Name: Ibrahim taha Ibrahim saleh

Sec : 1

Code : 220100273

Lab1

- **Code: -**

```
"use strict";

const SHOW = "SHOW_PRICE";
const UPDATE = "UPDATE_USD_PRICE";

let fs = require('fs');
let EventEmitter = require('events');

function readJsonFromFile(fileName) {
   try {
      let data = fs.readFileSync(fileName, 'utf8');
      return JSON.parse(data);
   } catch (err) {
      console.error("Error reading JSON file:", err);
      return {};
   }
}

class CurrencyConverter extends EventEmitter {
```

```
static calculateRates(usdPrices) {
    let rates = {};
    let usdMap = {};

    // Calculate exchange rates against USD
    for (let i in usdPrices) {
        let o = usdPrices[i];
        let sym = o['asset_id_quote'];
        let usdRate = o['rate'];

        rates[USD-${sym}] = usdRate;
        rates[${sym}-USD] = 1 / usdRate;
        usdMap[sym] = usdRate;
    }

    // Calculate direct crypto-to-crypto exchange rates
    let symbols = Object.keys(usdMap);
    for (let from of symbols) {
        for (let to of symbols) {
            if (from !== to) {
                let tag = ${from}-${to};
                rates[tag] = usdMap[to] / usdMap[from];
            }
        }
    }
```

```
    return rates;
}

constructor(coin2USD) {
    super();
    this.rates = this.constructor.calculateRates(coin2USD.rates);

    this.on(SHOW, (o) => {
        console.log("SHOW event received.");
        console.log(o);
        const { from, to } = o;
        try {
            let rate = this.convert(1, from, to);
            console.log(1 ${from} is worth ${rate} ${to});
        } catch (e) {
            console.error(e.message);
        }
    });

    this.on(UPDATE, (o) => {
        const { sym, usdPrice } = o;
        if (!sym || !usdPrice || usdPrice <= 0) {
            console.error("Invalid update parameters.");
            return;
        }
        console.log(Updating ${sym} price to ${usdPrice} USD.);
```

```
        // Update USD exchange rates
        this.rates[USD-${sym}] = usdPrice;
        this.rates[${sym}-USD] = 1 / usdPrice;

        // Recalculate crypto-to-crypto exchange rates
        const symbols = Object.keys(this.rates)
            .filter(key => key.startsWith('USD-'))
            .map(key => key.split('-')[1]);

        for (let from of symbols) {
            for (let to of symbols) {
                if (from !== to) {
                    this.rates[${from}-${to}] = this.rates[USD-${to}] / this.rates[USD-${from}];
                }
            }
        }

        console.log("Rates updated successfully.");
    });
}

convert(amount, fromUnits, toUnits) {
    let tag = ${fromUnits}-${toUnits};
    let rate = this.rates[tag];
    if (rate === undefined) {
        throw new Error(Rate for ${tag} not found);
    }
```

```
        return rate * amount;
    }

}

// JSON file path containing currency rates
const PATH = './rates.json';
let cnv = new CurrencyConverter(readJsonFromFile(PATH));

console.log(cnv.rates);

console.log("============================================================
============");

function test(amt, from, to) {
    console.log(${amt} ${from} is worth ${cnv.convert(amt, from, to)} ${to}.);
}

test(4000, 'ETH', 'BTC');
test(200, 'BTC', 'EOS');

console.log("============================================================
============");

// Test event handling
cnv.emit(SHOW, { from: "EOS", to: "BTC" });
console.log("============================================================
============");
```

```
cnv.emit(SHOW, { from: "EOS", to: "ETH" });

console.log("============================================================");


cnv.emit(SHOW, { from: "ETC", to: "ETH" });

console.log("============================================================");


cnv.emit(SHOW, { from: "LTC", to: "BTC" });

console.log("============================================================");


cnv.emit(UPDATE, { sym: "BTC", usdPrice: 50000 });

console.log("============================================================");


cnv.emit(SHOW, { from: "LTC", to: "BTC" });
```

- **Explanation: -**

1. Required Modules

- fs: Used to read data from a JSON file.
- events: Used to create custom events that the program can respond to.

2. Defining Constants

- SHOW = "SHOW_PRICE": Identifier for the event used to display currency prices.
- UPDATE = "UPDATE_USD_PRICE": Identifier for the event used to update the price of a specific currency against USD.

3. readJsonFromFile Function

This function attempts to read a JSON file and parse it into a JavaScript object.

If an error occurs during reading, it prints an error message and returns an empty object {}.

4. CurrencyConverter Class

This is the main class in the program, and it extends EventEmitter, allowing it to handle the SHOW and UPDATE events.

calculateRates Method

This method takes cryptocurrency prices against USD and calculates:

1. Conversion rates between USD and different cryptocurrencies (USD-Coin and Coin-USD).

2. Direct conversion rates between cryptocurrencies without needing to go through USD.

Constructor constructor

When an instance of CurrencyConverter is created:

1. It reads price data from the JSON file.

2. It calculates conversion rates using calculateRates.

3. It sets up event listeners for:

- SHOW: When triggered, it displays the conversion rate between two currencies.

- UPDATE: When triggered, it updates a currency's price against USD and recalculates conversion rates.

convert Method

This method converts a given amount from one currency to another using the stored exchange rates.

If no exchange rate is available, it throws an error.

5. Loading Exchange Rate Data

- The file path is set as PATH = './rates.json'.
- A CurrencyConverter instance is created using the data read from the file.

6. Testing Currency Conversion

The program runs some test conversions, such as:

test(4000, 'ETH', 'BTC');

test(200, 'BTC', 'EOS');

[4:25 pm, 01/03/2025] Ibrahim Taha: These convert 4000 ETH to BTC and 200 BTC to EOS.

7. Testing Events

The program triggers SHOW events to display exchange rates:cnv.emit(SHOW, { from: "EOS", to: "BTC" });

These convert 4000 ETH to BTC and 200 BTC to EOS.

7. Testing Events

The program triggers SHOW events to display exchange rates:cnv.emit(SHOW, { from: "EOS", to: "BTC" });

It also updates Bitcoin’s price in USD using:

```
cnv.emit(UPDATE, { sym: "BTC", usdPrice: 50000 });
```

How Does the Program Work?

1. It reads cryptocurrency exchange rates from a JSON file.
2. It calculates different exchange rates.
3. It allows displaying exchange rates using the SHOW event.
4. It allows updating exchange rates dynamically using the UPDATE event.
5. It enables converting an amount from one currency to another using the convert method.
