javafx.scene.layout.VBox;  
*import* javafx.stage.FileChooser;  
*import* javafx.stage.Stage;  
*//import org.python.antlr.ast.Str;  
  
import* java.util.ArrayList;  
  
*public class* MainWindow *extends* Application {  
  
 *public static* Scene *scene*;  
 *public static* Stage *stage*;  
 *public static* BorderPane *root* = *new* BorderPane();  
 *public static* VBox *contactsBox* = *new* VBox();  
 *static* ScrollPane *scrollPane* = *new* ScrollPane();  
*// public static List<Contact> contacts = retrieveContactsFromFile();  
 public static* ArrayList<Contact> *contacts* = *new* ArrayList<>();  
 HBox menu = *new* HBox();  
 *public static* ArrayList<ContactBox> *boxes* = *new* ArrayList<>();  
  
  
  
 *private* Boolean isVerified = *false*;  
  
  
  
  
 *//stuff that will be used for connecting with server  
 public static* String *userPhoneNumber*;  
 *private* String userName;  
 *public static* Client *client*;  
  
 @Override  
 *public void* start(Stage primaryStage) *throws* IOException, ClassNotFoundException {  
  
 verify(primaryStage);  
 *stage* = primaryStage;  
 TopBar topBar = *new* TopBar();  
 HBox menu = topBar.create();  
 *root*.setTop(menu);  
  
 *//Ai Integration* ClownAi ai = *new* ClownAi();  
 *contactsBox*.getChildren().add(ai.createClown(primaryStage));  
  
  
 *contactsBox*.setSpacing(5);  
 *contactsBox*.setPadding(*new* Insets(10, 10, 10, 10));  
  
 *scrollPane*.setContent(*contactsBox*);  
 *scrollPane*.setFitToWidth(*true*);  
  
  
  
 *root*.setCenter(*scrollPane*);  
 *scene* = *new* Scene(*root*, 400, 600);  
 *scene*.getStylesheets().add(getClass().getResource("styles/main.css").toExternalForm());  
  
 primaryStage.setScene(*scene*);  
 primaryStage.setTitle("Main");  
  
  
  
  
  
 *root*.requestFocus();  
 *client* = *new* Client();  
  
  
  
 *//funtion calls* contactsInitializtion();  
  
  
*//Action events here* topBar.getAddContact().setOnAction(actionEvent -> addContact());  
  
 *//adding event that when backspace is pressed the border pane's middle change contacts box  
 scene*.setOnKeyPressed(event -> {  
 *if* (event.getCode() == KeyCode.*BACK\_SPACE*){  
 *root*.setCenter(*scrollPane*);  
 *root*.setTop(menu);  
 *root*.setBottom(*null*);  
 }  
 });  
  
  
*//Styles here  
 contactsBox*.setStyle("-fx-background-color: rgb(30,31,34);");  
 *contactsBox*.setStyle("-fx-background-color: #005b52; -fx-border-color: #005b52; -fx-border-width: 2px; -fx-padding: 10;");  
 *scrollPane*.setStyle("-fx-background-color: #005b52; -fx-border-color: #005b52; -fx-border-width: 2px; -fx-padding: 10;");  
  
 }  
  
  
  
 *public void* addContact() {  
 Stage contactAddStage = *new* Stage();  
 contactAddStage.setTitle("Add Contact");  
  
 *// Create UI elements* Label nameLabel = *new* Label("Contact Name:");  
 TextField nameField = *new* TextField();  
  
 Label numberLabel = *new* Label("Contact Number:");  
 TextField numberField = *new* TextField();  
  
 Label pictureLabel = *new* Label("Profile Picture:");  
 Button choosePictureButton = *new* Button("Choose Picture");  
 Label picturePathLabel = *new* Label();  
  
 Button saveButton = *new* Button("Save");  
 Button cancelButton = *new* Button("Cancel");  
  
 *// Layout* VBox layout = *new* VBox(10);  
 layout.setPadding(*new* Insets(10));  
 layout.getChildren().addAll(nameLabel, nameField, numberLabel, numberField, pictureLabel, choosePictureButton, picturePathLabel, saveButton, cancelButton);  
  
 Scene scene = *new* Scene(layout, 300, 300);  
 contactAddStage.setScene(scene);  
  
 *// File chooser logic  
 final* FileChooser fileChooser = *new* FileChooser();  
 choosePictureButton.setOnAction(e -> {  
 File file = fileChooser.showOpenDialog(contactAddStage);  
 *if* (file != *null*) {  
 picturePathLabel.setText(file.getAbsolutePath());  
 }  
 });  
  
 *// Save button logic* saveButton.setOnAction(e -> {  
 String name = nameField.getText();  
 String number = numberField.getText();  
 String picturePath = picturePathLabel.getText();  
 ContactBox box = *new* ContactBox(*new* Contact(name,number));  
 box.setProfilePicture(*new* Image(picturePathLabel.getText()));  
 *boxes*.add(box);  
 *contactsBox*.getChildren().add(box.create(*stage*));  
 *if* (name.isEmpty() || number.isEmpty() || picturePath.isEmpty()) {  
 Alert alert = *new* Alert(Alert.AlertType.*WARNING*, "Please fill in all fields and select a picture.", ButtonType.*OK*);  
 alert.showAndWait();  
 } *else* {  
 *// Store the details* System.*out*.println("Name: " + name);  
 System.*out*.println("Number: " + number);  
 System.*out*.println("Picture Path: " + picturePath);  
  
  
  
 *// Close the stage* contactAddStage.close();  
 }  
 });  
  
 *// Cancel button logic* cancelButton.setOnAction(e -> contactAddStage.close());  
  
  
  
  
 *// Show the stage* contactAddStage.show();  
 }  
  
 *public void* contactsInitializtion(){  
 *contacts*.add(*new* Contact("Eman", "03193544291"));  
 *contacts*.get(0).setContactImage(*new* Image("file:eman.png"));  
 *contacts*.add(*new* Contact("John Doe", "123123"));  
 *contacts*.add(*new* Contact("gen Asim", "456456"));  
 *contacts*.add(*new* Contact("Gen Bajwa", "1213113"));  
 *contacts*.add(*new* Contact("Lumber 1", "23232333"));  
 *contacts*.add(*new* Contact("Lumber 2", "23232333"));  
 *contacts*.add(*new* Contact("Lumber 3", "23232333"));  
 *contacts*.add(*new* Contact("Lumber 4", "23232333"));  
 *contacts*.add(*new* Contact("Lumber 5", "23232333"));  
 *contacts*.add(*new* Contact("Lumber 6", "23232333"));  
  
 *contacts*.get(0).addMessage(*new* Message("hello","You"));  
 *contacts*.get(0).addMessage(*new* Message("wassup","Eman"));  
 *contacts*.get(0).addMessage(*new* Message("nigga","Eman"));  
 *contacts*.get(0).addMessage(*new* Message("how are you","Eman"));  
 *contacts*.get(0).addMessage(*new* Message("no","Eman"));  
 *contacts*.get(0).addMessage(*new* Message("yes","Eman"));  
  
