MINI PROJECT

STRUKTUR DATA D

Dosen Pengampu:
Muhammad Ilham Perdana, S.Tr.T. M.T.

Source Code:
https://github.com/hisyam99/mini-project-strukdat-2024/tree/main/MINI_PROJECT_1

Muhammad Hisyam Kamil
202210370311060

Daftar Isi

Program Enkripsi Password

Source Code: https://github.com/hisyam99/mini-project-strukdat-2024/tree/main/MINI_PROJECT_1

1. Enkripsi

Langkah 1: Metode encrypt(String password)

```
// Metode untuk memulai proses enkripsi
public static String encrypt(String password) {
    // Langkah 1: Enkripsi menggunakan Step 1
    String step1 = step1(password);
    System.out.println("Step 1 : " + step1);

    // Langkah 2: Enkripsi menggunakan Step 2
    String step2 = step2(step1);
    System.out.println("Step 2 : " + step2);

    // Langkah 3: Enkripsi menggunakan Step 3
    String step3 = step3(step2);
    System.out.println("Step 3 : " + step3);

    // Mengembalikan hasil enkripsi
    return step3;
}
```

- Metode ini merupakan titik masuk untuk memulai proses enkripsi.
- Parameter **password** yang diterima adalah string yang akan dienkripsi.
- Metode ini memanggil tiga metode enkripsi secara berurutan: **step1**, **step2**, dan **step3**.
- Setiap langkah enkripsi akan menampilkan hasilnya ke konsol.
- Metode ini mengembalikan string hasil enkripsi dari langkah terakhir.

Langkah 2: Metode step1(String password)

```
private static String step1(String password) {
    // Membuat antrian untuk menyimpan karakter-karakter hasil enkripsi
    Queue<Character> queue = new LinkedList<>();

    // Mengiterasi setiap karakter dalam password
    for (int i = 0; i < password.length(); i++) {
        char c = password.charAt(i);
        // Jika karakter adalah huruf, geser 5 posisi ke depan dalam urutan abjad
        if (Character.isLetter(c)) {
            char shiftedChar = (char) (c + 5);
            if (Character.isUpperCase(c)) {
                // Jika karakter adalah huruf kapital, atur ulang jika hasil
geserannya melebihi 'Z'
```

```
                if (shiftedChar > 'Z') {
                    shiftedChar = (char) (shiftedChar - 26);
                }
            } else {
                // Jika karakter adalah huruf kecil, atur ulang jika hasil
geserannya melebihi 'z'
                if (shiftedChar > 'z') {
                    shiftedChar = (char) (shiftedChar - 26);
                }
            }
            // Tambahkan karakter yang telah dienkripsi ke dalam antrian
            queue.add(shiftedChar);
        } else {
            // Jika bukan huruf, tambahkan langsung ke dalam antrian tanpa
perubahan
            queue.add(c);
        }
    }
    // Membangun kembali string hasil enkripsi dari antrian
    StringBuilder result = new StringBuilder();
    while (!queue.isEmpty()) {
        result.append(queue.poll());
    }
    return result.toString();
}
```

- Metode ini bertanggung jawab untuk langkah pertama dalam enkripsi.
- Menggunakan struktur data Queue (antrian) dari Java untuk menyimpan karakter-karakter hasil enkripsi.
- Setiap karakter dalam password akan diproses. Jika karakter tersebut merupakan huruf, maka karakter tersebut akan digeser lima posisi ke depan dalam urutan abjad.
- Jika karakter adalah huruf kapital dan hasil geserannya melebihi 'Z', maka akan dilakukan perhitungan ulang dengan memutar ke abjad awal setelah 'Z'.
- Jika karakter adalah huruf kecil dan hasil geserannya melebihi 'z', maka akan dilakukan perhitungan ulang dengan memutar ke abjad awal setelah 'z'.
- Jika karakter bukan huruf, karakter tersebut akan ditambahkan ke antrian tanpa perubahan.
- Setelah semua karakter diproses, hasil enkripsi disusun ulang dari antrian dan dikembalikan sebagai string.

