INTERNATIONAL HELLENIC UNIVERSITY
SCHOOL OF ENGINEERING
DEPARTMENT OF INFORMATICS, COMPUTER AND TELECOMMUNICATIONS ENGINEERING

CURRENCY CONVERTER WITH JAVA

Project Team:

Anastasiades Alkinoos (20003)

Zina Eleni (20046)

Lagiokapas Dimitrios (20079)

Makri Styliani (20060)

Supervisors:

Barber Argyrios

Lantzos Theodoros

SERRES, MARCH 2023

Contents

Summary

The purpose of this project was the team implementation of a currency converter/currency application using the Java programming language. The application receives the amount of money as input the user wants to convert, and returns the corresponding exchange rate in real time in the selected currency.

Introduction

In today's era of globalization, currency converters are an indispensable tool for businesses but also for the common person. With the rise and evolution of world trade, the need to convert amounts of money between countries' currencies is huge.

The purpose of this project is to create a JavaFX-based application where it makes conversions easy and fast. The application is designed to be user-friendly, with a simple User Interface that allows anyone who uses it to convert the desired amount from and to any currency they want.

The following sections develop the methodology used to create the application, including the tools used. The architecture of the application and its implementation will also be analyzed, highlighting its basic features and functionality.

Timetable

12/3/2023: We analyzed the specifications that our project should have, in order to think of simple ways to implement it.

13/3/2023: Each member of the team searched and found videos and various sources on the internet, which informed the team and thus facilitated us in simplifying the implementation of the program. Then we started testing and writing code.

14/3/2023: We started with a simple currency converter with 5 currencies, which received values (currencies) and displayed them in the console of our software development environment (**IntelliJ**). However, this display of results did not satisfy us and we thought of adding graphics.

3/15/2023: We looked for ways we can add graphics and ended up downloading an app that gave us this feature (**Scene Builder**). We adapted the previous code we had to the app perspective and created the GUI.

16/3/2023: We decided to create a text file (.txt), which would include all the currencies that exist and import them into the program. We implemented this too and proceeded to improve the code, thus finishing the project.

1. Methodology

To create the currency conversion application, we used a combination of tools and software, including IntelliJ IDEA, Scene Builder, JavaFX and an API for currency rates.

Initially, we used IntelliJ IDEA as the Integrated Development Environment (IDE) to write the Java code that forms the backbone of the program. We chose IntelliJ IDEA because of its easy-to-use features that allowed us to write an efficient, well-written code.

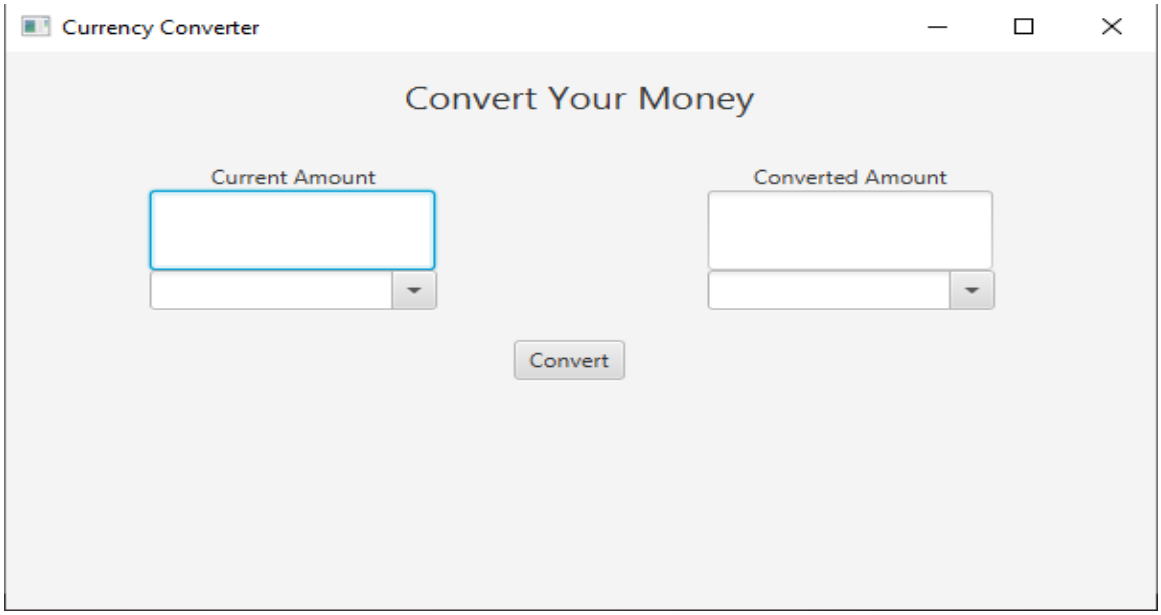
Furthermore, we used Scene Builder to create the UI of the app. Scene Builder is a drag-and-drop Graphical User Interface Builder that integrates JavaFX, allowing us to create a visually beautiful and easy-to-use program. That's the reason we chose Scene Builder which helped us design and modify the UI quickly and easily.

To implement the functionality of the currency converter we used JavaFX, a powerful free-to-use framework for building desktop and mobile applications.

Finally, we used an API to access currency rates in real time. The API, Application Programming Interface, is essentially a set of rules and protocols that allows different software applications to access data of another application or service. In the application we built, we used APIs because we would have easy access to all currencies and exchange rates, which would be time-consuming if we tried to make a database, or if we put all the exchange rates in variables manually.

