

**North South University**

Department of Electrical and Computer Engineering

**Lab Project Report**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Semester : NSU Spring 2023

Course Code : CSE 215L

Section : 16

Group Name : G

Faculty : Dr. Shamim Al Mamun (SAM3)

Lab Instructor : A. S. M. Sabiqul Hassan

Project Topic : Phonebook Management System

GitHub Repo Link : <https://www.google.com/url?q=https://github.com/MidhatShazzad/Phonebook-Management-System&sa=D&source=editors&ust=1686908644934301&usg=AOvVaw2uOpazdHeaWklzOL9d7Un1>

Submission Date : 16/06/2023

|  |  |
| --- | --- |
| Student Information | GitHub Account Links |
| Midhat Bin Shazzad  2222560642  [midhat.shazzad@northsouth.edu](mailto:midhat.shazzad@northsouth.edu) | <https://github.com/MidhatShazzad> |
| Mohammed Mostofa  2222874042  [mohammed.mostofa@northsouth.edu](mailto:mohammed.mostofa@northsouth.edu) | <https://github.com/mowtofa> |
| Md.Alamin Hasan Shawon  1912322642  [alamin.shawon1@northsouth.edu](mailto:alamin.shawon1@northsouth.edu) | <https://github.com/hasanshawon18> |
|  |  |

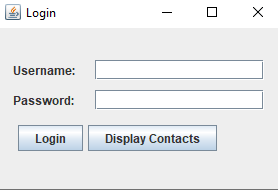
Introduction

The purpose of this project is to develop a Java-based phonebook application that allows users to add, edit, and remove contacts. The application also provides the ability to display contacts without requiring the user to log in. The project utilizes Java Swing for the user interface and employs file handling to store the contact information..

Implementation

The application consists of two main classes: “PhonebookLoginUI” and “PhonebookUI”. The “PhonebookLoginUI” class handles the login functionality, while the “PhonebookUI” class manages the main phonebook interface.

PhonebookLoginUI

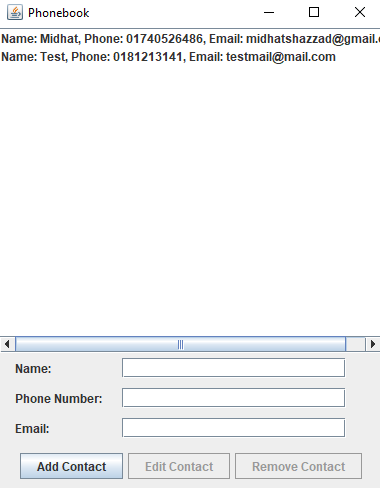


The “PhonebookLoginUI” class serves as the entry point of the application. It extends the “JFrame” class and displays a login window to the user. The window contains fields for entering a username and password, along with login and display contacts buttons.

The `createLoginPanel()` method creates the login panel with the necessary components such as labels, text fields, and buttons. The login button's action listener checks if the entered username and password match the hardcoded values ("Admin" and "Password" respectively). If the login is successful, the `launchPhonebookUI()` method is invoked to open the main phonebook window. Otherwise, an error message is displayed.

The `displayContactsButton` action listener invokes the `displayContactsFromFile()` method, which reads the contacts from a file ("contacts.txt") and displays them using a “JOptionPane”.

PhonebookUI



The “PhonebookUI” class represents the main phonebook program. It also extends the “JFrame” class and displays a list of contacts along with a form for adding, editing, and removing contacts.

The constructor initializes the UI components such as the contact list, form fields, and buttons. It also creates an instance of the “Phonebook” class, which manages the contact information.

The “createFormPanel()” method creates the form panel with labels and text fields for entering contact information. It also adds buttons for adding, editing, and removing contacts. The action listeners for these buttons invoke methods in the “Phonebook” class to perform the respective operations. After each operation, the contacts are saved to a file using the “saveContacts()” method, and the updated contact list is displayed using the “displayContacts()” method.

The “handleContactSelection()” method is invoked when a contact is selected from the contact list. It populates the form fields with the selected contact's information and enables the edit and remove buttons. If no contact is selected, the form fields are cleared, and the edit and remove buttons are disabled.

The “saveContacts()” method writes the contact information to a file ("contacts.txt"). It iterates over the contacts in the “Phonebook” class and writes each contact's name, phone number, and email on a separate line.

The “loadContacts()” method reads the contacts from the file using a “BufferedReader” and fills the “Phonebook” class with the retrieved contacts. Each line in the file is split into parts using a semicolon (;). If the line contains all three parts (name, phone number, and email), a new “Contact” object is created and added to the “Phonebook”.

