Tugas ke	: 1
Mata Kuliah	: Keamanan Komputer/Kriptografi

Kriptografi Klasik

Disusun Oleh :

NAMA	**: Sheila Dwi Yulianti Saputri**
STB	**: 222238**
KELAS	**: 5TKKO-G**

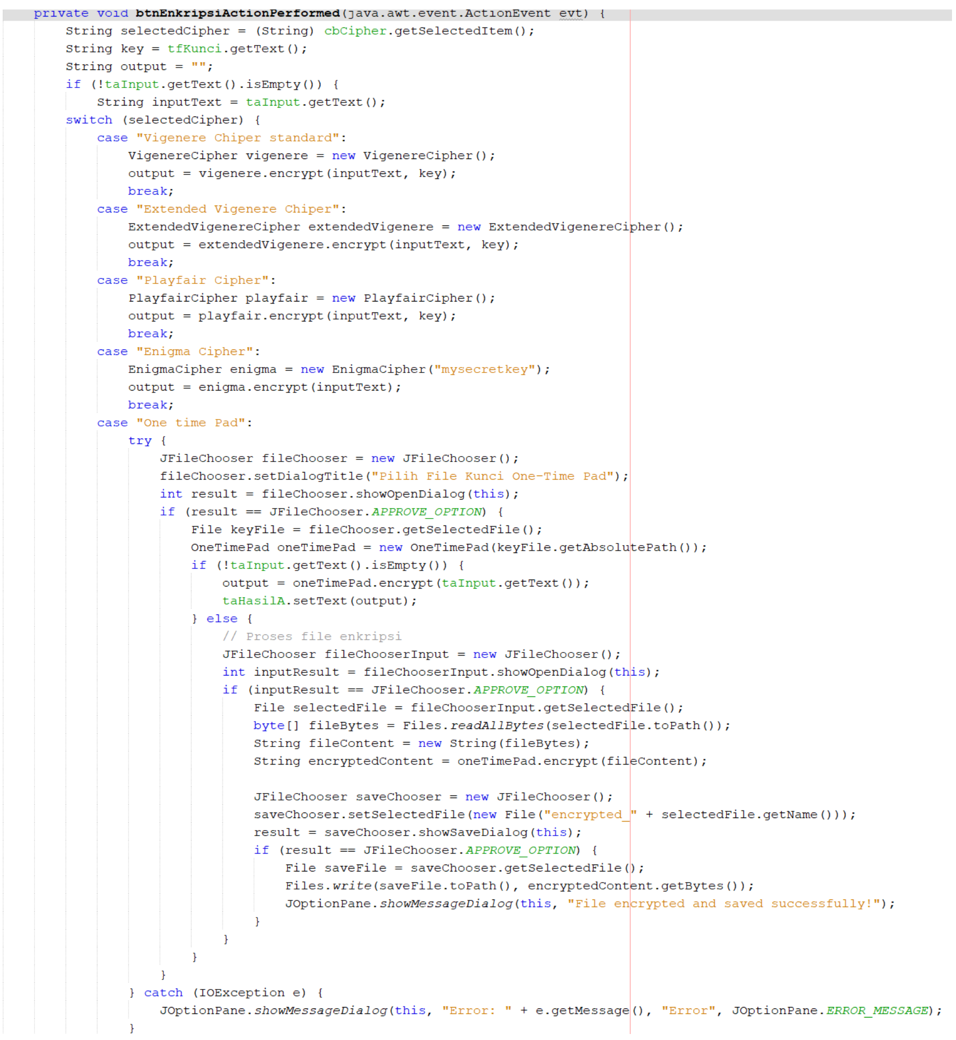
UNIVERSITAS DIPA MAKASSAR

MAKASSAR

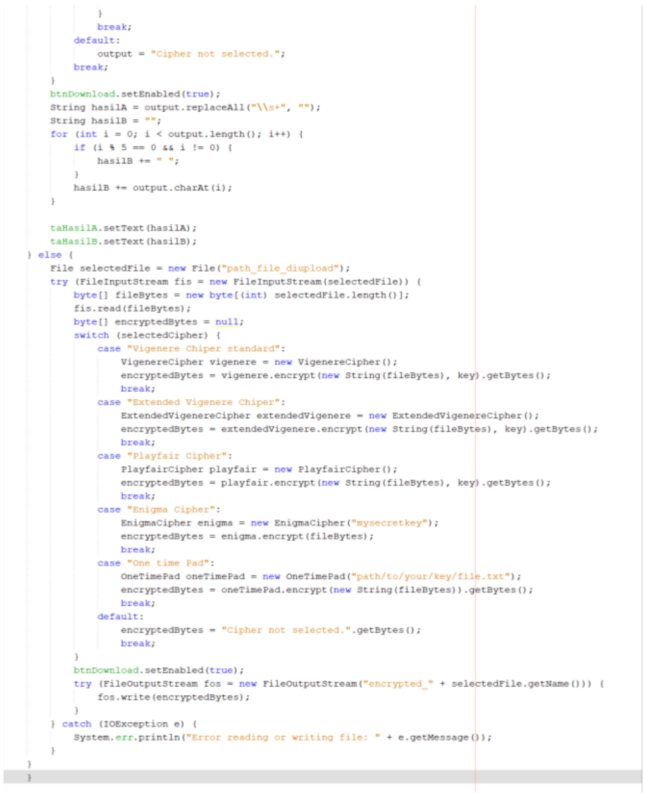
2024

No.	Spesifikasi	Berhasil (✓)	Kurang Berhasil (**X**)	keterangan
1.	Vigenere Cipher (Standar)	✓		Plaintext: hello world, Key: malam, Encrypt: TEWLAIOCLP, Decrypt: HELLOWORLD
2.	Extended Vigenere Cipher	✓		Plaintext: sheila, Key: valo, Encrypt: nhpwga, Decrypt: sheila
3.	Playfair Cipher	✓		Plaintext: undipa, Key: hii, Encrypt: SPIOLE, Decrypt: UNDIPA
4.	Enigma Cipher	✓		Plaintext: semester, Key: gelo, Encrypt: GQGKVIKI, Decrypt: SEMESTER
5.	One-Time Pad	✓		Plaintext: lima!!!, Key: file kunci, Encrypt: JQIN, Decrypt: LIMA