12. Implementation

Initially, we created a simple program in the terminal, without a window and forms to understand how the API works, since none of us had used it before. Then, after we managed to make it work, we applied the technique to the regular program. The application is a very simple window with 2 drop down lists [1] containing the denominations of coins located within a txt file [2], 2 textbox [3] where one is the user input and the other returns the result by pressing the "Convert" button.



The User Interface of the application was created using JavaFX and Scene Builder [4]. We used the .fxml file provided by Scene Builder to define the building blocks of the UI by design, as well as a Controller class that manages user interactions and the overall implementation of the program.

We used Alpha Vantage [5] CURRENCY_EXCHANGE_RATE, a free-to-use API that returns real-time exchange rate between two currencies. The call to the API is made using sendHttpGETRequest in Java, and the response is converted to Java Object for further processing:

```
public double sendHttpGETRequest(String fromCode, String toCode, double amount) throws IOException {
    String output;
    String GET_URL = "https://www.alphavantage.co/query?function=CURRENCY_EXCHANGE_RATE&from_currency=" + fromCode +
            "&to_currency=" + toCode + "&apikey=LPYNAN4P26VTCTY0";
    URL url = new URL(GET_URL);
    HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
    httpURLConnection.setRequestMethod("GET");
```

The sendHttpGETRequest() function is called by BtnConvert when the user presses the Convert button.

```
public void BtnConvert(ActionEvent e) throws IOException {
    fromCode = LeftUnitMenu.getValue();
    toCode = RightUnitMenu.getValue();
    String amountString = LeftTextField.getText();
    if (isValid(amountString) && isValid(fromCode) && isValid(toCode)) {
        amount = Double.parseDouble(amountString);
        try {
            rate = sendHttpGETRequest(fromCode, toCode, amount);
            DecimalFormat df = new DecimalFormat( pattern: "#.##");
            String resultString = df.format( number: amount * rate);
            RightTextField.setText(resultString);
        } catch (IOException ex) {
            System.out.println("An error occurred while converting currency. Please try again later.");
        }
    } else {
        System.out.println("Please enter a valid amount and select both unit codes.");
    }
    amount = Double.parseDouble(LeftTextField.getText());
    rate= sendHttpGETRequest(fromCode,toCode,amount);
    RightTextField.setText(Double.toString( d: amount * rate));
```

We used the JSON format returned by the API to retrieve exchange rates from a .txt file. JSON (JavaScript Object Notation) [6] is an open format that uses human-readable text to transmit data objects consisting of attribute and array data type pairs.

Our API returned JSON with the following structure:


```
{
    "Realtime Currency Exchange Rate": {
        "1. From_Currency Code": "USD",
        "2. From_Currency Name": "United States Dollar",
        "3. To_Currency Code": "JPY",
        "4. To_Currency Name": "Japanese Yen",
        "5. Exchange Rate": "132.72800000",
        "6. Last Refreshed": "2023-03-16 11:36:22",
        "7. Time Zone": "UTC",
        "8. Bid Price": "132.72630000",
        "9. Ask Price": "132.72960000"
    }
}
```


By using the keys "Realtime Currency Exchange Rate" and "5.Exchange Rate" we specifically select the value we are interested in, in this case the exchange rate.

```
JSONObject obj = new JSONObject(response.toString());
JSONObject exchangeRateObj = obj.getJSONObject( key: "Realtime Currency Exchange Rate");
String exchangeRateString = exchangeRateObj.getString( key: "5. Exchange Rate");
double exchangeRate = Double.parseDouble(exchangeRateString);
return exchangeRate;
```

The logic of the converter was made using the exchange rates from the API and with an operation that calculates the result by multiplying the initial amount by the exchange rate of the two currencies, and places the result in the right Text field:

```
RightTextField.setText(Double.toString( d: amount * rate));
```

3. Results-Conclusion

In conclusion, the application we developed is user friendly and has access to almost all currencies. Through the implementation of this application, we gained valuable experience working with JavaFX and the use of APIs, as well as the development of graphical applications using Scene Builder.

Future updates to this application could include additional features, such as support for various languages, better graphics, better error handling, and user communication.

Overall, the development of the currency converter has been a satisfactory and relatively demanding process, and we hope it will be the beginning of the development of more complex and innovative applications in the future.

Bibliography

[1] JavaFx Scene builder ComboBox Loading
https://www.youtube.com/watch?v=eY7wa3Ur4iQ&ab_channel=TutusFunny

[2] Publications Office – Interinstitutional Style Guide – Annex A7 – Currency codes
https://publications.europa.eu/code/en/en-5000700.htm#fn1

[3] JavaFX Scene Builder Tutorial 33 - TextField and Button
https://www.youtube.com/watch?v=vE5HhzqSOZ8&ab_channel=CodeAmir

[4] JavaFX Scene Builder Tutorial 36 ComboBox
https://www.youtube.com/watch?v=rKv8eavrAio&list=PLxaMIx7eqffLc9mkqFoBFANcZmJVBtzvp&index=38&ab_channel=CodeAmir

[5] Alpha Vantage API support
https://www.alphavantage.co/documentation/

[6] JSON Description
https://el.wikipedia.org/wiki/JSON

[7] Export JavaFX 11, 15 or 17 projects into an executable jar file with IntelliJ [2022]
https://www.youtube.com/watch?v=F8ahBtXkQzU&ab_channel=Randomcode