Langkah 3: Metode step2(String password)

```
private static String step2(String password) {
    // Memecah password menjadi tiga bagian
    String depan = password.substring(0, 3);
    String tengah = password.substring(3, password.length() - 3);
    String akhir = password.substring(password.length() - 3);
```

```
    // Memindahkan bagian akhir ke awal, bagian tengah tetap, dan bagian depan ke
belakang
    return akhir + tengah + depan;
}
```

- Metode ini adalah langkah kedua dalam enkripsi.
- Password dipecah menjadi tiga bagian: tiga karakter pertama (**depan**), sisa karakter di tengah (**tengah**), dan tiga karakter terakhir (**akhir**).
- Kemudian, bagian akhir dipindahkan ke awal, bagian tengah tetap, dan bagian depan dipindahkan ke belakang.
- Hasil pergeseran tersebut dikembalikan sebagai string.

Langkah 4: Metode step3(String password)

```
private static String step3(String password) {
    // Membuat tumpukan untuk membalik urutan karakter dalam password
    Stack<Character> stack = new Stack<>();
    for (char c : password.toCharArray()) {
        stack.push(c);
    }
    // Membangun kembali string hasil enkripsi dari tumpukan
    StringBuilder reversedPassword = new StringBuilder();
    while (!stack.isEmpty()) {
        reversedPassword.append(stack.pop());
    }
    return reversedPassword.toString();
}
```

- Metode ini merupakan langkah terakhir dalam enkripsi.
- Menggunakan struktur data Stack (tumpukan) dari Java untuk membalik urutan karakter dalam password.
- Setiap karakter dalam password dimasukkan ke dalam tumpukan.
- Kemudian, karakter-karakter tersebut dikeluarkan dari tumpukan satu per satu, sehingga password awal dibalik.
- Password yang telah dibalik dikembalikan sebagai string.

2. Dekripsi

Langkah 1: Metode decrypt(String password)

```
// Metode untuk memulai proses dekripsi
public static String decrypt(String password) {
    // Langkah 1: Dekripsi menggunakan Step 1
    String decryptedStep1 = step3(password);
    System.out.println("Step 1 : " + decryptedStep1);

    // Langkah 2: Dekripsi menggunakan Step 2
    String decryptedStep2 = step2(decryptedStep1);
    System.out.println("Step 2 : " + decryptedStep2);
```

```
    // Langkah 3: Dekripsi menggunakan Step 3
    String decryptedStep3 = step1(decryptedStep2);
    System.out.println("Step 3 : " + decryptedStep3);

    // Mengembalikan hasil dekripsi
    return decryptedStep3;
}
```

- Metode ini merupakan titik masuk untuk memulai proses dekripsi.
- Parameter **password** adalah string yang akan didekripsi.
- Metode ini memanggil tiga metode dekripsi secara berurutan: **step3**, **step2**, dan **step1**.
- Setiap langkah dekripsi akan menampilkan hasilnya ke konsol.
- Metode ini mengembalikan string hasil dekripsi dari langkah terakhir.

Langkah 2: Metode step1(String password)

```
private static String step1(String password) {
    // Membuat antrian untuk menyimpan karakter-karakter hasil dekripsi
    Queue<Character> queue = new LinkedList<>();

    // Mengiterasi setiap karakter dalam password
    for (int i = 0; i < password.length(); i++) {
        char c = password.charAt(i);
        // Jika karakter adalah huruf, geser 5 posisi ke belakang dalam urutan
abjad
        if (Character.isLetter(c)) {
            char shiftedChar = (char) (c - 5);
            if (Character.isUpperCase(c)) {
                // Jika karakter adalah huruf kapital, atur ulang jika hasil
geserannya kurang dari 'A'
                if (shiftedChar < 'A') {
                    shiftedChar = (char) (shiftedChar + 26);
                }
            } else {
                // Jika karakter adalah huruf kecil, atur ulang jika hasil
geserannya kurang dari 'a'
                if (shiftedChar < 'a') {
                    shiftedChar = (char) (shiftedChar + 26);
                }
            }
            // Tambahkan karakter yang telah didekripsi ke dalam antrian
            queue.add(shiftedChar);
        } else {
            // Jika bukan huruf, tambahkan langsung ke dalam antrian tanpa
perubahan
            queue.add(c);
        }
    }
    // Membangun kembali string hasil dekripsi dari antrian
    StringBuilder result = new StringBuilder();
    while (!queue.isEmpty()) {
        result.append(queue.poll());
```

