import javax.crypto.Cipher;

import javax.crypto.CipherInputStream;

import javax.crypto.CipherOutputStream;

import javax.crypto.spec.SecretKeySpec;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.security.MessageDigest;

import java.util.Arrays;

public class FileEncryptionTool extends JFrame {

private JTextField filePathField;

private JPasswordField passwordField;

private JButton browseButton, encryptButton, decryptButton;

private File selectedFile;

public FileEncryptionTool() {

setTitle("File Encryption/Decryption Tool");

setSize(500, 250);

setLocationRelativeTo(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new GridLayout(5, 1, 10, 10));

// UI Components

filePathField = new JTextField();

passwordField = new JPasswordField();

browseButton = new JButton("Browse File");

encryptButton = new JButton("Encrypt");

decryptButton = new JButton("Decrypt");

// Panel for file chooser

JPanel filePanel = new JPanel(new BorderLayout());

filePanel.add(filePathField, BorderLayout.CENTER);

filePanel.add(browseButton, BorderLayout.EAST);

// Panel for password

JPanel passwordPanel = new JPanel(new BorderLayout());

passwordPanel.add(new JLabel("Enter Password: "), BorderLayout.WEST);

passwordPanel.add(passwordField, BorderLayout.CENTER);

// Button panel

JPanel buttonPanel = new JPanel();

buttonPanel.add(encryptButton);

buttonPanel.add(decryptButton);

// Add to frame

add(new JLabel("Select File to Encrypt/Decrypt:", SwingConstants.CENTER));

add(filePanel);

add(passwordPanel);

add(buttonPanel);

add(new JLabel("© Secure File Tool", SwingConstants.CENTER));

// Button actions

browseButton.addActionListener(e -> chooseFile());

encryptButton.addActionListener(e -> processFile(true));

decryptButton.addActionListener(e -> processFile(false));

setVisible(true);

}

private void chooseFile() {

JFileChooser chooser = new JFileChooser();

int result = chooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

selectedFile = chooser.getSelectedFile();

filePathField.setText(selectedFile.getAbsolutePath());

}

}

private void processFile(boolean encrypt) {

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

if (selectedFile == null || password.isEmpty()) {

JOptionPane.showMessageDialog(this, "Select file and enter password!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

File outputFile;

if (encrypt) {

outputFile = new File(selectedFile.getAbsolutePath() + ".encrypted");

} else {

String name = selectedFile.getAbsolutePath();

if (name.endsWith(".encrypted")) {

name = name.substring(0, name.length() - 10);

}

outputFile = new File(name + ".decrypted");

}

try {

SecretKeySpec key = getKey(password);

Cipher cipher = Cipher.getInstance("AES");

cipher.init(encrypt ? Cipher.ENCRYPT\_MODE : Cipher.DECRYPT\_MODE, key);

FileInputStream fis = new FileInputStream(selectedFile);

FileOutputStream fos = new FileOutputStream(outputFile);

CipherInputStream cis = encrypt ? null : new CipherInputStream(fis, cipher);

CipherOutputStream cos = encrypt ? new CipherOutputStream(fos, cipher) : null;

byte[] buffer = new byte[4096];

int bytesRead;

if (encrypt) {

while ((bytesRead = fis.read(buffer)) != -1) {

cos.write(buffer, 0, bytesRead);

}

cos.flush();

cos.close();

} else {

while ((bytesRead = cis.read(buffer)) != -1) {

fos.write(buffer, 0, bytesRead);

}

fos.flush();

fos.close();

}

fis.close();

JOptionPane.showMessageDialog(this,

(encrypt ? "Encryption" : "Decryption") + " successful!\nSaved as: " + outputFile.getName(),

"Success", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "❌ Error: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private SecretKeySpec getKey(String password) throws Exception {

byte[] key = password.getBytes("UTF-8");

MessageDigest sha = MessageDigest.getInstance("SHA-256");

key = sha.digest(key); // SHA-256 hash of password

key = Arrays.copyOf(key, 16); // use only 128 bits for AES

return new SecretKeySpec(key, "AES");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(FileEncryptionTool::new);

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.