IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Advanced Level Currency Converter

Dr. J. Narendra Babu[1], Ankitha A[2], Anush Kumar[3], Anushree V[4], Archana G[5], Chandan HK[6]
[1]Professor, Department of Data Science, Sapthagiri NPS University, India
[2,3,4,5,6,]Students, Department of Data Science, Sapthagiri NPS University, India

***Abstract**:* on USD facilitating retrievals and providing ease of updates. This program permits users to convert a specified amount, between any pair of supported currencies check exchange rates and dynamically add or update rates. By using USD as a pivot currency the system guarantees uniform and dependable outcomes.The application follows a menu-driven structure, improving user interaction by guiding them through conversion, rate viewing, and update options. It demonstrates important Java concepts such as collections, modular programming, input handling, and formatted output. This project highlights how simple Java programs can be applied to real-world financial tasks and can be further expanded with features like live API-based rates, a graphical interface, or database support .Overall, the Advanced Currency Converter demonstrates how core programming concepts like data structures, loops, conditional statements, and formatted output can be used to build a real-world, functional application. The project not only performs accurate conversions but is also flexible for future enhancements such as integrating real-time exchange APIs, adding a graphical interface, or expanding currency support. Thus, it stands as a meaningful example of applying Java programming to solve everyday financial needs.

Index Terms: API-based rates., USD, Real-world, Exchange Rates

1. INTRODUCTION

EduLearn is a large Java web-based application created for modern e-learning experience. In this digital age, the demand for open and flexible learning environments is quite high. ~ EduLearn eliminates the instructor-student divide, and makes it easy for teachers to create, manage and share courses in a digital domain Educators can set up courses for students by registering them as users on their own EduLearn site.

The goal of this project is a collaborative digital learning environment centralized to support interactivity. The specific goals include:

- Developing a secure authentication system between students and instructors.
- For management and, of course, distribution of material.
- To explain the relevance of Object-Oriented Programming (OOP) principles: Encapsulation, Inheritance and Polymorphism using an example. ● To have an interface making use of HTML, CSS & Javascript.

2. LITERATURE SURVEY

Currency converter development has progressed from basic hardcoded exchange-rate applications to systems that incorporate live financial APIs and utilize high-accuracy computations. Simple converters typically keep currency rates stored locally within data structures such, as Hash Maps allowing retrievals and straightforward updates. This method works well for limited-scope applications, educational projects and offline environments. Nevertheless studies and current applications emphasize difficulties including keeping exchange rates current guaranteeing computational precision confirming user input validity and creating intuitive interfaces. In-depth research suggests utilizing data providers employing decimal-centric calculations (like Java's Big Decimal) and implementing systematic error management to avoid errors, in monetary transactions.

Contemporary currency exchange platforms additionally incorporate exchange-rate APIs, caching solutions and database management to enhance scalability, dependability and data retention. They frequently feature graphical or web-based user interfaces to boost ease of use and accessibility. Research indicates that converters designed for the future need to accommodate updates, multiple currency support and historical rate evaluations. In contrast basic Java console programs offer a grasp of data structures, modular architecture and financial calculations. Your console-based converter aligns with the core principles found in existing research and can be enhanced by incorporating live APIs, GUI features, and improved precision methods, making it a strong base for more advanced financial software.

3. P ROPOSED METHOD

3.1 Architecture

The proposed method focuses on building a simple yet flexible currency conversion system using Java. The program stores exchange rates in a HashMap, with each currency mapped to its value relative to USD. This allows fast lookups and makes updates easy when users want to modify or add new rates

The User Interface Layer handles all user interactions via a console menu utilizing the Scanner class directing the user to select an operation.

The Application Logic Layer includes the functions that handle converting data checking input validity showing exchange rates and modifying values.

