Digital Custody - Coding Challenge

Name - Divyank Sameer Shah
Email Id - divyankshah97@gmail.com

Task Overview -

Implement a Java program that monitors a blockchain address on Ethereum, fetches historical transactions, and computes some basic statistics, e.g., how many swaps have been performed, the overall transaction volume in USD of that address, ...
The program should be able to calculate the account balance of all assets in USD.

# Ethereum Address Monitor

A Java + Spring Boot application that monitors any Ethereum address, fetches its historical transactions using Infura, detects swap count, and computes total transaction volume in USD using real-time pricing from CoinGecko.

# Features

1) View account balance of all addresses in USD
2) Fetches historical transactions.
3) Calculate how many swaps have been performed.
4) Computing total transaction volume in USD using real-time pricing from CoinGecko.

# TechStack and Purpose

Techstack	Purpose
Java 17	Core lang with long term security support
Spring boot	Rest API
Web3j	Ethereum blockchain access via Infura
Infura	Ethereum JSON-RPC provider
CoinGecko	Real time usd conversion of ETH

# Approach

1. Created a simple spring boot application with a service layer named as EthereumService and a controller class Monitoring controller so that we could expose json response using REST APIs.
2. The application connects to Ethereum mainnet using Infura a JSON-RPC provider.
3. Created a free account at Infura. Signed up and added Ethereum, saved the API key and mainnet ID in our Java application.
4. The Java app connects to Ethereum mainnet using Web3Client dependency and Infura a JSON-RPC provider.
5. Tested the connection with /status call sending request.
6. Built below Rest API calls with their functionalities

# APIs defined

Sr.no	Rest API call	Purpose
1.	/status	Checks if our Java app is connected to Infura
2.	/address/{ethAddress}/balance	Returns ETH balance as well as its USD equivalent of the ethAddress provided.
3.	/transactions	Scans blocks and gets transaction history within start block and end block
4.	/binance/swaps	a) Iterates through all transactions between startBlock and endBlock. b) Checks each transaction’s input data to identify known swap function signatures (from Uniswap, SushiSwap, etc.). c) The function returns a list of all detected swap transactions, or can be used to compute the **total number of swaps** in the specified block range.
5.	/binance/usd-volume	a) Extracts the value of the transaction in ETH of binance hot wallet. b) Accumulates all values and converts it to realtime USD with help of coinbase API

Approaches for few important features -

1) **Fetching transaction history -**
   - Iterates through blocks from startBlock to endBlock in chunks of size chunkSize.
   - For each chunk:

     Fetches blocks one by one with full transaction details using web3j.ethGetBlockByNumber(...).
   - Scans all transactions in each block.
   - Filters transactions where the given address is either the sender (from) or recipient to.
   - For each matching transaction, it extracts:

     Block number, Transaction hash, From and To addresses, Value transferred in ETH and Timestamp of the block

Why Chunking?

- Improves performance and avoids timeouts when scanning large block ranges.
- Prevents Infura rate limiting by reducing the number of requests per second.
- Allows for parallelism or pagination in future extensions.

2) **Calculating number of swaps have been performed -**

The method getSwapTransactions(startBlock, endBlock) is used to scan Ethereum blocks and identify transactions that invoke known swap functions on popular DEXs (like Uniswap and SushiSwap).

- It Iterates through blocks from startBlock to endBlock.
- Retrieves all transactions within each block.
- Parses each transaction's input field to extract the function signature.
- Matches the signature against a known list of swap function signatures.
- Optionally filters by a known address (e.g., used Binance hot wallet in the code).

Method - Swap Function Signatures:
The method checks for the following well-known swap functions (using their method selectors):

3) **Computing total transaction volume in USD -**

- Extracts the ETH value transferred.
- Multiplies the ETH amount by the current ETH/USD price (fetched via getEthPriceUSD()).
- Aggregates the total USD value for all such transactions.
- ETH/USD price is assumed to be fetched via a separate method (getEthPriceUSD()), which can return static or real-time prices.
- A delay (Thread.sleep(300)) is used after each block to respect Infura rate limits.

How to run and test the application -

1) **Import the project as a gradle project or a spring boot application.**
2) **Run Gradle build command.**
3) **Right click and Run as a Spring boot application.**
4) **Test all APIs listed above - can change the startblock,endblock, ETH address.**