Assignment 2

Design a Crypto Trading Application. In the crypto market there are a lot of coins that have been listed and operations like updatePrice, buy, sell, and increaseVolume can be performed on them. Traders buy and sell coins on a daily basis and maintain a portfolio of every coin they own. Design a system with appropriate models to represent the data and improve data querying.

Resources:

1. You will be given a CSV file with the list of all the coins present in the market along with their details.
2. You will be given a CSV file with the list of all the Traders.
3. Another JSON file will be provided consisting of a large number of transaction requests that must be processed.:
    a. BUY
    b. SELL
    c. UPDATE PRICE
    d. ADD VOLUME

Your program must follow the following flow.

1. On load, data from the CSV files must be loaded into the memory.
2. A thread must be created to process the transactions where each transaction is processed by a separate thread to improve the efficiency of the system.
3. While the transactions are being processed, the user is shown a menu to run the following queries.
    a. Given the name or code of a coin, retrieve all its details.
    b. Display top 50 coins in the market based on price.
    c. For a given trader, show his portfolio.
    d. For a given trader, display the total profit or loss they have made trading in the crypto market.
    e. Show top 5 and bottom 5 traders based on their profit/loss.
4. Keep in mind that every transaction must be thread-safe to ensure data integrity,

The following constraints must be kept in mind

1. The volume of coins available for trade should never go below zero or exceed the maximum value.
2. If at any point, a BUY request is made where the number of coins available is less than the requested amount, the BUY request must transition to a pending state where it will wait for another trader to SELL their coins or wait for the volume to be increased.
3. Ensure that a user is only allowed to SELL coins that they have previously bought.

Since cryptocurrency transactions are compute-heavy, where for every transaction a block hash must be generated and stored, to simulate this behavior use the method provided below to generate a random transaction hashcode.

```java
/**
     * Method generates the unique block hash required
     * for transactions made using the cryptocurrencies
     * @return - string representing the transaction hashcode
     */
    private String getBlockHash() {
        String SALTCHARS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
        StringBuilder transactionHash = new StringBuilder();
        Random rnd = new Random();

        /**
         * Introducing delay mimicking complex
         * calculation being performed.
         */
        for(double i=0;i<199999999; i++){
            i = i;
        }

        while (transactionHash.length() < 128) {
            int index = (int) (rnd.nextFloat() * SALTCHARS.length());
            transactionHash.append(SALTCHARS.charAt(index));
        }

        String hashCode = transactionHash.toString();
        return "0x" + hashCode.toLowerCase();
    }
```

Example Transaction File

```
[
    {
        type: BUY,
        data: {
            coin: "BTC",
            quantity: 2000,
            wallet_address: 0x5a1fcde6a86ea0dd483f33d81f35000f
        }
    },
    {
        type: SELL,
        data: {
            coin: "BTC",
            quantity: 2000,
            wallet_address: 0x5a1fcde6a86ea0dd483f33d81f35000f
        }
    },
    {
        type: UPDATE_PRICE,
        data: {
            coin: "BTC",
            price: 24.34
        }
    },
    {
        type: ADD_VOLUME,
        data: {
            coin: "BTC",
            volume: 20000
        }
    }
]
```