Facharbeit

Passwortmanager mit lokaler Datenbank

Verfasser: Sami Zerfaoui

Fach: Informatik

Lehrer: Herr Weuffen

Java: 19.0.2

Schule: Kaiserin-Theophanu-Schule

Schuljahr: Q1

Einleitung

Ein Passwortmanager ist eine Software, die es Benutzern ermöglicht, ihre Passwörter sicher zu verwalten. In der heutigen digitalen Welt, in der fast jeder eine Vielzahl von Konten hat, ist es wichtig, starke und einzigartige Passwörter zu verwenden. Ein Passwortmanager kann dabei helfen, indem er dem Benutzer eine sichere Möglichkeit bietet, seine Passwörter zu speichern und zu organisieren.

In dieser Facharbeit wird ein Passwortmanager in Java und MySQL entwickelt. Java ist eine objektorientierte Programmiersprache, die für ihre Portabilität und ihre Fähigkeit, auf verschiedenen Betriebssystemen zu laufen, bekannt ist. MySQL ist ein relationales Datenbankverwaltungssystem, das für seine Zuverlässigkeit und Skalierbarkeit geschätzt wird.

Zunächst wird in dieser Facharbeit eine Einführung in die Konzepte von Java und MySQL gegeben. Dann wird der Prozess der Entwicklung des Passwortmanagers in Java und MySQL Schritt für Schritt erklärt. Dazu gehört die Erstellung der Datenbankstruktur, die Implementierung der Benutzeroberfläche und die Implementierung der Funktionen des Passwortmanagers wie das Speichern und Abrufen von Passwörtern.

Schließlich wird die Funktionalität des Passwortmanagers in Java und MySQL diskutiert und getestet sowie demonstriert. Dabei werden mögliche Erweiterungen und Verbesserungen des Passwortmanagers sowie mögliche Sicherheitsbedenken behandelt.

Java und mySql

Java ist eine objektorientierte Programmiersprache, die es Entwicklern ermöglicht, Anwendungen und Softwarepakete zu erstellen. MySQL ist eine relationale Datenbank, die verwendet wird, um große Datenmengen zu speichern

und zu verwalten. Zusammen können sie leistungsstarke Anwendungen erstellen.

Um eine Anwendung zu erstellen, benötigt man die Connector/J-Bibliothek von MySQL, die es dem Java-Programm ermöglicht, auf die MySQL-Datenbank zuzugreifen. Außerdem den DriverManager, dieser ist in Java ist ein Mechanismus, der eine Liste von Datenbanktreibern verwaltet und den geeigneten Treiber auswählt, um eine Verbindung zur Datenbank herzustellen.

DriverManager

In Java können wir den DriverManager in Ihrem Programm importieren, indem Sie die folgende Zeile am Anfang Ihres Codes, vor der Klassendeklaration, hinzufügen:

```
import java.sql.DriverManager;
```

Dies importiert die DriverManager-Klasse aus dem Java.sql-Paket, das für die Verwaltung von JDBC-Treibern und die Verbindungsherstellung mit der Datenbank verantwortlich ist. Zu beachten ist jedoch, dass der DriverManager allein nicht ausreicht, um eine Verbindung zur Datenbank herzustellen. Wir benötigen auch den passenden JDBC-Treiber für Ihre spezifische Datenbank, den Sie in Ihrem Projekt einbinden müssen.

Connector

Java Connector JAR ist eine Java-Bibliothek, die als JDBC-Treiber für MySQL-Datenbanken dient. Sie ermöglicht es Java-Anwendungen, eine Verbindung zu MySQL-Datenbanken herzustellen, Daten abzurufen und zu aktualisieren.

localhost

Mit Java und MySQL können Entwickler leistungsfähige Anwendungen erstellen, die Daten speichern, verwalten und anzeigen können. In diesem

Beispiel wird eine lokale Datenbank (localhost) benutzt, die normalerweise für Testzwecke und Debugging benutzt da es eine “Sichere bzw. Isolierte” Umgebung ist, somit kann man z.B. sichergehen das Fehler nicht vom Netzwerk ausgehen.

Datenbank

Es ist möglich, eine vorgefertigte Tabelle zur Erstellung der Datenbank zu verwenden, indem man eine .sq.zip Datei in das XAMPP phpMyAdmin Panel importiert. Dazu muss man zunächst das Panel über die URL http://localhost/dashboard/ aufrufen und eine neue Datenbank erstellen. Anschließend kann man die entsprechende .sq.zip Datei importieren, um die Tabelle zu erstellen. Es ist wichtig zu beachten, dass die erstellte Datenbank den Namen "PasswordManager" tragen sollte oder die db.java klasse auf den Namen modifiziert werden sollte.

Passwortmanager

How to setup

Anforderungen:

XAMPP – Programmpaket

Java SDK – 8.0+

mysql-connector-java-8.0.20.jar - mySql connector

Passwordmanager.sql.gz - Daten Tabelle

XAMPP:

Install Apache and mySql and Start the Server.
Username: root, Password = None

Connector:

IntellijIdea:
Project Structure -> Modules -> Dependencies -> Add

Ich benutzte nicht den Java Editor. Jedoch ist der connector eine Abhängigkeit.