Data Storage Layer uses a HashMap to store and retrieve currency rates efficiently

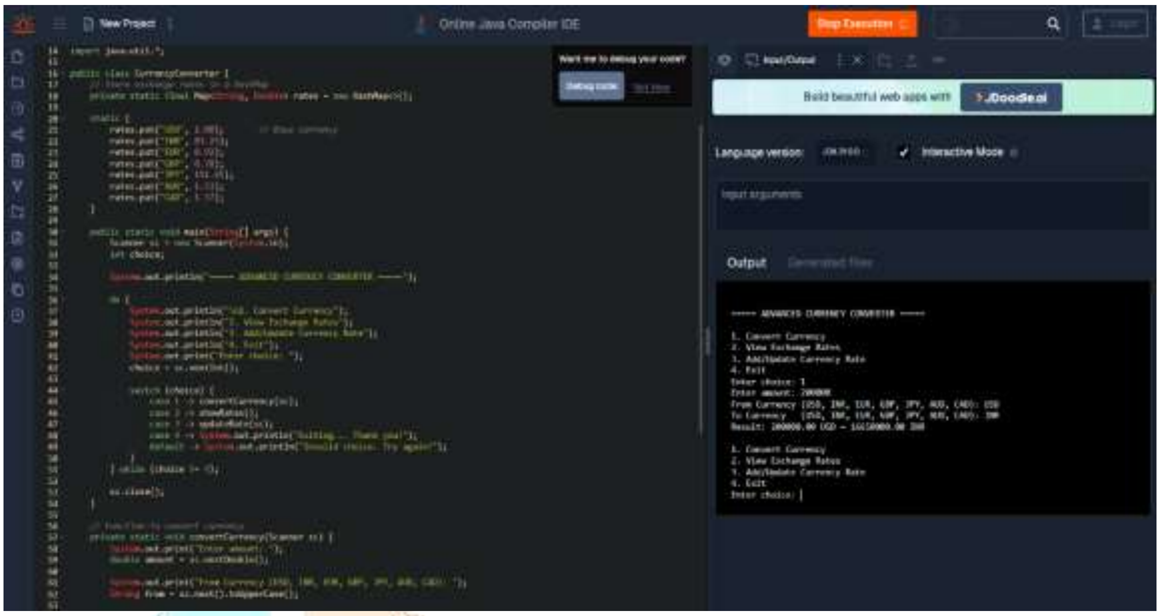
3.2 MODULES DESCRIPTIONS

- Menu Module: Shows all choices. Manages the overall program sequence by guiding the user to the chosen task.
- Input Handling Module: Collects user inputs such as currency codes, amounts, and updated rates using the Scanner class.
- Conversion Module: Executes the currency exchange by changing the source amount into USD and subsequently into the desired currency.
- Currency Exchange Display Module: Presents all currencies together, with their prevailing exchange rates.
- Rates Update Module: Enables users to change currency rates or insert new currency records.
- Data Storage Module: Uses a HashMap to store and retrieve all currency codes and their exchange values efficiently..

4. RESULTS

- The application effectively transforms currency amounts utilizing exchange rates kept within a HashMap.
- Users have the ability to input any value and choose source and target currencies, with the system showing the converted result.
- The two-stage conversion process (source → USD → target) yielded precise outcomes throughout the testing phase.
- The menu-based interface functioned seamlessly enabling users to select conversion check rates or modify options.
- Upon updating the exchange rates the program instantly implemented the figures without needing a restart.

