

LAPORAN PROJEK AKHIR PEMROGRAMAN BERORIENTASI OBJEK
Konversi Mata Uang Menggunakan API dan Konvensional Rate



Penyusun :

Adnan Dwi Kurniawan	: 2310506002
Wiyandra Syaiful Abidin	: 2310506009
Afifatul Ukhrowiyah	: 2310506011
Tafrikhan Rizki Praditya	: 2320506018

PROGRAM STUDI S1 TEKNOLOGI INFORMASI
UNIVERSITAS TIDAR
GANJIL 2023/2024

BAB I

PENDAHULUAN

1.1 Latar Belakang

Nilai tukar Rupiah terhadap mata uang acuan internasional, yaitu Dollar Amerika (USD), mencerminkan salah satu indikator pertumbuhan ekonomi Indonesia. Pertumbuhan ekonomi sendiri mencerminkan perkembangan yang terjadi di berbagai sektor ekonomi negara. Nilai tukar mata uang yang berbeda antar negara, yang dikenal dengan istilah KURS, dipengaruhi oleh hukum permintaan dan penawaran mata uang tersebut di pasar. Setiap tahunnya, kurs ini cenderung mengalami fluktuasi, sehingga perlu adanya penyesuaian terhadap nilai tukar USD. KURS atau nilai tukar adalah instrumen yang sangat penting dalam perekonomian negara dengan sistem ekonomi terbuka, di mana negara tersebut berinteraksi secara ekonomi dengan negara lain, baik dalam perdagangan ekspor maupun impor produk dari sektor-sektor utama perekonomian mereka. KURS menunjukkan nilai mata uang suatu negara terhadap mata uang negara lain, yang diukur dan dinyatakan dalam mata uang yang berbeda. Konversi mata uang mirip dengan mengubah satuan pengukuran, seperti mengonversi berat 1 kg menjadi 1000 gram, di mana yang berubah hanya satuan ukurannya tanpa memengaruhi nilai sebenarnya.

Dalam konteks kehidupan sehari-hari, masyarakat semakin membutuhkan informasi yang akurat dan terkini mengenai nilai tukar mata uang untuk berbagai keperluan, seperti perdagangan, perjalanan internasional, atau investasi. Mengingat peran pentingnya kurs dalam ekonomi global, masyarakat sering kali mengandalkan konversi mata uang yang tepat untuk memaksimalkan transaksi mereka, baik dalam skala individu maupun perusahaan. Oleh karena itu, pembuatan aplikasi konversi mata uang yang menggunakan data nilai tukar dari API serta metode konvensional menjadi sangat relevan. Aplikasi semacam ini memungkinkan pengguna untuk mendapatkan informasi nilai tukar yang real-time dan akurat berdasarkan sumber API yang terpercaya, sekaligus menyediakan opsi untuk menggunakan kurs konvensional sebagai referensi alternatif. Dengan adanya aplikasi semacam ini, pengguna dapat dengan mudah melakukan konversi mata uang sesuai kebutuhan, baik untuk transaksi internasional, investasi, ataupun kegiatan sehari-hari lainnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, rumusan masalah dalam penulisan ini adalah:

1. Bagaimana cara memperoleh data nilai tukar mata uang yang akurat dan terkini untuk keperluan konversi mata uang antar negara?
2. Apa saja kelebihan dan kekurangan dari pembuatan aplikasi konversi mata uang?
3. Bagaimana memastikan aplikasi dapat digunakan dengan mudah dan cepat oleh pengguna untuk kebutuhan konversi mata uang?
4. Apa perbedaan dari konversi mata uang menggunakan API dan Konvensional Rate?

1.1 Tujuan Penulisan

Tujuan penulisan penelitian ini adalah untuk:

1. Membuat aplikasi konversi mata uang yang mudah, cepat, dan tepat untuk digunakan pengguna.
2. Membandingkan kelebihan dan kekurangan antara konversi mata uang menggunakan API dan Konvensional Rate.
3. Mempermudah proses konversi mata uang yang sesuai dengan nilai tukar yang terbaru.

BAB II

HASIL DAN PEMBAHASAN

2.1 Pengertian Aplikasi

Aplikasi KonversiUang adalah sebuah program sederhana yang dirancang untuk membantu pengguna menghitung nilai tukar mata uang dari satu jenis ke jenis lainnya. Aplikasi ini menggunakan data nilai tukar terkini dari layanan API untuk memastikan hasil konversi yang akurat. Dengan fitur input jumlah uang, pilihan mata uang asal, dan tujuan melalui menu dropdown, aplikasi ini dirancang agar mudah digunakan oleh siapa saja. Selain itu, aplikasi ini juga memiliki nilai tukar default (konvensional rate) sebagai cadangan jika terjadi masalah saat mengambil data dari internet, sehingga tetap dapat digunakan kapan saja. Aplikasi ini cocok untuk berbagai keperluan, seperti membantu perhitungan saat bepergian ke luar negeri atau dalam kegiatan belajar. Aplikasi ini juga memiliki beberapa kelebihan dan kekurangan, yaitu :