Erklärung des Programms

UI

Für die UI wurde der netBeans Idea benutzt, der code für das UI ist automatisch generiert, wie beim Java Editor, also werde ich dort nicht ins Detail gehen.

db.java

```
// Definieren des Pakets, in dem sich die Klasse befindet

package database;

import java.sql.*; // Der "*" Importiert alle Klassen aus dem Package java.sql

/* Importieren von Klassen, die für die Datenbankkonnektivität erforderlich sind

import java.sql.Connection/DriverManager

*/



// Definieren der Klasse db ( database )

public class db {



    // Eine statische Methode, um eine Verbindung zur Datenbank herzustellen
```

```
    public static Connection conn(){



        // Variablen für Datenbank-URL, Benutzername und Passwort
definieren
        String url, user, pass; // nächste Seite
        url = "jdbc:mysql://localhost:3306/passwordManager"; // URL
der MySQL-Datenbank
        user = "root"; // Benutzername für den Datenbankzugriff
        pass = ""; // Passwort für den Datenbankzugriff



        Connection con = null; // Variablen für die Verbindung zur
Datenbank definieren



        try {
            //    Laden    des    MySQL-Datenbanktreibers    mit
Class.forName()
            Class.forName("com.mysql.cj.jdbc.Driver");



            //  Herstellen  der  Verbindung  zur  Datenbank  mit
DriverManager.getConnection()
            con = DriverManager.getConnection(url,user,pass);
            System.out.println("Verbunden");   //   Ausgabe   einer
Bestätigung, wenn die Verbindung erfolgreich hergestellt wurde
```

```
        } catch (Exception e) {
            // Fehlermeldung ausgeben, wenn die Verbindung zur
Datenbank nicht geklappt hat
            System.out.println(e);
        }


        // Rückgabe der Verbindung zur Datenbank
        return con;
    }
}
```

Erklärung

Die Klasse db enthält eine statische Methode conn, die eine Verbindung zur MySQL-Datenbank herstellt. Die Verbindungsdaten wie URL, Benutzername und Passwort sind im Code festgelegt. Wenn die Verbindung erfolgreich ist, wird eine Bestätigungsmeldung ausgegeben, andernfalls eine Fehlermeldung. Die Methode gibt die Verbindung als Connection-Objekt zurück, genaueres ist im Code so gut wie möglich kommentiert.

Info: db.java wird in einem eigenen Order gespeichert und anstatt vom Erben, das wir hatten, importieren Klassen diese klasse oben im code mit database.db :

(import Ordnername.Klassenname)

AESencryption.java

Theorie

AES (Advanced Encryption Standard) ist ein Verschlüsselungsalgorithmus, der zur sicheren Übertragung von Informationen im Internet verwendet wird. **AES** wurde 2001 von der National Security Agency (NSA) entwickelt und 2002 vom National Institute of Standards and Technology (NIST) als Standard akzeptiert.

AES (Advanced Encryption Standard) ist ein symmetrischer Verschlüsselungsalgorithmus, das bedeutet, dass der gleiche Schlüssel sowohl für die Verschlüsselung als auch für die Entschlüsselung von Daten verwendet wird. AES arbeitet mit Blockchiffren, was bedeutet, dass Daten in Blöcke

gleicher Größe aufgeteilt werden und jeder Block einzeln verschlüsselt wird. Die Blockgröße bei **AES** beträgt 128 Bit.

Der **AES-Algorithmus** unterstützt Schlüssellängen von 128, 192 und 256 Bit. Die Anzahl der benötigten Runden hängt von der gewählten Schlüssellänge ab. Wenn eine 128-Bit-Schlüssellänge verwendet wird, führt der Algorithmus 10 Runden von Operationen durch. Bei einer 192-Bit-Schlüssellänge sind es 12 Runden und bei einer 256-Bit-Schlüssellänge sind es 14 Runden.

Während des Verschlüsselungsprozesses verwendet **AES** verschiedene Transformationen wie Substitution, Permutation, Xor und Shift-Operationen. Diese Transformationen werden in jeder Runde wiederholt, um den Text weiter zu verschlüsseln. Dadurch wird es für Angreifer schwierig, den Klartext aus dem verschlüsselten Text zu extrahieren, da mehrere Transformationen angewendet werden.

Ein Byte besteht aus 8 Bits, was bedeutet, dass ein Byte 2^8 (256) mögliche Zustände speichern kann, in dieser Klasse wurde 16 Bytes für den Schlüssel und IV benutzt.

Vorteile

Die Vorteile von AES sind:

1. Sicherheit: AES ist ein sehr sicherer Algorithmus, der zur Verschlüsselung von sensiblen Informationen wie Finanzdaten oder persönlichen Informationen verwendet wird. Es ist bisher kein erfolgreicher Angriff auf AES bekannt.
2. Effizienz: AES ist sehr effizient und schnell, was bedeutet, dass große Datenmengen schnell verschlüsselt werden können.
3. Standardisierung: AES ist ein internationaler Standard, der von vielen Regierungen und Unternehmen auf der ganzen Welt verwendet wird. Dies bedeutet, dass Informationen, die mit AES verschlüsselt sind, von vielen verschiedenen Anwendungen und Systemen gelesen werden können.

4. Flexibilität: AES unterstützt verschiedene Schlüssellängen und Blockgrößen, was es flexibel macht, und auf verschiedene Anwendungen angepasst werden kann.

Insgesamt ist AES ein sehr sicherer und effizienter Verschlüsselungsalgorithmus, der weltweit eingesetzt wird.