 *contacts*.get(1).addMessage(*new* Message("Hello Qamar","You"));  
 *contacts*.get(1).addMessage(*new* Message("Bolo Bhai","Qamar"));  
 System.*out*.println(*contacts*.get(0).getChatHistory().size());  
 setContactsBox(*stage*);  
 }  
  
 *public void* setContactsBox(Stage primaryStage){  
 *for* (*int* i = 0; i < *contacts*.size(); i++) {  
 ContactBox box = *new* ContactBox(*contacts*.get(i));  
 *boxes*.add(box);  
 *contactsBox*.getChildren().add(box.create(primaryStage));  
 }  
 }  
  
  
  
 *//checks whether the user already exists or not , users Phone number will identify the  
 // user, user will give phone number to communicate , the return user will suggests whether  
 // the user exists or not  
 public void* verify(Stage stage) *throws* IOException {  
 File file = *new* File("exists.txt");  
 *if* (!file.exists()){  
 file.createNewFile();  
 System.*out*.println("new user detected");  
 register();  
 *return*;  
 }*else* {  
  
 *try*(BufferedReader br = *new* BufferedReader(*new* FileReader(file))) {  
 String line = br.readLine();  
 String data[] = line.split(",");  
 System.*out*.println(data[0] + " |||" + data[1]);  
 *userPhoneNumber* = data[1];  
 userName = data[0];  
  
 }*catch* (IOException e){  
 e.printStackTrace();  
 }  
  
 stage.show();  
 }  
  
 }  
  
  
 *// incase there is new user of application.  
 public* Boolean register() *throws* IOException {  
*// stage.hide();* Stage registerStage = *new* Stage();  
 VBox vbox = *new* VBox();  
 Scene registerScene = *new* Scene(vbox, 400, 400);  
 registerStage.setScene(registerScene);  
  
 vbox.setSpacing(5);  
 vbox.setPadding(*new* Insets(10, 10, 10, 10));  
 vbox.setAlignment(Pos.*CENTER*);  
  
  
 Label name = *new* Label("Name : ");  
 TextField nameField = *new* TextField();  
 name.setStyle("-fx-text-fill: white; -fx-font-weight: bold;");  
  
  
 HBox nameBox = *new* HBox();  
 nameBox.getChildren().addAll(name,nameField);  
 nameBox.setSpacing(5);  
 nameBox.setAlignment(Pos.*CENTER*);  
*// nameBox.setStyle("-fx-text-fill: #ffff");* Label phoneNumber = *new* Label("Phone :");  
 TextField phoneNumberField = *new* TextField();  
 phoneNumber.setStyle("-fx-text-fill: white; -fx-font-weight: bold;");  
  
  
 HBox phoneBox = *new* HBox();  
 phoneBox.getChildren().addAll(phoneNumber,phoneNumberField);  
 phoneBox.setSpacing(5);  
 phoneBox.setAlignment(Pos.*CENTER*);  
  
  
 Button cancelButton = *new* Button("Cancel");  
 cancelButton.setStyle("-fx-text-fill: red; -fx-font-weight: bold;");  
 Button registerButton = *new* Button("Register");  
 registerButton.setStyle("-fx-text-fill: blue");  
  
  
 HBox cancelregister = *new* HBox();  
 cancelregister.setSpacing(5);  
 cancelregister.setPadding(*new* Insets(10, 10, 10, 10));  
 cancelregister.setAlignment(Pos.*CENTER*);  
 cancelregister.getChildren().addAll(cancelButton,registerButton);  
  
  
 vbox.setStyle("-fx-background-color: black; -fx-font-family: Candara; -fx-text-fill: white;");  
 vbox.getChildren().addAll(nameBox,phoneBox,cancelregister);  
 registerStage.show();  
 *userPhoneNumber* = phoneNumberField.getText();  
  
  
  
 registerButton.setOnAction(e -> {  
  
 File file = *new* File("exists.txt");  
  
 *try*(BufferedWriter writer = *new* BufferedWriter(*new* FileWriter(file,*false*))){  
 writer.write(nameField.getText()+","+phoneNumberField.getText());  
 }*catch* (IOException p){  
 p.printStackTrace();  
 }  
  
 nameField.clear();  
 phoneNumberField.clear();  
  
 Alert alert = *new* Alert(Alert.AlertType.*CONFIRMATION*);  
 alert.setTitle("Confirmation");  
 alert.setHeaderText(*null*);  
 alert.setContentText("User Registered Successfully");  
 registerStage.close();  
 *stage*.show();  
 *return*;  
 });  
  
  
  
 cancelButton.setOnAction(e -> {  
 nameField.clear();  
 phoneNumberField.clear();  
 });  
  
  
 *return false*;  
  
 }  
  
 *public static void* main(String[] args) {  
 *launch*(args);  
 }  
  
}  
  
  
*class* Client *implements Runnable*{  
  
 *private* Socket socket; *// Changed to Socket for client-side connection  
 public static* PrintWriter *out*;  
 *public static* BufferedReader *in*;  
  
 *private* String serverAddress;  
 *private int* port = 12345;  
  
 *private* Boolean userNamePush = *false*;  
  
  
 *public* Client() {  
*// System.out.println("Client created");  
// System.out.println("Phone"+ MainWindow.userPhoneNumber);  
// this.port = port;* chatOnline();  
 }  
  
 *public void* sendMessage(String message) {  
 *if* (*out* != *null*) {  
 *out*.println(message);  
 }  
  
 }  
  
  
 *public void* chatOnline() {  
 serverAddress = "192.168.197.26";  
 *try* {  
 socket = *new* Socket(serverAddress, port); *// Establish connection to server  
 out* = *new* PrintWriter(socket.getOutputStream(), *true*); *// Initialize PrintWriter here  
 out*.println(MainWindow.*userPhoneNumber*);  
  
 *new* Thread(*this*).start(); *// Start the client listener thread  
// System.out.println("Successfully connected to Server" );* } *catch* (IOException e) {  
*// JOptionPane.showMessageDialog(this, "Could not connect to " + contact.getName() + " at " + serverAddress + ":" + port,  
// "Connection Error", JOptionPane.ERROR\_MESSAGE);* e.printStackTrace();  
 }  
 }  
  
 @Override  
 *public void* run() {  
 *try* {  
 *in* = *new* BufferedReader(*new* InputStreamReader(socket.getInputStream()));  
 *out* = *new* PrintWriter(socket.getOutputStream(), *true*);  
*//  
// if ((in.toString().contains(":"))) {  
// iLoveFilter(in.toString());  
// System.out.println("gya");  
// }else {  
// System.out.println("mhi gya");  
// }  
//* String inputLine;  
*// if (userNamePush){  
// inputLine = MainWindow.userPhoneNumber;  
// userNamePush = false;  
// }  
  
 while* ((inputLine = *in*.readLine()) != *null*) {  
 System.*out*.println(inputLine);  
*// test();  
 if* (inputLine.contains(":")){  
 iLoveFilter(inputLine);  
*// System.out.println("gya");* }  
  