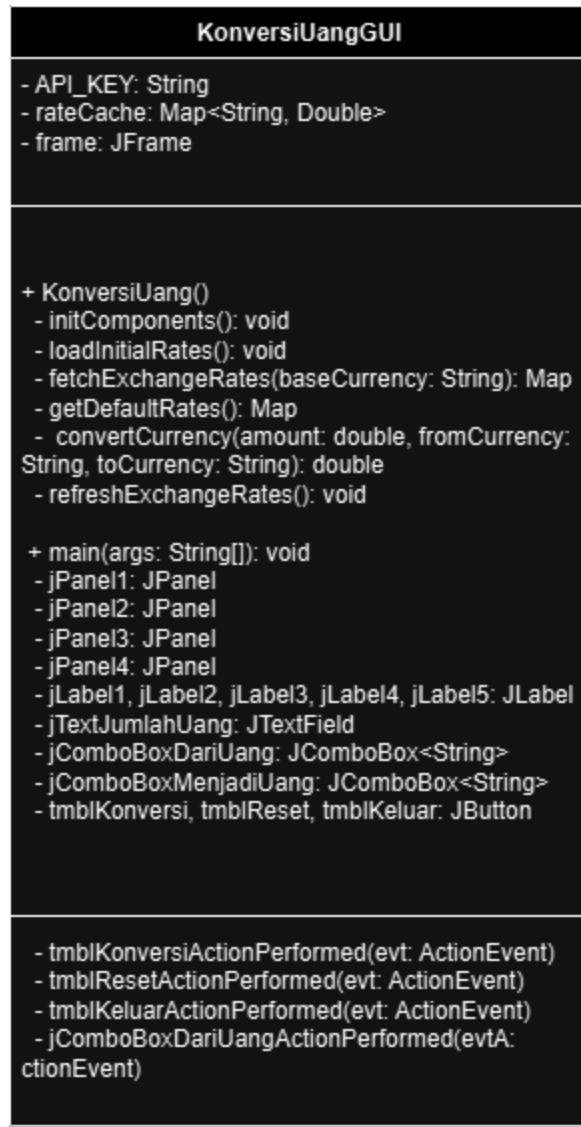
a. Kelebihan

1. Memiliki fitur *fallback* yang berarti dapat diakses secara online (API) dan offline (Konvensional Rate).
2. Memiliki banyak bahasa yang dapat dikonversi secara berulang-ulang.
3. Memiliki tampilan GUI yang mudah digunakan dan dimengerti oleh pengguna.
4. Pada saat menggunakan API, data rate yang digunakan merupakan data yang realtime.

b. Kekurangan

1. Tampilan GUI yang ditampilkan belum di desain secara menarik.
2. Belum terdapat grafik fluktuasi mata uang.

2.2 Diagram Kelas



Gambar 1 Diagram Kelas

Atribut:

1. API_KEY: Kunci API untuk data nilai tukar.
2. rateCache: Cache nilai tukar untuk efisiensi.

Metode:

1. KonversiUang(): Konstruktor, inisialisasi GUI dan data nilai tukar.
2. loadInitialRates(): Memuat nilai tukar awal (dari API atau default).
3. fetchExchangeRates(baseCurrency): Mengambil nilai tukar dari API.

4. `getDefaultRates()`: Menyediakan nilai tukar default jika API gagal.
5. `convertCurrency(amount, fromCurrency, toCurrency)`: Menghitung konversi mata uang.
6. `refreshExchangeRates()`: Memperbarui nilai tukar dari API.

Tampilan GUI:

Komponen: tombol (Konversi, Reset, Keluar), dropdown (mata uang asal dan tujuan), kotak teks untuk input jumlah dan hasil konversi.

2.3 Kode Program dan Hasil Aplikasi

a) Class Currency Rate

```
package KonversiUangTm1_ANT;

public class CurrencyRate {
    private String code;
    private String name;
    private double rate;

    public CurrencyRate(String code, String name, double rate) {
        this.code = code;
        this.name = name;
        this.rate = rate;
    }

    public String getCode() { return code; }
    public String getName() { return name; }
    public double getRate() { return rate; }
}
```

Gambar 2

Kode program tersebut mendefinisikan kelas `CurrencyRate` dalam Java, yang merepresentasikan data mata uang dengan atribut `code` (kode mata uang), `name` (nama mata uang), dan `rate` (nilai tukar). Kelas ini memiliki konstruktor untuk menginisialisasi atribut dan metode getter untuk mengakses nilai-nilai tersebut. Kelas ini digunakan untuk mengelola informasi mata uang dalam aplikasi, seperti konversi mata uang.

b) Class Currency Service

```
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
```

1. **import java.io.*** : Mengimpor semua kelas dari paket `java.io`, yang digunakan untuk operasi input dan output, seperti membaca dan menulis data ke file.
2. **import java.net.HttpURLConnection** : Mengimpor kelas `HttpURLConnection` dari paket `java.net`, yang digunakan untuk membuat koneksi HTTP dan mengelola komunikasi dengan server web.
3. **import java.net.URL** : Mengimpor kelas `URL`, yang digunakan untuk merepresentasikan URL (Uniform Resource Locator) dan memungkinkan program untuk berinteraksi dengan sumber daya di internet.
4. **import java.util.HashMap** : Mengimpor kelas `HashMap`, yang merupakan struktur data untuk menyimpan pasangan kunci-nilai, berguna untuk menyimpan data yang dapat diakses dengan cepat.
5. **import java.util.Map** : Mengimpor antarmuka `Map`, yang merupakan koleksi objek yang menyimpan pasangan kunci-nilai.
6. **import java.util.Properties** : Mengimpor kelas `Properties`, yang merupakan subclass dari `Hashtable` dan digunakan untuk menyimpan data konfigurasi dalam bentuk pasangan kunci-nilai, sering digunakan untuk membaca file konfigurasi.

```
public class CurrencyService {
    private static String API_KEY;
    private Map<String, Double> rateCache = new HashMap<>();
    private static final String CONFIG_FILE = "src/resources/config.properties";
    private static final String RATES_FILE = "src/resources/default_rates.json";
```