Klasse

```
package passwordmanager; // Paketname, in dem sich die Klasse befindet

import javax.crypto.Cipher; // Import für die Cipher-Klasse für die Verschlüsselung

import javax.crypto.spec.IvParameterSpec; // Import für die IvParameterSpec-Klasse für den Initialisierungsvektor

import javax.crypto.spec.SecretKeySpec; // Import für die SecretKeySpec-Klasse für den Schlüssel

import java.nio.charset.StandardCharsets; // Import für den Zeichensatz für die Kodierung von Strings

import java.util.Base64; // Import für die Base64-Kodierung für das Verschlüsselte



public class AESencryption { // Klassendefinition für die Verschlüsselung



    private static final String ALGORITHM = "AES/CBC/PKCS5PADDING"; // Algorithmus für die Verschlüsselung

    // Geheime Schlüssel und IV-Parameter (Initialisierungsvektor) für die Verschlüsselung

    private static final String SECRET_KEY = "U=Agp7ILG6R}yTGL";

    private static final String IV_PARAMETER = "D87JU3F^x|0~9#&B";



    public static String encrypt(String input) throws Exception { // Methode für die Verschlüsselung
```

```
        Cipher cipher = Cipher.getInstance(ALGORITHM); // Erstelle
ein Cipher-Objekt für die Verschlüsselung

        SecretKeySpec             key             =             new
SecretKeySpec(SECRET_KEY.getBytes(StandardCharsets.UTF_8), "AES");
// Erstelle einen Schlüssel für die Verschlüsselung

        IvParameterSpec             iv             =             new
IvParameterSpec(IV_PARAMETER.getBytes(StandardCharsets.UTF_8)); //
Erstelle einen Initialisierungsvektor

        cipher.init(Cipher.ENCRYPT_MODE, key, iv); // Initialisiere
den Cipher-Modus mit Schlüssel und IV

        byte[]  encrypted  =  cipher.doFinal(input.getBytes());  //
Verschlüssele den Text

        return  Base64.getEncoder().encodeToString(encrypted);  //
Gib das verschlüsselte Ergebnis als Base64-kodierten String zurück

    }



    public static String decrypt(String input) throws Exception {
// Methode für die Entschlüsselung



        Cipher cipher = Cipher.getInstance(ALGORITHM); // Erstelle
ein Cipher-Objekt für die Entschlüsselung

        SecretKeySpec             key             =             new
SecretKeySpec(SECRET_KEY.getBytes(StandardCharsets.UTF_8), "AES");
// Erstelle einen Schlüssel für die Entschlüsselung

        IvParameterSpec             iv             =             new
IvParameterSpec(IV_PARAMETER.getBytes(StandardCharsets.UTF_8)); //
Erstelle einen Initialisierungsvektor
```

```
        cipher.init(Cipher.DECRYPT_MODE, key, iv); // Initialisiere
den Cipher-Modus mit Schlüssel und IV

        byte[]                      decrypted                      =
cipher.doFinal(Base64.getDecoder().decode(input));              //
Entschlüssele den Base64-kodierten String

        return  new  String(decrypted);  //  Gib  das  entschlüsselte
Ergebnis als String zurück

    }

}
```

Erklärung, “in a nutshell”

Dieser Code definiert eine Klasse namens **AESencryption**, die Methoden zum Verschlüsseln und Entschlüsseln von Text mithilfe des AES-Verschlüsselungsalgorithmus bereitstellt. Die Klasse importiert mehrere Klassen aus den Paketen **javax.crypto** und **java.util** für die Implementierung von Verschlüsselung und Kodierung.

Die Klasse **AESencryption** definiert zwei Methoden:

1. **encrypt**: Diese Methode nimmt einen Eingabestring entgegen und gibt die verschlüsselte Version als Base64-kodierten String zurück. Die Methode erstellt zuerst ein **Cipher**-Objekt für die Verschlüsselung mithilfe des AES/CBC/PKCS5PADDING Algorithmus. Es werden auch ein geheimer Schlüssel und ein Initialisierungsvektor für die Verschlüsselung verwendet. Der Eingabestring wird dann in Bytes konvertiert und mit dem **Cipher**-Objekt verschlüsselt. Das Ergebnis wird als Base64-kodierter String zurückgegeben.
2. **decrypt**: Diese Methode nimmt einen Base64-kodierten String entgegen und gibt die entschlüsselte Version als String zurück. Die Methode erstellt zuerst ein **Cipher**-Objekt für die Entschlüsselung mithilfe des AES/CBC/PKCS5PADDING Algorithmus. Es werden auch ein geheimer Schlüssel und ein Initialisierungsvektor für die Entschlüsselung verwendet. Der Base64-kodierte String wird dann in Bytes konvertiert und mit dem **Cipher**-Objekt entschlüsselt. Das Ergebnis wird als String zurückgegeben.

Kryptographie ist ein sehr komplexes Thema und würde eine

Eigene Facharbeit sein, daher habe ich es hier sehr einfach und kurz erklärt.

Weiteres ist detailliert in der Klasse kommentiert.