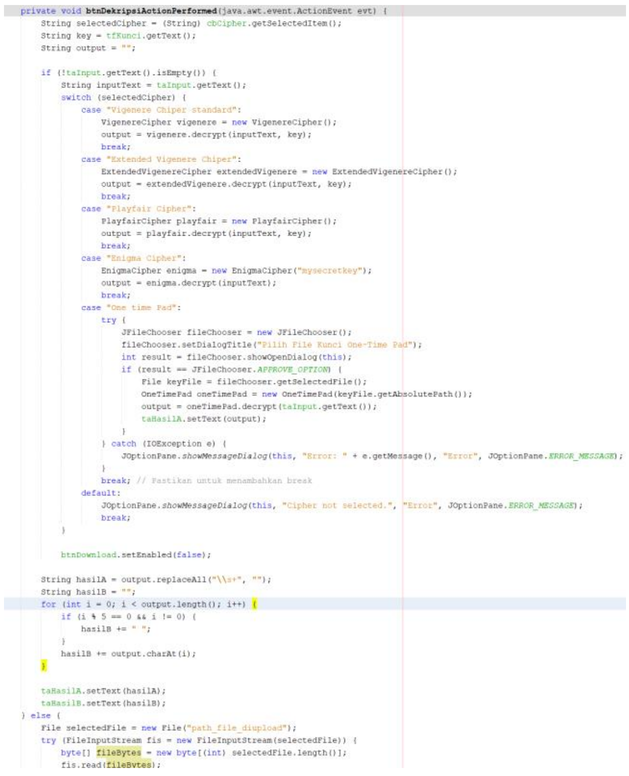
Source Kode



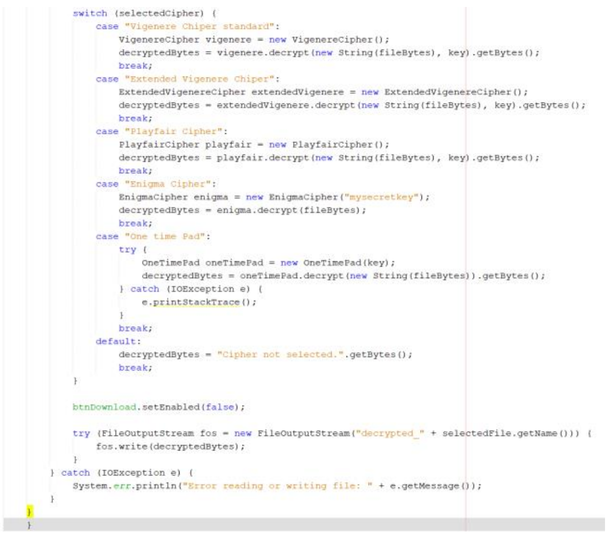
```java
    private void btnEnkripsiActionPerformed(java.awt.event.ActionEvent evt) {
        String selectedCipher = (String) cbCipher.getSelectedItem();
        String key = tfKunci.getText();
        String output = "";
        if (!taInput.getText().isEmpty()) {
            String inputText = taInput.getText();
        switch (selectedCipher) {
            case "Vigenere Chiper standard":
                VigenereCipher vigenere = new VigenereCipher();
                output = vigenere.encrypt(inputText, key);
                break;
            case "Extended Vigenere Chiper":
                ExtendedVigenereCipher extendedVigenere = new ExtendedVigenereCipher();
                output = extendedVigenere.encrypt(inputText, key);
                break;
            case "Playfair Cipher":
                PlayfairCipher playfair = new PlayfairCipher();
                output = playfair.encrypt(inputText, key);
                break;
            case "Enigma Cipher":
                EnigmaCipher enigma = new EnigmaCipher("mysecretkey");
                output = enigma.encrypt(inputText);
                break;
            case "One time Pad":
                try {
                    JFileChooser fileChooser = new JFileChooser();
                    fileChooser.setDialogTitle("Pilih File Kunci One-Time Pad");
                    int result = fileChooser.showOpenDialog(this);
                    if (result == JFileChooser.APPROVE_OPTION) {
                        File keyFile = fileChooser.getSelectedFile();
                        OneTimePad oneTimePad = new OneTimePad(keyFile.getAbsolutePath());
                        if (!taInput.getText().isEmpty()) {
                            output = oneTimePad.encrypt(taInput.getText());
                            taHasilA.setText(output);
                        } else {
                            // Proses file enkripsi
                            JFileChooser fileChooserInput = new JFileChooser();
                            int inputResult = fileChooserInput.showOpenDialog(this);
                            if (inputResult == JFileChooser.APPROVE_OPTION) {
                                File selectedFile = fileChooserInput.getSelectedFile();
                                byte[] fileBytes = Files.readAllBytes(selectedFile.toPath());
                                String fileContent = new String(fileBytes);
                                String encryptedContent = oneTimePad.encrypt(fileContent);

                                JFileChooser saveChooser = new JFileChooser();
                                saveChooser.setSelectedFile(new File("encrypted_" + selectedFile.getName()));
                                result = saveChooser.showSaveDialog(this);
                                if (result == JFileChooser.APPROVE_OPTION) {
                                    File saveFile = saveChooser.getSelectedFile();
                                    Files.write(saveFile.toPath(), encryptedContent.getBytes());
                                    JOptionPane.showMessageDialog(this, "File encrypted and saved successfully!");
                                }
                            }
                        }
                    }
                } catch (IOException e) {
                    JOptionPane.showMessageDialog(this, "Error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                }
```



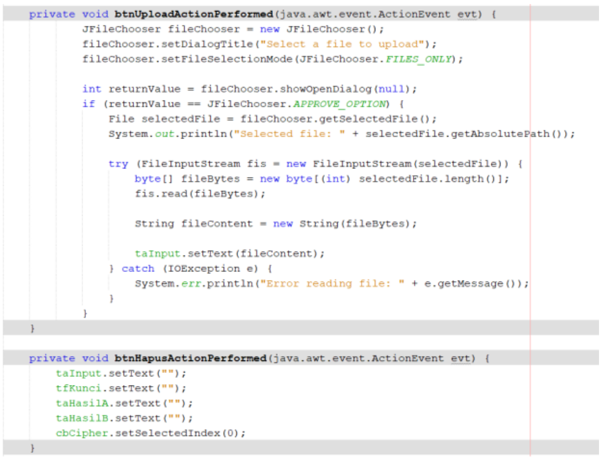
```java
                }
                break;
            default:
                output = "Cipher not selected.";
            break;
        }
        btnDownload.setEnabled(true);
        String hasilA = output.replaceAll("\\s+", "");
        String hasilB = "";
        for (int i = 0; i < output.length(); i++) {
            if (i % 5 == 0 && i != 0) {
                hasilB += " ";
            }
            hasilB += output.charAt(i);
        }

        taHasilA.setText(hasilA);
        taHasilB.setText(hasilB);
    } else {
        File selectedFile = new File("path_file_diupload");
        try (FileInputStream fis = new FileInputStream(selectedFile)) {
            byte[] fileBytes = new byte[(int) selectedFile.length()];
            fis.read(fileBytes);
            byte[] encryptedBytes = null;
            switch (selectedCipher) {
                case "Vigenere Chiper standard":
                    VigenereCipher vigenere = new VigenereCipher();
                    encryptedBytes = vigenere.encrypt(new String(fileBytes), key).getBytes();
                    break;
                case "Extended Vigenere Chiper":
                    ExtendedVigenereCipher extendedVigenere = new ExtendedVigenereCipher();
                    encryptedBytes = extendedVigenere.encrypt(new String(fileBytes), key).getBytes();
                    break;
                case "Playfair Cipher":
                    PlayfairCipher playfair = new PlayfairCipher();
                    encryptedBytes = playfair.encrypt(new String(fileBytes), key).getBytes();
                    break;
                case "Enigma Cipher":
                    EnigmaCipher enigma = new EnigmaCipher("mysecretkey");
                    encryptedBytes = enigma.encrypt(fileBytes);
                    break;
                case "One time Pad":
                    OneTimePad oneTimePad = new OneTimePad("path/to/your/key/file.txt");
                    encryptedBytes = oneTimePad.encrypt(new String(fileBytes)).getBytes();
                    break;
                default:
                    encryptedBytes = "Cipher not selected.".getBytes();
                    break;
            }
            btnDownload.setEnabled(true);
            try (FileOutputStream fos = new FileOutputStream("encrypted_" + selectedFile.getName())) {
                fos.write(encryptedBytes);
            }
        } catch (IOException e) {
            System.err.println("Error reading or writing file: " + e.getMessage());
        }
    }
    }
```