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

- Metode ini adalah langkah pertama dalam proses dekripsi.
- Menggunakan struktur data Queue (antrian) untuk menyimpan karakter-karakter hasil dekripsi.
- Setiap karakter dalam password akan diproses. Jika karakter tersebut adalah huruf, maka karakter tersebut akan digeser lima posisi ke belakang dalam urutan abjad untuk mendapatkan karakter aslinya.
- Jika karakter adalah huruf kapital dan hasil geserannya kurang dari 'A', maka akan dilakukan perhitungan ulang dengan menggeser kembali ke abjad akhir setelah 'Z'.
- Jika karakter adalah huruf kecil dan hasil geserannya kurang dari 'a', maka akan dilakukan perhitungan ulang dengan menggeser kembali ke abjad akhir setelah 'z'.
- Jika karakter bukan huruf, karakter tersebut akan ditambahkan ke dalam antrian tanpa perubahan.
- Setelah semua karakter diproses, hasil dekripsi disusun ulang dari antrian dan dikembalikan sebagai string.

Langkah 3: Metode step2(String password)

```
private static String step2(String password) {
    // Memecah password menjadi tiga bagian
    String akhir = password.substring(0, 3);
    String tengah = password.substring(3, password.length() - 3);
    String depan = password.substring(password.length() - 3);
    // Memindahkan bagian depan ke belakang, bagian tengah tetap, dan bagian akhir
ke awal
    return depan + tengah + akhir;
}
```

- Metode ini adalah langkah kedua dalam proses dekripsi.
- Password akan dibagi menjadi tiga bagian: tiga karakter pertama (**akhir**), sisa karakter di tengah (**tengah**), dan tiga karakter terakhir (**depan**).
- Kemudian, bagian akhir dipindahkan ke belakang, bagian tengah tetap, dan bagian depan dipindahkan ke awal.
- Hasil pergeseran tersebut dikembalikan sebagai string.

Langkah 4: Metode step3(String password)

```
private static String step3(String password) {
    // Membuat tumpukan untuk membalik urutan karakter dalam password
    Stack<Character> stack = new Stack<>();
    for (char c : password.toCharArray()) {
        stack.push(c);
```

```
    }
    // Membangun kembali string hasil dekripsi dari tumpukan
    StringBuilder reversedPassword = new StringBuilder();
    while (!stack.isEmpty()) {
        reversedPassword.append(stack.pop());
    }
    return reversedPassword.toString();
}
```

- Metode ini adalah langkah terakhir dalam proses dekripsi.
- Menggunakan struktur data Stack (tumpukan) untuk membalik urutan karakter dalam password.
- Setiap karakter dalam password dimasukkan ke dalam tumpukan.
- Kemudian, karakter-karakter tersebut dikeluarkan dari tumpukan satu per satu, sehingga password awal dibalik.
- Password yang telah dibalik dikembalikan sebagai string.