Login.java

in der Klasse Login.java ist autogenerierter Code enthalten, der von der Netbeans IDE für die Benutzeroberfläche generiert wurde. Aus diesem Grund werde ich diesen Code entfernen.

```
package passwordmanager;



import javax.swing.JOptionPane;
import java.sql.*;



import database.db;



import java.awt.Color;
import java.net.URL;
import javax.swing.ImageIcon;



public class Login extends javax.swing.JFrame {
    /**
     * Erstellt eine neue Form "Login".
     */
    Connection conn; // Deklariert eine Verbindungsvariable, um die
Verbindung hinzuzufügen
```

```
    Statement st;// Deklariert eine Statement-Variable, um die Anweisung hinzuzufügen


    public Login() {
        initComponents(); // Initialisiert die Komponenten
        conn = db.conn(); // Initialisiert die Verbindungsvariable
        st = null; // Initialisiert die Anweisungsvariable


        // Setzt ein Icon
        URL image = getClass().getResource("/source/ic.png"); // Holt das Bild
        ImageIcon img = new ImageIcon(image); // Erstellt ein ImageIcon


        this.setIconImage(img.getImage()); // Setzt das Icon


        Color col = new Color(35, 35, 35); // Erstellt eine Farbe


        getContentPane().setBackground(col);    //    Setzt    die Hintergrundfarbe
    }


    /**
```

```
     * This method is called from within the constructor to
initialize the     form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    //   <editor-fold   defaultstate="collapsed"   desc="Generated
Code">//GEN-BEGIN:initComponents



    // </editor-fold>//GEN-END:initComponents




    private void jLabel3MouseClicked(java.awt.event.MouseEvent evt)
{
        this.dispose(); //aktuelle Form schließen
        SignUp obj = new SignUp(); //neue Form erstellen
        obj.show(); //neue Form anzeigen
    }



    private                                                  void
btnLoginActionPerformed(java.awt.event.ActionEvent   evt)   {//GEN-
FIRST:event_btnLoginActionPerformed
        // TODO Füge hier deinen Code für die Verarbeitung hinzu:
        String     userName     =     txtLoginUserName.getText();
//Benutzername abrufen
```

```
        String password = txtLoginPassword.getText(); //Passwort
abrufen



        try {

            //Passwort abrufen, das zum Benutzernamen gehört, und
prüfen, ob es übereinstimmt oder nicht

            String query = "select password from users where
userName = '" + userName + "'"; //Passwort abrufen, das zum
Benutzernamen gehört

            st = conn.createStatement(); //Erstelle eine Anweisung

            ResultSet rs = st.executeQuery(query); //Führe die
Abfrage aus



            //Wenn es Daten im Resultset gibt, überprüfe, ob das
Passwort übereinstimmt oder nicht

            if (rs.next()) {

                if
(password.equals(AESencryption.decrypt(rs.getString("password"))))
{ // Wenn das Passwort übereinstimmt, dann gehe zum Passwortmanager-
Formular

                    ManageForm obj = new ManageForm(); //neue Form
erstellen

                    obj.show(); //neue Form anzeigen

                    obj.getUser(userName); //Benutzernamen abrufen

                    this.dispose(); //aktuelle Form schließen

                } else {

                    //Wenn das Passwort nicht übereinstimmt, zeige
eine Fehlermeldung an
```

```
                    JOptionPane.showMessageDialog(this,
"Benutzername oder Passwort stimmen nicht überein!", "Fehler!",
JOptionPane.ERROR_MESSAGE);
                }
            } else {
                //Wenn es keine Daten im Resultset gibt, zeige eine
Fehlermeldung an
                JOptionPane.showMessageDialog(this, "Sie haben kein
Konto!", "Fehler!", JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this,  e);  //zeige  die
Fehlermeldung an
        }
    }//GEN-LAST:event_btnLoginActionPerformed

    // Main methode
    public static void main(String args[]) {

        // set the look and feel of the form - AutoGenerated

        //</editor-fold>

        // Create and display the form
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
```

```
                new Login().setVisible(true); // Zeigt die login
form an.
            }
        });
    }


    // Variables declaration - do not modify//GEN-BEGIN:variables


    // End of variables declaration//GEN-END:variables
}
```

Erklärung

Die Klasse "Login" ist Teil eines Passwort-Managers und ist dafür zuständig, die Benutzeranmeldung durchzuführen. Sie enthält Methoden zum Abrufen von Benutzernamen und Passwort, zur Überprüfung, ob das eingegebene Passwort dem im System gespeicherten Passwort entspricht, und zur Anzeige von Fehlermeldungen, falls die Anmeldung fehlschlägt.

Die Klasse hat eine Verbindung mit der Datenbank aufgebaut und verwendet ein Statement-Objekt, um Abfragen auszuführen. Sie initialisiert auch das Icon der Anwendung und setzt die Hintergrundfarbe.

Die Main-Methode der Klasse startet die Anwendung und zeigt das Login-Formular an. Insgesamt bietet die Klasse die Start Form. hier sollte das Programm gestartet werden.

