****

**North South University**

Department of Electrical and Computer Engineering

**Lab Project Report**

Semester : NSU Spring 2023

Course Code : CSE 215L

Section : 16

Group Name : A

Faculty : Dr. Shamim Al Mamun (SAM3)

Lab Instructor : A. S. M. Sabiqul Hassan

Project Topic : Metro\_Rail\_Ticketing\_Management System

GitHub Repo Link :

Submission Date : 16/06/2023

| Student Information | GitHub Account Links |
| --- | --- |
| // 2222472042  Md.Rayhan Sharif Rafsun  // rayhan.rafsun@northsouth.edu | // https://github.com/rsrrafsun |
|  |  |
|  |  |
|  |  |

We are from group D.

Here is my lab report explaining every method , class, variable , and also their purpose on this project

The first class in this project is Home.java

In this class there are Button.

adminButton

signUpButton

loginButton

and every button has actionlister to do their particular task.

This code represents a graphical user interface (GUI) for a Metro Ticketing System. It creates a window with buttons for different functionalities.

The code begins with importing necessary packages:

javax.swing.\* provides classes for creating GUI components.

java.awt.\* provides classes for basic GUI functionality.

java.awt.event.\* provides classes for handling GUI events.

The Home class is declared, which extends the JFrame class and implements the ActionListener interface. This means that the Home class can handle events generated by GUI components.

Inside the Home class, various GUI components are declared as private member variables:

adminButton, signUpButton, and loginButton are JButton objects representing buttons in the GUI.

headingLabel is a JLabel object representing a label in the GUI.

The Home class contains a constructor Home(), which initializes the GUI components and sets up the window:

The background color of the window is set to white using getContentPane().setBackground(Color.WHITE).

The layout manager is set to null using setLayout(null), which means components will be manually positioned.

The size of the window is set to 600x400 pixels using setSize(600, 400).

The default close operation is set to JFrame.EXIT\_ON\_CLOSE, which exits the application when the window is closed.

The window is centered on the screen using setLocationRelativeTo(null).

Inside the constructor, the GUI components are initialized and added to the window:

The headingLabel is created with the text "Welcome to Metro Ticketing System", positioned at (120, 0) with a width of 1000 pixels and a height of 150 pixels. The font, color, and style of the label are also set.

The adminButton, signUpButton, and loginButton are created with their respective labels. They are positioned and styled accordingly.

Event listeners are attached to the buttons using addActionListener(this), indicating that the Home class will handle the button click events.

The actionPerformed method is implemented from the ActionListener interface, which handles button click events:

The method checks which button generated the event using ae.getSource().

If the signUpButton is clicked, the current window is disposed of (closed) using dispose(), and a new Register object is created. This likely opens a registration form for the user.

If the loginButton is clicked, the current window is disposed of, and a new Login object is created. This likely opens a login form for the user.

If the adminButton is clicked, the current window is disposed of, and a new AdminLogin object is created. This likely opens a login form for an administrator.

The main method creates an instance of the Home class, which initializes and displays the GUI window.

Overall, this code sets up a simple GUI for a Metro Ticketing System with buttons for user registration, user login, and administrator login.

The next class is Register.java

This code represents a graphical user interface (GUI) for user registration in a Metro Ticketing System. It allows users to enter their information, including username, email, password, security question, and security answer.

The code begins with importing necessary packages:

javax.swing.\* provides classes for creating GUI components.

java.awt.\* provides classes for basic GUI functionality.

java.awt.event.\* provides classes for handling GUI events.

java.io.BufferedReader and java.io.FileReader are used for reading files.

java.io.FileWriter is used for writing to files.

java.io.IOException handles input/output exceptions.

java.util.Arrays provides utility methods for arrays.

The Register class is declared, which extends the JFrame class and implements the ActionListener interface. This means that the Register class can handle events generated by GUI components.

Inside the Register class, various GUI components and utility variables are declared as private member variables:

Labels: headingLabel, userLabel, emailLabel, passwordLabel, confirmPasswordLabel, securityQuestionLabel, securityAnswerLabel

Buttons: saveButton, cancelButton

Text fields: usernameField, emailField, securityQuestionField, securityAnswerField

Password fields: passwordField, confirmPasswordField

The Register class contains a constructor Register(), which initializes the GUI components and sets up the window:

The background color of the window is set to white using getContentPane().setBackground(Color.WHITE).

The layout manager is set to null using setLayout(null), which means components will be manually positioned.

The size of the window is set to 600x500 pixels using setSize(600, 500).

The default close operation is set to JFrame.EXIT\_ON\_CLOSE, which exits the application when the window is closed.

The window is centered on the screen using setLocationRelativeTo(null).

Inside the constructor, the GUI components are initialized and added to the window:

Labels and text fields are created and positioned accordingly.

Buttons are created, styled, and positioned accordingly. Event listeners are attached to the buttons using addActionListener(this), indicating that the Register class will handle the button click events.

The revalidate() and repaint() methods are called to update the GUI and make it visible using setVisible(true).