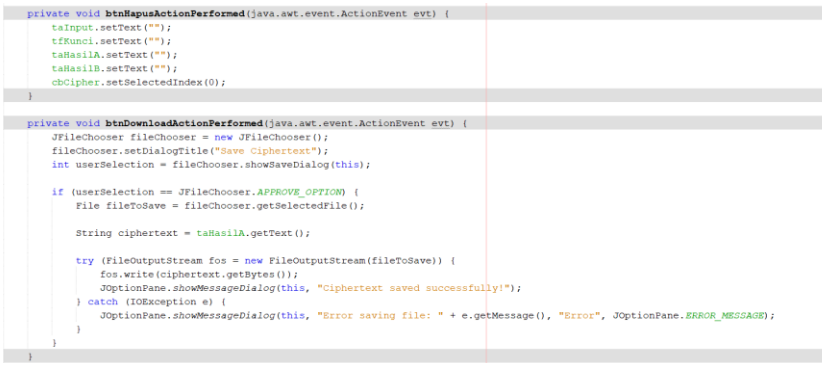
```java
private void btnDekripsiActionPerformed(java.awt.event.ActionEvent evt) {
    String selectedCipher = (String) cbCipher.getSelectedItem();
    String key = tfKunci.getText();
    String output = "";

    if (!taInput.getText().isEmpty()) {
        String inputText = taInput.getText();
        switch (selectedCipher) {
            case "Vigenere Chiper standard":
                VigenereCipher vigenere = new VigenereCipher();
                output = vigenere.decrypt(inputText, key);
                break;
            case "Extended Vigenere Chiper":
                ExtendedVigenereCipher extendedVigenere = new ExtendedVigenereCipher();
                output = extendedVigenere.decrypt(inputText, key);
                break;
            case "Playfair Cipher":
                PlayfairCipher playfair = new PlayfairCipher();
                output = playfair.decrypt(inputText, key);
                break;
            case "Enigma Cipher":
                EnigmaCipher enigma = new EnigmaCipher("mysecretkey");
                output = enigma.decrypt(inputText);
                break;
            case "One time Pad":
                try {
                    JFileChooser fileChooser = new JFileChooser();
                    fileChooser.setDialogTitle("Pilih File Kunci One-Time Pad");
                    int result = fileChooser.showOpenDialog(this);
                    if (result == JFileChooser.APPROVE_OPTION) {
                        File keyFile = fileChooser.getSelectedFile();
                        OneTimePad oneTimePad = new OneTimePad(keyFile.getAbsolutePath());
                        output = oneTimePad.decrypt(taInput.getText());
                        taHasilA.setText(output);
                    }
                } catch (IOException e) {
                    JOptionPane.showMessageDialog(this, "Error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                }
                break; // Pastikan untuk menambahkan break
            default:
                JOptionPane.showMessageDialog(this, "Cipher not selected.", "Error", JOptionPane.ERROR_MESSAGE);
                break;
        }

        btnDownload.setEnabled(false);

    String hasilA = output.replaceAll("\\s+", "");
    String hasilB = "";
    for (int i = 0; i < output.length(); i++) {
        if (i % 5 == 0 && i != 0) {
            hasilB += " ";
        }
        hasilB += output.charAt(i);
    }

    taHasilA.setText(hasilA);
    taHasilB.setText(hasilB);
} else {
    File selectedFile = new File("path_file_diupload");
    try (FileInputStream fis = new FileInputStream(selectedFile)) {
        byte[] fileBytes = new byte[(int) selectedFile.length()];
        fis.read(fileBytes);
```



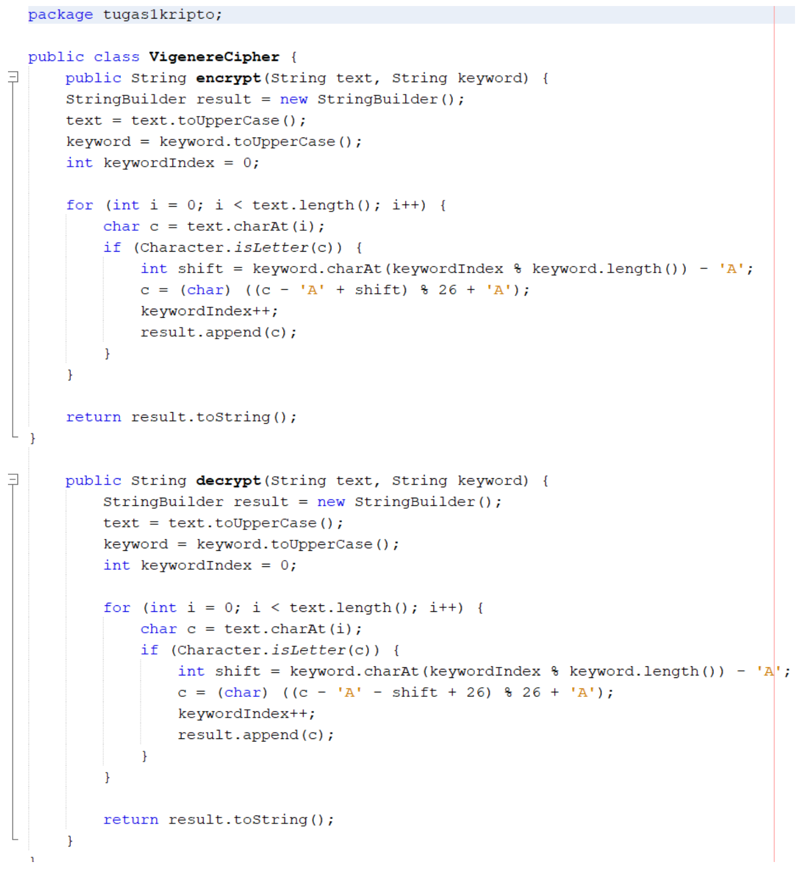
```java
            switch (selectedCipher) {
                case "Vigenere Chiper standard":
                    VigenereCipher vigenere = new VigenereCipher();
                    decryptedBytes = vigenere.decrypt(new String(fileBytes), key).getBytes();
                    break;
                case "Extended Vigenere Chiper":
                    ExtendedVigenereCipher extendedVigenere = new ExtendedVigenereCipher();
                    decryptedBytes = extendedVigenere.decrypt(new String(fileBytes), key).getBytes();
                    break;
                case "Playfair Cipher":
                    PlayfairCipher playfair = new PlayfairCipher();
                    decryptedBytes = playfair.decrypt(new String(fileBytes), key).getBytes();
                    break;
                case "Enigma Cipher":
                    EnigmaCipher enigma = new EnigmaCipher("mysecretkey");
                    decryptedBytes = enigma.decrypt(fileBytes);
                    break;
                case "One time Pad":
                    try {
                        OneTimePad oneTimePad = new OneTimePad(key);
                        decryptedBytes = oneTimePad.decrypt(new String(fileBytes)).getBytes();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    break;
                default:
                    decryptedBytes = "Cipher not selected.".getBytes();
                    break;
            }

            btnDownload.setEnabled(false);

            try (FileOutputStream fos = new FileOutputStream("decrypted_" + selectedFile.getName())) {
                fos.write(decryptedBytes);
            }
        } catch (IOException e) {
            System.err.println("Error reading or writing file: " + e.getMessage());
        }
    }
    }
```



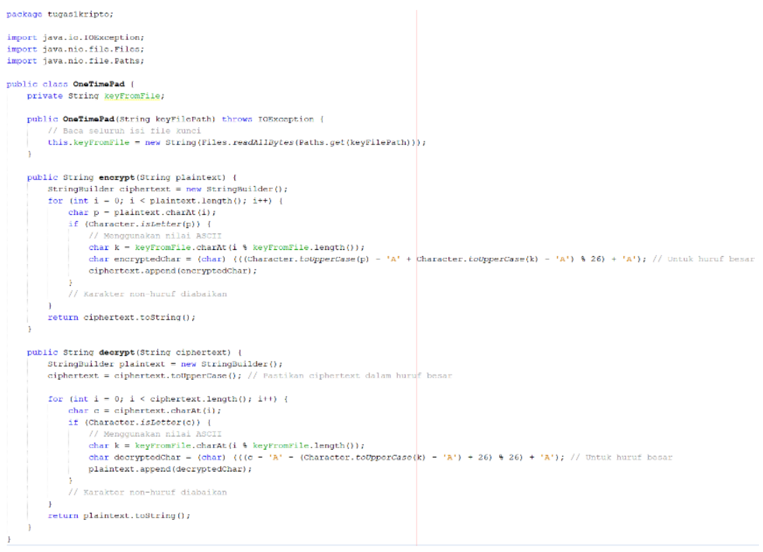
```java
private void btnUploadActionPerformed(java.awt.event.ActionEvent evt) {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Select a file to upload");
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);

        int returnValue = fileChooser.showOpenDialog(null);
        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            System.out.println("Selected file: " + selectedFile.getAbsolutePath());

            try (FileInputStream fis = new FileInputStream(selectedFile)) {
                byte[] fileBytes = new byte[(int) selectedFile.length()];
                fis.read(fileBytes);

                String fileContent = new String(fileBytes);

                taInput.setText(fileContent);
            } catch (IOException e) {
                System.err.println("Error reading file: " + e.getMessage());
            }
        }
}

private void btnHapusActionPerformed(java.awt.event.ActionEvent evt) {
    taInput.setText("");
    tfKunci.setText("");
    taHasilA.setText("");
    taHasilB.setText("");
    cbCipher.setSelectedIndex(0);
}
```