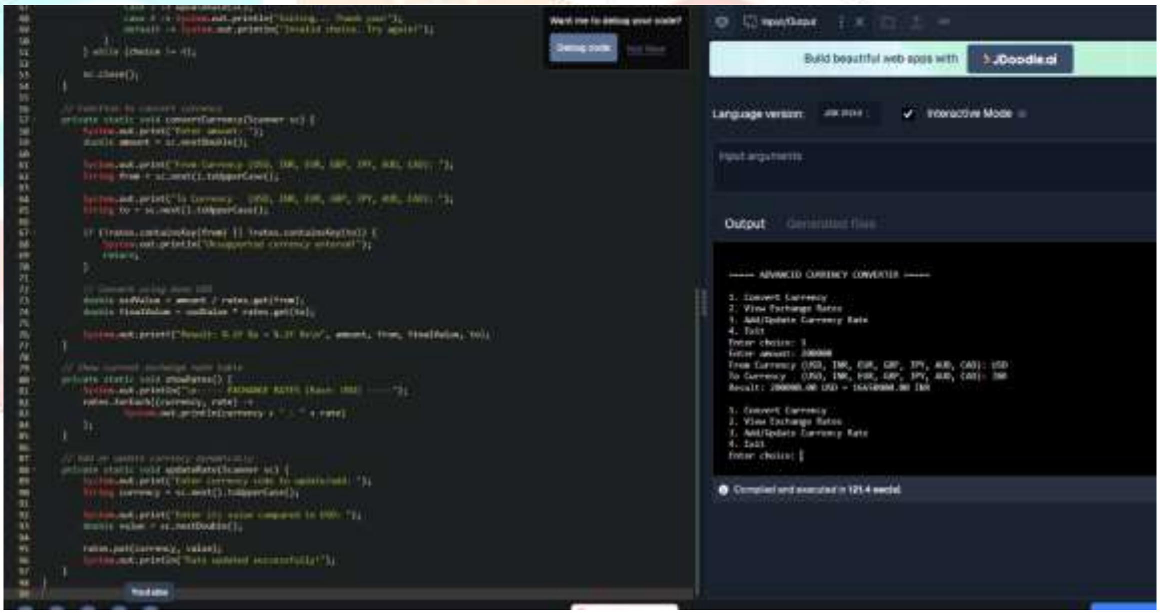
- Every supported currency reacted appropriately. The software managed valid user entries efficiently.
- Overall, the results show the system is functional, reliable, and user-friendly for basic currency conversion tasks.



```java
import java.util.*;

public class CurrencyConverter {
    // Store exchange rates in a HashMap
    private static final Map<String, Double> rates = new HashMap<>();

    static {
        rates.put("USD", 1.00);          // Base currency
        rates.put("INR", [illegible]);
        rates.put("EUR", [illegible]);
        rates.put("GBP", [illegible]);
        rates.put("JPY", [illegible]);
        rates.put("AUD", [illegible]);
        rates.put("CAD", [illegible]);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice;

        System.out.println("----- ADVANCED CURRENCY CONVERTER -----");

        do {
            System.out.println("\n1. Convert Currency");
            System.out.println("2. View Exchange Rates");
            System.out.println("3. Add/Update Currency Rate");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1 -> convertCurrency(sc);
                case 2 -> showRates();
                case 3 -> updateRate(sc);
                case 4 -> System.out.println("Exiting... Thank you!");
                default -> System.out.println("Invalid choice. Try again!");
            }
        } while (choice != 4);

        sc.close();
    }

    // Function to convert currency
    private static void convertCurrency(Scanner sc) {
        System.out.print("Enter amount: ");
        double amount = sc.nextDouble();

        System.out.print("From Currency (USD, INR, EUR, GBP, JPY, AUD, CAD): ");
        String from = sc.next().toUpperCase();
```

```
----- ADVANCED CURRENCY CONVERTER -----

1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: 1
Enter amount: 200000
From Currency (USD, INR, EUR, GBP, JPY, AUD, CAD): USD
To Currency   (USD, INR, EUR, GBP, JPY, AUD, CAD): INR
Result: 200000.00 USD = 16650000.00 INR

1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: |
```