SignUp.java

```
package passwordmanager;


import database.db; //Importieren der db-Klasse aus dem database-Paket
import java.awt.Color; //Importieren der Color-Klasse aus dem java.awt-Paket
import java.awt.event.ActionEvent; //Importieren der ActionEvent-Klasse aus dem java.awt.event-Paket
import java.net.URL; //Importieren der URL-Klasse aus dem java.net-Paket
import java.sql.*; //Importieren der JDBC-Klassen aus dem java.sql-Paket
import javax.swing.ImageIcon; //Importieren der ImageIcon-Klasse aus dem javax.swing-Paket
import javax.swing.JOptionPane; //Importieren der JOptionPane-Klasse aus dem javax.swing-Paket


public class SignUp extends javax.swing.JFrame {


    Connection conn; //Deklaration einer Verbindungsvariablen, um die Verbindung hinzuzufügen
```

```
    Statement st; //Deklaration einer Statement-Variable, um die
Anweisung hinzuzufügen



    /**
     * Creates new form SignUp
     */
    public SignUp() {
        initComponents();



        conn = db.conn(); //Holen der Datenbankverbindung aus der
db-Klasse



        //Setzen eines Symbols
        URL image = getClass().getResource("/source/ic.png");
//Holen der URL für das Symbolbild
        ImageIcon img = new ImageIcon(image); //Erstellen eines
ImageIcon-Objekts aus der URL



        this.setIconImage(img.getImage()); //Setzen des Symbols für
das Fenster
        Color col = new Color(35, 35, 35); //Erstellen einer neuen
Farbe für den Hintergrund



        getContentPane().setBackground(col); //Setzen der Farbe für
den Hintergrund des Fensters
```

```
    }


    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    //   <editor-fold   defaultstate="collapsed"   desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {
    }// </editor-fold>//GEN-END:initComponents
private  void  btnSignUpActionPerformed(ActionEvent  evt)  {//GEN-
FIRST:event_btnSignUpActionPerformed
    //   TODO   Hier   wird   der   Code   zur   Verarbeitung   des
Registrierungsformulars hinzugefügt:



    //Holen der Werte aus den Textfeldern (Benutzername, Passwort,
Passwortbestätigung)
    String userName = txtSignUserName.getText();
    String password = txtSignPassword.getText();
    String rePassword = txtSignRePassword.getText();



    try {
```

```
        //Überprüfen, ob das eingegebene Passwort und die Passwortbestätigung übereinstimmen
        if (password.equals(rePassword)) {


            //Erstellen der SQL-Abfrage zum Einfügen des neuen Benutzers in die Datenbank
            String query = "insert into users values('" + userName + "','" + passwordmanager.AESencryption.encrypt(password) + "')";
            st = conn.createStatement();
            st.executeUpdate(query);
            System.out.println("done");


            //Schließen des Registrierungsfensters und Öffnen des Anmeldefensters
            this.dispose();
            Login obj = new Login();
            obj.show();


        } else {
            //Anzeigen einer Fehlermeldung, wenn das Passwort und die Passwortbestätigung nicht übereinstimmen
            JOptionPane.showMessageDialog(this, "Password is not matching!", "Error!", JOptionPane.ERROR_MESSAGE);
        }
```

```
    } catch (Exception e) {
        //Anzeigen einer Fehlermeldung, wenn bei der Verarbeitung ein Fehler auftritt
        JOptionPane.showMessageDialog(this, e);
    }

}//GEN-LAST:event_btnSignUpActionPerformed

private void jLabel6MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_jLabel6MouseClicked
    // TODO Hier wird der Code zum Schließen des Registrierungsfensters und zum Öffnen des Anmeldefensters hinzugefügt:
    this.dispose();
    Login obj = new Login();
    obj.show();

}//GEN-LAST:event_jLabel6MouseClicked

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
```

```
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Windows".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(SignUp.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(SignUp.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(SignUp.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(SignUp.class.getName()).log(jav
a.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new SignUp().setVisible(true); // Sign up form
anzeigen
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
}
```

Auch hier sind wieder bestimmte Teile entfernt worden aufgrund von Autogenerierung.

Erklärung

Dieser Code ist ein Teil eines Password Managers. Es importiert verschiedene Klassen aus verschiedenen Paketen, um bestimmte Funktionen zu implementieren.

Es gibt eine Klasse namens "SignUp", die ein Fenster für die Registrierung von Benutzern öffnet. Diese Klasse hat eine Verbindungsvariable und eine Anweisungsvariable, um eine Verbindung zur Datenbank herzustellen und Datenbankabfragen auszuführen.

Das Fenster wird mit verschiedenen Eigenschaften wie Symbolbild, Hintergrundfarbe und Schaltflächeninitialisierung konfiguriert. Wenn der Benutzer auf die Schaltfläche "SignUp" klickt, wird der eingegebene Benutzername und das Passwort aus den Textfeldern geholt und dann wird geprüft, ob das Passwort und die Passwortbestätigung übereinstimmen. Wenn ja, wird eine SQL-Abfrage zum Einfügen des neuen Benutzers in die Datenbank ausgeführt. Andernfalls wird eine Fehlermeldung angezeigt.

Wenn der Benutzer auf das Label "Anmelden" klickt, wird das Registrierungsfenster geschlossen und das Anmeldefenster geöffnet.