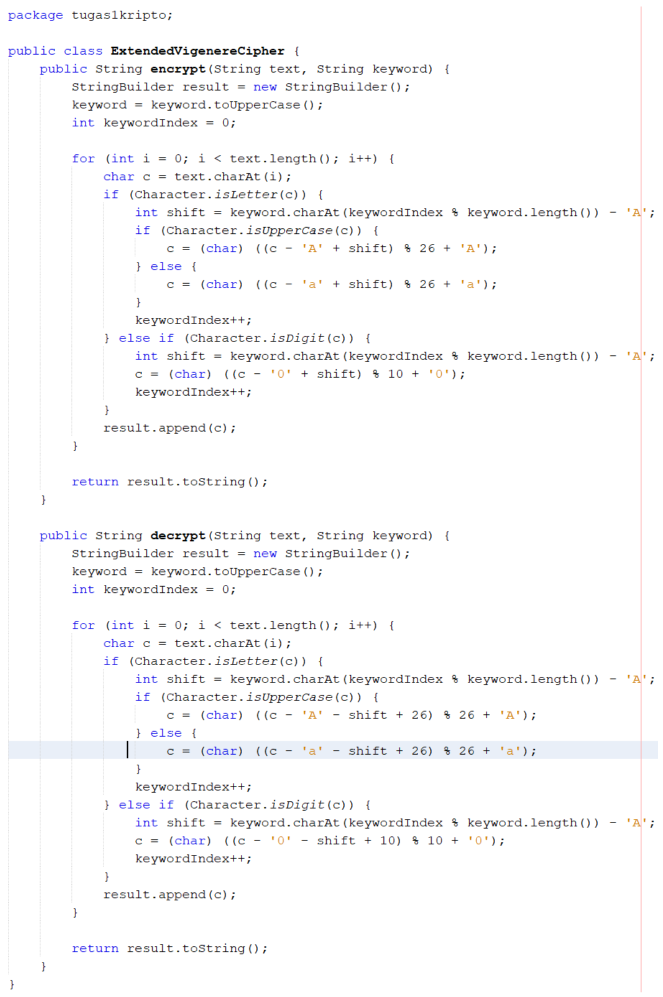
```java
    private void btnHapusActionPerformed(java.awt.event.ActionEvent evt) {
        taInput.setText("");
        tfKunci.setText("");
        taHasilA.setText("");
        taHasilB.setText("");
        cbCipher.setSelectedIndex(0);
    }

    private void btnDownloadActionPerformed(java.awt.event.ActionEvent evt) {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Save Ciphertext");
        int userSelection = fileChooser.showSaveDialog(this);

        if (userSelection == JFileChooser.APPROVE_OPTION) {
            File fileToSave = fileChooser.getSelectedFile();

            String ciphertext = taHasilA.getText();

            try (FileOutputStream fos = new FileOutputStream(fileToSave)) {
                fos.write(ciphertext.getBytes());
                JOptionPane.showMessageDialog(this, "Ciphertext saved successfully!");
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Error saving file: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
```



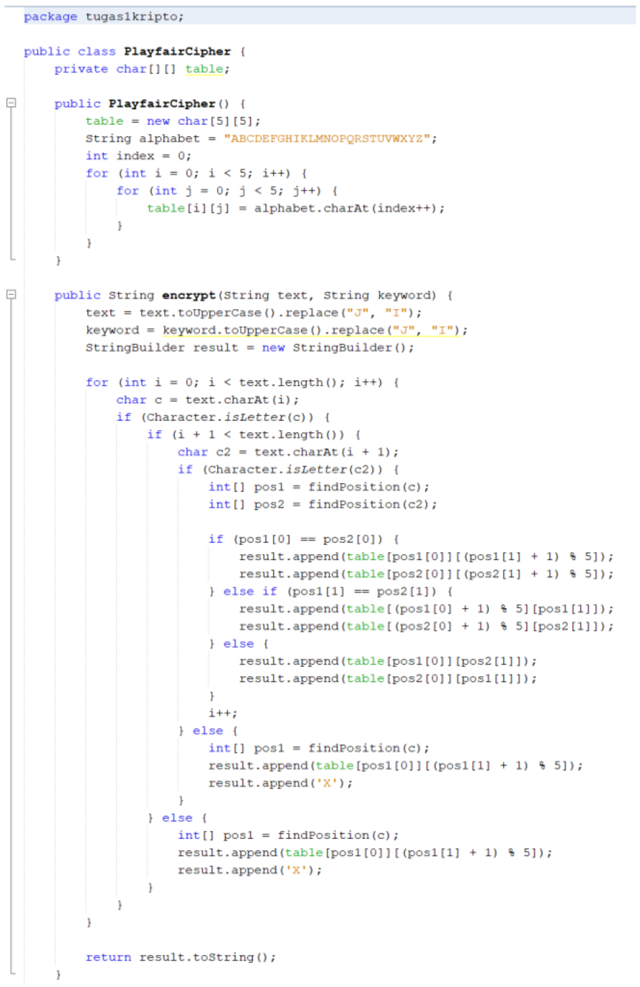
```java
package tugas1kripto;

public class VigenereCipher {
    public String encrypt(String text, String keyword) {
    StringBuilder result = new StringBuilder();
    text = text.toUpperCase();
    keyword = keyword.toUpperCase();
    int keywordIndex = 0;

    for (int i = 0; i < text.length(); i++) {
        char c = text.charAt(i);
        if (Character.isLetter(c)) {
            int shift = keyword.charAt(keywordIndex % keyword.length()) - 'A';
            c = (char) ((c - 'A' + shift) % 26 + 'A');
            keywordIndex++;
            result.append(c);
        }
    }

    return result.toString();
}

    public String decrypt(String text, String keyword) {
        StringBuilder result = new StringBuilder();
        text = text.toUpperCase();
        keyword = keyword.toUpperCase();
        int keywordIndex = 0;

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (Character.isLetter(c)) {
                int shift = keyword.charAt(keywordIndex % keyword.length()) - 'A';
                c = (char) ((c - 'A' - shift + 26) % 26 + 'A');
                keywordIndex++;
                result.append(c);
            }
        }

        return result.toString();
    }
}
```



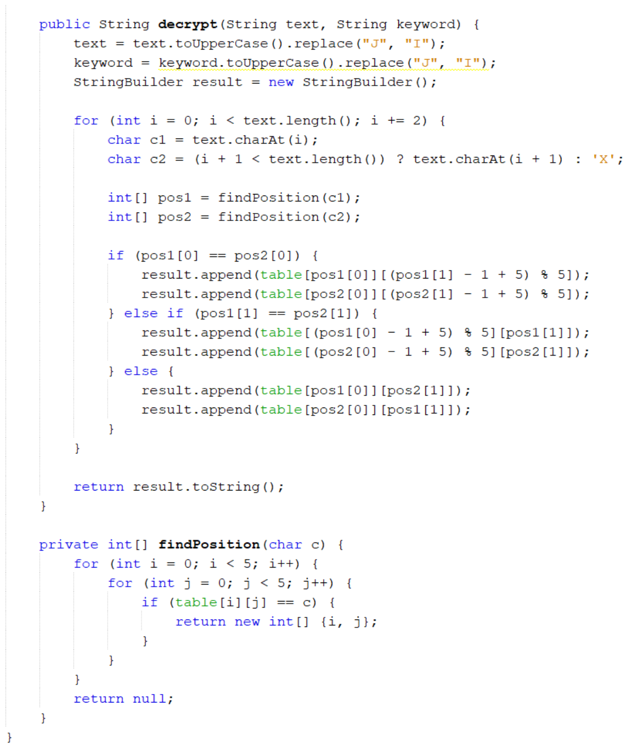
```java
package tugas1kripto;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class OneTimePad {
    private String keyFromFile;

    public OneTimePad(String keyFilePath) throws IOException {
        // Baca seluruh isi file kunci
        this.keyFromFile = new String(Files.readAllBytes(Paths.get(keyFilePath)));
    }

    public String encrypt(String plaintext) {
        StringBuilder ciphertext = new StringBuilder();
        for (int i = 0; i < plaintext.length(); i++) {
            char p = plaintext.charAt(i);
            if (Character.isLetter(p)) {
                // Menggunakan nilai ASCII
                char k = keyFromFile.charAt(i % keyFromFile.length());
                char encryptedChar = (char) (((Character.toUpperCase(p) - 'A' + Character.toUpperCase(k) - 'A') % 26) + 'A'); // Untuk huruf besar
                ciphertext.append(encryptedChar);
            }
            // Karakter non-huruf diabaikan
        }
        return ciphertext.toString();
    }

    public String decrypt(String ciphertext) {
        StringBuilder plaintext = new StringBuilder();
        ciphertext = ciphertext.toUpperCase(); // Pastikan ciphertext dalam huruf besar

        for (int i = 0; i < ciphertext.length(); i++) {
            char c = ciphertext.charAt(i);
            if (Character.isLetter(c)) {
                // Menggunakan nilai ASCII
                char k = keyFromFile.charAt(i % keyFromFile.length());
                char decryptedChar = (char) (((c - 'A' - (Character.toUpperCase(k) - 'A') + 26) % 26) + 'A'); // Untuk huruf besar
                plaintext.append(decryptedChar);
            }
            // Karakter non-huruf diabaikan
        }
        return plaintext.toString();
    }
}
```