The writeToFile method is a utility function that writes user registration information to a file named "userInfo.txt":

It generates a unique card number using generateUniqueCardNumber() method.

It appends the user information, including username, password, email, card number, initial balance, security question, and security answer, to the file.

The generateUniqueCardNumber method generates a random 9-digit card number and checks if it already exists in the file:

It generates a random card number using Math.random() and some arithmetic operations.

It reads the file "userInfo.txt" line by line and checks if the generated card number already exists.

If the card number exists, it generates a new one recursively by calling itself.

If the card number doesn't exist, it returns the generated card number.

The actionPerformed method is implemented from the ActionListener interface, which handles button click events:

The method checks which button generated the event using ae.getSource().

If the cancelButton is clicked, the current window is disposed of (closed) using dispose(), and a new Home object is created, returning to the home screen.

If the saveButton is clicked, the method retrieves the entered information from the text fields.

It performs validation checks on the entered information, such as empty fields, password length, email format, and matching passwords.

If the validation checks pass, a success message is displayed, the information is written to the file using writeToFile method, and a new Login object is created, likely redirecting the user to the login screen.

The isValidEmail method checks if an email address is in a valid format using regular expressions.

The showSuccessMessage method creates a new window to display a success message after successful registration.

The showErrorDialog method displays an error message dialog with a specific error message.

The main method creates an instance of the Register class, which initializes and displays the registration GUI window.

Another class is Login.java

This code represents the login functionality for the Metro Ticketing System. It provides a graphical user interface (GUI) for users to enter their credentials and login to the system. Let's go through the code step by step:

The code is in the package metro.home.

The required packages are imported:

metro.user.UserHome is imported, indicating that this class interacts with the UserHome class.

javax.swing.\* provides classes for creating GUI components.

java.awt.\* provides classes for basic GUI functionality.

java.awt.event.\* provides classes for handling GUI events.

java.io.BufferedReader and java.io.FileReader are used for reading files.

java.io.IOException handles input/output exceptions.

java.util.Arrays provides utility methods for arrays.

The Login class is declared, which extends the JFrame class and implements the ActionListener interface. This means that the Login class can handle events generated by GUI components.

Inside the Login class, various GUI components and utility variables are declared as private member variables:

Buttons: cancelButton, nextButton, forgetPasswordButton

Text fields: usernameField

Password field: passwordField

Labels: headingLabel, userLabel, passwordLabel

signUpFilePath stores the path of the file containing user registration information.

The Login class contains a constructor Login(), which initializes the GUI components and sets up the login window:

The background color of the window is set to white using getContentPane().setBackground(Color.WHITE).

The layout manager is set to null using setLayout(null), which means components will be manually positioned.

The size of the window is set to 600x400 pixels using setSize(600, 400).

The default close operation is set to JFrame.EXIT\_ON\_CLOSE, which exits the application when the window is closed.

The window is centered on the screen using setLocationRelativeTo(null).

Inside the constructor, the GUI components are initialized and added to the window:

Labels, text fields, and buttons are created and positioned accordingly.

Event listeners are attached to the buttons using addActionListener(this), indicating that the Login class will handle the button click events.

The revalidate() and repaint() methods are called to update the GUI and make it visible using setVisible(true).

The verifyCredentials method checks if the entered username and password match the saved credentials in the file:

It reads the file "userInfo.txt" line by line.

For each line starting with "username=", it extracts the saved username and password.

If the entered username and password match the saved credentials, it returns true.

If the file is not found or an I/O exception occurs, it prints the stack trace and returns false.

The actionPerformed method is implemented from the ActionListener interface, which handles button click events:

Here is the code:

private boolean verifyCredentials(String enteredUsername, String enteredPassword) {

try {

BufferedReader reader = new BufferedReader(new FileReader(signUpFilePath));

String line;

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

if (line.startsWith("username=")) {

String savedUsername = line.substring(9);

String passwordLine = reader.readLine();

String savedPassword = passwordLine.substring(9);

if (enteredUsername.equals(savedUsername) && enteredPassword.equals(savedPassword)) {

reader.close();

return true;

}

}

}

reader.close();

} catch (IOException e) {

e.printStackTrace();

}

return false;

}

If the cancelButton is clicked, the current window is disposed of (closed), and a new Home object is created, returning to the home screen.

If the nextButton is clicked, the method retrieves the entered username and password.

It verifies the credentials using the verifyCredentials method.

If the credentials are valid, the current window is disposed of, and a new UserHome object is created, likely redirecting the user to the user home screen.

If the credentials are invalid, an error message dialog is displayed using JOptionPane.showMessageDialog.

If the forgetPasswordButton is clicked, the current window is disposed of, and a new Home object is created, returning to the home screen.

The main method creates an instance of the Login class, which initializes and displays the login GUI window.

The next class is AdminLogin.java

This code represents the admin login functionality for the Metro Ticketing System. It provides a graphical user interface (GUI) for the admin to enter their username and password and login to the system.