 }  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
  
  
 *public void* test(){  
 *for* (ContactBox b: MainWindow.*boxes*){  
 System.*out*.println(b.getContactNumber());  
 }  
 }  
  
  
 *public void* iLoveFilter(String content){  
  
  
 *if* (!content.isEmpty()) {  
  
 System.*out*.println(content);  
 *int* indexOfColon = content.indexOf(':');  
 *if* (indexOfColon == -1) {  
 *throw new* IllegalArgumentException("Invalid input format. Expected ':' delimiter.");  
 }  
  
 String numberPart = content.substring(0, indexOfColon).trim();  
 String messagePart = content.substring(indexOfColon + 1).trim();  
  
 System.*out*.println(numberPart);  
 System.*out*.println(messagePart);  
  
 ContactBox box = findContactHbox(numberPart.trim(),messagePart);  
  
  
 }  
  
  
 }  
  
  
  
  
  
  
  
  
 *public* ContactBox findContactHbox(String contactNumber, String message) {  
 *for* (ContactBox box : MainWindow.*boxes*) {  
 *if* (box.getContactNumber().equals(contactNumber)) {  
 System.*out*.println(box.getContactNumber());  
*// System.out.println("true hogya");  
  
 // Use Platform.runLater to safely update UI components or observable lists* Platform.*runLater*(() -> {  
 box.getChatHistory().add(*new* Message(message, contactNameFinder(contactNumber)));  
 });  
  
 *return* box;  
 }  
 }  
  
 *return null*;  
 }  
  
  
  
  
 *public* String contactNameFinder(String number){  
 String name;  
 *for* (Contact c: MainWindow.*contacts*){  
 *if* (c.getPhoneNumber().equals(number)){  
 name = c.getName();  
 *return* name;  
 }  
 }  
 *return* "Error";  
 }  
  
}

**2. ContactBox**

*package* com.example.chattingappelements;  
  
*import* javafx.collections.*ObservableList*;  
*import* javafx.geometry.Insets;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Node;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.image.Image;  
*import* javafx.scene.image.ImageView;  
*import* javafx.scene.input.KeyCode;  
*import* javafx.scene.layout.\*;  
*import* javafx.scene.shape.Circle;  
*import* javafx.stage.FileChooser;  
*import* javafx.stage.Popup;  
*import* javafx.stage.Stage;  
  
*import* java.io.File;  
*import* java.io.PrintWriter;  
*import* java.net.Socket;  
  
*public class* ContactBox {  
  
 *private* String contactName;  
 *private* String contactNumber;  
 *private* String messageContent;  
  
 *private* Image profilePicture;  
 *private final* Image defaultProfilePicture = *new* Image("file:defaultProfile.png");  
  
 *private* ChatView chatView = *new* ChatView();  
 *private* ScrollPane messageListView = *new* ScrollPane();  
  
 *private* Boolean onlineStatus = *false*;  
  
  
 *private ObservableList*<Message> chatHistory;  
  
 *public* ContactBox(String contactName, String contactNumber, String messageContent, Image profilePicture) {  
 *this*.contactName = contactName;  
 *this*.contactNumber = contactNumber;  
 *this*.messageContent = messageContent;  
 *this*.profilePicture = profilePicture;  
 }  
  
 *public* ContactBox(Contact contact) {  
 *this*.contactName = contact.getName();  
 *this*.contactNumber = contact.getPhoneNumber();  
 *if* (contact.getContactImage() != *null*) {  
 *this*.profilePicture = contact.getContactImage();  
 }*else  
 this*.profilePicture = defaultProfilePicture;  
*// if (contact.getChatHistory().getLast().getContent() != null) {  
// this.messageContent = contact.getChatHistory().getLast().getContent();  
// }else  
// this.messageContent = "";  
 this*.chatHistory = contact.getChatHistory();  
  
  
 messageListView = chatView.create(chatHistory);  
  
  
 }  
 *public* Node create(Stage stage) {  
 HBox hbox = *new* HBox();  
 hbox.setStyle("-fx-background-color: grey; -fx-padding: 10; -fx-background-radius: 15");  
 hbox.setAlignment(Pos.***CENTER\_LEFT***);  
  
 *// Create a circular profile picture view* ImageView contactImage = *new* ImageView(profilePicture);  
 contactImage.setFitWidth(30);  
 contactImage.setFitHeight(30);  
 Circle circleClip = *new* Circle(15, 15, 15);  
 contactImage.setClip(circleClip);  
  
 Button profileButton = *new* Button();  
 profileButton.setGraphic(contactImage);  
 profileButton.setBackground(Background.***EMPTY***);  
  
 *// Handle hover effect* profileHoverEffect(profileButton);  
  
 VBox labelsBox = *new* VBox(5);  
 Label nameLabel = *new* Label(contactName);  
 nameLabel.setStyle("-fx-text-fill: white; -fx-font-weight: bold; -fx-font-size: 14px;");  
 Label messageLabel = *new* Label(" ");  
  
 *if* (chatHistory.size() > 0) {  
 messageLabel.setText(chatHistory.getLast().onlyMessageContent());  
 }  
  
 labelsBox.getChildren().addAll(nameLabel, messageLabel);  
  
 *// Create the timeDateOfLastMessage VBox* VBox timeDateOfLastMessage = *new* VBox(5);  
 Label dateLabel = *new* Label();  
 Label timeLabel = *new* Label();  
 *if* (chatHistory.size() > 0) {  
 dateLabel.setText(chatHistory.getLast().getMessageDate());  
 timeLabel.setText(chatHistory.getLast().getMessageTime());  
 }  
 timeDateOfLastMessage.getChildren().addAll(dateLabel,timeLabel);  
 timeDateOfLastMessage.setStyle("-fx-text-fill: white; -fx-font-size: 12px;");  
  
  
  
 *// Add a listener to chatHistory to dynamically update UI on change* chatHistory.addListener((javafx.collections.*ListChangeListener*<Message>) c -> {  
 *if* (!chatHistory.isEmpty()) {  
 Message latestMessage = chatHistory.getLast();  
 dateLabel.setText(latestMessage.getMessageDate());  
 timeLabel.setText(latestMessage.getMessageTime());  
 messageLabel.setText(latestMessage.onlyMessageContent());  
 } *else* {  
 dateLabel.setText("");  
 timeLabel.setText("");  
 messageLabel.setText("");  
 }  
 });  
  
  
 *// Spacer to push the timeDateOfLastMessage VBox to the right* Region spacer = *new* Region();  
 HBox.*setHgrow*(spacer, Priority.***ALWAYS***);  
  
 *// Add elements to the hbox* hbox.getChildren().addAll(profileButton, labelsBox, spacer, timeDateOfLastMessage);  
  
 profileButton.setOnAction(e -> showImagePopup(profilePicture, stage));  
 hbox.setOnMouseClicked(e -> openChat(stage, chatHistory));  
 highlightBox(hbox);  
  
 *return* hbox;  
 }  
  
  
 *private void* profileHoverEffect(Button button) {  
 button.setOnMouseEntered(event -> {  
 button.setScaleX(1.1);  
 button.setScaleY(1.1);  
 });  
 button.setOnMouseExited(event -> {  
 button.setScaleX(1.0);  
 button.setScaleY(1.0);  
 });  
 }  
  