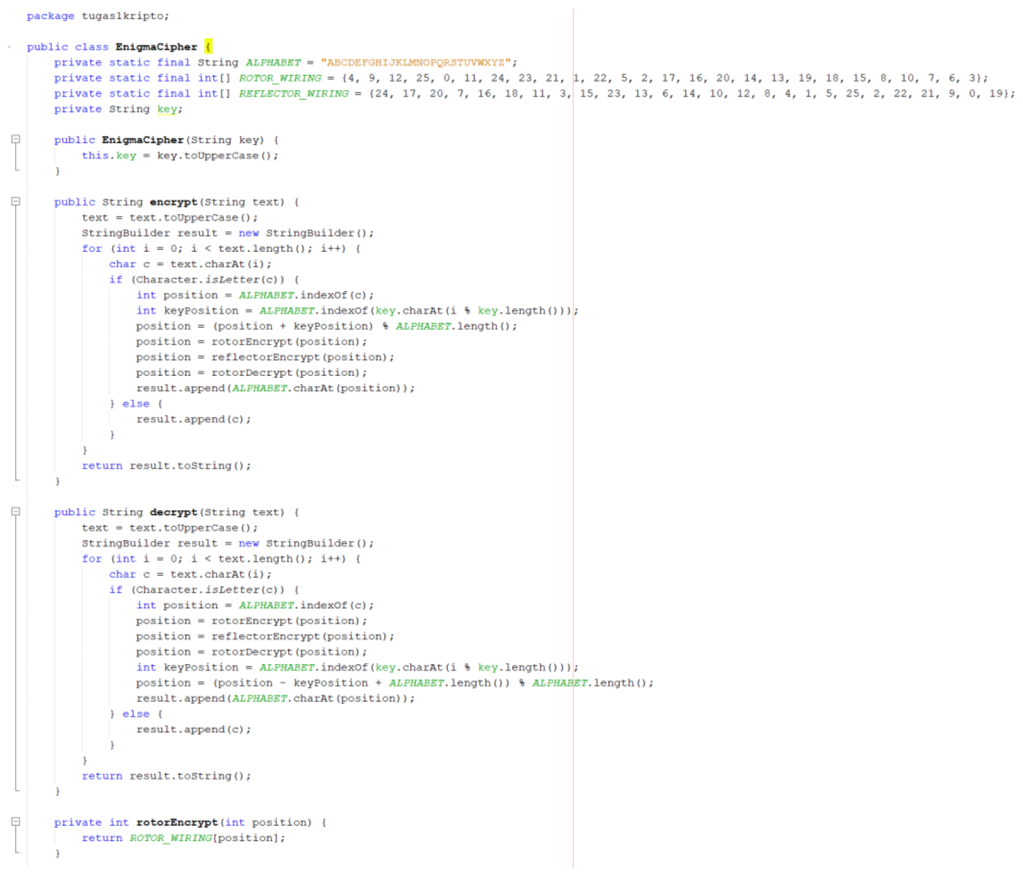
```java
package tugas1kripto;

public class ExtendedVigenereCipher {
    public String encrypt(String text, String keyword) {
        StringBuilder result = new StringBuilder();
        keyword = keyword.toUpperCase();
        int keywordIndex = 0;

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (Character.isLetter(c)) {
                int shift = keyword.charAt(keywordIndex % keyword.length()) - 'A';
                if (Character.isUpperCase(c)) {
                    c = (char) ((c - 'A' + shift) % 26 + 'A');
                } else {
                    c = (char) ((c - 'a' + shift) % 26 + 'a');
                }
                keywordIndex++;
            } else if (Character.isDigit(c)) {
                int shift = keyword.charAt(keywordIndex % keyword.length()) - 'A';
                c = (char) ((c - '0' + shift) % 10 + '0');
                keywordIndex++;
            }
            result.append(c);
        }

        return result.toString();
    }

    public String decrypt(String text, String keyword) {
        StringBuilder result = new StringBuilder();
        keyword = keyword.toUpperCase();
        int keywordIndex = 0;

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (Character.isLetter(c)) {
                int shift = keyword.charAt(keywordIndex % keyword.length()) - 'A';
                if (Character.isUpperCase(c)) {
                    c = (char) ((c - 'A' - shift + 26) % 26 + 'A');
                } else {
                    c = (char) ((c - 'a' - shift + 26) % 26 + 'a');
                }
                keywordIndex++;
            } else if (Character.isDigit(c)) {
                int shift = keyword.charAt(keywordIndex % keyword.length()) - 'A';
                c = (char) ((c - '0' - shift + 10) % 10 + '0');
                keywordIndex++;
            }
            result.append(c);
        }

        return result.toString();
    }
}
```



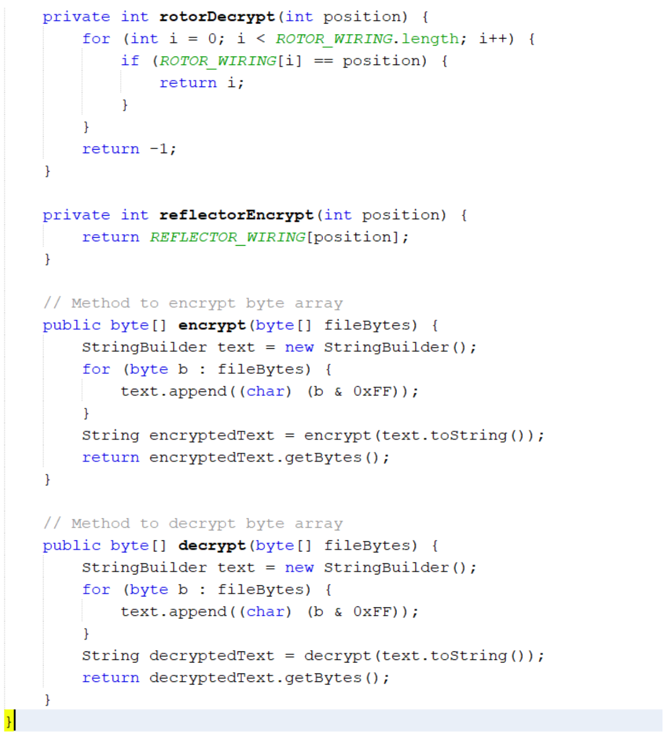
```java
package tugas1kripto;

public class PlayfairCipher {
    private char[][] table;

    public PlayfairCipher() {
        table = new char[5][5];
        String alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ";
        int index = 0;
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                table[i][j] = alphabet.charAt(index++);
            }
        }
    }

    public String encrypt(String text, String keyword) {
        text = text.toUpperCase().replace("J", "I");
        keyword = keyword.toUpperCase().replace("J", "I");
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (Character.isLetter(c)) {
                if (i + 1 < text.length()) {
                    char c2 = text.charAt(i + 1);
                    if (Character.isLetter(c2)) {
                        int[] pos1 = findPosition(c);
                        int[] pos2 = findPosition(c2);

                        if (pos1[0] == pos2[0]) {
                            result.append(table[pos1[0]][(pos1[1] + 1) % 5]);
                            result.append(table[pos2[0]][(pos2[1] + 1) % 5]);
                        } else if (pos1[1] == pos2[1]) {
                            result.append(table[(pos1[0] + 1) % 5][pos1[1]]);
                            result.append(table[(pos2[0] + 1) % 5][pos2[1]]);
                        } else {
                            result.append(table[pos1[0]][pos2[1]]);
                            result.append(table[pos2[0]][pos1[1]]);
                        }
                        i++;
                    } else {
                        int[] pos1 = findPosition(c);
                        result.append(table[pos1[0]][(pos1[1] + 1) % 5]);
                        result.append('X');
                    }
                } else {
                    int[] pos1 = findPosition(c);
                    result.append(table[pos1[0]][(pos1[1] + 1) % 5]);
                    result.append('X');
                }
            }
        }

        return result.toString();
    }
```



```java
    public String decrypt(String text, String keyword) {
        text = text.toUpperCase().replace("J", "I");
        keyword = keyword.toUpperCase().replace("J", "I");
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < text.length(); i += 2) {
            char c1 = text.charAt(i);
            char c2 = (i + 1 < text.length()) ? text.charAt(i + 1) : 'X';

            int[] pos1 = findPosition(c1);
            int[] pos2 = findPosition(c2);

            if (pos1[0] == pos2[0]) {
                result.append(table[pos1[0]][(pos1[1] - 1 + 5) % 5]);
                result.append(table[pos2[0]][(pos2[1] - 1 + 5) % 5]);
            } else if (pos1[1] == pos2[1]) {
                result.append(table[(pos1[0] - 1 + 5) % 5][pos1[1]]);
                result.append(table[(pos2[0] - 1 + 5) % 5][pos2[1]]);
            } else {
                result.append(table[pos1[0]][pos2[1]]);
                result.append(table[pos2[0]][pos1[1]]);
            }
        }

        return result.toString();
    }

    private int[] findPosition(char c) {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (table[i][j] == c) {
                    return new int[] {i, j};
                }
            }
        }
        return null;
    }
}
```