Contact and Phonebook Classes

The “Contact” class represents an individual contact. It contains fields for name, phone number, and email. It provides getter methods for accessing the contact information and overrides the “toString()” method to display the contact details in a formatted string.

The “Phonebook” class manages a list of “Contact” objects. It provides methods for adding, editing, and removing contacts. The class also has a method for retrieving all contacts. The

contact list is implemented using an arraylist.

Conclusion

The phonebook application developed in this project allows users to add, edit, and remove contacts. The application provides a login screen for authentication and stores the contact information in a file. Additionally, it allows users to display contacts without requiring them to log in. It has certain limitations such as a hardcored username and password and the lack of a search functionality. However, it handles the basic functions of a typical phonebook.

CODE

package com.mycompany.phonebookloginui;

/\*\*

\*

\* @author Midhat

\*/

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class PhonebookLoginUI extends JFrame {

private JButton loginButton;

private JButton displayContactsButton;

public PhonebookLoginUI() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setTitle("Login");

setSize(300, 200);

setLayout(new BorderLayout());

JPanel loginPanel = createLoginPanel();

add(loginPanel, BorderLayout.CENTER);

}

private JPanel createLoginPanel() {

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints constraints = new GridBagConstraints();

constraints.insets = new Insets(5, 10, 5, 10);

constraints.anchor = GridBagConstraints.WEST;

JLabel usernameLabel = new JLabel("Username:");

JTextField usernameField = new JTextField(15);

addFormField(panel, constraints, usernameLabel, usernameField);

JLabel passwordLabel = new JLabel("Password:");

JPasswordField passwordField = new JPasswordField(15);

addFormField(panel, constraints, passwordLabel, passwordField);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

loginButton = new JButton("Login");

loginButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

if (username.equals("Admin") && password.equals("Password")) {

launchPhonebookUI();

} else {

JOptionPane.showMessageDialog(null, "Invalid username or password.", "Login Failed", JOptionPane.ERROR\_MESSAGE);

}

}

});

displayContactsButton = new JButton("Display Contacts");

displayContactsButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

displayContactsFromFile();

}

});

buttonPanel.add(loginButton);

buttonPanel.add(displayContactsButton);

constraints.gridx = 0;

constraints.gridy++;

constraints.gridwidth = 2;

panel.add(buttonPanel, constraints);

return panel;

}

private void addFormField(JPanel panel, GridBagConstraints constraints, JLabel label, JTextField textField) {

constraints.gridx = 0;

constraints.gridy++;

constraints.gridwidth = 1;

panel.add(label, constraints);

constraints.gridx = 1;

panel.add(textField, constraints);

}

private void launchPhonebookUI() {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

PhonebookUI phonebookUI = new PhonebookUI();

phonebookUI.setVisible(true);

dispose();

}

});

}

private void displayContactsFromFile() {

try (BufferedReader reader = new BufferedReader(new FileReader("contacts.txt"))) {

StringBuilder contactsBuilder = new StringBuilder();

String line;

while ((line = reader.readLine()) != null) {

contactsBuilder.append(line).append("\n");

}

JOptionPane.showMessageDialog(null, contactsBuilder.toString(), "Contacts", JOptionPane.PLAIN\_MESSAGE);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Error reading contacts from file.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

PhonebookLoginUI loginUI = new PhonebookLoginUI();

loginUI.setVisible(true);

}

});

}

}

class PhonebookUI extends JFrame {

private Phonebook phonebook;

private JTextField nameField;

private JTextField phoneNumberField;

private JTextField emailField;

private JList<Contact> contactList;

private DefaultListModel<Contact> listModel;

private JButton addButton;

private JButton editButton;

private JButton removeButton;

public PhonebookUI() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setTitle("Phonebook");

setSize(400, 500);

setLayout(new BorderLayout());

phonebook = new Phonebook();

listModel = new DefaultListModel<>();

contactList = new JList<>(listModel);

contactList.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

contactList.addListSelectionListener(e -> handleContactSelection());

JScrollPane scrollPane = new JScrollPane(contactList);

add(scrollPane, BorderLayout.CENTER);

JPanel formPanel = createFormPanel();

add(formPanel, BorderLayout.SOUTH);

loadContacts();

displayContacts();

}

private JPanel createFormPanel() {

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints constraints = new GridBagConstraints();

constraints.insets = new Insets(5, 10, 5, 10);

constraints.anchor = GridBagConstraints.WEST;

JLabel nameLabel = new JLabel("Name:");

nameField = new JTextField(20);

addFormField(panel, constraints, nameLabel, nameField);