  
 *public void* openChat(Stage stage, *ObservableList*<Message> chatHistory) {  
 MainWindow.*root*.setCenter(messageListView);  
 MainWindow.*root*.setTop(contactInfoTile());  
 MainWindow.*root*.setBottom(messageControl());  
 }  
  
 *public* HBox messageControl() {  
 HBox hbox = *new* HBox();  
 Button sendButton = *new* Button("Send");  
 Button sendFileButton = *new* Button("Send File");  
 TextField textInput = *new* TextField();  
  
 *// String to store the selected file path  
 final* String[] path = {""};  
  
  
 hbox.getChildren().addAll(textInput, sendButton, sendFileButton);  
 hbox.setSpacing(10);  
 hbox.setAlignment(Pos.***BASELINE\_CENTER***);  
 hbox.setPadding(*new* Insets(10, 0, 10, 0));  
  
 *// Handle sending messages* sendButton.setOnAction(e -> {  
 String message = textInput.getText();  
 *if* (!message.isEmpty()) {  
 chatHistory.add(*new* Message(message, "You"));  
 MainWindow.*client*.sendMessage(contactNumber+":"+message);  
 textInput.clear();  
 }  
 });  
  
 *// Handle sending a file* sendFileButton.setOnAction(e -> {  
 FileChooser fileChooser = *new* FileChooser();  
 File file = fileChooser.showOpenDialog(MainWindow.*stage*);  
  
 *if* (file != *null*) {  
 path[0] = file.getAbsolutePath();  
 System.***out***.println("File path selected: " + path[0]);  
 *// You can now send this file path or perform further logic as needed.* }  
 });  
  
 *// Handle key events for quick sending with the Enter key* textInput.setOnKeyPressed(keyEvent -> {  
 *if* (keyEvent.getCode() == KeyCode.***ENTER***) {  
 sendButton.fire();  
 }  
 });  
  
 *//hbox.setStyle("-fx-background-color: #02b07c; -fx-border-color: #128C7E; -fx-border-width: 2px; -fx-padding: 10;");*hbox.setStyle("-fx-background-color: #02b07c; -fx-border-color: #005b52; -fx-border-width: 2px; -fx-padding: 10;");  
  
 MainWindow.*root*.requestFocus();  
 *return* hbox;  
 }  
  
*//  
// public static void updateRecord(){  
//  
// }  
  
  
 private void* showImagePopup(Image image, Stage primaryStage) {  
 Popup popup = *new* Popup();  
  
 *// Setup full-screen image view in the popup* ImageView fullScreenImage = *new* ImageView(image);  
 fullScreenImage.setPreserveRatio(*true*);  
 fullScreenImage.setFitWidth(400);  
 fullScreenImage.setFitHeight(400);  
  
 Button closeButton = *new* Button("Close");  
 closeButton.setOnAction(e -> popup.hide());  
  
 Button chatButton = *new* Button("Chat");  
 chatButton.setOnAction(e -> {  
 popup.hide();  
 openChat(primaryStage,chatHistory);  
 });  
  
 Button conntactInfo = *new* Button("Contact Info");  
  
  
  
 HBox popupButtons = *new* HBox();  
 popupButtons.setAlignment(Pos.***CENTER***);  
  
 popupButtons.setSpacing(10);  
 popupButtons.getChildren().addAll( chatButton, conntactInfo,closeButton);  
  
 VBox popupLayout = *new* VBox(10);  
 popupLayout.getChildren().addAll(fullScreenImage, popupButtons);  
 popupLayout.setStyle("-fx-background-color: #005b52; -fx-border-color: #005b52; -fx-border-width: 2px; -fx-padding: 10; -fx-border-radius: 10");  
 closeButton.setStyle("-fx-text-fill: white; -fx-font-weight: bold; -fx-background-color: #02b07c;");  
 chatButton.setStyle("-fx-text-fill: white;-fx-font-weight: bold; -fx-background-color: #02b07c;");  
 conntactInfo.setStyle("-fx-text-fill: white; -fx-font-weight: bold;-fx-background-color: #02b07c;");  
 popupButtons.setStyle("-fx-background-color: #005b52; -fx-border-color: #005b52; -fx-border-width: 2px; -fx-padding: 10;");  
  
 popup.getContent().add(popupLayout);  
 popup.show(primaryStage);  
  
 }  
  
  
 *public* String getContactName() {  
 *return* contactName;  
 }  
  
 *public void* setContactName(String contactName) {  
 *this*.contactName = contactName;  
 }  
  
 *public* String getContactNumber() {  
 *return* contactNumber;  
 }  
  
 *public void* setContactNumber(String contactNumber) {  
 *this*.contactNumber = contactNumber;  
 }  
  
 *public* String getMessageContent() {  
 *return* messageContent;  
 }  
  
 *public void* setMessageContent(String messageContent) {  
 *this*.messageContent = messageContent;  
 }  
  
 *public* Image getProfilePicture() {  
 *return* profilePicture;  
 }  
  
 *public void* setProfilePicture(Image profilePicture) {  
 *this*.profilePicture = profilePicture;  
 }  
  
 *public* Image getDefaultProfilePicture() {  
 *return* defaultProfilePicture;  
 }  
  
 *public ObservableList*<Message> getChatHistory() {  
 *return* chatHistory;  
 }  
  
 *public void* setChatHistory(*ObservableList*<Message> chatHistory) {  
 *this*.chatHistory = chatHistory;  
 }  
  
 *public void* highlightBox(HBox hbox) {  
 *// Add a mouse enter effect* hbox.setOnMouseEntered(event -> {  
 hbox.setStyle("-fx-background-color: grey; -fx-padding: 10; -fx-background-radius: 15; -fx-effect: dropshadow(gaussian, #00c276, 10, 0.5, 0, 0);");  
*// System.out.println("entered");* });  
  
 *// Add a mouse exit effect* hbox.setOnMouseExited(event -> {  
 hbox.setStyle("-fx-background-color: grey; -fx-padding: 10; -fx-background-radius: 15; -fx-effect: None;");  
  
 });  
 }  
  
  
  
  
 *public* HBox contactInfoTile(){  
 HBox hbox = *new* HBox();  
 ImageView contactImage = *new* ImageView(profilePicture);  
 contactImage.setFitWidth(40);  
 contactImage.setFitHeight(40);  
  
 Circle clip = *new* Circle(20, 20, 20);  
 contactImage.setClip(clip);  
  
  
 VBox box = *new* VBox();  
  
 Label contactName = *new* Label(getContactName());  
 contactName.setStyle("-fx-font-style: bold; -fx-font-weight: bold;");  
 Label status = *new* Label("");  
 *if* (onlineStatus){  
 status.setText("Online");  
 }*else* status.setText("Online");  
  
  
 box.getChildren().addAll(contactName, status);  
  
 hbox.setSpacing(20);  
 hbox.setPadding(*new* Insets(10,10,10,10));  
 hbox.getChildren().addAll(contactImage,box);  
  
 hbox.setStyle("-fx-background-color: #02b07c; -fx-border-color: #02b07c; -fx-border-width: 2px; -fx-padding: 10;");  
  