Di dalam kelas ini, terdapat variabel statis API_KEY yang kemungkinan digunakan untuk autentikasi saat mengakses API mata uang. Kelas ini juga memiliki rateCache, sebuah HashMap yang menyimpan nilai tukar mata uang untuk meningkatkan efisiensi dengan menghindari permintaan berulang ke API. Selain itu, terdapat dua konstanta, CONFIG_FILE dan RATES_FILE, yang menunjukkan lokasi file konfigurasi dan file default untuk nilai tukar, masing-masing, yang digunakan dalam pengoperasian layanan ini.

```

public CurrencyService() {
    loadConfiguration();
    updateRatesOnStartup();
}

```

Dalam konstruktor ini, dua metode dipanggil. Metode `loadConfiguration()` bertanggung jawab untuk memuat pengaturan yang diperlukan dari file konfigurasi, seperti API key dan parameter lainnya yang diperlukan untuk operasi layanan. Sementara itu, `updateRatesOnStartup()` digunakan untuk memperbarui nilai tukar mata uang saat layanan pertama kali dijalankan, memastikan bahwa data yang digunakan adalah yang terbaru.

```

private void loadConfiguration() {
    Properties prop = new Properties();
    try (InputStream input = new FileInputStream(CONFIG_FILE)) {
        prop.load(input);
        API_KEY = prop.getProperty("api.key");
    } catch (IOException ex) {
        System.err.println("Could not load configuration: " + ex.getMessage());
        // Fallback to default API key if config file is not available
        API_KEY = "7646clf680793787184f9f59";
    }
}

```

Metode `loadConfiguration()` dalam kelas `CurrencyService` memuat pengaturan dari file konfigurasi. Program membuat objek `Properties` untuk menyimpan data, kemudian mencoba membuka file menggunakan `FileInputStream`. Jika berhasil, metode ini memuat properti dan mengambil nilai `API_KEY` dengan kunci `"api.key"`. Jika terjadi kesalahan, seperti file tidak ditemukan, metode ini menangkap pengecualian dan menetapkan `API_KEY` ke nilai default, memastikan aplikasi tetap berjalan meskipun file konfigurasi tidak tersedia.


```

private void updateRatesOnStartup() {
    try {
        // Try to get fresh rates using USD as base currency
        Map<String, Double> freshRates = fetchExchangeRates("USD");
        if (!freshRates.isEmpty()) {
            // If successful, update cache and save to file
            rateCache = freshRates;
            saveRatesToFile(freshRates);
            System.out.println("Exchange rates updated successfully from API");
        } else {
            // If API call returns empty, load from file
            loadDefaultRates();
            System.out.println("Using cached exchange rates from file");
        }
    } catch (Exception e) {
        // If any error occurs, fall back to file/default rates
        loadDefaultRates();
        System.err.println("Failed to update rates from API: " + e.getMessage());
    }
}

```

Pertama, program mencoba mendapatkan nilai tukar terbaru dengan menggunakan USD sebagai mata uang dasar melalui metode `fetchExchangeRates()`. Jika berhasil dan `freshRates` tidak kosong, maka cache nilai tukar (`rateCache`) diperbarui dan nilai tukar disimpan ke file menggunakan `saveRatesToFile()`, disertai pesan sukses. Jika tidak ada data yang diperoleh dari API, metode ini memuat nilai tukar default dari file dengan memanggil `loadDefaultRates()`. Jika terjadi kesalahan selama proses, metode ini juga akan memuat nilai tukar default dan mencetak pesan kesalahan, memastikan aplikasi tetap memiliki data yang dapat digunakan.

```

private void loadDefaultRates() {
    rateCache = loadRatesFromFile();
    if (rateCache.isEmpty()) {
        rateCache = createDefaultRates();
        saveRatesToFile(rateCache);
    }
}

public static String getAPI_KEY() {
    return API_KEY;
}

```

Program memanggil `loadRatesFromFile()` untuk mengisi `rateCache`. Jika `rateCache` kosong, metode ini akan membuat nilai tukar default baru dengan memanggil `createDefaultRates()` dan menyimpannya ke file menggunakan `saveRatesToFile()`. Selain itu, terdapat metode statis `getAPI_KEY()` yang mengembalikan nilai dari `API_KEY`, memungkinkan akses ke kunci API dari

luar kelas. Ini memberikan cara untuk mendapatkan informasi konfigurasi tanpa mengubah status internal kelas.

```
// Save current rates to file
private void saveRatesToFile(Map<String, Double> rates) {
    try (PrintWriter out = new PrintWriter(new FileWriter(RATES_FILE))) {
        out.println("{");
        rates.forEach((currency, rate) ->
            out.printf("  \"%s\": %.2f,%n", currency, rate)
        );
        out.println("}");
    } catch (IOException e) {
        System.err.println("Error saving rates: " + e.getMessage());
    }
}
```

