//Client Code

package application;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.input.KeyCode;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.paint.Color;

import javafx.stage.Stage;

import java.io.IOException;

import java.io.PrintWriter;

import java.net.Socket;

import java.util.NoSuchElementException;

import java.util.Optional;

import java.util.Scanner;

/\*\*

 \* Chat

 \*/

public class Chat extends Application {

    /\*\*

     \* constructor

     \*/

    public Chat(){}

    private final TextArea appMsg = new TextArea();

    private PrintWriter printServer;

    private TextArea newMessage = new TextArea();

    private Stage stage;

    private Socket socket;

    class NewMessage implements Runnable {

        public NewMessage(Socket s) {

            socket = s;

            Thread thread = new Thread(this);

            thread.start();

        }

        public void run() {

            try {

                try (Scanner getText = new Scanner(socket.getInputStream())) {

                    while (true) {

                        String message = getText.nextLine();

                        Platform.runLater(() -> appMsg.appendText(message+"\n"));//blank line on Chat window

                    }

                }

            } catch (IOException e) {

                e.printStackTrace();

            } catch (NoSuchElementException e) {

                Platform.runLater(() -> appMsg.appendText("Server is closed.\n" +

                        "Your chat log has been saved. Please feel free to close this window."));

            }

        }

    }

    @Override

    public void start(Stage primaryStage) {

        BorderPane pane = new BorderPane();

        pane.setPadding(new Insets(15, 15, 15, 15));

        pane.setPrefWidth(Double.MAX\_VALUE);

        pane.setPrefHeight(Double.MAX\_VALUE);

        appMsg.setEditable(false);

        Scene scene = new Scene(pane, 650, 400);

        scene.setFill(Color.TRANSPARENT); // make the background of the scene transparent

        scene.getRoot().setStyle("-fx-background-color: steelblue;");

        primaryStage.setScene(scene);

        primaryStage.show();

        this.stage = primaryStage;

        try {

            Socket socket = new Socket("localhost", 8080);

            if (socket.isConnected()){

                appMsg.appendText("Connected successfully, running on: " + socket + "\n\n");

                TextInputDialog dialog = new TextInputDialog();

                dialog.setTitle("Welcome");

                dialog.setHeaderText("Enter your name:");

                Optional<String> result = dialog.showAndWait();

                if (result.isPresent()) {

                    String title = result.get();

                    primaryStage.setTitle(title);

                }

            }

            printServer = new PrintWriter(socket.getOutputStream());

            new NewMessage(socket);

        }

        catch (IOException e) {

            appMsg.appendText("Problem to connect to server / server not found!\n");

        }

        ScrollPane scrollPane = new ScrollPane(appMsg);

        scrollPane.setFitToWidth(true);

        scrollPane.setFitToHeight(true);

        pane.setCenter(scrollPane);

        Button m\_btn = new Button("Send");

        m\_btn.setPrefWidth(80);

        GridPane m\_panel = new GridPane();

        m\_panel.setHgap(5);

        m\_panel.setVgap(5);

        m\_panel.setAlignment(Pos.CENTER);

        newMessage = new TextArea();

        newMessage.setPrefColumnCount(50);

        newMessage.setPrefHeight(80);

        m\_panel.add(newMessage, 0, 1);

        m\_panel.add(m\_btn, 1, 1);

        pane.setBottom(m\_panel);

        m\_btn.setOnAction(e -> sendMessage());

        newMessage.setOnKeyPressed(e -> {

            if (e.getCode() == KeyCode.ENTER && e.isShiftDown()) {

                // Shift + Enter was pressed, insert a new line without sending the message

                int caretPos = newMessage.getCaretPosition();

                newMessage.setText(newMessage.getText().substring(0, caretPos) + "\n" + newMessage.getText().substring(caretPos));

                newMessage.positionCaret(caretPos + 1);

                e.consume();

            } else if (e.getCode() == KeyCode.ENTER) {

                // Enter was pressed, send the message

                sendMessage();

                e.consume();

            }

        });

        stage.setOnCloseRequest(e -> {

            e.consume();

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

            alert.setTitle("Exit");

            alert.setHeaderText(null);

            alert.setContentText("Do you want to exit?");

            Optional<ButtonType> result = alert.showAndWait();

            if (result.isPresent() && result.get() == ButtonType.OK) {

                if (socket != null) {

                    try {

                        socket.close();

                        System.out.println("Chat closed");

                    } catch (IOException ioException) {

                        ioException.printStackTrace();

                    }

                }

                Platform.exit();

            }

        });

    }

    private void sendMessage() {

        try {

            if (!newMessage.getText().equals("")) {

                printServer.print(stage.getTitle() + ": " + newMessage.getText());

                newMessage.setText("");

                printServer.flush();

                newMessage.requestFocus();

            } else {

                javafx.scene.control.Alert alert = new javafx.scene.control.Alert(javafx.scene.control.Alert.AlertType.WARNING);

                alert.setTitle("Warning");

                alert.setHeaderText(null);

                alert.setContentText("Please input a message");

                alert.showAndWait();

