Simple Blockchain Project

Autors: Jacek Kwieciński, Przemysław Musiał

Project requirements - what has been achieved

- The blockchain has been implemented using javascript.
- The blockchain supports the addition of new blocks containing transaction data, ensuring secure storage.
- Cryptographic mechanisms such as hashing and digital signatures are employed to ensure the integrity and security of the data.
- Consensus mechanism 'proof-of-work' has been implemented to ensure the validity of the blocks.

Remaining projects requirements

- The blockchain should include a user interface allowing users to add and view transactions, as well as to observe the current state of the blockchain.
- Basic security features such as authentication and access control should be incorporated to prevent unauthorized access to the blockchain.
- The blockchain should be thoroughly tested and evaluated for both its performance and security.

Implementation

Key generation

```
const EC = require('elliptic').ec;
const ec = new EC('secp256k1');

const key = ec.genKeyPair();
const publicKey = key.getPublic('hex');
const privateKey = key.getPrivate('hex');

console.log();
console.log('Private key: ', privateKey);

console.log();
console.log('Public key: ', publicKey);
```

Implementation

Blockchain attributes

```
class Block{
  constructor(timestamp, transactions, previousHash=''){
    this.timestamp=timestamp;
    this.transactions= transactions;
    this.previousHash=previousHash;
    this.hash=this.calculateHash();
    this.nonce = 0;
  }
  ...
  class Blockchain{
  constructor(){
    this.chain=[this.createGenesisBlock()];
    this.difficulty = 4;
    this.pendingTransactions =[];
    this.miningReward = 100;
  }
```

Implementation

Main method

```
const {Blockchain, Transaction} = require('./blockchain');
const EC = require('elliptic').ec;
const ec = new EC('secp256k1');
let JPCoin = new Blockchain();

const jacKey = ec.keyFromPrivate('087a54abd0190bc80c44f0349c07fbf23b589d19f9119c954b48d2ab3c2ca92d');
const jacekWalletAddress = jacKey.getPublic('hex');
const przKey = ec.keyFromPrivate('097a54abd0190bc80c44f0sjgc07fbf23b589d19f9119c954b48d2ab3c2ca92d');
const przemekWalletAddress = przKey.getPublic('hex');

const tr1 = new Transaction(jacekWalletAddress, przemekWalletAddress, 10);
tr1.signTransaction(jacKey);
JPCoin.addTransaction(tr2);
JPCoin.minePendingTransactions(jacekWalletAddress);

console.log('\nBalance of Jack\'s wallet is', JPCoin.getBalanceOfAddress(jacekWalletAddress));
console.log('Is chain valid?', JPCoin.isChainValid());
```

Tests

Testing the simple implementation of a blockchain.


```
        {
            "index": 1,
            "timestamp": "10/06/2023",
            "data": {
                "amount": 4
            },
            "previousHash": "76f5030ca77763c12fb12f0918dfe88c6ad739862c457738a56b6001c640de8e",
            "hash": "76440b8be8f2df0f85e80de620c11670c1f7d5c846ec2e8f8ed083d464d4be03"
        },
        {
            "index": 2,
            "timestamp": "12/06/2023",
            "data": {
                "amount": 10
            },
            "previousHash": "76440b8be8f2df0f85e80de620c11670c1f7d5c846ec2e8f8ed083d464d4be03",
            "hash": "4302a7aaaf52fdc34bfd637acf4753c1ce5df8e138061887917821833df90b4a"
        }
}
```

Tests

Testing the transactions.

```
      Starting the miner...
Block mined: 00151f3c89d6623a6df1bfc8a2945c3351d369064a2052eb48da8e8cc1fd3329
Block successfully mined!

Balance of Jack's wallet is 90
```

Tests

Testing validity of the chain after modifying the transaction.

```
jackCoin.chain[1].transactions[0].amount = 1;
```

```
       Starting the miner...
Block mined: 00a130bb7d3ab8e0f54a54a39216507648064669afb73b07abcee4c075ee48cc
Block successfully mined!

Balance of Jack's wallet is 90
Is chain valid? false
```

Tests

```
const tr1 = new Transaction(jacekWalletAddress, przemekWalletAddress, 10);
const tr2 = new Transaction(przemekWalletAddress, jacekWalletAddress, 15);
tr1.signTransaction(jacKey);
JPCoin.addTransaction(tr1);
tr2.signTransaction(przKey);
JPCoin.addTransaction(tr2);

console.log('\n Starting the miner...');
JPCoin.minePendingTransactions(przemekWalletAddress);

console.log('\nBalance of Jack\'s wallet is', JPCoin.getBalanceOfAddress(jacekWalletAddress));
console.log('\nBalance of Przemek\'s wallet is', JPCoin.getBalanceOfAddress(przemekWalletAddress));
```

Tests

```
...
tr1.signTransaction(jacKey);
JPCoin.addTransaction(tr1);

tr2.signTransaction(jacKey);
JPCoin.addTransaction(tr2);
...
```

```
Error: You cannot sign transactions fo other wallets!
```

Tests

```
...
const tr3 = new Transaction(przemekWalletAddress, jacekWalletAddress, 25);
tr3.signTransaction(przKey);
JPCoin.addTransaction(tr3);
...
```

```
Block successfully mined!
Balance of Jack's wallet is 35

Balance of Przemek's wallet is 65
Is chain valid? true
```

Tests

```
...
for(let i = 1; i< 100; i++){
  const tr3 = new Transaction(przemekWalletAddress, jacekWalletAddress, 25+i);
  tr3.signTransaction(przKey);
  JPCoin.addTransaction(tr3);
  JPCoin.minePendingTransactions(przemekWalletAddress);
}
...
```

```
Block mined: 008b4a4a8d7d6607bf5aafd0eba53ec9a33d562d0e13a27c7db9966465b6bc41
Block successfully mined!
Block mined: 0000cac0456a928f283fa3d1cb919287cda2c9081eecbbff6c3039b345cf0297
Block successfully mined!

Balance of Jack's wallet is 7535

Balance of Przemek's wallet is 2465
Is chain valid? true
```

Tests

```
...
class Blockchain{
  constructor(){
    this.chain=[this.createGenesisBlock()];
    this.difficulty = 4;
    this.pendingTransactions =[];
    this.miningReward = 100;
  }
...
```

```
Balance of Jack's wallet is 380

Balance of Przemek's wallet is 620
Is chain valid? true
```
