

## COMPUTER SCIENCE DEPARTMENT

**CS0053**

(PROGRAMMING TOOLS AND TECHNIQUES)

EXERCISE

3

GUI-Based Login Module

|  |  |
| --- | --- |
| **Name: John Paul L. Besagas** | **Professor: Dr. Beau Habal** |
| **Date Performed : 10/1/2024** | **Date Submitted:10/2/2024** |

1. **OBJECTIVES**

At the end of this exercise, students must be able to:

Cognitive

1. Understand the topics they have learned from lesson 3.

.

Psychomotor:

1. Apply functions and recursive functions.
2. Create a GUI-based Java program using functions and recursive functions.

Affective

1. Appreciate the concept behind this exercise.
2. **BACKGROUND INFORMATION**

In order to accomplish this exercise, the student must have a clear understanding of the following topics:

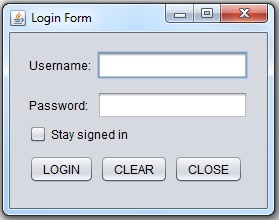
* Java method signatures
* Recursion
* Import packages or libraries

1. **LABORATORY PROCEDURE**

1. Create a new program.

Program Name: Login.java

2. Design your layout as shown below



3. Requirements

- Username format: Initial of First Name, Initial of Middle Name and complete Last Name ex. ADAquino

* The program will force the user to input correct value.
* The program will display notification for successful and unsuccessful attempt.
* The program must apply the password criteria:

- must be at least 8 characters

- must have at least one char both lower and uppercase, one number and one special character.

* Apply coding conventions.
* The program must be free from any errors.

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.util.prefs.Preferences;  public class Login {      JFrame frameLogin;      JTextField fieldDisplayUsername;      JPasswordField fieldDisplayPassword;      JCheckBox checkSignedIn;      JButton buttonLogin, buttonClear, buttonCloseButton, buttonLogout;      Font fontDisplay = new Font("Arial", Font.PLAIN, 16);      Preferences preferences = Preferences.userRoot().node(this.getClass().getName());      public Login() {          // user is already logged in          if (isUserLoggedIn()) {              frameLogin = new JFrame("Welcome");              frameLogin.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);              frameLogin.setSize(400, 200);              frameLogin.getContentPane().setBackground(new Color(60, 63, 65));              frameLogin.setLayout(new BorderLayout());              JLabel welcome = new JLabel("Welcome back, " + preferences.get("username", "") + "!", JLabel.CENTER);              welcome.setFont(new Font("Arial", Font.BOLD, 16));              welcome.setForeground(Color.WHITE);              buttonLogout = new JButton("LOGOUT");              buttonLogout.setPreferredSize(new Dimension(110, 30));              buttonLogout.setBackground(new Color(220, 20, 60));              buttonLogout.setForeground(Color.WHITE);              buttonLogout.setFocusPainted(false);              buttonLogout.addActionListener(new ActionListener() {                  public void actionPerformed(ActionEvent e) {                      clearLogin();                      JOptionPane.showMessageDialog(frameLogin, "Logged out.");                      frameLogin.dispose();                      new Login();                  }              });              JPanel buttonPanel = new JPanel();              buttonPanel.setBackground(new Color(60, 63, 65));              buttonPanel.add(buttonLogout);              frameLogin.add(welcome, BorderLayout.CENTER);              frameLogin.add(buttonPanel, BorderLayout.SOUTH);              frameLogin.setLocationRelativeTo(null);              frameLogin.setVisible(true);              return;          }          frameLogin = new JFrame("Login");          frameLogin.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);          frameLogin.setSize(400, 300);          frameLogin.getContentPane().setBackground(new Color(60, 63, 65));          frameLogin.setLayout(new BorderLayout());          JPanel mainPanel = new JPanel();          mainPanel.setBackground(new Color(60, 63, 65));          mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y\_AXIS));          fieldDisplayUsername = new JTextField();          fieldDisplayPassword = new JPasswordField();          checkSignedIn = new JCheckBox("Stay signed in");          checkSignedIn.setForeground(Color.WHITE);          checkSignedIn.setBackground(new Color(60, 63, 65));          JPanel formPanel = new JPanel(new GridLayout(3, 2, 10, 10));          formPanel.setBackground(new Color(60, 63, 65));          formPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));          JLabel userLabel = new JLabel("Username:");          userLabel.setForeground(Color.WHITE);          JLabel passLabel = new JLabel("Password:");          passLabel.setForeground(Color.WHITE);          formPanel.add(userLabel);          formPanel.add(fieldDisplayUsername);          formPanel.add(passLabel);          formPanel.add(fieldDisplayPassword);          formPanel.add(checkSignedIn);          JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));          buttonPanel.setBackground(new Color(60, 63, 65));          buttonLogin = new JButton("LOGIN");          buttonLogin.setPreferredSize(new Dimension(110, 30));          buttonLogin.setBackground(new Color(30, 144, 255));          buttonLogin.setForeground(Color.WHITE);          buttonLogin.setFocusPainted(false);          buttonClear = new JButton("CLEAR");          buttonClear.setPreferredSize(new Dimension(110, 30));          buttonClear.setBackground(new Color(30, 144, 255));          buttonClear.setForeground(Color.WHITE);          buttonClear.setFocusPainted(false);          buttonCloseButton = new JButton("CLOSE");          buttonCloseButton.setPreferredSize(new Dimension(110, 30));          buttonCloseButton.setBackground(new Color(220, 20, 60));          buttonCloseButton.setForeground(Color.WHITE);          buttonCloseButton.setFocusPainted(false);          buttonPanel.add(buttonLogin);          buttonPanel.add(buttonClear);          buttonPanel.add(buttonCloseButton);          buttonLogin.addActionListener(new ActionListener() {              public void actionPerformed(ActionEvent e) {                  String username = fieldDisplayUsername.getText();                  String password = new String(fieldDisplayPassword.getPassword());                  if (checkUser(username) && checkPassword(password)) {                      JOptionPane.showMessageDialog(frameLogin, "Login successful");                      if (checkSignedIn.isSelected()) {                          checkedLogin(username);                      }                      frameLogin.dispose();                  } else if (!checkUser(username)) {                      JOptionPane.showMessageDialog(frameLogin,                              "Invalid username format. Username format: Initial of First Name, Initial of Middle Name and complete Last Name ex. ADAquino");                  } else if (!checkPassword(password)) {                      JOptionPane.showMessageDialog(frameLogin,                              "Invalid password. The password must have at least 8 characters, must have at least one char both lower and uppercase, one number and one special character.");                  }              }          });          buttonClear.addActionListener(new ActionListener() {              public void actionPerformed(ActionEvent e) {                  fieldDisplayUsername.setText("");                  fieldDisplayPassword.setText("");                  checkSignedIn.setSelected(false);                  clearLogin();              }          });          buttonCloseButton.addActionListener(new ActionListener() {              public void actionPerformed(ActionEvent e) {                  frameLogin.dispose();              }          });          mainPanel.add(formPanel);          mainPanel.add(buttonPanel);          frameLogin.add(mainPanel, BorderLayout.CENTER);          frameLogin.setLocationRelativeTo(null);          frameLogin.setVisible(true);      }      // Validate username      public boolean checkUser(String username) {          return username.matches("[A-Z]{2}[A-Z][a-zA-Z]+");      }      // Validate password      public boolean checkPassword(String password) {          return password.length() >= 8 &&                  password.matches(".\*[a-z].\*") &&                  password.matches(".\*[A-Z].\*") &&                  password.matches(".\*[0-9].\*") &&                  password.matches(".\*[!@#$%^&\*(),.?\":{}|<>].\*");      }      // Preferences      public void checkedLogin(String username) {          preferences.putBoolean("loggedIn", true);          preferences.put("username", username);      }      // Check if the user is logged in      public boolean isUserLoggedIn() {          return preferences.getBoolean("loggedIn", false);      }      // Clear login      public void clearLogin() {          preferences.putBoolean("loggedIn", false);          preferences.remove("username");      }      public static void main(String[] args) {          new Login();      }  } |