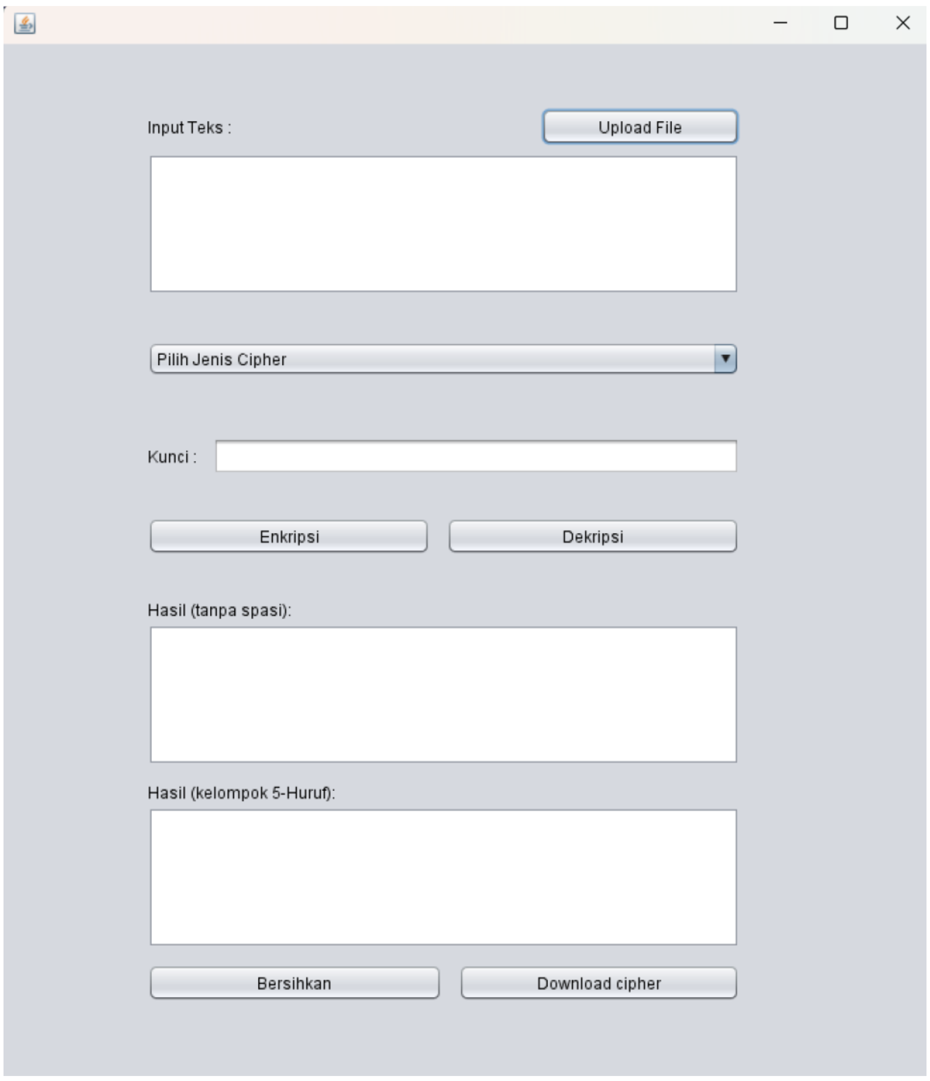


```java
package tugas1kripto;

public class EnigmaCipher {
    private static final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private static final int[] ROTOR_WIRING = {4, 9, 12, 25, 0, 11, 24, 23, 21, 1, 22, 5, 2, 17, 16, 20, 14, 13, 19, 18, 15, 8, 10, 7, 6, 3};
    private static final int[] REFLECTOR_WIRING = {24, 17, 20, 7, 16, 18, 11, 3, 15, 23, 13, 6, 14, 10, 12, 8, 4, 1, 5, 25, 2, 22, 21, 9, 0, 19};
    private String key;

    public EnigmaCipher(String key) {
        this.key = key.toUpperCase();
    }

    public String encrypt(String text) {
        text = text.toUpperCase();
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (Character.isLetter(c)) {
                int position = ALPHABET.indexOf(c);
                int keyPosition = ALPHABET.indexOf(key.charAt(i % key.length()));
                position = (position + keyPosition) % ALPHABET.length();
                position = rotorEncrypt(position);
                position = reflectorEncrypt(position);
                position = rotorDecrypt(position);
                result.append(ALPHABET.charAt(position));
            } else {
                result.append(c);
            }
        }
        return result.toString();
    }

    public String decrypt(String text) {
        text = text.toUpperCase();
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (Character.isLetter(c)) {
                int position = ALPHABET.indexOf(c);
                position = rotorEncrypt(position);
                position = reflectorEncrypt(position);
                position = rotorDecrypt(position);
                int keyPosition = ALPHABET.indexOf(key.charAt(i % key.length()));
                position = (position - keyPosition + ALPHABET.length()) % ALPHABET.length();
                result.append(ALPHABET.charAt(position));
            } else {
                result.append(c);
            }
        }
        return result.toString();
    }

    private int rotorEncrypt(int position) {
        return ROTOR_WIRING[position];
    }
```



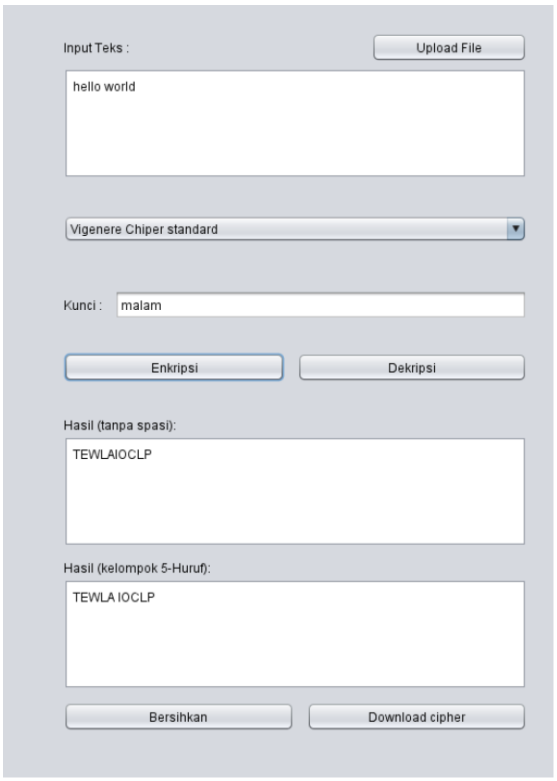
```
    private int rotorDecrypt(int position) {
        for (int i = 0; i < ROTOR_WIRING.length; i++) {
            if (ROTOR_WIRING[i] == position) {
                return i;
            }
        }
        return -1;
    }

    private int reflectorEncrypt(int position) {
        return REFLECTOR_WIRING[position];
    }

    // Method to encrypt byte array
    public byte[] encrypt(byte[] fileBytes) {
        StringBuilder text = new StringBuilder();
        for (byte b : fileBytes) {
            text.append((char) (b & 0xFF));
        }
        String encryptedText = encrypt(text.toString());
        return encryptedText.getBytes();
    }

    // Method to decrypt byte array
    public byte[] decrypt(byte[] fileBytes) {
        StringBuilder text = new StringBuilder();
        for (byte b : fileBytes) {
            text.append((char) (b & 0xFF));
        }
        String decryptedText = decrypt(text.toString());
        return decryptedText.getBytes();
    }
}
```