Metode ini menggunakan `PrintWriter` dan `FileWriter` untuk menulis data ke file yang ditentukan oleh `RATES_FILE`.

```
// Load rates from file
private Map<String, Double> loadRatesFromFile() {
    Map<String, Double> rates = new HashMap<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(RATES_FILE))) {
        StringBuilder content = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            content.append(line);
        }
        parseRatesFromResponse(content.toString(), rates);
    } catch (IOException e) {
        System.err.println("Error loading rates: " + e.getMessage());
    }
    return rates;
}
```

Metode `loadRatesFromFile()` bertugas untuk memuat nilai tukar dari file yang ditentukan oleh `RATES_FILE`. Metode ini menginisialisasi sebuah `HashMap` untuk menyimpan pasangan mata uang dan nilai tukar. Kemudian, dengan menggunakan `BufferedReader`, ia membuka file dan membaca setiap baris hingga akhir, menggabungkan semua konten ke dalam sebuah `StringBuilder`. Setelah itu, konten yang dibaca diparsing menjadi format yang dapat digunakan dengan memanggil metode `parseRatesFromResponse()`, yang mengisi `rates` dengan

data yang sesuai. Jika terjadi kesalahan saat membaca file, pengecualian IOException ditangkap dan pesan kesalahan dicetak ke konsol.

```
public Map<String, Double> fetchExchangeRates(String baseCurrency) {
    Map<String, Double> rates = new HashMap<>();

    try {
        String apiUrl = String.format("https://v6.exchangerate-api.com/v6/%s/latest/%s", API_KEY, baseCurrency);
        URL url = new URL(apiUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");

        if (connection.getResponseCode() != HttpURLConnection.HTTP_OK) {
            throw new RuntimeException("HTTP error code : " + connection.getResponseCode());
        }

        BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        StringBuilder response = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            response.append(line);
        }
        reader.close();

        parseRatesFromResponse(response.toString(), rates);
        if (!rates.isEmpty()) {
            saveRatesToFile(rates); // Save the latest rates only if successful
        }
        return rates;
    } catch (Exception e) {
        System.err.println("API Error: " + e.getMessage());
        return new HashMap<>(); // Return empty map to trigger fallback
    }
}
```

Metode ini menginisialisasi sebuah HashMap untuk menyimpan nilai tukar. Kemudian, ia membangun URL API menggunakan API_KEY dan baseCurrency, dan membuka koneksi HTTP dengan metode GET. Jika respons dari server tidak OK, metode ini akan melempar pengecualian. Setelah itu, metode ini membaca respons dari API menggunakan BufferedReader dan menyimpan hasilnya dalam StringBuilder. Konten yang dibaca kemudian diparsing menjadi nilai tukar dengan memanggil parseRatesFromResponse(). Jika nilai tukar berhasil dimuat, metode ini menyimpan data terbaru ke file menggunakan saveRatesToFile(). Jika terjadi kesalahan selama proses, pengecualian ditangkap, pesan kesalahan dicetak, dan metode ini mengembalikan HashMap kosong untuk memicu fallback.

```

private void parseRatesFromResponse(String response, Map<String, Double> rates) {
    String[] supportedCurrencies = {
        "IDR", "USD", "EUR", "GBP", "JPY", "CNY", "INR",
        "RUB", "BRL", "ZAR", "ZMK", "CAD", "NGN", "MXN", "CHF", "AUD"
    };

    for (String currency : supportedCurrencies) {
        String searchStr = "\"" + currency + "\"";
        int currencyIndex = response.indexOf(searchStr);
        if (currencyIndex != -1) {
            int valueStart = currencyIndex + searchStr.length();
            int valueEnd = response.indexOf(",", valueStart);
            if (valueEnd == -1) {
                valueEnd = response.indexOf("}", valueStart);
            }

            try {
                double rate = Double.parseDouble(
                    response.substring(valueStart, valueEnd).trim()
                );
                rates.put(currency, rate);
            } catch (NumberFormatException e) {
                // Skip if rate can't be parsed
            }
        }
    }
}

```

Untuk setiap mata uang dalam array tersebut, metode ini mencari string yang sesuai dalam respons menggunakan `indexOf()`. Jika mata uang ditemukan, metode ini menentukan posisi awal dan akhir dari nilai tukar dengan mencari tanda koma atau kurung tutup. Setelah menemukan nilai tukar, metode ini mencoba untuk mengonversinya menjadi tipe `double` dan menyimpannya dalam `rates`. Jika terjadi kesalahan saat parsing nilai (misalnya, jika nilai tidak dapat dikonversi menjadi angka), pengecualian `NumberFormatException` ditangkap dan proses dilanjutkan tanpa menambahkan nilai tersebut.

```

public Map<String, Double> getDefaultRates() {
    Map<String, Double> rates = loadRatesFromFile();
    if (rates.isEmpty()) {
        rates = createDefaultRates();
        saveRatesToFile(rates);
    }
    return rates;
}

