COSC 1200 Assignment 5

/\*

\* Author: Christian Powlette

\* Due Date: April 15, 2024

\* Professor: Yuvaraj Sambandan

\* Course: COSC-1200-06 Object-Oriented Programming

\* Description of class: A program for patients’ registration and veterinarian assignment.

\*/

**User Interface Window**

A screenshot of a computer

Description automatically generated

**Unsuccessful Registration**

A screenshot of a computer

Description automatically generated

**Successful Write to File**

A screenshot of a computer

Description automatically generated

**Newpatientfile.txt write.**

A screenshot of a computer

Description automatically generated

**Register Button Tooltip**

A screenshot of a computer

Description automatically generated

**Clear Button Tooltip**

**A screenshot of a computer

Description automatically generated**

**Exit Button Tooltip**

**A screenshot of a computer

Description automatically generated**

**Successful Patient Registration + Write to file.**

**A screenshot of a computer

Description automatically generated**

\*\*After the realization that the message **“Patient registered successfully”** and **“Successfully wrote to file”** would overlap because they were in the same status Label, I created another, so they came be shown simultaneously\*\*

**A screenshot of a computer screen

Description automatically generated**

**Picture Used as the Icon**

**A black and white logo

Description automatically generated**

Code

/\*

\* Author: Christian Powlette

\* Due Date: April 15, 2024

\* Professor: Yuvaraj Sambandan

\* Course: COSC-1200-06 Object-Oriented Programming

\* Description of class: A program for patients registration and veterinarian assignment.

\*/

// Imports

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.io.PrintWriter;

import java.util.Date;

import java.util.regex.\*;

import java.awt.event.InputEvent;

//Clas Defintion

public class NewPatientForm extends JFrame {

//Instance Variables

private JTextField patientNameField, ownerNameField, emailField; //Initiates the 3 required text fields

private JRadioButton vet1Button, vet2Button;

private JButton registerButton, clearButton, exitButton;

private JLabel statusLabel1, statusLabel2;

//Creation of The Patient Registration Program Forum / Constructor

public NewPatientForm() {

setTitle("New Patient Registry"); //Form has custom title "New Patient Registry"

setIconImage(new ImageIcon("veterinarian-program-logo.jpeg").getImage()); //// Form sets the Icon in the top left replacing the standard Java Logo

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //Form sets exit button in top right as operational

setResizable(false); //Set as not able to resize

// Set layout

JPanel panel = new JPanel();

GroupLayout layout = new GroupLayout(panel);

panel.setLayout(layout);

layout.setAutoCreateGaps(true);

layout.setAutoCreateContainerGaps(true);

// Create components

//Text Field Labels

JLabel patientLabel = new JLabel("Patient Name:"); //Label for Patient Name Text Field

patientNameField = new JTextField(20); //Creates Patient Name Text Field

JLabel ownerLabel = new JLabel("Owner Name:"); //Label for Owner

ownerNameField = new JTextField(20); //Creates Owner Text Field

JLabel emailLabel = new JLabel("Email Address:"); //Label for Email Address

emailField = new JTextField(20); //Creates Email Address Text field

//Radio Button Labels

JLabel vetLabel = new JLabel("Assign Veterinarian:");

vet1Button = new JRadioButton("Christian Doe", true);

vet2Button = new JRadioButton("John Powlette");

//Creating Radio Button group

ButtonGroup vetGroup = new ButtonGroup();

vetGroup.add(vet1Button); //Assigns to Christian Doe

vetGroup.add(vet2Button); //Assigns to John Powlette

//Create Buttons

//Creation of Register Button

registerButton = new JButton("Register");

registerButton.setToolTipText("Click to register new patient (Alt+R)"); //Sets tool-tip for Button

//Add Action Listener to Register Button

registerButton.addActionListener(\_ -> registerPatient());

//Set Key binding (ALT + R) for Register Button

InputMap registerInputMap = registerButton.getInputMap(JComponent.WHEN\_IN\_FOCUSED\_WINDOW);

ActionMap registerActionMap = registerButton.getActionMap();

KeyStroke altR = KeyStroke.getKeyStroke(KeyEvent.VK\_R, InputEvent.ALT\_DOWN\_MASK);

registerInputMap.put(altR, "registerAction");

registerActionMap.put("registerAction", new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

registerPatient();

//System.out.println("Attempt at Registration of Patient performed via (ALT + R) key binding!"); // Left Behind to show my testing

}

});

//Creation of Clear Button

clearButton = new JButton("Clear");

clearButton.setToolTipText("Click to Clear the form (Alt+C)"); //Sets tool-tip for Button

//Add Action Listener to Clear Button

clearButton.addActionListener(\_ -> clearForm());

//Set Key binding for Clear Button

InputMap clearInputMap = clearButton.getInputMap(JComponent.WHEN\_IN\_FOCUSED\_WINDOW);