  
 *return* hbox;  
 }  
}

**3. TopBar**

*package* com.example.chattingappelements;  
  
*import* javafx.geometry.Insets;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Node;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.layout.HBox;  
*import* javafx.scene.layout.Region;  
  
*public class* TopBar {  
 *private* Button searchButton;  
 *private* Button tilesButton;  
  
 *public* TopBar() {  
 }  
  
*// public HBox create() {  
// // Create the HBox (top bar)  
// HBox topBar = new HBox();  
//  
// // Initialize buttons  
// searchButton = new Button("Search");  
// tilesButton = new Button("Tile");  
//  
// // Create a flexible spacer that will expand to push the tile button to the right corner  
// Region spacer = new Region();  
// HBox.setHgrow(spacer, javafx.scene.layout.Priority.ALWAYS);  
//  
//  
// /\*  
// \* The Region spacer = new Region(); acts as a flexible, resizable space. HBox.setHgrow  
// \* (spacer, javafx.scene.layout.Priority.ALWAYS); ensures that this region takes up all available  
// \*  
// \* \*/  
//  
// // Add elements to the HBox: searchButton, spacer, tilesButton  
// topBar.getChildren().addAll(searchButton, spacer, tilesButton);  
//  
// // Set HBox properties  
// topBar.setPrefWidth(MainWindow.stage.getWidth());  
// topBar.setAlignment(Pos.CENTER\_LEFT);  
// topBar.setPadding(new Insets(5, 10, 5, 10)); //top //right //bottom //left  
//  
// return topBar;  
// }  
  
  
 // Menu Bar* MenuBar menuBar = *new* MenuBar();  
 Menu optionsMenu = *new* Menu("Contacts");  
 MenuItem addContact = *new* MenuItem("Add Contact");  
 MenuItem deleteContact = *new* MenuItem("Delete Contact");  
 MenuItem modifyContact = *new* MenuItem("Modify Contact");  
 MenuItem userManual = *new* MenuItem("User Manual");  
  
  
 TextField search = *new* TextField();  
  
  
  
 *public* HBox create(){  
  
 search.setPromptText("Search");  
  
  
 optionsMenu.getItems().addAll(addContact, deleteContact, modifyContact, userManual);  
 menuBar.getMenus().addAll(optionsMenu);  
  
  
 HBox box = *new* HBox();  
 box.setAlignment(Pos.***CENTER***);  
 box.setSpacing(10);  
 box.setPadding(*new* Insets(10, 10, 10, 10));  
 box.getChildren().addAll(menuBar,search);  
  
box.setStyle("-fx-background-color: #005b52; -fx-border-color: #004112; -fx-border-width: 2px; -fx-padding: 10;");  
  
  
  
 *return* box;  
 }  
  
  
 *public* Button getSearchButton() {  
 *return* searchButton;  
 }  
  
 *public void* setSearchButton(Button searchButton) {  
 *this*.searchButton = searchButton;  
 }  
  
 *public* Button getTilesButton() {  
 *return* tilesButton;  
 }  
  
 *public void* setTilesButton(Button tilesButton) {  
 *this*.tilesButton = tilesButton;  
 }  
  
 *public* MenuBar getMenuBar() {  
 *return* menuBar;  
 }  
  
 *public void* setMenuBar(MenuBar menuBar) {  
 *this*.menuBar = menuBar;  
 }  
  
 *public* Menu getOptionsMenu() {  
 *return* optionsMenu;  
 }  
  
 *public void* setOptionsMenu(Menu optionsMenu) {  
 *this*.optionsMenu = optionsMenu;  
 }  
  
 *public* MenuItem getAddContact() {  
 *return* addContact;  
 }  
  
 *public void* setAddContact(MenuItem addContact) {  
 *this*.addContact = addContact;  
 }  
  
 *public* MenuItem getDeleteContact() {  
 *return* deleteContact;  
 }  
  
 *public void* setDeleteContact(MenuItem deleteContact) {  
 *this*.deleteContact = deleteContact;  
 }  
  
 *public* MenuItem getModifyContact() {  
 *return* modifyContact;  
 }  
  
 *public void* setModifyContact(MenuItem modifyContact) {  
 *this*.modifyContact = modifyContact;  
 }  
  
 *public* MenuItem getUserManual() {  
 *return* userManual;  
 }  
  
 *public void* setUserManual(MenuItem userManual) {  
 *this*.userManual = userManual;  
 }  
  
 *public* TextField getSearch() {  
 *return* search;  
 }  
  
 *public void* setSearch(TextField search) {  
 *this*.search = search;  
 }  
}

**4. ChatView**

*package* com.example.chattingappelements;  
  
*import* javafx.collections.FXCollections;  
*import* javafx.collections.*ListChangeListener*;  
*import* javafx.collections.*ObservableList*;  
*import* javafx.geometry.Insets;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.image.Image;  
*import* javafx.scene.image.ImageView;  
*import* javafx.scene.input.KeyCode;  
*import* javafx.scene.layout.Background;  
*import* javafx.scene.layout.BackgroundFill;  
*import* javafx.scene.layout.HBox;  
*import* javafx.scene.layout.VBox;  
*import* javafx.scene.paint.Color;  
  
*public class* ChatView {  
  
 *private ObservableList*<Message> chat = FXCollections.*observableArrayList*();  
  
  
  
  
  
 *public* ScrollPane create(*ObservableList*<Message> messages) {  
 VBox vbox = *new* VBox(10); *// VBox with spacing between messages* vbox.setPadding(*new* Insets(10));  
  
 ScrollPane scrollPane = *new* ScrollPane(vbox);  
 scrollPane.setFitToWidth(*true*); *// Make the ScrollPane fit to the VBox's width* scrollPane.setHbarPolicy(ScrollPane.ScrollBarPolicy.***NEVER***); *// Hide horizontal scrollbar* scrollPane.setVbarPolicy(ScrollPane.ScrollBarPolicy.***AS\_NEEDED***); *// Show vertical scrollbar as needed  
  
 // Add existing messages to the VBox  
 for* (Message m : messages) {  
 vbox.getChildren().add(createMessageLabel(m));  
 }  
  
 *// Add listener to the ObservableList* messages.addListener((*ListChangeListener*<Message>) change -> {  
 *while* (change.next()) {  
 *if* (change.wasAdded()) {  
 *for* (Message m : change.getAddedSubList()) {  
 vbox.getChildren().add(createMessageLabel(m));  
 }  
 *// Scroll to the bottom to show the latest message* vbox.layout(); *// Ensure the layout is updated before scrolling* scrollPane.setVvalue(1.0); *// Scroll to the bottom* }  
 }  
 });  
  
*// vbox.setStyle("-fx-background-color: #02b07c");  
 return* scrollPane;  
 }  
  
 *private* HBox createMessageLabel(Message message) {  
 Label label = *new* Label(message.getContent());  
 label.setStyle("-fx-background-color: lightgray; -fx-padding: 10; -fx-border-radius: 5; -fx-background-radius: 5;");  
  