Figure 1: java code of currency converter

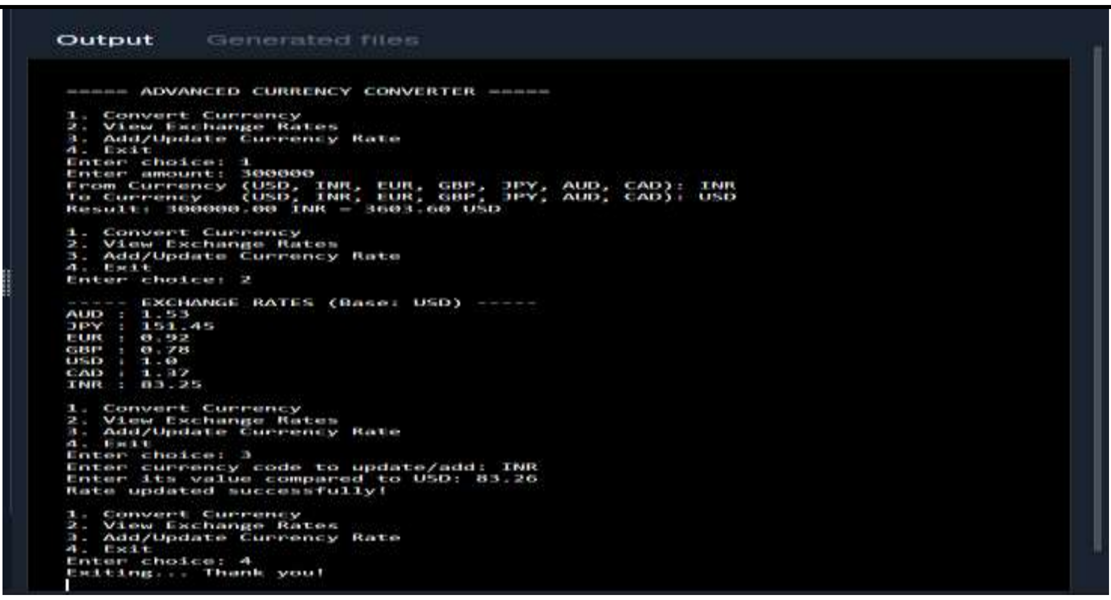


```java
                case 3 -> updateRate(sc);
                case 4 -> System.out.println("Exiting... Thank you!");
                default -> System.out.println("Invalid choice. Try again!");
            }
        } while (choice != 4);

        sc.close();
    }

    // Function to convert currency
    private static void convertCurrency(Scanner sc) {
        System.out.print("Enter amount: ");
        double amount = sc.nextDouble();

        System.out.print("From Currency (USD, INR, EUR, GBP, JPY, AUD, CAD): ");
        String from = sc.next().toUpperCase();

        System.out.print("To Currency   (USD, INR, EUR, GBP, JPY, AUD, CAD): ");
        String to = sc.next().toUpperCase();

        if (!rates.containsKey(from) || !rates.containsKey(to)) {
            System.out.println("Unsupported currency entered!");
            return;
        }

        // Convert using base USD
        double usdValue = amount / rates.get(from);
        double finalValue = usdValue * rates.get(to);

        System.out.printf("Result: %.2f %s = %.2f %s\n", amount, from, finalValue, to);
    }

    // Show current exchange rate table
    private static void showRates() {
        System.out.println("\n----- EXCHANGE RATES (Base: USD) -----");
        rates.forEach((currency, rate) ->
                System.out.println(currency + " : " + rate)
        );
    }

    // Add or update currency dynamically
    private static void updateRate(Scanner sc) {
        System.out.print("Enter currency code to update/add: ");
        String currency = sc.next().toUpperCase();

        System.out.print("Enter its value compared to USD: ");
        double value = sc.nextDouble();

        rates.put(currency, value);
        System.out.println("Rate updated successfully!");
    }
}
```

```
----- ADVANCED CURRENCY CONVERTER -----

1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: 1
Enter amount: 200000
From Currency (USD, INR, EUR, GBP, JPY, AUD, CAD): USD
To Currency   (USD, INR, EUR, GBP, JPY, AUD, CAD): INR
Result: 200000.00 USD = 16650000.00 INR

1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: |
```

Compiled and executed in 121.4 sec(s)

Figure 2: java code of currency converter



```
===== ADVANCED CURRENCY CONVERTER =====
1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: 1
Enter amount: 300000
From Currency (USD, INR, EUR, GBP, JPY, AUD, CAD): INR
To Currency   (USD, INR, EUR, GBP, JPY, AUD, CAD): USD
Result: 300000.00 INR = 3603.60 USD

1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: 2

----- EXCHANGE RATES (Base: USD) -----
AUD : 1.53
JPY : 151.45
EUR : 0.92
GBP : 0.78
USD : 1.0
CAD : 1.37
INR : 83.25

1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: 3
Enter currency code to update/add: INR
Enter its value compared to USD: 83.26
Rate updated successfully!

1. Convert Currency
2. View Exchange Rates
3. Add/Update Currency Rate
4. Exit
Enter choice: 4
Exiting... Thank you!
```

