PACKAGE: lotto; FILE: ClientListener, ColoredButton, DigitListener, Display, DownloaderThread, NumberInput, Principale, Reader

//CLIENTLISTENER.JAVA

package lotto;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.Scanner;

import javax.swing.JOptionPane;

public class ClientListener implements ActionListener{

//ALCUNI CAMPI: COSTANTI PER GESTIRE GLI STATI

public static final String START = "start";

public static final String STOP = "stop";

public static final String CONNECT = "connect";

public static final String DISCONNECT = "disconnect";

public static final String CLEAR = "clear";

//CLIENT LISTENER HA BISOGNO DI ACCEDERE ALLE INFO DELLA FINESTRA

//PER FARE QUESTO POSSIAMO IMPLEMENTARE DEI METODI PUBBLICI

//IN DISPLAY.JAVA OPPURE RENDERE NON PRIVATE I CAMPI DI INTERESSE

//DEL DISPLAY COSI DA POTERVI ACCEDERE DA QUI

//ipAddressField, portField, pulsantiera SONO DI INTERESSE

//QUINDI TOLGO IL PRIVATE

//FATTO QUESTO PASSO LA FINESTRA NEL COSTRUTTORE

//LA FINESTRA è QUINDI UN CAMPO

private Display displayAscoltato = null;

//CAMPI PER CONNESSIONE E LETTURA SCRITTURA

//CHE USI IN setupConnection() MA CHE TI SERVONO ANCHE FUORI

//QUINDI INIZIALIZZI QUI

private Socket sock = null;

private PrintWriter pw = null;

private Scanner sc = null;

//COSTRUTTORE

public ClientListener(Display display) {

this.displayAscoltato = display;

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals(START)) {

//GESTIONE RESET COLORE PULSANTIERA

for (ColoredButton cb : this.displayAscoltato.pulsantiera) {

cb.changeColor(Color.LIGHT\_GRAY);

}

//GESTIONE CASO NON TUTTI I NUMERI SONO INSERITI

for (ColoredButton cb : this.displayAscoltato.pulsantiera) {

if (cb.getDigit().equals("")) {

JOptionPane.showMessageDialog(null, "tutti i numeri devono essere inseriti");

return;

}

}

//GESTIONE VISIBILITà PULSANTI VANNO FATTE DOPO LE OPERAZIONI

//DI VERIFICA SOPRA!!!

this.displayAscoltato.started = true;

this.displayAscoltato.setButtons();

//INVIO COMANDO AL SERVER PER RICEVERE NUMERI

pw.println("start");

pw.flush();

//LETTURA NUMERI IN INGRESSO CON THREAD

DownloaderThread dt = new DownloaderThread(displayAscoltato.pulsantiera,

sc, displayAscoltato);

Thread thread = new Thread(dt);

thread.start();

}else if(e.getActionCommand().equals(STOP)){

//GESTIONE VISIBILITà PULSANTI

this.displayAscoltato.started = false;

this.displayAscoltato.setButtons();

//INVIO COMANDO AL SERVER PER INTERROMPERE NUMERI

pw.println("interrompi");

pw.flush();

}else if(e.getActionCommand().equals(CONNECT)){

try {

//AVVIO CONNESSIONE CON SERVER

this.setupConnection();

}catch(IOException ex) {

JOptionPane.showMessageDialog(null, "impossibile connettersi al server");

return;

}

//GESTIONE VISIBILITà PULSANTI

this.displayAscoltato.connected = true;

this.displayAscoltato.setButtons();

}else if(e.getActionCommand().equals(DISCONNECT)){

try {

//DISCONNESSIONE DA SERVER

this.closeConnection();

}catch(IOException ex) {

JOptionPane.showMessageDialog(null, "impossibile disconnettersi dal server");

return;

}

//GESTIONE VISIBILITà PULSANTI

this.displayAscoltato.connected = false;

this.displayAscoltato.setButtons();

}else if(e.getActionCommand().equals(CLEAR)){

//GESTIONE VISIBILITà PULSANTI

for (ColoredButton cb : this.displayAscoltato.pulsantiera) {

cb.changeColor(Color.LIGHT\_GRAY);

cb.setTextDigit("");

}

}

}

//METODI AUSILIARI

private void setupConnection() throws IOException {

//CREA SOCKET DI CONNESSIONE VERSO SERVER

sock = new Socket(this.displayAscoltato.ipAddressField.getText(),

Integer.parseInt(this.displayAscoltato.portField.getText()));;

//SU QUESTO SOCKET DEVO SIA SCRIVERE CHE LEGGERE

pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()));

sc = new Scanner(sock.getInputStream());

}

private void closeConnection() throws IOException {

sc.close();

pw.close();

sock.close();

}

}

//COLOREDBUTTON.JAVA

package lotto;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

@SuppressWarnings("serial")

public class ColoredButton extends JButton{

//stringa che mantiene l'informazione sulla posizione della casella nella griglia

private String position;