1. **QUESTION AND ANSWER**
2. What are the methods you used in the program?

**In my program, I used several methods to handle different functionalities. One key method is checkUser, which validates the username format. I also used checkPassword to ensure the password meets specific criteria. Additionally, I used checkedLogin to store login state and the username in the preferences. To check if the user is logged in when the program starts, I used the isUserLoggedIn method. Lastly, the clearLogin method logs the user out and removes their information from the preferences.**

1. When you use standard methods in OOP? Cite an example and explain.

**In my program, I apply standard OOP methods to encapsulate reusable functionality in my programs. For example, the checkUser method validates username formats, eliminating the need to rewrite validation logic. This aligns with the core OOP principle of encapsulation, simplifying future code maintenance and modifications.**

1. What are the methods you use to compare literal value?

**In my program, I used String.matches to compare input values of usernames and passwords against specific regular expressions. This ensures that the username follows a required pattern and that the password includes various character types that followed and meets the criteria.**

1. **QUESTION AND ANSWER**

|  |  |
| --- | --- |
| Department | Computer Science |
| Subject Code | CSSSPEC2 |
| Description | Programming Tools and Techniques |
| Term/Academic Year | 1st Term SY 2016-2017 |

|  |  |
| --- | --- |
| Topic | Functions and Recursive Functions |
| Lab Activity No | 3 |
| Lab Activity | **GUI-Based Login Module** |
| CLO | **1, 2** |

**Note: The following rubrics/metrics will be used to grade students’ output in the lab exercise 3.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Exceptional | Acceptable | Amateur | Unsatisfactory |
| Specifications  (40%) | The program works and meets all of the specifications. (40) | The program works and produces the correct results and displays them correctly. It also meets most of the other specifications. (35-39) | The program produces correct results but does not display them correctly. (30-34) | The program is producing incorrect results. (20-29) |
| Design  (15 %) | The design is exceptionally attractive. Program is "user-friendly" with informative and consistent prompts and messages. (15) | The design is fairly attractive. Program is "user-friendly" with informative and consistent prompts and messages. (13-14) | The design is fairly attractive. Program is not "user-friendly" but still provide informative and consistent prompts and messages. (10-12) | The design is unattractive and not user-friendly (8-9) |
| Efficiency (20%) | The code is extremely efficient without sacrificing readability and understanding. (20) | The code is fairly efficient without sacrificing readability and understanding. (17-19) | The code is brute force and unnecessarily long. (14-16) | The code is huge and appears to be patched together. (10-13) |
| Readability  (10 %) | The code is exceptionally well organized and very easy to follow. (10) | The code is fairly easy to read. (8-9) | The code is readable only by someone who knows what it is supposed to be doing. (6-7) | The code is poorly organized and very difficult to read. (4-5) |
| Delivery  (15%) | The program was delivered on time. (15) | The program was delivered within a day of the due date. (13-14) | The code was within 2 days of the due date. (10-12) | The code was within a week of the due date. (8-9) |
| Total: 100% |  |  |  |  |