Der Code verwendet auch das Nimbus-Look-and-Feel, um ein besseres Aussehen zu bieten

Manageform.java

```
package passwordmanager;


import database.db;


import java.awt.Color;
import java.net.URL;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;


public class ManageForm extends javax.swing.JFrame {


    /**
     * Creates new form ManageForm
     */
    int userNameUpdateId;
    Connection conn; // Variable für die Verbindung zur Datenbank
    Statement st; // Variable für den SQL-Statement
    String user; // Variable für den Benutzernamen des
angemeldeten Benutzers
```

```
    public ManageForm() {
        initComponents();
        conn = db.conn(); // Datenbankverbindung herstellen
        st = null;

        // Programm-Icon setzen
        URL image = getClass().getResource("/source/ic.png");
        ImageIcon img = new ImageIcon(image);
        this.setIconImage(img.getImage());

        Color col = new Color(35, 35, 35);
        getContentPane().setBackground(col);

    }

    // Methode zum Abrufen des aktuellen Benutzers und Anzeigen
einer Begrüßungsnachricht
    public void getUser(String user) {
        this.user = user;
        greetinglabel.setText("Willkommen zurück, " + user + "!");
        // alle Daten in der Datenbanktabelle abrufen
        getTableDetails();
    }

    /**
     * This method is called from within the constructor to
initialize the form.
```

```
     * WARNING: Do NOT modify this code. The content of this
method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {



    }// </editor-fold>//GEN-END:initComponents



    // Methode zum Abrufen aller Details in der Datenbanktabelle
    public void getTableDetails() {



        try {
            // Abfrage, um alle Daten zu einem Benutzer abzurufen
            String query = "select * from userdata where user = '"
+ user + "'";
            st = conn.createStatement();
            ResultSet rs = st.executeQuery(query); // Abrufen
aller Daten und Zuweisen zu ResultSet



            while (rs.next()) {
                // Daten hinzufügen, bis alle Daten der
Datenbanktabelle durchlaufen wurden
                String name = rs.getString("userName");
                String pass = rs.getString("enPass");
                String des = (rs.getString("des"));
                String id = Integer.toString(rs.getInt("uid"));
```

```
                // String-Array zum Speichern der Daten in der
Tabelle
                String tbData[] = {id, des, name, pass};
                DefaultTableModel tblModel = (DefaultTableModel)
tbl.getModel();


                // Daten in der Tabelle einfügen
                tblModel.addRow(tbData);
            }


        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, e);
        }


    }


    // Methode, die ausgeführt wird, wenn der "Hinzufügen" Button
geklickt wird
/////////////////////////////////////////////////////private
void btnAddActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnAddActionPerformed
    // Hier wird der Button "btnAdd" geklickt und die folgenden
Aktionen werden ausgeführt


    // Die Werte der Textfelder (Benutzername, Passwort) werden
abgerufen
    String userName = txtUserName.getText();
    String password = txtPassword.getText();
    String des = txtdes.getText();
```

```
    // Die Variable encryptedpassword wird initialisiert
    String encryptedpassword = null;

    try {
        // MessageDigest-Instanz für MD5 wird erstellt
        MessageDigest m = MessageDigest.getInstance("MD5");

        // Die Klartext-Passwort-Bytes werden dem Digest mit der
MD5-Update()-Methode hinzugefügt
        m.update(password.getBytes());

        // Der Hash-Wert wird in Bytes konvertiert
        byte[] bytes = m.digest();

        // Der Bytes-Array hat Bytes im Dezimalformat.
Konvertieren in Hexadezimalformat
        StringBuilder s = new StringBuilder();
        for (int i = 0; i < bytes.length; i++) {
            s.append(Integer.toString((bytes[i] & 0xff) + 0x100,
16).substring(1));
        }

        // Vollständig gehashtes Passwort im Hexadezimalformat
        encryptedpassword = s.toString();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
```

```
    try {
        // Datenbank-Abfrage wird erstellt, um Benutzerdaten in
der Datenbank zu speichern
        String query = "insert into userdata(userName, password,
enPass, des, user) values('" + userName + "','" +
AESencryption.encrypt(password) + "','" + encryptedpassword +
"','" + des + "','" + user + "')";


        st = conn.createStatement();
        st.executeUpdate(query);


        // Meldung wird angezeigt, dass die Passwortdaten
erfolgreich hinzugefügt wurden
        JOptionPane.showMessageDialog(this, "Passwortdaten
erfolgreich hinzugefügt!", "Erfolg",
JOptionPane.INFORMATION_MESSAGE);


        // Die Textfelder werden nach dem Hinzufügen geleert
        txtPassword.setText("");
        txtUserName.setText("");
        txtdes.setText("");


        // Die Tabelle wird aktualisiert
        DefaultTableModel tModel = (DefaultTableModel)
tbl.getModel();
        tModel.setRowCount(0);


        // Alle Daten aus der Datenbank-Tabelle werden abgerufen
        getTableDetails();
```

```
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, e);
        System.out.println(e);
    }
}//GEN-LAST:event_btnAddActionPerformed


private void tblMouseClicked(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_tblMouseClicked
    // Die Methode wird aufgerufen, wenn eine Mausklick-Aktion auf
einer JTable (mit dem Namen `tbl`) ausgeführt wird.


    int row = tbl.getSelectedRow(); // Die Variable `row` wird mit
dem Index der ausgewählten Zeile zugewiesen.
    DefaultTableModel model = (DefaultTableModel) tbl.getModel();
// Das `DefaultTableModel`-Objekt wird erstellt und der Tabelle
zugewiesen.


    String idString = model.getValueAt(row, 0).toString(); // Der
Wert der ersten Spalte der ausgewählten Zeile wird als String
abgerufen.
    int id = Integer.parseInt(idString); // Der String-Wert der
`idString`-Variable wird in eine Ganzzahl (int) umgewandelt und
der Variable `id` zugewiesen.


    // Details der ausgewählten Zeile in die Textfelder laden.
    try {
        String query = "select * from userdata where uid = " + id;
// SQL-Abfrage, um die Datensätze der Tabelle mit der
entsprechenden `uid`-Nummer zu selektieren.
        st = conn.createStatement(); // Erstellt eine Instanz des