```

Metode `getDefaultRates()` bertugas untuk mendapatkan nilai tukar default. Pertama, metode ini memanggil `loadRatesFromFile()` untuk mencoba memuat nilai tukar dari file. Jika peta rates yang dihasilkan kosong, ini menunjukkan bahwa tidak ada nilai tukar yang tersedia, sehingga metode ini akan memanggil `createDefaultRates()` untuk membuat nilai tukar default. Setelah itu, nilai tukar default yang baru dibuat disimpan ke dalam file menggunakan `saveRatesToFile()`. Selanjutnya, metode ini mengembalikan peta rates, yang berisi nilai tukar yang dimuat dari file atau nilai tukar default yang baru dibuat.

```
private Map<String, Double> createDefaultRates() {  
    Map<String, Double> rates = new HashMap<>();  
    rates.put("USD", 1.0);  
    rates.put("IDR", 20160.50);  
    rates.put("EUR", 1.20);  
    rates.put("GBP", 1.15);  
    rates.put("JPY", 190.60);  
    // ... add other default rates ...  
    return rates;  
}
```

Metode `createDefaultRates()` bertugas untuk membuat dan mengembalikan peta nilai tukar default. Di dalam metode ini, sebuah `HashMap` baru diinisialisasi untuk menyimpan pasangan mata uang dan nilai tukarnya

```

public double convertCurrency(double amount, String fromCurrency, String toCurrency) {
    // Always try to fetch fresh rates from API first
    try {
        Map<String, Double> currentRates = fetchExchangeRates(fromCurrency);
        if (!currentRates.isEmpty()) {
            rateCache = currentRates;
        } else {
            // If API fetch returns empty, fall back to cached rates
            if (rateCache.isEmpty()) {
                rateCache = loadRatesFromFile();
            }
        }
    } catch (Exception e) {
        // If API fails, use cached rates
        if (rateCache.isEmpty()) {
            rateCache = loadRatesFromFile();
        }
    }

    if (!rateCache.containsKey(fromCurrency) || !rateCache.containsKey(toCurrency)) {
        throw new IllegalArgumentException("Invalid Currency");
    }

    double fromRate = rateCache.get(fromCurrency);
    double toRate = rateCache.get(toCurrency);

    return amount * (toRate / fromRate);
}

```

Metode `convertCurrency(double amount, String fromCurrency, String toCurrency)` bertugas untuk mengonversi jumlah uang dari satu mata uang ke mata uang lain. Metode ini mencoba untuk mengambil nilai tukar terbaru dari API dengan memanggil `fetchExchangeRates(fromCurrency)`. Jika nilai tukar berhasil diambil dan tidak kosong, peta nilai tukar tersebut disimpan dalam `rateCache`. Jika pengambilan dari API gagal atau hasilnya kosong, metode ini akan memeriksa apakah `rateCache` juga kosong; jika ya, maka nilai tukar akan dimuat dari file menggunakan `loadRatesFromFile()`.

Setelah memastikan bahwa `rateCache` memiliki nilai tukar yang diperlukan, metode ini memeriksa apakah mata uang yang diberikan valid dengan memastikan bahwa kedua mata uang (`fromCurrency` dan `toCurrency`) ada dalam `rateCache`. Jika salah satu dari mata uang tersebut tidak valid, metode ini akan melempar pengecualian `IllegalArgumentException`. Jika kedua mata uang valid, metode ini mengambil nilai tukar untuk masing-masing mata uang dari `rateCache` dan menghitung nilai konversi dengan rumus $\text{amount} * (\text{toRate} / \text{fromRate})$. Hasil konversi kemudian

dikembalikan. Dengan cara ini, metode ini memastikan bahwa konversi mata uang dilakukan dengan menggunakan nilai tukar yang paling akurat dan terkini yang tersedia.

```
5 package KonversiUangTim1_ANT;
6
7 import javax.swing.*;
8 import java.io.IOException;
9 import java.io.InputStream;
10 import java.io.BufferedReader;
11 import java.io.InputStreamReader;
12 import java.net.HttpURLConnection;
13 import java.net.URL;
14 import javax.swing.JFrame;
15 import javax.swing.JOptionPane;
16 import java.util.HashMap;
17 import java.util.Map;
```

Gambar 3 Import Library

Program dimulai dengan mendefinisikan sebuah package bernama KonversiUangTim1_ANT. Package ini digunakan untuk mengelompokkan kode sehingga lebih terstruktur. Selanjutnya, beberapa library penting diimpor untuk mendukung fungsi-fungsi yang akan digunakan dalam program ini. Berikut library yang digunakan :

1. **javax.swing.*** : Bagian dari Java Standard Library yang menyediakan berbagai komponen GUI (Graphical User Interface), seperti jendela, tombol, label, dan input teks.
2. **java.io.IOException** : Untuk menangani kesalahan yang terjadi selama operasi I/O (*Input/Output*).
3. **java.io.InputStream** : Untuk membaca data dalam bentuk *byte* dari berbagai sumber, seperti file atau koneksi jaringan.
4. **java.io.BufferedReader** : Untuk membaca karakter teks secara efisien dari suatu input stream (misalnya dari file atau jaringan) dengan menggunakan buffer.
5. **java.io.InputStreamReader** : Penghubung antara InputStream (seperti file atau jaringan) dan aliran data karakter, memungkinkan pembacaan byte stream sebagai karakter.
6. **java.net.HttpURLConnection** : Untuk membuat koneksi HTTP (Hypertext Transfer Protocol) dengan server dan mengirimkan permintaan HTTP (seperti GET atau POST).