 HBox messageContainer = *new* HBox();  
 messageContainer.setPadding(*new* Insets(5));  
  
 *if* (!message.getSender().equals("You")) {  
 messageContainer.setAlignment(Pos.***CENTER\_LEFT***); *// Align to left* } *else* {  
 messageContainer.setAlignment(Pos.***CENTER\_RIGHT***); *// Align to right* label.setStyle("-fx-background-color: lightblue; -fx-padding: 10; -fx-border-radius: 5; -fx-background-radius: 5;");  
 }  
  
 messageContainer.getChildren().add(label);  
 *return* messageContainer;  
 }  
*// public ListView<Message> create(ObservableList<Message> messages) {  
// this.chat = messages;  
//  
//  
//  
  
//  
//  
// ListView<Message> view = new ListView<>();  
// view.setItems(messages);  
//  
// view.setStyle("-fx-font-family: Candara; -fx-text-fill: white; -fx-background-color: #005b52");  
//  
//// Background background = new Background(new BackgroundFill(Color.RED, null, null));  
//// view.setBackground(background);  
//  
// // Attach listener to scroll when the observable list changes  
// messages.addListener((javafx.collections.ListChangeListener<Message>) c -> {  
// if (!messages.isEmpty()) {  
// view.scrollTo(messages.size() - 1);  
// }  
// });  
//  
//// vbox.getChildren().addAll(view, messageControl(view, messages));  
// return view;  
// }  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
//  
// public HBox messageControl(ListView<Message> view, ObservableList<Message> messages) {  
// HBox hbox = new HBox();  
// Button sendButton = new Button("Send");  
// TextField textInput = new TextField();  
// hbox.getChildren().addAll(textInput, sendButton);  
// hbox.setSpacing(10);  
// hbox.setAlignment(Pos.BASELINE\_CENTER);  
//  
// sendButton.setOnAction(e -> {  
// String message = textInput.getText();  
// if (!message.isEmpty()) {  
// messages.add(new Message(message, "Nigga"));  
// textInput.clear();  
// }  
// });  
//  
// textInput.setOnKeyPressed(keyEvent -> {  
// if (keyEvent.getCode() == KeyCode.ENTER) {  
// sendButton.fire();  
// }  
// });  
//  
// return hbox;  
// }*}

**5. ClownAI**

*package* com.example.chattingappelements;  
  
*import* javafx.geometry.Insets;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.control.Button;  
*import* javafx.scene.control.Label;  
*import* javafx.scene.control.TextArea;  
*import* javafx.scene.control.TextField;  
*import* javafx.scene.image.Image;  
*import* javafx.scene.image.ImageView;  
*import* javafx.scene.input.KeyCode;  
*import* javafx.scene.layout.Background;  
*import* javafx.scene.layout.HBox;  
*import* javafx.scene.layout.VBox;  
*import* javafx.scene.shape.Circle;  
*import* javafx.stage.Popup;  
*import* javafx.stage.Stage;  
  
*import* java.io.BufferedReader;  
*import* java.io.InputStreamReader;  
  
*public class* ClownAi {  
*//  
// public static void main(String[] args) {  
// try {  
// // Python script and the prompt  
// String pythonScript = "generate\_response.py";  
// String prompt = "Explain how computer works";  
//  
// // Build the process to execute the Python script  
// ProcessBuilder pb = new ProcessBuilder("python3", pythonScript, prompt);  
// Process process = pb.start();  
//  
// // Capture the script output  
// BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));  
// StringBuilder output = new StringBuilder();  
// String line;  
// while ((line = reader.readLine()) != null) {  
// output.append(line);  
// }  
//  
// // Wait for the process to complete  
// process.waitFor();  
//  
// // Store the output in a Java string  
// String response = output.toString();  
// System.out.println("Python function returned: " + response);  
// } catch (Exception e) {  
// e.printStackTrace();  
// }  
// }  
  
  
 public* HBox createClown(Stage clownStage) {  
 HBox hbox = *new* HBox();  
 hbox.setStyle("-fx-background-color: grey; -fx-padding: 10; -fx-background-radius: 15");  
 hbox.setAlignment(Pos.***CENTER\_LEFT***);  
  
  
 Image img = *new* Image("file:clownAi.png");  
 *// Create a circular profile picture view* ImageView contactImage = *new* ImageView(img);  
 contactImage.setFitWidth(50);  
 contactImage.setFitHeight(50);  
 Circle circleClip = *new* Circle(25, 25, 25);  
 contactImage.setClip(circleClip);  
*// contactImage.setPreserveRatio(true);* Button profileButton = *new* Button();  
 profileButton.setGraphic(contactImage);  
 profileButton.setBackground(Background.***EMPTY***);  
  
 *// Handle hover effect* profileHoverEffect(profileButton);  
  
 *// Label elements* VBox labelsBox = *new* VBox(5);  
 Label nameLabel = *new* Label("Clown AI");  
 nameLabel.setStyle("-fx-text-fill: white; -fx-font-weight: bold; -fx-font-size: 14px;");  
 Label messageLabel = *new* Label("I am clown");  
 messageLabel.setStyle("-fx-text-fill: white; -fx-font-weight: normal; -fx-font-size: 12px;");  
  
 labelsBox.getChildren().addAll(nameLabel, messageLabel);  
  
 *// Add everything to the hbox* hbox.getChildren().addAll(profileButton, labelsBox);  
  
  
*// hbox.setOnKeyPressed(event->{  
// if (event.getCode() == KeyCode.BACK\_SPACE){  
// MainWindow.root.setCenter(MainWindow.scrollPane);  
// }  
// });* profileButton.setOnAction(e -> showImagePopup(img, clownStage));  
 hbox.setOnMouseClicked(event->{  
 chatAI();  
 });  
 *return* hbox;  
 }  
  
  
 *public void* chatAI(){  
 Button sendButton = *new* Button("Send");  
 TextField prompt = *new* TextField();  
  
 HBox control = *new* HBox();  
 control.setSpacing(10);  
 control.setPadding(*new* Insets(10, 10, 10, 10));  
 control.setAlignment(Pos.***CENTER***);  
 control.getChildren().addAll(prompt,sendButton);  
  
  
 TextArea chat = *new* TextArea();  
 chat.setEditable(*false*);  
 chat.setWrapText(*true*);  
 chat.appendText("Ai : Hi there!!! How may I assist You?\n");  
  
 chat.setOnKeyPressed(event->{  
 *if* (event.getCode() == KeyCode.***BACK\_SPACE***){  
 MainWindow.*root*.setCenter(MainWindow.*scrollPane*);  
 }  
 });  
  
 MainWindow.*root*.setCenter(chat);  
 MainWindow.*root*.setBottom(control);  
  
 sendButton.setOnAction(e->{  
 chat.appendText("You : "+prompt.getText()+ "\n");  
 System.***out***.println("prompt: "+prompt.getText());  
 chat.appendText(req( "Ai : " +prompt.getText())+"\n");  
 prompt.clear();  
  