3. Class Main

```
package org.project.v1;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Membuat objek Scanner untuk
input dari pengguna
        boolean exit = false; // Variabel untuk menentukan apakah program harus
keluar

        // Perulangan menu
        while (!exit) {
            System.out.println("Menu:");
            System.out.println("1. Encrypt");
            System.out.println("2. Decrypt");
            System.out.println("3. Keluar");
            System.out.print("Pilih menu (1/2/3): ");
            int menuOption = scanner.nextInt(); // Menerima pilihan menu dari
pengguna
            scanner.nextLine();

            switch (menuOption) {
                case 1: // Jika pilihan adalah Encrypt
                    System.out.println("Masukkan Password untuk Enkripsi: ");
                    String inputEncrypt = scanner.nextLine(); // Menerima password
dari pengguna
                    if (isValidPassword(inputEncrypt)) { // Memeriksa apakah
password valid
                        String encryptedPassword = Encryptor.encrypt(inputEncrypt);
// Melakukan enkripsi password
                        System.out.println("Password terenkripsi: " +
```

```
encryptedPassword); // Menampilkan password terenkripsi
                        exit = askForMainMenu(scanner); // Meminta pengguna apakah
ingin kembali ke menu utama
                    } else {
                        System.out.println("Panjang password harus antara 8 hingga
15 karakter"); // Pesan kesalahan jika password tidak valid
                    }
                    break;
                case 2: // Jika pilihan adalah Decrypt
                    System.out.println("Masukkan Password untuk Dekripsi: ");
                    String inputDecrypt = scanner.nextLine(); // Menerima password
dari pengguna
                    if (isValidPassword(inputDecrypt)) { // Memeriksa apakah
password valid
                        String decryptedPassword = Decryptor.decrypt(inputDecrypt);
// Melakukan dekripsi password
                        System.out.println("Password terdekripsi: " +
decryptedPassword); // Menampilkan password terdekripsi
                        exit = askForMainMenu(scanner); // Meminta pengguna apakah
ingin kembali ke menu utama
                    } else {
                        System.out.println("Panjang password harus antara 8 hingga
15 karakter"); // Pesan kesalahan jika password tidak valid
                    }
                    break;
                case 3: // Jika pilihan adalah Keluar
                    exit = true; // Menandakan agar program keluar dari loop utama
                    break;
                default: // Jika pilihan tidak valid
                    System.out.println("Menu tidak valid."); // Pesan kesalahan
jika pilihan menu tidak valid
                    break;
            }
        }
    }

    // Metode untuk memeriksa validitas password
    private static boolean isValidPassword(String password) {
        return password.length() >= 8 && password.length() <= 15 &&
password.matches("[a-zA-Z]+");
    }

    // Metode untuk meminta pengguna apakah ingin kembali ke menu utama
    private static boolean askForMainMenu(Scanner scanner) {
        System.out.println("Apakah Anda ingin kembali ke menu utama? (Y/n)");
        String response = scanner.nextLine(); // Menerima jawaban dari pengguna
        return response.equalsIgnoreCase("n"); // Mengembalikan false jika jawaban
adalah "n", true jika jawaban adalah yang lainnya
    }
}
```

Penjelasan Class Main:

1. Program dimulai dengan membuat objek **Scanner** untuk menerima input dari pengguna dan variabel **exit** yang digunakan untuk mengontrol apakah program harus keluar dari loop utama.

2. Program memasuki loop utama yang terus berjalan sampai variabel **exit** menjadi **true**.
3. Di dalam loop utama, program menampilkan menu untuk pengguna dan meminta pengguna memilih opsi dengan input angka (1 untuk Enkripsi, 2 untuk Dekripsi, dan 3 untuk Keluar).
4. Bergantung pada pilihan pengguna, program akan:
    - Meminta pengguna memasukkan password.
    - Memeriksa apakah password valid menggunakan metode **isValidPassword**.
    - Jika password valid, program akan menggunakan kelas **Encryptor** atau **Decryptor** untuk melakukan enkripsi atau dekripsi.
    - Hasil enkripsi atau dekripsi kemudian ditampilkan.
    - Pengguna ditanya apakah ingin kembali ke menu utama menggunakan metode **askForMainMenu**.
5. Jika pengguna memilih Keluar (opsi 3), program akan keluar dari loop utama dan berakhir.

Metode **isValidPassword** digunakan untuk memastikan bahwa password yang dimasukkan pengguna memenuhi kriteria panjang (antara 8 hingga 15 karakter) dan hanya terdiri dari huruf alfabet.

Metode **askForMainMenu** digunakan untuk meminta pengguna apakah ingin kembali ke menu utama setelah proses enkripsi atau dekripsi selesai.

4. Output Program

```
"C:\Program Files\OpenJDK\jdk-21.0.1\bin\java.exe"
Menu:
1. Encrypt
2. Decrypt
3. Keluar
Pilih menu (1/2/3): 1
Masukkan Password untuk Enkripsi:
ILOVECILOK
Step 1 : NQTAJHNQTP
Step 2 : QTPAJHNNQT
Step 3 : TQNNHJAPTQ
Password terenkripsi: TQNNHJAPTQ
Apakah Anda ingin kembali ke menu utama? (Y/n)

Menu:
1. Encrypt
2. Decrypt
3. Keluar
Pilih menu (1/2/3): 2
Masukkan Password untuk Dekripsi:
TQNNHJAPTQ
Step 1 : QTPAJHNNQT
Step 2 : NQTAJHNQTP
Step 3 : ILOVECILOK
Password terdekripsi: ILOVECILOK
Apakah Anda ingin kembali ke menu utama? (Y/n)
n

Process finished with exit code 0
```