//colore della casella

private Color color;

//cifra visualizzata all'interno della casella

private String digit;

public ColoredButton(String position, Color color) {

this.position = position;

this.color = color;

this.digit = "";

DigitListener listener = new DigitListener();

this.addActionListener(listener);

changeColor(color);

}

public void setTextDigit(String digit) {

// metodo per gestire l'informazione riguardo la cifra visualizzata all'interno della casella

// passare come parametro la stringa vuota "" per ripristinare il contenuto di una casella

this.digit=digit;

super.setText(digit);

}

public void changeColor(Color c) {

// metodo per gestire l'informazione riguardo ai cambi di colore della casella

this.color = c;

super.setBackground(c);

}

public boolean isGreen() {

// metodo che indica se la casella in questione contiene un numero vincente

return color.equals(Color.GREEN);

}

public String getDigit() {

//metodo utile per verificare a-posteriori se la casella in questione contiene una cifra o una stringa vuota ""

return digit;

}

@Override

public void setBackground(Color bg) {

// NON UTILIZZARE setBackground per cambiare colore ad un'istanza di ColoredButton.

// Usare invece il metodo changeColor(Color color)

return;

}

@Override

public void setText(String str) {

// NON UTILIZZARE setText per cambiare il testo visualizzato ad un'istanza di ColoredButton.

// Usare invece il metodo setTextDigit(String n)

return;

}

}

//DIGITLISTENER.JAVA

package lotto;

import java.awt.event.\*;

import javax.swing.\*;

import java.awt.\*;

final class DigitListener implements ActionListener {

//listener per cambio cifra con il click

protected DigitListener(){

super();

}

public void actionPerformed(ActionEvent e) {

ColoredButton cb = (ColoredButton) e.getSource();

Reader r = new NumberInput();

String digit = r.read();

if (digit!=null) {

cb.setTextDigit(""+digit);

cb.changeColor(Color.LIGHT\_GRAY);

}

}

}

//DISPLAY.JAVA

package lotto;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

public class Display extends JFrame{

//COMPONENTI

private JButton startButton = new JButton("Start");

JTextField ipAddressField = new JTextField("127.0.0.1");

JTextField portField = new JTextField("4400");

private JButton stopButton = new JButton("Stop");

private JButton connectButton = new JButton("Connect");

private JButton disconnectButton = new JButton("Disconnect");

private JButton clear = new JButton("Clear");

ColoredButton[] pulsantiera = new ColoredButton[5];

//QUESTI DUE CAMPI BOOLEAN SERVONO SIA PER IL METODO

//SET BUTTON CHE NEL CLIENT LISTENER

public boolean connected = false;

public boolean started = false;

//COSTRUTTORE

public Display() {

super("Nome Cognome Matricola");

JPanel northPanel = new JPanel();

northPanel.add(startButton);

northPanel.add(new JLabel("IP Address"));

northPanel.add(ipAddressField);

northPanel.add(new JLabel("Port"));

northPanel.add(portField);

northPanel.add(stopButton);

this.getContentPane().add(northPanel, BorderLayout.NORTH);

JPanel middlePanel = new JPanel(new GridLayout(1,5));

for (int i=0; i<5; i++) {

pulsantiera[i]=new ColoredButton(i+"", Color.LIGHT\_GRAY);

middlePanel.add(pulsantiera[i]);

}

//DEVI AGGIUNGERE SET PREFFERRED SIZE ALTRIMENTI VENGONO MINI

middlePanel.setPreferredSize(new Dimension(600, 100));

this.getContentPane().add(middlePanel, BorderLayout.CENTER);

JPanel southPanel = new JPanel();

southPanel.add(connectButton);

southPanel.add(disconnectButton);

southPanel.add(clear);

this.getContentPane().add(southPanel, BorderLayout.SOUTH);

//PARTE DI EVENTI E LISTENER

ClientListener cl = new ClientListener(this);

startButton.addActionListener(cl);

startButton.setActionCommand(ClientListener.START);

stopButton.addActionListener(cl);

stopButton.setActionCommand(ClientListener.STOP);

connectButton.addActionListener(cl);

connectButton.setActionCommand(ClientListener.CONNECT);

disconnectButton.addActionListener(cl);

disconnectButton.setActionCommand(ClientListener.DISCONNECT);

clear.addActionListener(cl);

clear.setActionCommand(ClientListener.CLEAR);

//PARTE DI IMPOSTAZIONI VISIBILITà

this.setSize(700, 200);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setLocationRelativeTo(null);

this.pack();

this.setButtons();//VEDI METODO SOTTO

this.setVisible(true);