Tampilan GUI



Vigenere Cipher Standard

Enkripsi


Input Teks : hello world

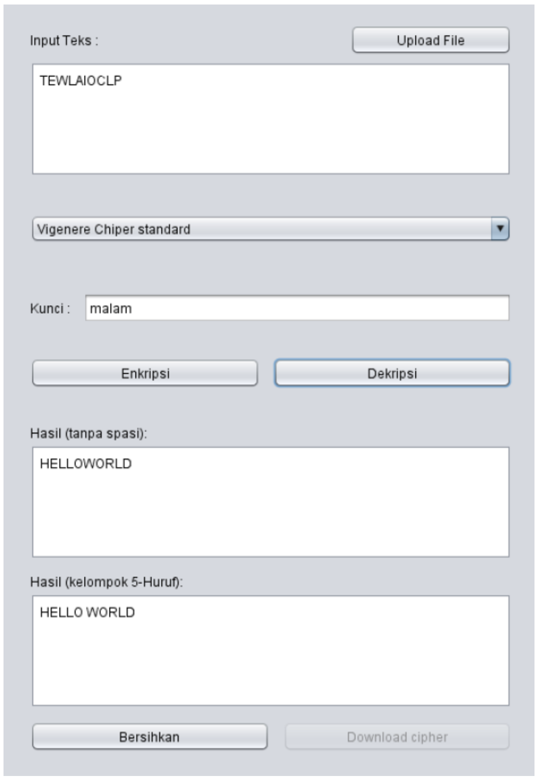
Vigenere Chiper standard

Kunci : malam

Hasil (tanpa spasi): TEWLAIOCLP

Hasil (kelompok 5-Huruf): TEWLA IOCLP

Dekripsi


Input Teks : TEWLAIOCLP

Vigenere Chiper standard

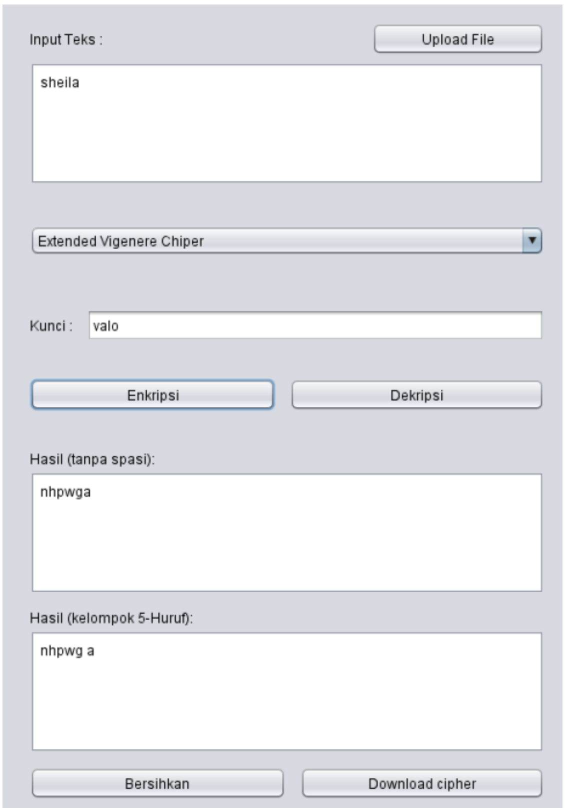
Kunci : malam

Hasil (tanpa spasi): HELLOWORLD

Hasil (kelompok 5-Huruf): HELLO WORLD

Extended Vigenere Cipher

Enkripsi


Input Teks : sheila

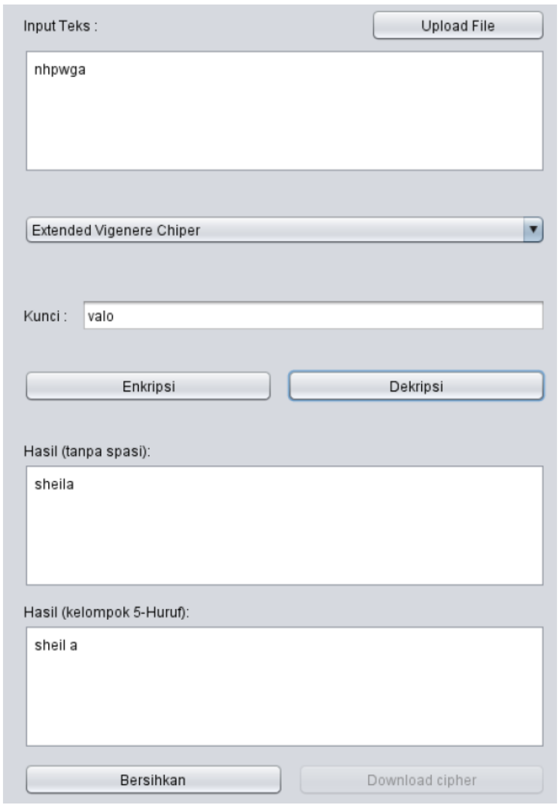
Extended Vigenere Chiper

Kunci : valo

Hasil (tanpa spasi): nhpwga

Hasil (kelompok 5-Huruf): nhpwg a

Dekripsi


Input Teks : nhpwga

Extended Vigenere Chiper

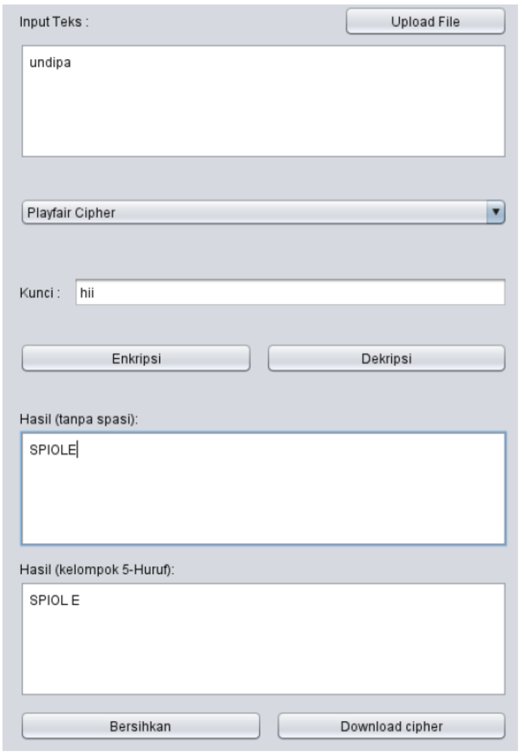
Kunci : valo

Hasil (tanpa spasi): sheila

Hasil (kelompok 5-Huruf): sheil a

Playfair Cipher

Enkripsi


Input Teks : undipa

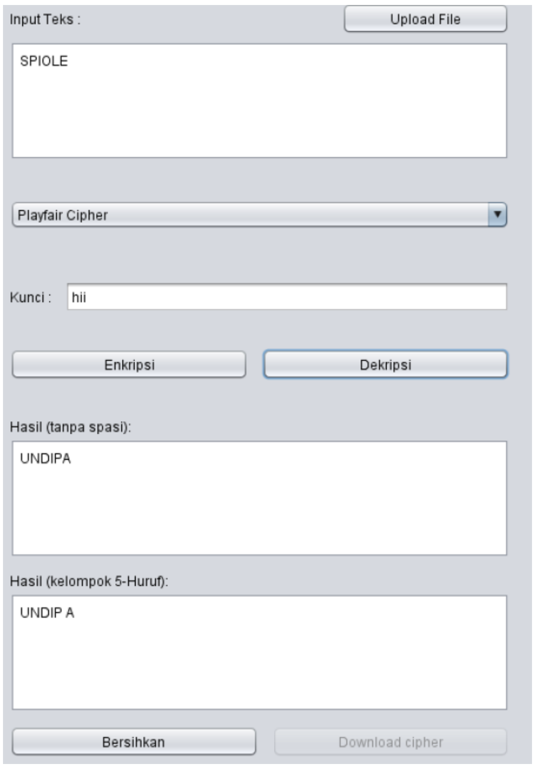
Playfair Cipher

Kunci : hii

Hasil (tanpa spasi): SPIOLE

Hasil (kelompok 5-Huruf): SPIOL E

Dekripsi