`Statement`-Objekts, um die SQL-Abfrage auszuführen.
```

```
        ResultSet rs = st.executeQuery(query); // Führt die
Abfrage mit der Methode `executeQuery()` aus und speichert das
Ergebnis in das ResultSet-Objekt `rs`.



        if (rs.next()) { // Wenn `rs` mindestens einen Datensatz
enthält,


txtPassword.setText(AESencryption.decrypt(rs.getString("password")
)); // wird der Wert der Spalte `password` entschlüsselt und im
Textfeld `txtPassword` angezeigt.

            txtUserName.setText(rs.getString("userName")); // Der
Wert der Spalte `userName` wird im Textfeld `txtUserName`
angezeigt.

            txtdes.setText(rs.getString("des")); // Der Wert der
Spalte `des` wird im Textfeld `txtdes` angezeigt.

            userNameUpdateId = id; // Die Variable
`userNameUpdateId` wird mit der `id`-Variable zugewiesen.

        }

    } catch (Exception e) { // Falls es einen Fehler gibt,

        JOptionPane.showMessageDialog(this, "WARNING", "Something
went wrong", JOptionPane.WARNING_MESSAGE); // wird eine Warnung
angezeigt.

        System.out.println(e); // Der Fehler wird in der Konsole
ausgegeben.

    }

}
```

```
//////////////////////////////////////////////////////

private void txtUpdateActionPerformed(java.awt.event.ActionEvent
evt) throws Exception {//GEN-FIRST:event_txtUpdateActionPerformed

        // TODO add your handling code here:

        String userName = txtUserName.getText();
```

```
        String pass =
AESencryption.encrypt(txtPassword.getText()); // Passwort wird
verschlüsselt

        String des = txtdes.getText();

        System.out.println(userNameUpdateId);

        try {

            // SQL-Abfrage zum Aktualisieren von Benutzerdaten in
der Datenbank

            String query = "update userdata set userName = '" +
userName + "', password = '" + pass + "', des = '" + des + "'
where uid = " + userNameUpdateId;

            st = conn.createStatement();

            st.executeUpdate(query);



            // Textfelder werden geleert, nachdem die Daten
aktualisiert wurden

            txtPassword.setText("");

            txtUserName.setText("");

            txtdes.setText("");



            // Alle Daten in der Tabelle werden entfernt und die
neuen Daten werden hinzugefügt

            DefaultTableModel tModel = (DefaultTableModel)
tbl.getModel();

            tModel.setRowCount(0);



            // Alle Daten in der Datenbank-Tabelle werden
abgerufen und in die Tabelle hinzugefügt

            getTableDetails();

        } catch (Exception e) {

            JOptionPane.showMessageDialog(this, "Something went
wrong!", "WARNING!", JOptionPane.WARNING_MESSAGE);

            System.out.println(e);
```

```
        }
    }//GEN-LAST:event_txtUpdateActionPerformed


    private void chckVisMouseClicked(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_chckVisMouseClicked
        // TODO add your handling code here:
        if (chckVis.isSelected()) {


            txtPassword.setEchoChar((char) 0); // Passwort wird
angezeigt
        } else {


            txtPassword.setEchoChar('\u25cf'); // Passwort wird
verschleiert
        }
    }//GEN-LAST:event_chckVisMouseClicked


    private void txtSearchKeyReleased(java.awt.event.KeyEvent evt)
{//GEN-FIRST:event_txtSearchKeyReleased
        // TODO add your handling code here:
        String search = txtSearch.getText();


        // Alle Daten in der Tabelle werden entfernt und die neuen
Daten werden hinzugefügt
        DefaultTableModel tModel = (DefaultTableModel)
tbl.getModel();
        tModel.setRowCount(0);


        try {
```

```
            // SQL-Abfrage zum Abrufen aller Daten, die mit dem
eingegebenen Suchbegriff übereinstimmen und die dem angemeldeten
Benutzer gehören

            String query = "select * from userdata where (userName
LIKE '%" + search + "%' OR password LIKE '%" + search + "%' OR des
LIKE '%" + search + "%') AND user = '" + user + "'";

            st = conn.createStatement();

            ResultSet rs = st.executeQuery(query); //alle Daten
werden einem ResultSet zugewiesen



            while (rs.next()) {

                // Daten werden hinzugefügt, bis alle Daten der
Datenbank-Tabelle abgearbeitet sind

                String name = rs.getString("userName");

                String pass = rs.getString("enPass"); // Passwort
wird entschlüsselt

                String des = (rs.getString("des"));

                String id = Integer.toString(rs.getInt("uid"));



                // Ein String-Array wird erstellt, um Daten in der
Tabelle zu speichern

                String tbData[] = {id, des, name, pass};

                DefaultTableModel tblModel = (DefaultTableModel)
tbl.getModel();



                // String array der Tabelle hinzufügen

                tblModel.addRow(tbData);

            }



        } catch (Exception e) {

            JOptionPane.showMessageDialog(this, e); // Error
message im UI anzeigen
```

```
            System.out.println(e); // Error message printen ( zum
cmd )
        }



    }//GEN-LAST:event_txtSearchKeyReleased


    private void jLabel5MouseClicked(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_jLabel5MouseClicked
        // TODO add your handling code here:


    }//GEN-LAST:event_jLabel5MouseClicked


    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and
feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/pla
f.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
```

```
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(ManageForm.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(ManageForm.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(ManageForm.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(ManageForm.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Form und anzeige erstellen */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ManageForm().setVisible(true); // Form
anzeigen
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnAdd;
```

```
    private javax.swing.JCheckBox chckVis;
    private javax.swing.JLabel greetinglabel;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tbl;
    private javax.swing.JPasswordField txtPassword;
    private javax.swing.JTextField txtSearch;
    private javax.swing.JButton txtUpdate;
    private javax.swing.JTextField txtUserName;
    private javax.swing.JTextField txtdes;
    // End of variables declaration//GEN-END:variables
}
```