}

//METODO CENTRALIZZATO CHE GESTISCE I VARI STATI DELLA FINESTRA

//STATO INIZIALE: CLIENT NON CONNESSO AL SERVER

//STATO CONNESSIONE PRIMA DI PREMERE START

//STATO CONNESSIONE DOPO LA PRESSIONE DI START

public void setButtons() {

if (!connected) {

//la connessione non è ancora avvenuta

//voglio solo i pulsanti connect e clear

connectButton.setEnabled(true);

clear.setEnabled(true);

startButton.setEnabled(false);

disconnectButton.setEnabled(false);

stopButton.setEnabled(false);

}else{

connectButton.setEnabled(false);

if (started) {

startButton.setEnabled(false);

stopButton.setEnabled(true);

clear.setEnabled(false);

disconnectButton.setEnabled(false);

//non voglio disconnettermi a gioco iniziato

}else {

startButton.setEnabled(true);

stopButton.setEnabled(false);

clear.setEnabled(true);

//posso cancellare ciò che ho inserito

disconnectButton.setEnabled(true);

//non sto giocando posso disconnettermi

}

}

}

}

//DOWNLOADERTHREAD.JAVA

package lotto;

import java.awt.Color;

import java.util.Scanner;

import javax.swing.JOptionPane;

//QUESTO THREAD LEGGE I DATI E LI ANALIZZA

//CAMBIA I COLORI E MANDA UN MESSAGGIO DI CONSEGUENZA

public class DownloaderThread implements Runnable{

private ColoredButton[] pulsantiera;

private Scanner sc;

private Display display;

public DownloaderThread(ColoredButton[] pulsantiera,

Scanner sc, Display display) {

this.pulsantiera = pulsantiera;

this.sc = sc;

this.display = display;

}

@Override

public void run() {

//MENTRE RICEVIAMO I NUMERI DOBBIAMO DISABILITARE I PULSANTI

setEnabled(false);

boolean running = true;

//USIAMO RUNNING PER CAPIRE QUANDO INTERROMPERE

//IL CICLO DI LETTURA

boolean interrupted = false;

//USIAMO INTERRUPTED PER TENERE IN MEMORIA SE ABBIAMO INTERROTTO

//PERCHè VUOL DIRE CHE ABBIAMO PERSO

while(running) {

String cmd = sc.nextLine();

//TOKENIZZAZIONE DEI COMANDI RICEVUTI

String commands[] = cmd.split(";");

//SPLIT DIVIDE SUL CARATTERE PASSATO

//E RESTITUISCE UN ARRAY

if (commands[0].equals("\*") && commands[1].equals("\*")) {

//SIAMO ALLA FINE DEI DATI

running = false;

continue;

}

if (commands[0].equals("-1") && commands[1].equals("-1")) {

//ABBIAMO PREMUTO DISCONNECT, ESTRAZIONE INTERROTTA

running = false;

interrupted = true;

continue;

}

int posizione = Integer.parseInt(commands[0]);

String numeroEstratto = commands[1];

if (pulsantiera[posizione].getDigit().equals(numeroEstratto)) {

pulsantiera[posizione].changeColor(Color.GREEN);

}else {

pulsantiera[posizione].changeColor(Color.RED);

}

}

if (interrupted) {

//L'UTENTE HA PERSO

JOptionPane.showMessageDialog(null, "you lost");

}else {

boolean greenFound = false;

for (ColoredButton cb : pulsantiera) {

if (cb.isGreen()) {

greenFound = true;

}

}

if (greenFound) {

//L'UTENTE HA VINTO

JOptionPane.showMessageDialog(null, "you won");

}else {

//L'UTENTE HA PERSO

JOptionPane.showMessageDialog(null, "you lost");

}

}

setEnabled(true);

display.started = false;

//ALLA FINE DEL THREAD METTO STARTED A FALSE

display.setButtons();

//E RISETTO I PULSANTI

}

public void setEnabled(boolean state) {

for (ColoredButton cb : pulsantiera) {

cb.setEnabled(state);

}

}

}

//NUMBERINPUT.JAVA

package lotto;

import javax.swing.JOptionPane;

import java.io.\*;

import java.lang.\*;

import java.util.\*;

public class NumberInput implements Reader {

@Override

public String read() {

String digit = JOptionPane.showInputDialog("inserire numero: ");

if (digit == null) {return null;}

boolean correctDigit = false; //gestione car non validi

while (!correctDigit) { //gestione car non validi

try {

while (Integer.parseInt(digit)<0 || Integer.parseInt(digit)>9) {

digit = JOptionPane.showInputDialog("inserire numero: ");

if (digit == null) {return null;}

}

correctDigit = true; //gestione car non validi

}catch(NumberFormatException e) {

digit = JOptionPane.showInputDialog("inserire numero: ");

//gestione car non validi

}

}

return digit;

}

}

//READER.JAVA

package lotto;

public interface Reader {

String read();

}

//PRINCIPALE.JAVA

package lotto;

public class Principale {

public static void main(String[] args) {

Display display = new Display();

}

}