  
 });  
  
  
  
 MainWindow.*root*.setOnKeyPressed(event->{  
 *if* (event.getCode() == KeyCode.***ENTER***){  
 sendButton.fire();  
 }  
 });  
 }  
  
  
  
 *public* String req(String prompt){  
  
 System.***out***.println(" req prompt: "+prompt);  
 *try* {  
 System.***out***.println("in try");  
 *// Python script and the prompt* String pythonScript = "generate\_response.py";  
  
 *// Build the process to execute the Python script* ProcessBuilder pb = *new* ProcessBuilder("python3", pythonScript, prompt);  
 Process process = pb.start();  
  
 *// Capture the script output* BufferedReader reader = *new* BufferedReader(*new* InputStreamReader(process.getInputStream()));  
 StringBuilder output = *new* StringBuilder();  
 String line;  
 *while* ((line = reader.readLine()) != *null*) {  
 output.append(line);  
 }  
 System.***out***.println("wait start");  
  
 *// Wait for the process to complete* process.waitFor();  
  
 System.***out***.println("wait over");  
  
 *// Store the output in a Java string* String response = output.toString();  
 System.***out***.println("Python function returned: " + response);  
 *return* response;  
*// System.out.println("Python function returned: " + response);* } *catch* (Exception e) {  
 e.printStackTrace();  
 }  
  
 *return* "Ai : Something went wrong Indeed i am a clown" ;  
 }  
  
  
 *private void* profileHoverEffect(Button button) {  
 button.setOnMouseEntered(event -> {  
 button.setScaleX(1.1);  
 button.setScaleY(1.1);  
 });  
 button.setOnMouseExited(event -> {  
 button.setScaleX(1.0);  
 button.setScaleY(1.0);  
 });  
 }  
  
  
  
 *private void* showImagePopup(Image image, Stage primaryStage) {  
 Popup popup = *new* Popup();  
  
 *// Setup full-screen image view in the popup* ImageView fullScreenImage = *new* ImageView(image);  
 fullScreenImage.setPreserveRatio(*true*);  
 fullScreenImage.setFitWidth(400);  
 fullScreenImage.setFitHeight(400);  
  
 Button closeButton = *new* Button("close");  
 closeButton.setOnAction(e -> popup.hide());  
  
  
 HBox popupButtons = *new* HBox();  
 popupButtons.setAlignment(Pos.***CENTER***);  
  
 popupButtons.getChildren().add(closeButton);  
  
 VBox popupLayout = *new* VBox(10);  
 popupLayout.getChildren().addAll(fullScreenImage, popupButtons);  
 popupLayout.setStyle("-fx-padding: 10; -fx-background-color: white; -fx-alignment: center;");  
  
 popup.getContent().add(popupLayout);  
 popup.show(primaryStage);  
 }  
  
  
  
}

**6. ChatClient**

*package* com.example.chattingappelements;  
  
*import* java.io.\*;  
*import* java.net.\*;  
*import* java.util.Scanner;  
  
*public class* ChatClient {  
 *private static final* String ***SERVER\_ADDRESS*** = "127.0.0.1";  
 *private static final int* ***PORT*** = 12345;  
 *private* Socket socket;  
 *private* PrintWriter out;  
 *private* BufferedReader in;  
 *public static void* main(String[] args) {  
 ChatClient client = *new* ChatClient();  
 client.start();  
 }  
  
 *public void* start() {  
 *try* {  
 *// Establish connection to the server* socket = *new* Socket(***SERVER\_ADDRESS***, ***PORT***);  
 out = *new* PrintWriter(socket.getOutputStream(), *true*);  
 in = *new* BufferedReader(*new* InputStreamReader(socket.getInputStream()));  
  
 *// Start a thread to listen for messages from the server  
 new* Thread(*new* IncomingMessageListener()).start();  
  
 *// Handle user input* Scanner scanner = *new* Scanner(System.***in***);  
 System.***out***.println(in.readLine()); *// Read initial server prompt  
 while* (*true*) {  
 String userInput = scanner.nextLine();  
 out.println(userInput);  
 }  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 } *finally* {  
 *try* {  
 *if* (socket != *null*) {  
 socket.close();  
 }  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 *// Inner class to listen for messages from the server  
 class* IncomingMessageListener *implements Runnable* {  
 @Override  
 *public void* run() {  
 *try* {  
 String serverMessage;  
 *while* ((serverMessage = in.readLine()) != *null*) {  
 System.***out***.println(serverMessage);  
 }  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

**7. Contacts**

*package* com.example.chattingappelements;  
  
*import* javafx.collections.FXCollections;  
*import* javafx.collections.*ObservableList*;  
*import* javafx.scene.image.Image;  
  
*import* java.io.*Serializable*;  
*import* java.util.*Comparator*;  
  
*public class* Contact *implements Serializable* {  
 *private static final long* ***serialVersionUID*** = 1L;  
  
 *private* String id;  
 *private* String name;  
 *private* String phoneNumber;  
 *private transient ObservableList*<Message> chatHistory = FXCollections.*observableArrayList*(); *// Fixed list initialization  
 private* Image contactImage;  
  
 *public static int contactIdGenerator* = 1;  
  
 *public* Contact() {}  
  
 *public* Contact(String name, String phoneNumber, Image contactImage) {  
 *this*.name = name;  
 *this*.phoneNumber = phoneNumber;  
 *this*.contactImage = contactImage;  
 *this*.id = String.*format*("%d", *contactIdGenerator*++);  
 }  
  
 *public* Contact(String name, String phoneNumber) {  
 *this*(name, phoneNumber, *null*);  
 }  
  
 *public* Image getContactImage() {  
 *return* contactImage;  
 }  
  
 *public void* setContactImage(Image contactImage) {  
 *this*.contactImage = contactImage;  
 }  
  
 *public* String getId() {  
 *return* id;  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public* String getPhoneNumber() {  
 *return* phoneNumber;  
 }  
  
 *public ObservableList*<Message> getChatHistory() {  
 *return* chatHistory; *// Return ObservableList directly* }  
  
 *public void* setName(String name) {  
 *this*.name = name;  
 }  
  
 *public void* setPhoneNumber(String phoneNumber) {  
 *this*.phoneNumber = phoneNumber;  
 }  
  
 *// Add a message to chat history  
 public void* addMessage(Message message) {  
 chatHistory.add(message);  
 }  
  
 @Override  
 *public* String toString() {  
 *return* String.*format*("%s. %-25s\n %-25s", id, name, phoneNumber);  
 }  
  
 *public void* smsSorter() {  
 chatHistory.sort(*Comparator*.*comparing*(Message::getTimestamp).reversed());  
 }  
  
 *public* String getContactsInfo() {  
 *return* String.*format*("\tID: %-03d | %-12s | %-22s", getId(), getName(), getPhoneNumber());  
 }  
}

**8. Message**

*package* com.example.chattingappelements;  
  
*import* java.io.*Serializable*;  
*import* java.time.LocalDateTime;  
*import* java.time.format.DateTimeFormatter;  
  