JLabel phoneNumberLabel = new JLabel("Phone Number:");

phoneNumberField = new JTextField(20);

addFormField(panel, constraints, phoneNumberLabel, phoneNumberField);

JLabel emailLabel = new JLabel("Email:");

emailField = new JTextField(20);

addFormField(panel, constraints, emailLabel, emailField);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

addButton = new JButton("Add Contact");

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String name = nameField.getText();

String phoneNumber = phoneNumberField.getText();

String email = emailField.getText();

Contact contact = new Contact(name, phoneNumber, email);

phonebook.addContact(contact);

nameField.setText("");

phoneNumberField.setText("");

emailField.setText("");

saveContacts();

displayContacts();

}

});

editButton = new JButton("Edit Contact");

editButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int selectedIndex = contactList.getSelectedIndex();

if (selectedIndex != -1) {

String name = nameField.getText();

String phoneNumber = phoneNumberField.getText();

String email = emailField.getText();

Contact contact = new Contact(name, phoneNumber, email);

phonebook.editContact(selectedIndex, contact);

nameField.setText("");

phoneNumberField.setText("");

emailField.setText("");

saveContacts();

displayContacts();

}

}

});

editButton.setEnabled(false);

removeButton = new JButton("Remove Contact");

removeButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int selectedIndex = contactList.getSelectedIndex();

if (selectedIndex != -1) {

phonebook.removeContact(selectedIndex);

nameField.setText("");

phoneNumberField.setText("");

emailField.setText("");

saveContacts();

displayContacts();

}

}

});

removeButton.setEnabled(false);

buttonPanel.add(addButton);

buttonPanel.add(editButton);

buttonPanel.add(removeButton);

constraints.gridx = 0;

constraints.gridy++;

constraints.gridwidth = 2;

panel.add(buttonPanel, constraints);

return panel;

}

private void addFormField(JPanel panel, GridBagConstraints constraints, JLabel label, JTextField textField) {

constraints.gridx = 0;

constraints.gridy++;

constraints.gridwidth = 1;

panel.add(label, constraints);

constraints.gridx = 1;

panel.add(textField, constraints);

}

private void handleContactSelection() {

int selectedIndex = contactList.getSelectedIndex();

if (selectedIndex != -1) {

Contact contact = contactList.getSelectedValue();

if (contact != null) {

nameField.setText(contact.getName());

phoneNumberField.setText(contact.getPhoneNumber());

emailField.setText(contact.getEmail());

editButton.setEnabled(true);

removeButton.setEnabled(true);

}

} else {

nameField.setText("");

phoneNumberField.setText("");

emailField.setText("");

editButton.setEnabled(false);

removeButton.setEnabled(false);

}

}

private void displayContacts() {

listModel.clear();

List<Contact> contacts = phonebook.getAllContacts();

for (Contact contact : contacts) {

listModel.addElement(contact);

}

}

private void saveContacts() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter("contacts.txt"))) {

List<Contact> contacts = phonebook.getAllContacts();

for (Contact contact : contacts) {

writer.write(contact.getName() + ";" + contact.getPhoneNumber() + ";" + contact.getEmail());

writer.newLine();

}

} catch (IOException e) {

e.printStackTrace();

}

}

private void loadContacts() {

try (BufferedReader reader = new BufferedReader(new FileReader("contacts.txt"))) {

String line;

while ((line = reader.readLine()) != null) {

String[] parts = line.split(";");

if (parts.length == 3) {

String name = parts[0];

String phoneNumber = parts[1];

String email = parts[2];

Contact contact = new Contact(name, phoneNumber, email);

phonebook.addContact(contact);

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

PhonebookLoginUI loginUI = new PhonebookLoginUI();

loginUI.setVisible(true);

}

});

}

}

class Contact {

private String name;

private String phoneNumber;

private String email;

public Contact(String name, String phoneNumber, String email) {

this.name = name;

this.phoneNumber = phoneNumber;

this.email = email;

}

public String getName() {

return name;

}

public String getPhoneNumber() {

return phoneNumber;

}

public String getEmail() {

return email;

}

@Override

public String toString() {

return "Name: " + name + ", Phone: " + phoneNumber + ", Email: " + email;

}

}

class Phonebook {

private List<Contact> contacts;

public Phonebook() {

contacts = new ArrayList<>();

}

public void addContact(Contact contact) {

contacts.add(contact);

}

public void editContact(int index, Contact contact) {

if (index >= 0 && index < contacts.size()) {

contacts.set(index, contact);

}

}

public void removeContact(int index) {

if (index >= 0 && index < contacts.size()) {

contacts.remove(index);

}

}

public List<Contact> getAllContacts() {

return contacts;

}

}