```
import java.io.IOException;

import java.util.*;

import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import com.google.gson.Gson;

public class CurrencyConverter {
    Map<String, Double> exchangeRates;
    List<String> favoriteCurrencies;


    private final String apiUrl = "https://v6.exchangerate-api.com/v6/c3489c88f5f2648e0cf0f360/latest/";

    public CurrencyConverter() {
        exchangeRates = new HashMap<>(); // Initialize the exchangeRates map
        favoriteCurrencies = new ArrayList<>();
    }
       private Map<String, Double> fetchExchangeRates() throws IOException {
        String url = apiUrl + "USD";



        try (CloseableHttpClient client = HttpClients.createDefault()) {
            HttpGet request = new HttpGet(url);

            try (CloseableHttpResponse response = client.execute(request)) {
                String jsonResponse = EntityUtils.toString(response.getEntity());

                // Print the response to see its structure
                System.out.println("API Response: " + jsonResponse);

                // If the response is a string, you need to handle it accordingly
                if (jsonResponse.startsWith("{")) {
                    ExchangeRateApiResponse apiResponse = new Gson().fromJson(jsonResponse, Exchange
RateApiResponse.class);
                    if (apiResponse.conversion_rates != null) {
                        return apiResponse.conversion_rates;
                    } else {
                        System.out.println("No rates found in the API response.");
                        return Collections.emptyMap();
                    }
                } else {
                    System.out.println("Unexpected response format: " + jsonResponse);
                    return Collections.emptyMap();
                }
            }
        } catch (IOException e) {
            System.out.println("Error fetching exchange rates from the API: " + e.getMessage());
            return Collections.emptyMap();
```

```
    }
}

private static class ExchangeRateApiResponse {
    Map<String, Double> conversion_rates;

}

public void addFavoriteCurrency(String currency) {
    if (!exchangeRates.containsKey(currency)) {
        System.out.println("Invalid currency code. Cannot add to favorites.");
    } else if (!favoriteCurrencies.contains(currency)) {
        favoriteCurrencies.add(currency);
        System.out.println(currency + " added to favorites.");
    } else {
        System.out.println(currency + " is already in favorites.");
    }
}

public void viewFavoriteCurrencies() {
    System.out.println("Favorite currencies:");
    for (String currency : favoriteCurrencies) {
        System.out.println(currency);
    }
}

public void updateExchangeRate(String currency, double newRate) {
    if (exchangeRates.containsKey(currency)) {
        exchangeRates.put(currency, newRate);
        System.out.println("Exchange rate for " + currency + " updated.");
    } else {
        System.out.println(currency + " is not found in the exchange rates.");
    }
}

public double convert(String fromCurrency, String toCurrency, double amount) {
    if (!exchangeRates.containsKey(fromCurrency) || !exchangeRates.containsKey(toCurrency)) {
        System.out.println("Invalid currency code(s). Cannot perform conversion.");
        return -1; // Indicates an error
    }

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

    return (amount / fromRate) * toRate;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    CurrencyConverter converter = new CurrencyConverter();
    try {
        converter.exchangeRates = converter.fetchExchangeRates();
    } catch (IOException e) {
        System.out.println("Error fetching exchange rates from the API: " + e.getMessage());
        scanner.close();
```

```
            return;
        }

        int choice;
        do {
            System.out.println("\n1. Add Favorite Currency");
            System.out.println("2. View Favorite Currencies");
            System.out.println("3. Update Exchange Rate");
            System.out.println("4. Convert Currency");
            System.out.println("0. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter the currency code to add to favorites: ");
                    String addCurrency = scanner.next().toUpperCase();
                    converter.addFavoriteCurrency(addCurrency);
                    break;
                case 2:
                    converter.viewFavoriteCurrencies();
                    break;
                case 3:
                    System.out.print("Enter the currency code to update exchange rate: ");
                    String updateCurrency = scanner.next().toUpperCase();
                    System.out.print("Enter the new exchange rate: ");
                    double newRate = scanner.nextDouble();
                    converter.updateExchangeRate(updateCurrency, newRate);
                    break;
                case 4:
                    System.out.print("Enter the amount: ");
                    double amount = scanner.nextDouble();
                    System.out.print("Enter the currency to convert from: ");
                    String from = scanner.next().toUpperCase();
                    System.out.print("Enter the currency to convert to: ");
                    String to = scanner.next().toUpperCase();
                    double convertedAmount = converter.convert(from, to, amount);
                    if (convertedAmount != -1) {
                        System.out.printf("%.2f %s is equal to %.2f %s\n", amount, from, convertedAmount, to);
                    }
                    break;
                case 0:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice!");
                    break;
            }
        } while (choice != 0);

        scanner.close();
    }

}
```