*public class* Message *implements Comparable*, *Serializable* {  
  
 *private* String content;  
 *private* LocalDateTime timestamp;  
 *private boolean* isRead;  
 *private* String sender; *// Added sender field to track message origin  
 private static int idCounter* = 1;  
 *private* String smsId;  
  
  
  
  
  
 *public* Message(String content, String sender) {  
 *this*.content = content;  
 *this*.timestamp = LocalDateTime.*now*();  
 *this*.isRead = *false*;  
 *this*.sender = sender;  
 *this*.smsId = String.*format*("%d", *idCounter*++);  
 }  
  
 *public void* markAsRead() {  
 *this*.isRead = *true*;  
 }  
  
 *public* String getsmsId() {  
 *return* smsId;  
 }  
  
 *public* LocalDateTime getTimestamp() {  
 *return* timestamp;  
 }  
  
 *public* String getContent() {  
 *return* content;  
 }  
  
 *public* String getSender() {  
 *return* sender;  
 }  
  
 *public boolean* isRead() {  
 *return* isRead;  
 }  
  
 *public* String detailedMsg(){  
 DateTimeFormatter dateFormatter = DateTimeFormatter.*ofPattern*("HH:mm:ss");  
  
 *return* String.*format*("\t%-15s | sent time : %s" ,getContent(),timestamp.format(dateFormatter));  
 }  
  
  
 @Override  
 *public* String toString() {  
 DateTimeFormatter dateFormatter = DateTimeFormatter.*ofPattern*("dd/MM/yyyy HH:mm:ss");  
 *return* "[" + timestamp.format(dateFormatter) + "] " + sender + ": " + content;  
 }  
  
 @Override  
 *public int* compareTo(Object o) {  
 Message s = (Message) o;  
 *return* s.getTimestamp().compareTo(*this*.timestamp); *// Descending order* }  
  
 *public* String detail(){  
 *return* String.*format*("%s : %-15s | sender : %-8s (%s)",getsmsId(),getContent(),sender,timestamp.format(DateTimeFormatter.*ofPattern*("dd/MM/yyyy HH:mm:ss")));  
 }  
  
 *public* String onlyMessageContent(){  
 *return* String.*format*("%s",content);  
 }  
  
 *public* String getMessageDate(){  
 DateTimeFormatter dateFormatter = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");  
 *return* String.*format*("%s",timestamp.format(dateFormatter));  
 }  
  
 *public* String getMessageTime(){  
 DateTimeFormatter timeFormatter = DateTimeFormatter.*ofPattern*("HH:mm:ss");  
 *return* String.*format*("%s",timestamp.format(timeFormatter));  
 }  
}

**9. TopBar**

*package* com.example.chattingappelements;  
  
*import* javafx.geometry.Insets;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Node;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.layout.HBox;  
*import* javafx.scene.layout.Region;  
  
*public class* TopBar {  
 *private* Button searchButton;  
 *private* Button tilesButton;  
  
 *public* TopBar() {  
 }  
  
*// public HBox create() {  
// // Create the HBox (top bar)  
// HBox topBar = new HBox();  
//  
// // Initialize buttons  
// searchButton = new Button("Search");  
// tilesButton = new Button("Tile");  
//  
// // Create a flexible spacer that will expand to push the tile button to the right corner  
// Region spacer = new Region();  
// HBox.setHgrow(spacer, javafx.scene.layout.Priority.ALWAYS);  
//  
//  
// /\*  
// \* The Region spacer = new Region(); acts as a flexible, resizable space. HBox.setHgrow  
// \* (spacer, javafx.scene.layout.Priority.ALWAYS); ensures that this region takes up all available  
// \*  
// \* \*/  
//  
// // Add elements to the HBox: searchButton, spacer, tilesButton  
// topBar.getChildren().addAll(searchButton, spacer, tilesButton);  
//  
// // Set HBox properties  
// topBar.setPrefWidth(MainWindow.stage.getWidth());  
// topBar.setAlignment(Pos.CENTER\_LEFT);  
// topBar.setPadding(new Insets(5, 10, 5, 10)); //top //right //bottom //left  
//  
// return topBar;  
// }  
  
  
 // Menu Bar* MenuBar menuBar = *new* MenuBar();  
 Menu optionsMenu = *new* Menu("Contacts");  
 MenuItem addContact = *new* MenuItem("Add Contact");  
 MenuItem deleteContact = *new* MenuItem("Delete Contact");  
 MenuItem modifyContact = *new* MenuItem("Modify Contact");  
 MenuItem userManual = *new* MenuItem("User Manual");  
  
  
 TextField search = *new* TextField();  
  
  
  
 *public* HBox create(){  
  
 search.setPromptText("Search");  
  
  
 optionsMenu.getItems().addAll(addContact, deleteContact, modifyContact, userManual);  
 menuBar.getMenus().addAll(optionsMenu);  
  
  
 HBox box = *new* HBox();  
 box.setAlignment(Pos.***CENTER***);  
 box.setSpacing(10);  
 box.setPadding(*new* Insets(10, 10, 10, 10));  
 box.getChildren().addAll(menuBar,search);  
  
box.setStyle("-fx-background-color: #005b52; -fx-border-color: #004112; -fx-border-width: 2px; -fx-padding: 10;");  
  
  
  
 *return* box;  
 }  
  
  
 *public* Button getSearchButton() {  
 *return* searchButton;  
 }  
  
 *public void* setSearchButton(Button searchButton) {  
 *this*.searchButton = searchButton;  
 }  
  
 *public* Button getTilesButton() {  
 *return* tilesButton;  
 }  
  
 *public void* setTilesButton(Button tilesButton) {  
 *this*.tilesButton = tilesButton;  
 }  
  
 *public* MenuBar getMenuBar() {  
 *return* menuBar;  
 }  
  
 *public void* setMenuBar(MenuBar menuBar) {  
 *this*.menuBar = menuBar;  
 }  
  
 *public* Menu getOptionsMenu() {  
 *return* optionsMenu;  
 }  
  
 *public void* setOptionsMenu(Menu optionsMenu) {  
 *this*.optionsMenu = optionsMenu;  
 }  
  
 *public* MenuItem getAddContact() {  
 *return* addContact;  
 }  
  
 *public void* setAddContact(MenuItem addContact) {  
 *this*.addContact = addContact;  
 }  
  
 *public* MenuItem getDeleteContact() {  
 *return* deleteContact;  
 }  
  
 *public void* setDeleteContact(MenuItem deleteContact) {  
 *this*.deleteContact = deleteContact;  
 }  
  
 *public* MenuItem getModifyContact() {  
 *return* modifyContact;  
 }  
  
 *public void* setModifyContact(MenuItem modifyContact) {  
 *this*.modifyContact = modifyContact;  
 }  
  
 *public* MenuItem getUserManual() {  
 *return* userManual;  
 }  
  
 *public void* setUserManual(MenuItem userManual) {  
 *this*.userManual = userManual;  
 }  
  
 *public* TextField getSearch() {  
 *return* search;  
 }  
  
 *public void* setSearch(TextField search) {  
 *this*.search = search;  
 }  
}