

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 [ExchangeRate-API](#). It features a simple interface, interactive buttons, API integration, and clear error handling.

📦 Imported Packages+

```
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5 import java.io.BufferedReader;
6 import java.io.InputStreamReader;
7 import java.net.HttpURLConnection;
8 import java.net.URL;
9 import java.util.Map;
10 import com.google.gson.Gson;
```

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

```
12
13 /**
14
15  *A simple data class to map the JSON response from the EXchangeRate API.
16  */
17 class ExchangeRates {
18     String base;
19     Map<String, Double> conversion_rates;
20 }
21
22
23 }
```

- 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:

```
160 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:

```
33 JComboBox<String> to, from;
34 JTextField amount;
35 JButton swap, submit;
36 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**

 Handles errors gracefully by returning -1 and printing a warning if the API call fails.

5. public void actionPerformed(ActionEvent e)

Handles:

- **Submit button click:**
 - Reads the amount input
 - Calls convert() with from, to, and amount
 - 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
 - Centered labels and result output
 - Styled buttons for clarity
- **Real-time data** via API
- **Error Handling:**
 - Graceful failure if input is invalid
 - 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