7. **java.net.URL** : kelas yang merepresentasikan alamat sumber daya di web, misalnya URL untuk mengakses halaman web atau API.
8. **javax.swing.JFrame** : kelas yang digunakan untuk membuat jendela utama dalam aplikasi GUI berbasis Swing.
9. **javax.swing.JOptionPane** : kelas yang menyediakan metode statis untuk menampilkan dialog seperti pesan, konfirmasi, atau input kepada pengguna.
10. **java.util.HashMap** : kelas implementasi dari interface Map yang menyimpan pasangan kunci-nilai (key-value pairs). Ini memungkinkan akses cepat ke elemen berdasarkan kunci.
11. **java.util.Map** : interface yang mendefinisikan struktur data yang menyimpan pasangan kunci-nilai. HashMap adalah salah satu implementasinya.

```
23 public class KonversiUang extends javax.swing.JFrame {
24     // API key for ExchangeRate-API (you MUST replace this)
25     private static final String API_KEY = "7646c1f680793787184f9f59"; // Replace with your actual API key
26
27     // Cache to store exchange rates to reduce API calls
28     private Map<String, Double> rateCache = new HashMap<>();
29
30     // Keep all your existing constructor and initComponents() method unchanged
31     public KonversiUang() {
32         initComponents();
33         // Optional: Load rates when application starts
34         loadInitialRates();
35     }
36
37     private void loadInitialRates() {
38         try {
39             rateCache = fetchExchangeRates("USD");
40         } catch (Exception e) {
41             // Fallback to default rates if API call fails
42             rateCache = getDefaultRates();
43         }
44     }
```

Gambar 4 Kelas Konversi Uang

Setelah itu pada gambar dibuat kelas KonversiUang yang mengimplementasikan JFrame yang memungkinkan kelas ini berfungsi sebagai jendela GUI. Terdapat fungsi **initComponents** yang berguna untuk merancang elemen-elemen yang diperlukan dalam GUI. Lalu di dalam kelas tersebut dibuat atribut **API_KEY** untuk memasukkan API data dan **rateCache** untuk cache nilai tukar untuk efisiensi proses. Lalu terdapat konstruktor untuk inisialisasi komponen dan membaca proses yang akan digunakan.


```

46 // Method to fetch exchange rates from API
47 private Map<String, Double> fetchExchangeRates(String baseCurrency) {
48     Map<String, Double> rates = new HashMap<>();
49
50     try {
51         String apiUrl = String.format("https://v6.exchangerate-api.com/v6/%s/latest/%s", API_KEY, baseCurrency);
52         URL url = new URL(apiUrl);
53         HttpURLConnection connection = (HttpURLConnection) url.openConnection();
54         connection.setRequestMethod("GET");
55
56         // Check for successful connection
57         if (connection.getResponseCode() != HttpURLConnection.HTTP_OK) {
58             throw new RuntimeException("HTTP error code : " + connection.getResponseCode());
59         }
60
61         // Read the response
62         BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
63         StringBuilder response = new StringBuilder();
64         String line;
65         while ((line = reader.readLine()) != null) {
66             response.append(line);
67         }
68         reader.close();
69
70         // Manually parse the JSON-like response
71         String responseStr = response.toString();
72
73         // Supported currencies
74         String[] supportedCurrencies = {
75             "IDR", "USD", "EUR", "GBP", "JPY", "CNY", "INR",
76             "RUB", "BRL", "ZAR", "ZMK", "CAD", "NGN", "MXN", "CHF", "AUD"
77         };

```

Gambar 5 Metode fetchExchangeRates

```

79 // Manually extract rates for supported currencies
80 for (String currency : supportedCurrencies) {
81     String searchStr = "\"" + currency + "\"";
82     int currencyIndex = responseStr.indexOf(searchStr);
83     if (currencyIndex != -1) {
84         // Extract the rate value
85         int valueStart = currencyIndex + searchStr.length();
86         int valueEnd = responseStr.indexOf(",", valueStart);
87         if (valueEnd == -1) {
88             valueEnd = responseStr.indexOf("}", valueStart);
89         }
90
91         try {
92             double rate = Double.parseDouble(
93                 responseStr.substring(valueStart, valueEnd).trim()
94             );
95             rates.put(currency, rate);
96         } catch (NumberFormatException e) {
97             // Skip if rate can't be parsed
98         }
99     }
100 }
101
102 return rates;
103
104 } catch (Exception e) {
105     JOptionPane.showMessageDialog(this,
106         "Error fetching exchange rates: " + e.getMessage(),
107         "Connection Error",
108         JOptionPane.ERROR_MESSAGE);
109     return getDefaultRates(); // Fallback to default rates
110 }
111 }