            }

        } catch (NullPointerException n) {

            javafx.scene.control.Alert alert = new javafx.scene.control.Alert(javafx.scene.control.Alert.AlertType.ERROR);

            alert.setTitle("Error");

            alert.setHeaderText(null);

            alert.setContentText("Server is not running, client closing...");

            alert.showAndWait();

            System.exit(0);

        }

    }

}

/////////////////Server/////////////////////////////////

package application;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.time.LocalDateTime;

import java.time.ZoneId;

import java.time.ZonedDateTime;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.NoSuchElementException;

import java.util.Optional;

import java.util.Scanner;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ScrollPane;

import javafx.scene.control.TextArea;

import javafx.scene.layout.BorderPane;

import javafx.scene.paint.Color;

import javafx.stage.Stage;

/\*\*

 \* Server

 \*/

public class Server extends Application {

    /\*\*

     \* constructor

     \*/

    public Server(){}

    private final ArrayList<Client> clients = new ArrayList<>();

    private final TextArea textArea = new TextArea();

    private ServerSocket serverSocket;

    private Socket socket;

    @Override

    public void start(Stage primaryStage) {

        BorderPane pane = new BorderPane();

        pane.setPadding(new Insets(15, 15, 15, 15));

        pane.setPrefWidth(Double.MAX\_VALUE);

        pane.setPrefHeight(Double.MAX\_VALUE);

        ScrollPane scrollPane = new ScrollPane(textArea);

        scrollPane.setFitToWidth(true);

        scrollPane.setFitToHeight(true);

        pane.setCenter(scrollPane);

        Scene scene = new Scene(pane, 650, 250);

        scene.setFill(Color.TRANSPARENT); // make the background of the scene transparent

        scene.getRoot().setStyle("-fx-background-color: steelblue;"); // set the color of the window title bar

        primaryStage.setScene(scene);

        primaryStage.setTitle("Multi-thread Server");

        primaryStage.show();

        new Thread(() -> {

            try {

                serverSocket = new ServerSocket(8080);

                Platform.runLater(() ->

                        textArea.appendText("MultiThreadServer started at " +

                                DateTimeFormatter.ofPattern("EEEE MMMM d kk:mm:ss z u")

                                        .format(ZonedDateTime.now(ZoneId.systemDefault())) + "\n\n"));

                while (true) {

                    socket = serverSocket.accept();

                    clients.add(new Client(socket));

                }

            } catch (IOException e) {

                e.printStackTrace();

            }

        }).start();

        primaryStage.setOnCloseRequest(e -> {

            e.consume();

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

            alert.setTitle("Exit");

            alert.setHeaderText(null);

            alert.setContentText("Do you want to exit?");

            Optional<ButtonType> result = alert.showAndWait();

            if (result.isPresent() && result.get() == ButtonType.OK) {

                try {

                    // close all clients

                    for (Client client : clients) {

                        client.close();

                        System.out.println("Chat has disconnected");

                    }

                    // close the server socket

                    if (socket != null) {

                        socket.close();

                        System.out.println("Server closed");

                    }

                    System.exit(0);

                } catch (IOException ioException) {

                    ioException.printStackTrace();

                }

            }

        });

    }

    private void broadcast(String message) {

        Platform.runLater(() -> textArea.appendText(message + "\n"));//blank line on Server window

        LocalDateTime now = LocalDateTime.now();

        String fileName = "chat\_log\_" + DateTimeFormatter.ofPattern("yyyy-MM-dd").format(now) + ".txt";

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

            writer.write("[" + DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss").format(now) + "] " + message);

        } catch (IOException e) {

            e.printStackTrace();

        }

        for (Client client : clients) {

            client.sendMessage(message);

        }

    }

    private class Client implements Runnable {

        private final Socket socket;

        private PrintWriter writer;

        public Client(Socket socket) {

            this.socket = socket;

            new Thread(this).start();

        }

        @Override

        public void run() {

            try (Scanner scanner = new Scanner(socket.getInputStream())) {

                writer = new PrintWriter(socket.getOutputStream());

                Platform.runLater(() ->

                        textArea.appendText("Connection from " + socket  + " at " +

                                DateTimeFormatter.ofPattern("EEEE MMMM d - kk:mm:ss z u")

                                        .format(ZonedDateTime.now(ZoneId.systemDefault())) + "\n\n"));

                while (true) {

                    String message = scanner.nextLine();

                    broadcast(message + "\n");

                }

            } catch (IOException | NoSuchElementException e) {

                Platform.runLater(() -> textArea.appendText("Chat has disconnected.\n"));

                clients.remove(this);

            }

        }

        public void sendMessage(String message) {

            writer.println(message);

            writer.flush();

        }

        public void close() throws IOException{

            if (socket != null) {

                socket.close();

            }

        }

    }

    /\*\*

     \* main

     \* @param args  launch

     \*/

    public static void main(String[] args) {

        launch(args);

    }

}