Documentation for CurrencyConverter application

Overview

This is a **Java Swing GUI application** that allows users to convert an amount of money from one currency to another using real-time data from the <u>ExchangeRate-API</u>. It features a simple interface, interactive buttons, API integration, and clear error handling.

Imported Packages+

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Map;
import com.google.gson.Gson;

import java.util.Map;
import java.util.Map;
import java.util.Map;
```

Each import serves a purpose:

- javax.swing.* and java.awt.*: For building and styling the GUI.
- java.awt.event.*: For handling button click events.
- java.io.* and java.net.*: For HTTP requests to the API.
- java.util.Map: For handling key-value currency data.
- com.google.gson.Gson: For parsing JSON responses from the API.

🗱 Class Breakdown

1. ExchangeRates

• A **plain data holder** ("POJO") class used by GSON to automatically map the JSON response from the ExchangeRate API.

- base: The base currency code (e.g., "USD").
- conversion_rates: A map with keys as currency codes (e.g., "ZAR") and values as exchange rates.

This class interacts only internally when deserializing the JSON response using:

```
ExchangeRates rates = gson.fromJson(json.toString(), ExchangeRates.class);
```

2. CurrencyConverter (extends JFrame)

This is the **main class** that builds the entire app:

- GUI layout
- Event handling
- API communication
- User input validation
- Currency conversion logic

Implements: ActionListener

Attributes:

```
JComboBox<String> to, from;

JTextField amount;

JButton swap, submit;

JLabel resultLabel;
```

These components handle user inputs and display results.

Method Descriptions

1. public CurrencyConverter()

Constructor that calls setupUI() to initialize the window.

2. public void setupUI()

Sets up all **GUI components**:

- Main panel and input panel
- Fonts, colors (including **light blue theme**)

- Dropdowns (JComboBox) for currency selection
- Buttons for converting and swapping currencies
- Result label for output

Creative decisions:

- Uses a calming light blue color scheme
- Buttons and dropdowns are styled for readability
- Grid layout used for structured, clean form inputs

3. private String[] getCurrencies()

Returns a fixed list of commonly used currencies:

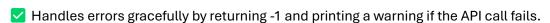
```
{"ZAR", "USD", "EUR", "GBP", "JPY", "AUD", "CAD", "CHF", "CNY", "HKD"}
```

This keeps the dropdown menus simple and avoids overloading users with all 160+ global currencies.

4. public double convert(String base, String target, double amount)

Responsible for:

- Constructing the API URL using the selected base
- Making a GET request
- Reading the JSON response
- Parsing it using Gson
- Getting the conversion rate for the target
- Returning the final converted value



5. public void actionPerformed(ActionEvent e)

Handles:

- Submit button click:
 - o Reads the amount input
 - o Calls convert() with from, to, and amount
 - o Formats and displays result using JLabel
- Swap button click:

- Swaps selections of from and to dropdowns using their indexes
- This improves UX by avoiding the need to manually reselect currencies when the user wants to reverse the conversion.

6. public static void main(String[] args)

Entry point of the application.

Just runs:

new CurrencyConverter();

Which shows the GUI.

Interactions Between Components

Component	Interacts With	Purpose
CurrencyConverter	ExchangeRates	Maps JSON response to Java objects
submit button	convert() + JLabel	Triggers conversion and displays result
swap button	JComboBox from + JComboBox to	Swaps selected currencies
JTextField amount	User input	Input validation before conversion

Creativity & Enhancements

- User-friendly GUI with:
 - Light blue theme
 - o Centered labels and result output
 - o Styled buttons for clarity
- Real-time data via API
- Error Handling:
 - o Graceful failure if input is invalid
 - o Handles API errors without crashing
- Swap functionality makes it intuitive

Conclusion

This project demonstrates a **practical Java GUI app** that combines:

- Frontend (Swing UI)
- Backend (API logic)
- JSON parsing (GSON)
- Good UX decisions (e.g., swapping, color scheme)
- Real-world application of programming concepts