```

Gambar 6 Metode fetchExchangeRates 2

Pada Gambar 4 dan 5 merupakan metode *fetchExchangeRates* yang berguna untuk menghubungkan cara kerja program kedalam API yang sudah terhubung sehingga proses konversi mata uang aplikasi dapat berjalan dengan baik menggunakan API. Prosesnya pertama-tama program membuat URL yang digunakan untuk meminta data kurs dari API berdasarkan pilihan mata uang asal. Kemudian program akan mengirim permintaan atau request ke API, apabila berhasil, API akan mengirimkan data kurs mata uang dalam bentuk JSON. Setelahnya program akan memproses data yang telah dikirim.

```
113 // Fallback method with default rates
114 private Map<String, Double> getDefaultRates() {
115     Map<String, Double> conversionRate = new HashMap<>();
116     // Base currency is USD (1.0)
117     conversionRate.put("USD", 1.0);
118
119     // Conversion rates (as of November 2024, approximate values)
120     conversionRate.put("IDR", 20160.50); // Indonesian Rupiah
121     conversionRate.put("USD", 1.27); // US Dollar
122     conversionRate.put("EUR", 1.20); // Euro
123     conversionRate.put("JPY", 190.60); // Japanese Yen
124     conversionRate.put("CNY", 9.22); // Chinese Yuan
125     conversionRate.put("INR", 107.50); // Indian Rupee
126     conversionRate.put("RUB", 135.75); // Russian Ruble
127     conversionRate.put("BRL", 7.60); // Brazilian Real
128     conversionRate.put("ZAR", 23.01); // South African Rand
129     conversionRate.put("ZMK", 34.20); // Zambian Kwacha
130     conversionRate.put("CAD", 1.76); // Canadian Dollar
131     conversionRate.put("NGN", 2144.20); // Nigerian Naira
132     conversionRate.put("MXN", 25.35); // Mexican Peso
133     conversionRate.put("CHF", 1.12); // Swiss Franc
134     conversionRate.put("AUD", 1.95); // Australian Dollar
135
136     return conversionRate;
137 }
```

Gambar 7 Metode getDefaultRates

Pada Gambar 6 merupakan metode *getDefaultRates* yang berguna untuk metode alternatif jika nantinya pada saat API error, tidak aktif, dan tidak memiliki internet pengguna dapat tetap menjalankan program dengan baik dan benar. Kurs ini disimpan menggunakan struktur data HashMap yang menyimpan pasangan kode mata uang dan nilai tukarnya.

```
256 private JFrame frame;
257 private void tmb1KeluarActionPerformed(java.awt.event.ActionEvent evt) {
258     // TODO add your handling code here:
259     frame = new JFrame("Exit");
260
261     if(JOptionPane.showConfirmDialog(frame,"Apakah anda ingin keluar?", "Currency Converter",
262         JOptionPane.YES_NO_OPTION)==JOptionPane.YES_NO_OPTION){
263         System.exit(0);
264     }
265
266 }
```

Gambar 8 Tombol Keluar

Pada Gambar 7 merupakan metode **tmblKeluarActionPerformed** yang berguna untuk memberikan aksi untuk keluar dari aplikasi pada saat tombol keluar ditekan oleh pengguna. Saat tombol “Keluar” ditekan, program akan memunculkan pesan “Apakah Anda ingin keluar?”. Jika pengguna memilih “Yes” maka program akan langsung menutup aplikasi.

```

269 private void tmblKonversiActionPerformed(java.awt.event.ActionEvent evt) {
270     double amount;
271     String fromCurrency;
272     String toCurrency;
273     double convertAmount;
274
275     try {
276         amount = Double.parseDouble(jTextJumlahUang.getText());
277     } catch (NumberFormatException e) {
278         JOptionPane.showMessageDialog(this, "Masukkan angka yang valid", "Eksklamasi", JOptionPane.ERROR_MESSAGE);
279         jTextJumlahUang.setText("");
280         jTextJumlahUang.requestFocus();
281         return;
282     }
283
284     fromCurrency = jComboBoxDariUang.getSelectedItem().toString().substring(0, 3);
285     toCurrency = jComboBoxMenjadiUang.getSelectedItem().toString().substring(0, 3);
286
287     try {
288         convertAmount = convertCurrency(amount, fromCurrency, toCurrency);
289         jTextJumlahTerkonversi.setText(String.format("%.2f %s = %.2f %s", amount, fromCurrency, convertAmount, toCurrency));
290     } catch (IllegalArgumentException ex) {
291         JOptionPane.showMessageDialog(this, "Konversi mata uang tidak valid!", "Kesalahan", JOptionPane.ERROR_MESSAGE);
292     }
293 }

```

Gambar 9 Metode Tombol Konversi

Pada Gambar 8 merupakan metode **tmblKonversiActionPerformed** yang berguna untuk memberikan aksi untuk melakukan konversi mata uang yang diinginkan pada saat tombol konversi ditekan oleh pengguna. Program akan memeriksa terlebih dahulu apakah jumlah uang yang dimasukkan adalah benar dan yang dimasukkan adalah angka atau bukan. Apabila bukan , akan ada pesan “Masukkan angka yang valid”.

```

295 private void tmblResetActionPerformed(java.awt.event.ActionEvent evt) {
296     // TODO add your handling code here:
297     jTextJumlahUang.setText("");
298     jTextJumlahTerkonversi.setText("");
299     jComboBoxDariUang.setSelectedIndex(-1);
300     jComboBoxMenjadiUang.setSelectedIndex(-1);
301 }