Input Teks : SPIOLE

Playfair Cipher

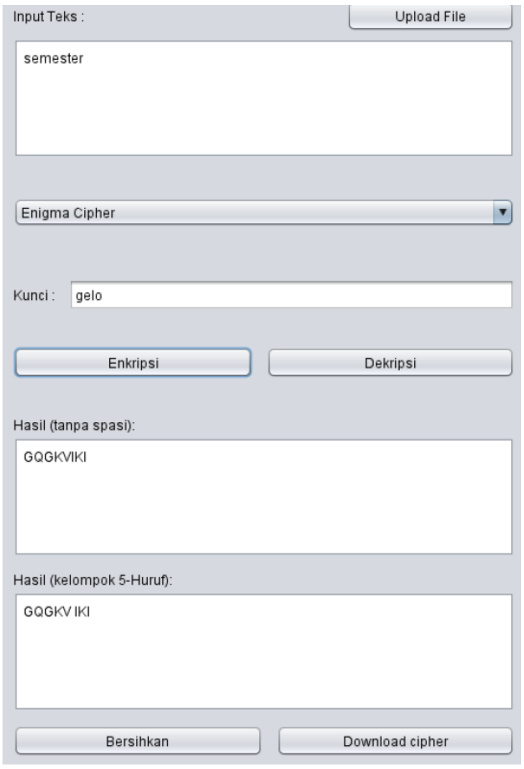
Kunci : hii

Hasil (tanpa spasi): UNDIPA

Hasil (kelompok 5-Huruf): UNDIP A

Enigma Cipher

Enkripsi


Input Teks : semester

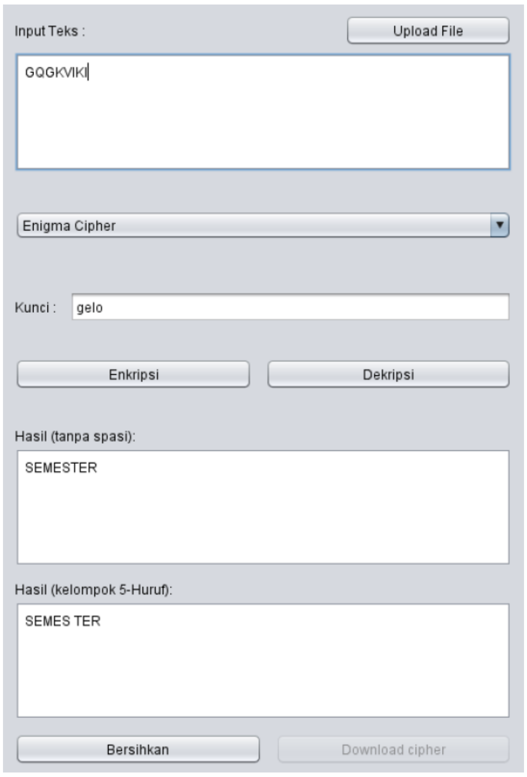
Enigma Cipher

Kunci : gelo

Hasil (tanpa spasi): GQGKVIKI

Hasil (kelompok 5-Huruf): GQGKV IKI

Dekripsi


Input Teks : GQGKVIKI

Enigma Cipher

Kunci : gelo

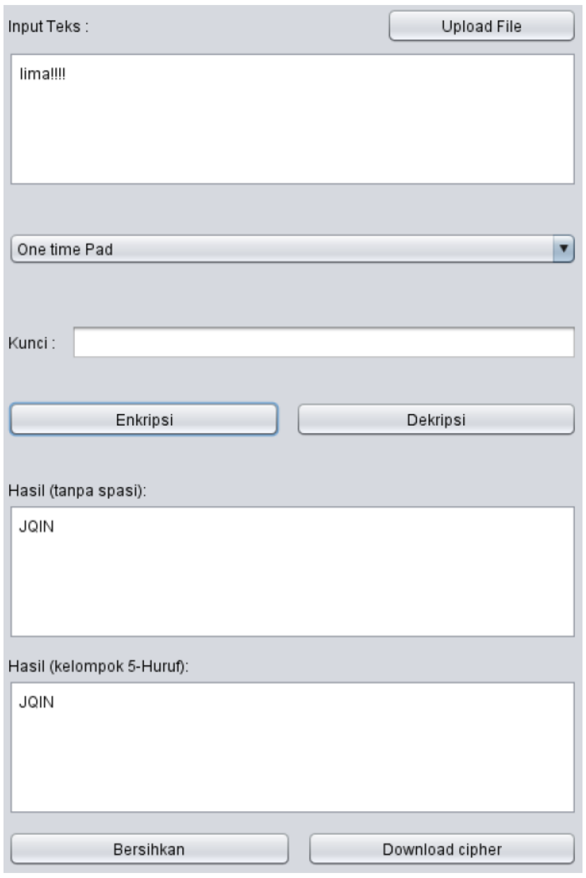
Hasil (tanpa spasi): SEMESTER

Hasil (kelompok 5-Huruf): SEMES TER

One-Time Pad

Pada one-time pad saya, tidak perlu untuk memasukkan kunci manual, jadi jika menekan enkripsi, maka langsung diarahkan untuk memasukkan file kunci ekstension (.txt)

Enkripsi


Input Teks :

lima!!!!

One time Pad

Kunci :

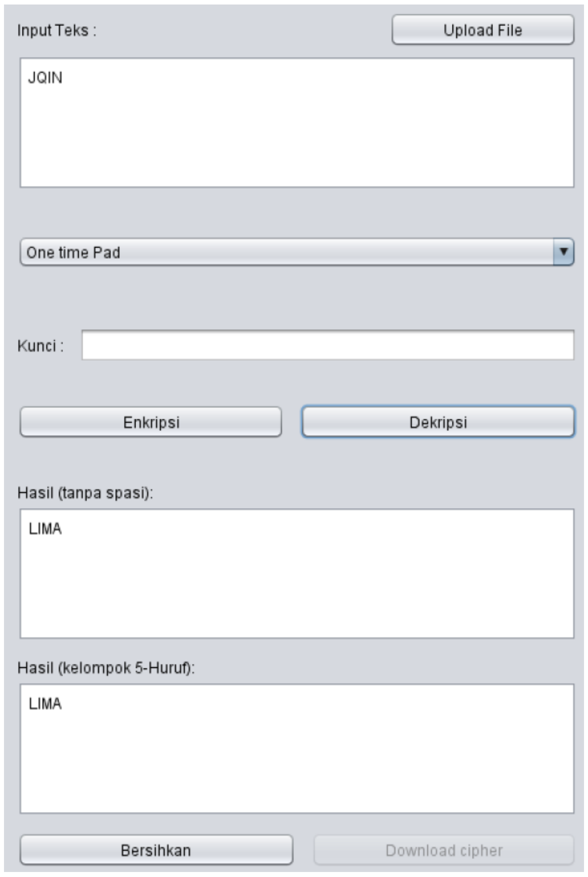
Hasil (tanpa spasi):

JQIN

Hasil (kelompok 5-Huruf):

JQIN

Dekripsi


Input Teks :

JQIN

One time Pad

Kunci :

Hasil (tanpa spasi):

LIMA

Hasil (kelompok 5-Huruf):

LIMA

Link github: repository tugas pertama