Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 1

з дисципліни «Захист інформації в комп’ютерних системах»

на тему: «Шифр моноалфавітної заміни (шифр Цезаря)»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірив:

Муляревич О.В.

**Мета роботи:** ознайомитись з основами класичної техніки шифрування – шифрами моноалфавітної заміни та типовим прикладом шифрів даного виду - шифром Цезаря.

**Завдання:**

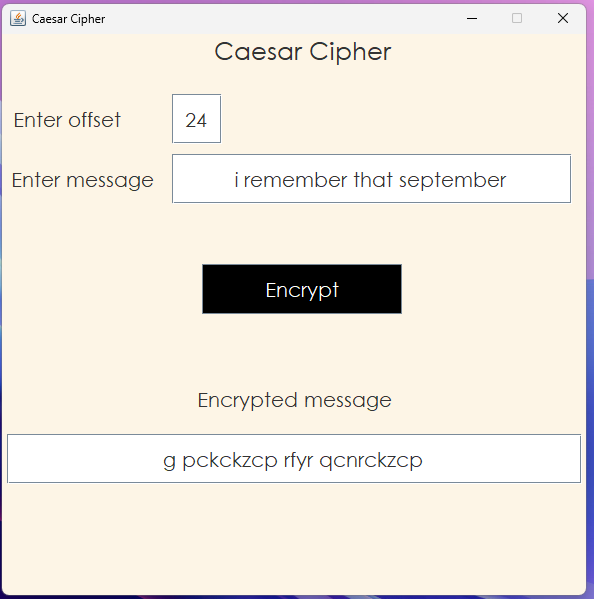
Створити програму, що реалізує шифрування вихідного повідомлення за допомогою шифру Цезаря.

**Варіант:** 22.

**Виконання завдання:**

Для виконання завдання було вибрано мову Java та бібліотеку Swing для створення графічного інтерфейсу додатку.

*Програма – Caesar Cipher*



*Рис. 1 – вікно програми*

Код файлу *CaesarCipher.java*, у якому міститься реалізація заданого алгоритму шифрування:

Лістинг 1

|  |
| --- |
| package application.encryptor;  public class CaesarCipher {  private static final int *ALPHABET\_LENGTH* = 26;  private static final int *UPPERCASE\_ASCII\_START* = 65;  private static final int *LOWERCASE\_ASCII\_START* = 97;   public String encrypt(String message, int offset) {  StringBuilder builder = new StringBuilder();  for (char c : message.toCharArray()) {  if (Character.*isLowerCase*(c)) {  encryptCharConsideringCase(builder, c, offset, *LOWERCASE\_ASCII\_START*);  } else {  encryptCharConsideringCase(builder, c, offset, *UPPERCASE\_ASCII\_START*);  }  }  return builder.toString();  }   private void encryptCharConsideringCase(StringBuilder builder,  char character,  int offset,  int caseAsciiStart) {  if (character != ' ') {  int newCharPosition = (character + offset - caseAsciiStart)  % *ALPHABET\_LENGTH* + caseAsciiStart;  builder.append((char) newCharPosition);  } else {  builder.append(" ");  }  } } |

Код файлу *Frame.java*, у якому міститься реалізація коду графічного інтерфейсу програми:

Лістинг 2

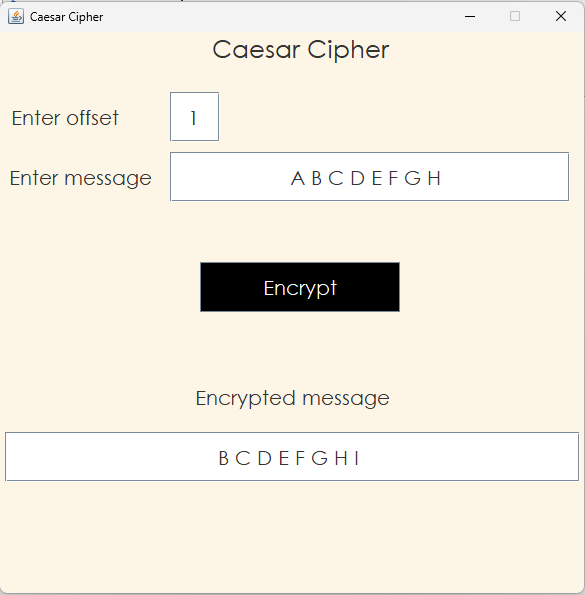
|  |
| --- |
| package application.gui;  import application.encryptor.CaesarCipher; import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField; import java.awt.Color; import java.awt.Dimension; import java.awt.Font; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;  public class Frame extends JFrame implements ActionListener {  private CaesarCipher caesarCipher;   private static final String *FRAME\_TITLE* = "Caesar Cipher";  private static final int *DIMENSION* = 600;   private JFrame frame;   private JLabel headLabel;  private JLabel offsetLabel;  private JLabel inputMessageLabel;  private JLabel outputMessageLabel;   private JTextField offsetTextField;  private JTextField inputTextField;  private JTextField outputTextField;   private JButton encryptionButton;   public Frame() {  headLabel = new JLabel();  adjustHeadLabelSettings(headLabel);   offsetLabel = new JLabel();  adjustOffsetLabelSettings(offsetLabel);   offsetTextField = new JTextField();  adjustOffsetTextFieldSettings(offsetTextField);   inputMessageLabel = new JLabel();  adjustInputMessageLabelSettings(inputMessageLabel);   inputTextField = new JTextField();  adjustInputTextFieldSettings(inputTextField);   encryptionButton = new JButton();  adjustEncryptButtonSettings(encryptionButton);   outputMessageLabel = new JLabel();  adjustOutputMessageLabelSettings(outputMessageLabel);   outputTextField = new JTextField();  adjustOutputTextFieldSettings(outputTextField);   frame = new JFrame();  adjustFrameSettings(frame);   frame.add(headLabel);  frame.add(offsetLabel);  frame.add(offsetTextField);  frame.add(inputMessageLabel);  frame.add(inputTextField);  frame.add(encryptionButton);  frame.add(outputMessageLabel);  frame.add(outputTextField);  }   private void adjustFrameSettings(JFrame frame) {  frame.setTitle(*FRAME\_TITLE*);  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setResizable(false);  frame.setSize(*DIMENSION*,*DIMENSION*);  frame.getContentPane().setBackground(new Color(253,245,230));  frame.setLayout(null);  frame.setVisible(true);  }   private void adjustHeadLabelSettings(JLabel headLabel) {  headLabel.setText(*FRAME\_TITLE*);  headLabel.setFont(new Font("Century Gothic", Font.*PLAIN*, 25));  headLabel.setVerticalAlignment(JLabel.*TOP*);  headLabel.setHorizontalAlignment(JLabel.*CENTER*);  headLabel.setBounds(0,0, *DIMENSION*,50);  }   private void adjustOffsetLabelSettings(JLabel offsetLabel) {  offsetLabel.setText("Enter offset");  offsetLabel.setFont(new Font("Century Gothic", Font.*PLAIN*, 20));  offsetLabel.setHorizontalAlignment(JLabel.*CENTER*);  offsetLabel.setBounds(5, 60, 120, 50);  }   private void adjustOffsetTextFieldSettings(JTextField offsetTextField) {  offsetTextField.setPreferredSize(new Dimension(250,40));  offsetTextField.setBounds(170, 60, 50, 50);  offsetTextField.setFont(new Font("Century Gothic", Font.*PLAIN*, 20));  offsetTextField.setHorizontalAlignment(JLabel.*CENTER*);  }   private void adjustInputMessageLabelSettings(JLabel inputMessageLabel) {  inputMessageLabel.setText("Enter message");  inputMessageLabel.setFont(new Font("Century Gothic", Font.*PLAIN*, 20));  inputMessageLabel.setHorizontalAlignment(JLabel.*CENTER*);  inputMessageLabel.setBounds(5, 120, 150, 50);  }   private void adjustInputTextFieldSettings(JTextField inputTextField) {  inputTextField.setPreferredSize(new Dimension(250,40));  inputTextField.setBounds(170, 120, 400, 50);  inputTextField.setFont(new Font("Century Gothic", Font.*PLAIN*, 20));  inputTextField.setHorizontalAlignment(JLabel.*CENTER*);  }   private void adjustEncryptButtonSettings(JButton encryptionButton) {  encryptionButton.setBounds(200, 230, 200, 50);  encryptionButton.setText("Encrypt");  encryptionButton.setFont(new Font("Century Gothic", Font.*PLAIN*, 20));  encryptionButton.setForeground(Color.*WHITE*);  encryptionButton.setFocusable(false);  encryptionButton.setBackground(Color.*black*);  encryptionButton.addActionListener(this);  }   private void adjustOutputMessageLabelSettings(JLabel outputMessageLabel) {  outputMessageLabel.setText("Encrypted message");  outputMessageLabel.setFont(new Font("Century Gothic", Font.*PLAIN*, 20));  outputMessageLabel.setHorizontalAlignment(JLabel.*CENTER*);  outputMessageLabel.setBounds(5, 340, 575, 50);  }   private void adjustOutputTextFieldSettings(JTextField outputTextField) {  outputTextField.setPreferredSize(new Dimension(575,50));  outputTextField.setBounds(5, 400, 575, 50);  outputTextField.setFont(new Font("Century Gothic", Font.*PLAIN*, 20));  outputTextField.setHorizontalAlignment(JLabel.*CENTER*);  }   @Override  public void actionPerformed(ActionEvent e) {  if (e.getSource().equals(encryptionButton)) {  caesarCipher = new CaesarCipher();  String encryptedMessage = caesarCipher.encrypt(inputTextField.getText(),  Integer.*parseInt*(offsetTextField.getText()));  outputTextField.setText(encryptedMessage);  }  } } |

Код головного файлу програми - Main.java:

Лістинг 3

|  |
| --- |
| package application;  import application.gui.Frame;  public class Main {  public static void main(String[] args) {  Frame frame = new Frame();  } } |

Результат роботи програми:



*Рис. 2 – вікно програми*

Програма шифрує текст обох регістрів та має можливість динамічно встановлювати зсув.

**Висновок:** у ході виконання лабораторної роботи було вивчено основи класичної техніки шифрування – шифрами моноалфавітної заміни та типовим прикладом шифрів даного виду - шифром Цезаря. Було створено програму, що реалізує шифрування вихідного повідомлення за допомогою шифру Цезаря мовою програмування Java.