```

Gambar 10 Metode Tombol Reset

Pada Gambar 9 merupakan metode **tmblResetActionPerformed** yang berguna untuk memberikan aksi untuk melakukan reset mata uang yang sudah dipilih pada saat tombol reset ditekan oleh pengguna.

```

306 public double convertCurrency(double amount, String fromCurrency, String toCurrency) {
307     // If cache is empty or doesn't contain rates, fetch new rates
308     if (rateCache.isEmpty()) {
309         rateCache = fetchExchangeRates("USD"); // Use USD as base currency
310     }
311
312     // If still no rates, use default rates
313     if (rateCache.isEmpty()) {
314         rateCache = getDefaultRates();
315     }
316
317     // Validate currencies
318     if (!rateCache.containsKey(fromCurrency) || !rateCache.containsKey(toCurrency)) {
319         throw new IllegalArgumentException("Invalid Currency: " +
320             (!rateCache.containsKey(fromCurrency) ? fromCurrency : toCurrency));
321     }
322
323     // Perform conversion using fetched rates
324     double fromRate = rateCache.get(fromCurrency);
325     double toRate = rateCache.get(toCurrency);
326
327     return amount * (toRate / fromRate);
328 }

```

Gambar 11 Metode convertCurrency

Pada Gambar 10 merupakan metode *convertCurrency* yang berguna untuk melakukan proses pengkonversian mata uang satu ke mata uang lainnya dengan melakukan pemilihan proses menggunakan API atau konvensional rate terlebih dahulu secara otomatis.

```

350 public static void main(String args[]) {
351     /* Set the Nimbus look and feel */
352     Look and feel setting code (optional)
353
354     /* Create and display the form */
355     java.awt.EventQueue.invokeLater(new Runnable() {
356         public void run() {
357             new KonversiUang().setVisible(true);
358         }
359     });
360 }

```

Gambar 12 Driver Code

Pada Gambar 11 merupakan driver code **main** yang berfungsi untuk menjalankan program aplikasi yang telah dibuat.



Konversi Mata Uang

Jumlah Uang

Dari Mata Uang

Menjadi Mata Uang

Jumlah Terkonversi

Konversi **Reset** **Keluar**

Gambar 13 Tampilan Aplikasi

Pada Gambar 12 merupakan tampilan aplikasi yang telah dibuat dan dijalankan programnya.

BAB III

KESIMPULAN

Program KonversiUang adalah aplikasi konversi mata uang yang memanfaatkan nilai tukar dari layanan API eksternal untuk memberikan hasil yang akurat. Aplikasi ini mengintegrasikan GUI berbasis Swing untuk memudahkan pengguna dalam mengakses fungsionalitas konversi. Aplikasi dirancang dengan fokus pada efisiensi dan pengalaman pengguna, termasuk penggunaan cache (*rateCache*) untuk menyimpan nilai tukar sementara, sehingga mengurangi ketergantungan pada akses internet secara terus-menerus.

Kelas ini memiliki struktur modular yang jelas. Metode *loadInitialRates()* bertugas memuat data nilai tukar awal, baik dari API maupun data default saat API tidak tersedia. Selain itu, *fetchExchangeRates()* digunakan untuk mengambil nilai tukar dari API berdasarkan mata uang dasar yang dipilih. Mekanisme cadangan melalui *getDefaultRates()* memastikan aplikasi tetap berfungsi meski terjadi kendala jaringan atau API tidak dapat diakses.

Antarmuka pengguna dirancang sederhana dengan komponen utama seperti kotak teks untuk input jumlah uang, dropdown untuk memilih mata uang asal dan tujuan, serta tombol untuk melakukan konversi, mereset data, atau keluar dari aplikasi. Fungsi utama *convertCurrency()* menghitung konversi berdasarkan nilai tukar yang tersedia, menjadikan aplikasi ini praktis untuk kebutuhan sehari-hari.

BAB IV

REFLEKSI DIRI

Dalam pengerjaan proyek akhir PBO, saya berperan dalam mengembangkan tombol-tombol dan operasinya untuk memastikan fungsi konversi mata uang berjalan dengan baik. Saya bekerja sama dengan anggota kelompok, menggunakan GitHub untuk mempermudah pembaruan versi file serta memungkinkan fitur undo jika terjadi kesalahan.

Saya juga terlibat dalam proses brainstorming untuk menentukan konsep dan fitur yang akan diterapkan pada aplikasi konversi mata uang ini. Selain itu, saya merancang diagram kelas (class diagram) yang mencakup variabel, metode, serta kelas yang akan digunakan, termasuk hubungan antar kelas.

Di sisi dokumentasi, saya turut berkontribusi dalam pembuatan laporan, khususnya pada bagian diagram kelas dan kesimpulan.

BAB V

DAFTAR PUSTAKA

- Nainggola, Daniel. (2012). *APLIKASI KONVERSI NILAI MATA UANGN MENGGUNAKAN J2ME*. <https://eprints.utdi.ac.id/1476/>.
- Winata, Ardani A. (2019). KONVERSI MATA UANG DAN SUHU MENGGUNAKAN BAHASA PEMPROGRAMAN C++ DENGAN MENGGUNAKAN CODE BLOCKS. https://www.researchgate.net/profile/Andy-Prasetyo/publication/330511877_KONVERSI_MATA_UANG_DAN_SUHU_MENGGUNAKAN_BAHASA_PEMPROGRAMAN_C_DENGAN_MENGGUNAKAN_CODE_BLOCKS/links/5c454a8aa6fdccd6b5bcb8cf/KONVERSI-MATA-UANG-DAN-SUHU-MENGGUNAKAN-BAHASA-PEMPROGRAMAN-C-DENGAN-MENGGUNAKAN-CODE-BLOCKS.pdf.