ActionMap clearActionMap = clearButton.getActionMap();

KeyStroke altC = KeyStroke.getKeyStroke(KeyEvent.VK\_C, InputEvent.ALT\_DOWN\_MASK);

clearInputMap.put(altC, "clearAction");

clearActionMap.put("clearAction",

new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

clearForm();

//System.out.println("Clear Form performed via (ALT + C) key binding!"); // Left Behind to show my testing

}

});

//Creation of Exit Button

exitButton = new JButton("Exit");

exitButton.setToolTipText("Exit the form (Alt+E)"); //Sets Tool-tip for Button

//Adds Action Listener for Button

exitButton.addActionListener(\_ -> System.exit(0));

//Set Key binding (ALT + R) for Register Button

InputMap exitInputMap = exitButton.getInputMap(JComponent.WHEN\_IN\_FOCUSED\_WINDOW);

ActionMap exitActionMap = exitButton.getActionMap();

KeyStroke altE = KeyStroke.getKeyStroke(KeyEvent.VK\_E, InputEvent.ALT\_DOWN\_MASK);

exitInputMap.put(altE, "registerExit");

exitActionMap.put("registerExit",

new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

//System.out.println("Program terminated via (ALT + E) key binding!"); // Left Behind to show my testing

System.exit(0);

}

});

//Creating Status Labels

statusLabel1 = new JLabel(" ");

statusLabel1.setForeground(Color.RED); // Set Error messages in Red color

statusLabel2 = new JLabel(" ");

statusLabel2.setForeground(Color.BLACK); // Set Error messages in Black color

//Create Layout for the placement of components Horizontally

layout.setHorizontalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(patientLabel)

.addComponent(patientNameField)

.addComponent(ownerLabel)

.addComponent(ownerNameField)

.addComponent(emailLabel)

.addComponent(emailField)

.addComponent(vetLabel)

.addComponent(vet1Button)

.addComponent(vet2Button)

.addGroup(layout.createSequentialGroup()

.addComponent(registerButton)

.addComponent(clearButton)

.addComponent(exitButton))

.addComponent(statusLabel1)

.addComponent(statusLabel2));

//Create layout for placement of components Vertically

layout.setVerticalGroup(layout.createSequentialGroup()

.addComponent(patientLabel)

.addComponent(patientNameField)

.addComponent(ownerLabel)

.addComponent(ownerNameField)

.addComponent(emailLabel)

.addComponent(emailField)

.addComponent(vetLabel)

.addComponent(vet1Button)

.addComponent(vet2Button)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)

.addComponent(registerButton)

.addComponent(clearButton)

.addComponent(exitButton))

.addComponent(statusLabel1)

.addComponent(statusLabel2));

add(panel);

pack();

setSize(400, 410); //Set as not able to resize

setLocationRelativeTo(null); // Center the form on the screen Upon Execution

}

//Method to Register Patients

private void registerPatient() {

String patientName = patientNameField.getText().trim();

String ownerName = ownerNameField.getText().trim();

String email = emailField.getText().trim();

String vetName = vet1Button.isSelected() ? "Christian Doe" : "John Powlette";

if (patientName.isEmpty() || ownerName.isEmpty() || email.isEmpty() || !isValidEmail(email)) {

statusLabel1.setText("Error: Please fill in all fields correctly."); // Un-Success Message Displayed

return;

}

statusLabel1.setText("Patient registered successfully."); //Success Message Displayed

try {

// Writing to file

try (PrintWriter fileOut = new PrintWriter("newpatientfile.txt")) {

fileOut.println("\*\*Patient Registration Document\*\*");

fileOut.println(STR."Patient Name: \{patientName}");

fileOut.println(STR."Owner Name: \{ownerName}");

fileOut.println(STR."email address: \{email}");

fileOut.println(STR."Selected Veterinarian: \{vetName}");

fileOut.println(STR."Current Date: \{new Date()}");

statusLabel2.setText("Successfully wrote to file.");

}

} catch (Exception e) {

statusLabel2.setText("There was an error writing to file."); //file Write Error Message

}

}

//Incorporated Valid email checker

private boolean isValidEmail(String email) {

String regexPattern = "^[\\w!#$%&'\*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'\*+/=?`{|}~^-]+)\*" +

"@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$";

Pattern pattern = Pattern.compile(regexPattern);

Matcher matcher = pattern.matcher(email);

return matcher.matches();

}

//Method to clear the Form, and reset as Beginning

private void clearForm() {

patientNameField.setText("");

ownerNameField.setText("");

emailField.setText("");

vet1Button.setSelected(true);

statusLabel1.setText(" ");

patientNameField.requestFocus();

}

//Main Method

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

new NewPatientForm().setVisible(true); //Sets Interface as Visible

});

}

}