The code is in the package metro.home.

The required packages are imported:

javax.swing.\* provides classes for creating GUI components.

metro.admin.AdminHome is imported, indicating that this class interacts with the AdminHome class.

java.awt.\* provides classes for basic GUI functionality.

java.awt.event.\* provides classes for handling GUI events.

The AdminLogin class is declared, which extends the JFrame class and implements the ActionListener interface. This means that the AdminLogin class can handle events generated by GUI components.

Inside the AdminLogin class, various GUI components are declared as private member variables:

Buttons: cancelButton, nextButton

Text fields: usernameField

Password field: passwordField

Labels: headingLabel, userLabel, passwordLabel

The AdminLogin class contains a constructor AdminLogin(), which initializes the GUI components and sets up the admin login window:

The background color of the window is set to white using getContentPane().setBackground(Color.WHITE).

The layout manager is set to null using setLayout(null), which means components will be manually positioned.

The size of the window is set to 600x400 pixels using setSize(600, 400).

The default close operation is set to JFrame.EXIT\_ON\_CLOSE, which exits the application when the window is closed.

The window is centered on the screen using setLocationRelativeTo(null).

Inside the constructor, the GUI components are initialized and added to the window:

Labels, text fields, and buttons are created and positioned accordingly.

Event listeners are attached to the buttons using addActionListener(this), indicating that the AdminLogin class will handle the button click events.

The revalidate() and repaint() methods are called to update the GUI and make it visible using setVisible(true).

The actionPerformed method is implemented from the ActionListener interface, which handles button click events:

If the cancelButton is clicked, the current window is disposed of (closed), and a new Home object is created, returning to the home screen.

If the nextButton is clicked, the method retrieves the entered username and password.

It compares the entered username and password with the admin credentials ("admin" for both username and password).

If the credentials are valid, the current window is disposed of, and a new AdminHome object is created, redirecting the admin to the admin home screen.

If the credentials are invalid, an error message dialog is displayed using JOptionPane.showMessageDialog, and the entered username and password fields are cleared.

The main method creates an instance of the AdminLogin class, which initializes and displays the admin login GUI window.

Another class is AdminHome.java

This code represents the admin home screen for the Metro Ticketing System. It provides a graphical user interface (GUI) for the admin with various options.

The code is in the package metro.admin.

The required packages are imported:

metro.home.Home is imported, indicating that this class interacts with the Home class.

javax.swing.\* provides classes for creating GUI components.

java.awt.\* provides classes for basic GUI functionality.

java.awt.event.\* provides classes for handling GUI events.

The AdminHome class is declared, which extends the JFrame class and implements the ActionListener interface. This means that the AdminHome class can handle events generated by GUI components.

Inside the AdminHome class, various GUI components are declared as private member variables:

Buttons: rechargeButton, editNoticeBoardButton, backButton

Note: There is a commented out button, timeTableButton, which is not used in the current code.

The AdminHome class contains a constructor AdminHome(), which initializes the GUI components and sets up the admin home screen:

The background color of the window is set to white using getContentPane().setBackground(Color.WHITE).

The layout manager is set to null using setLayout(null), which means components will be manually positioned.

The size of the window is set to 600x400 pixels using setSize(600, 400).

The default close operation is set to JFrame.EXIT\_ON\_CLOSE, which exits the application when the window is closed.

The window is centered on the screen using setLocationRelativeTo(null).

Inside the constructor, the GUI components are initialized and added to the window:

Buttons are created and positioned accordingly.

Event listeners are attached to the buttons using addActionListener(this), indicating that the AdminHome class will handle the button click events.

The revalidate() and repaint() methods are called to update the GUI and make it visible using setVisible(true).

The actionPerformed method is implemented from the ActionListener interface, which handles button click events:

public void actionPerformed(ActionEvent ae) {

if (ae.getSource() == cancelButton) {

dispose();

new Home();

} else if (ae.getSource() == nextButton) {

String enteredUsername = usernameField.getText();

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

if (enteredUsername.equals("admin") && enteredPassword.equals("admin")) {

dispose();

new AdminHome();

} else {

JOptionPane.showMessageDialog(this, "Invalid usernameField or passwordField. Please try again.",

"Authentication Failed", JOptionPane.ERROR\_MESSAGE);

usernameField.setText("");

passwordField.setText("");

}

}

}

If the rechargeButton is clicked, the current window is disposed of (closed), and a new RechargeMoney object is created, redirecting the admin to the recharge money screen.

If the editNoticeBoardButton is clicked, the current window is disposed of, and a new EditNoticeBoard object is created, redirecting the admin to the notice board editing screen.

If the backButton is clicked, the current window is disposed of, and a new Home object is created, returning to the home screen.

The main method creates an instance of the AdminHome class, which initializes and displays the admin home GUI screen.

Overall, this code provides an admin home screen for the Metro Ticketing System, allowing the admin to perform actions such as recharging money and editing the notice board. The buttons on the home screen are associated with specific actions, and clicking on them triggers the corresponding event handling code.