Figure 3: Output of Currency Converter

5. CONCLUSION

The creation of this Java-based currency converter illustrates how basic programming principles can be successfully integrated to create an user-friendly application. The system offers a platform for converting amounts, between various currencies by keeping exchange rates in a HashMap facilitating quick retrievals and simple modifications. The two-phase conversion process—first changing the source currency to USD and then converting USD to the destination currency—guarantees precision and uniformity throughout all supported conversions. The menu-based interface improves user experience by allowing navigation through functions, like currency conversion checking exchange rates and modifying rates.

During the development process significant focus was directed towards modularity and clarity in design. Every component of the system—rate storage, conversion, input processing and updates—was distinctly isolated, ensuring the program is straightforward to maintain and expand. Applying OOP concepts such, as abstraction and encapsulation contributed to organizing the code in a clear comprehensible way. Testing verified that the system converts accurately reacts promptly to user input. Updates exchange rates instantly. These findings emphasize the efficiency of the selected architecture and design strategy.

Overall, this project successfully achieves its objective of creating a functional, flexible, and user-friendly currency converter. It serves as a solid foundation for future enhancements such as integrating real-time exchange rate APIs, incorporating graphical user interfaces, storing data in external files or databases, and improving error handling. With such additions, the system has the potential to evolve from a simple educational tool into a more robust and fully featured financial application.

REFERENCES

1. Oracle. *Java Platform, Standard Edition Documentation*. Available at: https://docs.oracle.com/javase/ (Accessed 2025).
2. Oracle.*ClassHashMapDocumentation*.Availableat: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html
3. Dr.J.NarendraBabu, Steganography based message hiding in image, International journal of advance research in science and engineering, vol.no.4, Issue No.12, December 2015, ISSN-2319-8354.
4. Oracle. *Scanner Class Documentation*. Available at: https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html
5. Schildt, H. (2018). *Java: The Complete Reference*. McGraw-Hill Education.
6. Eck, David J. (2022). *Introduction to Programming Using Java*. Online textbook.
7. GeeksforGeeks. *Currency Converter in Java – Implementation Guide*. Available at: https://www.geeksforgeeks.org

8. Dr.J.NarendraBabu, upcoming Strengths on Internet of Things Journal of Advanced Research in Dynamical & Control systems Vol.11,N0.11,2019
9. Dr.J.Naredra Babu, Enhancement of RVM Using FPGA Journal of Advanced Research in Dynamical & Control systems Vol.11,No.11,2019
10. Dr.J.NarendrBabu, BrainSignal processing : Analysis, Technologies and Application Journal of Advanced Research in Dynamical & Control systems (JARDCS)Vol.11,No: 12,2019.
11. TutorialsPoint. *Java Programming Language Tutorials*. Available at: https://www.tutorialspoint.com/java/
12. W3Schools.*Java Basics and Methods*. Available at: https://www.w3schools.com/java/
13. Baeldung. *Guide to Java Collections and HashMap*. Available at: https://www.baeldung.com/java-hashmap

14.Dr.J.Narendra Babu, et.al- Experimental detection of vehicles for toll gate billing, International journal of Science technology and management, vol.no.4, Issue No.12, December 2015, ISSN 2394-1537

AUTHORS BIOGRAPHY

Dr. J. Narendra Babu is a seasoned academician with over 28 years of experience in teaching and the software industry. He currently serves as a Professor in the Department of Data Science at Sapthagiri NPS University,Bangalore. He holds B.Tech, M.Tech, and Ph.D. degrees. He has published extensively in reputed journals and conferences and plays a key role in mentoring students and coordinating academic activities.