Die Klasse **ManageForm** hat mehrere Methoden, die für die Verwaltung von Passwörtern zuständig sind.

Die Methode **getUser** erhält den Benutzernamen als Parameter und zeigt eine Begrüßungsnachricht für den Benutzer an. Sie ruft auch die Methode **getTableDetails** auf, um alle Passwortdaten des Benutzers aus der Datenbank abzurufen und in der Tabelle anzuzeigen.

Die Methode **getTableDetails** ruft alle Passwortdaten des angemeldeten Benutzers aus der Datenbank ab und fügt sie der Tabelle hinzu.

Die Methode **btnAddActionPerformed** wird aufgerufen, wenn der "Hinzufügen"-Button geklickt wird. Diese Methode ruft die eingegebenen Passwortdaten aus den Textfeldern ab und verschlüsselt das Passwort mit MD5-Hashing-Algorithmus. Es erstellt dann eine SQL-Abfrage, um die Passwortdaten in der Datenbank zu speichern. Wenn die Passwortdaten erfolgreich hinzugefügt wurden, wird eine Meldung angezeigt, die Textfelder werden geleert und die Tabelle wird aktualisiert.

Schluss

Das war meine Facharbeit zum Thema Passwortmanager. Es gibt zahlreiche Möglichkeiten, das Programm zu erweitern, wie beispielsweise die Implementierung dynamischer Schlüssel, die sich bei jedem Öffnen des Programms ändern. Je nach Design könnte auch die Login-Form angepasst werden, um die Benutzerfreundlichkeit zu verbessern. Darüber hinaus könnten auch die Sicherheitsfunktionen weiter optimiert werden und das Programm könnte in eine Cloud-Umgebung integriert werden. Die Möglichkeiten sind nahezu unbegrenzt. Es ist beispielsweise möglich, zu überprüfen, ob ein Passwort bereits verwendet wurde und gegebenenfalls ein stärkeres Passwort vorzuschlagen. Es besteht auch die Möglichkeit, das Design zu verbessern oder zusätzliche Funktionen hinzuzufügen. Während meiner Facharbeit konnte ich nur einen Teil dieser Möglichkeiten umsetzen. In Zukunft plane ich, das Programm privat weiterzuentwickeln und zu nutzen. Im Anhang finden Sie den Passwort-Manager, die Gates-Datei für die Datenbanktabelle sowie die Connector-Datei, die ebenfalls benötigt wird. Insgesamt ist es ein solides Programm, das auch privat genutzt werden könnte, jedoch würde ich persönlich

einen Passwortmanager wie den von Google, der in der Cloud funktioniert und direkt mit dem Browser integriert ist, bevorzugen. Java ist nicht unbedingt die beste Programmiersprache, um ein Programm in einen Browser einzubetten.

Demo (Video)

Das Demo-Video zeigt, wie Passwörter hinzugefügt, bearbeitet, gesehen und gelöscht werden können.

2023.02.18 PasswordManager

Sicherheitsbedenken/Vorteile/Nachteile

Ein Vorteil eines privaten, lokalen Passwortmanagers ist, dass er weniger anfällig für gezielte Angriffe durch Viren und Malware ist. Da die meisten Viren auf bekannte Passwort-Manager wie den von Google abzielen, ist es relativ unwahrscheinlich, dass Passwörter aus der Datenbank eines lokalen Passwortmanagers entwendet werden. Zusätzlich sind alle Passwörter in der Datenbank des Programms verschlüsselt. Das verwendete Verschlüsselungssystem ist schnell und sicher, aber Kryptographie ist ein sehr

komplexes Thema, auf das ich in meiner Facharbeit nicht näher eingehen konnte. Der Passwort-Manager ist jedoch nicht besonders geschützt, da die Quellcode-Dateien zugänglich sind. Ein Hardcoded-Schlüssel wird verwendet, aber normalerweise würden dynamische Verschlüsselungsschlüssel verwendet werden, um die Sicherheit weiter zu erhöhen. Diese Schlüssel werden dann mehrfach verschlüsselt und gesichert, um maximale Sicherheit zu gewährleisten. Aufgrund der zeitlichen Begrenzungen meiner Facharbeit konnte ich diese zusätzlichen Sicherheitsfunktionen jedoch nicht umsetzen.

Selbstständigkeitserklärung Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die den verwendeten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

01.03.2023 Köln

Sami Zerfaoui