Blockchain Voting

Each node can connect to any node on the network and then each node can broadcast to everyone any new transaction, block, or vote event. Each node then checks if that information is new or existing in its own pending transactions or blocks or blockchain and then if it's not it adds the new information to its own "database" (I didn't implement a database). The blockchain makes sure that each vote transaction to be valid needs to have been verified by at least 3 people at the city of the event. Each voter can only vote once for a specific event. The blocks are in a list in the blockchain and the blockchain is valid if all the block's calculated hash is the same as the hash it has, but also if the previous Hash is indeed the hash of the previous block.

Class Block

Class Variables:

- index: The position of the block in the blockchain.
- timestamp: The time when the block was created.
- previousHash: The hash of the previous block in the chain.
- nonce: A value used in the Proof of Work (mining) process.
- difficulty: The mining difficulty.
- hash: The hash of the block.
- VoteEvents: A HashMap that stores VoteEvent objects.
- votetransactions: An ArrayList that stores VoteTransaction objects.
- merkleRoot_vt: The Merkle root of the vote transactions.
- merkleRoot_vvt: The Merkle root of the voter verification transactions.
- voterverificationtransactions: An ArrayList that stores VoterVerificationTransaction objects.

Constructor:

Block(int index, String previousHash, int difficulty): Initializes a new Block with the given index, previousHash, and mining difficulty.

Methods:

- getHash(): Returns the hash of the block.
- calculateHash(): Calculates and returns the hash of the block based on its properties.
- setNonce(int nonce): Sets the nonce value for the block.
- addVoterVerificationTransaction(VoterVerificationTransaction vvt): Adds a VoterVerificationTransaction to the block if it is valid and its signature is verified.
- addVoteEvent(VoteEvent newevent, Blockchain blockchain): Adds a VoteEvent to the block if it does not already exist.
- getVoteEvents(): Returns the VoteEvents HashMap.
- getEventFromId(String eventid): Returns a VoteEvent object with the given event ID.
- addVoteTransaction(VoteTransaction vt, Blockchain blockchain): Adds a VoteTransaction to the block if it is valid and its signature is verified.
- validateTransactions(Blockchain blockchain): Validates all the vote transactions in the block.
- getVerificationsFor(PublicKey voter): Returns the number of verifications for a given voter.

- getCityVerificationsFor(PublicKey voter, String city): Returns the number of verifications for a given voter in a specific city.
- checkIfVoted(PublicKey voter, String eventId): Checks if the given voter has already voted for the specified event.
- voteResults(String eventId, VoteEvent event): Returns the voting results for a given event.
- print(): Prints the block's details.
- toString(): Returns a string representation of the block.
- getVoteTransactions(): Returns the ArrayList of vote transactions.
- getVoterVerificationTransactions(): Returns the ArrayList of voter verification transactions.
- getPreviousHash(): Returns the previous hash of the block.
- getIndex(): Returns the index of the block.
- getDifficulty(): Returns the mining difficulty.
- isValid(Blockchain blockchain): Checks if the block is valid based on the hash, transactions, and voter verifications.
- mineBlock(): Mines the block by finding a valid nonce that satisfies the mining difficulty.

This Block class represents a building block in a blockchain-based voting system. Each block contains vote transactions, voter verification transactions, and vote events, along with other necessary information for maintaining the integrity of the blockchain.

Blockchain class

Represents the core blockchain structure and contains an ArrayList of Block objects.

Class Variables:

- blocks: An ArrayList that stores the blocks in the blockchain.
- genesisBlock: The first block in the blockchain (the genesis block).

Constructor:

The Blockchain class has two constructors, one that accepts a Block object as the genesis block and another that creates a default genesis block.

Methods:

- hasVoted(): Determines if a voter, represented by a public key, has already voted in a given voting event.
- isValid(): Checks if the blockchain is valid, including checking the genesis block, linking blocks, and validating hashes.
- getSize(): Returns the size of the blockchain (i.e., the number of blocks).
- isInSameCity(): Determines if a voter is voting in their city by verifying that they have been verified by at least 3 other voters in the same city.
- isKnownByAtLeastThree(): Determines if a voter is known (verified) by at least 3 other voters.
- addBlock(): Adds a new block to the blockchain.
- getBlock(): Retrieves a block by its index in the blockchain.
- getLastBlock(): Retrieves the last block in the blockchain.
- getEventResults(): Calculates the results of a specific voting event.
- contains(): Checks if a given block is already in the blockchain.
- getChain(): Returns the ArrayList of Block objects representing the entire blockchain.

This implementation is designed to support a voting system with event creation, voter verification, and voting transactions.

VoteEvent class

Represents a voting event within the blockchain voting system.

Class Variables:

- voteEventId: A unique identifier for the voting event.
- initializerKey: The public key of the event initializer.
- choices: An ArrayList of strings representing the choices for the voting event.
- startTime: The start time of the voting event.
- endTime: The end time of the voting event.
- signed: A boolean flag indicating if the event has been signed.
- signature: A byte array representing the digital signature of the event.
- voteCity: The city where the voting event takes place.

Constructor:

The VoteEvent constructor initializes a new instance with the initializer's public key, the city, the event ID, and the event duration.

Methods:

- addOption(): Adds a voting option to the event if it hasn't been signed yet.
- getEventId(): Returns the unique event ID.
- getVoteCity(): Returns the city where the event takes place.
- getChoices(): Returns the list of choices for the event.
- signTransaction(): Signs the event data using the initializer's private key, using the SHA256withRSA algorithm.
- verifySignature(): Verifies the digital signature of the event using the initializer's public key.
- isEventActive(): Determines if the event is currently active based on the current time and the start and end times of the event.
- createDataString(): Creates a string representation of the event data for signing and verifying the signature.
- toString(): Returns a string representation of the VoteEvent object.

The VoteEvent class is designed to store and manage voting event information and to ensure the integrity of the event data by signing and verifying digital signatures.

VoterVerificationTransaction class

Represents a voter verification transaction within the blockchain voting system.

Class Variables:

- transactionId: A unique identifier for the transaction.
- verifier: The public key of the verifier (e.g., an election authority).

- verifey: The public key of the voter being verified.
- timestamp: The timestamp at which the transaction is created.
- city: The city in which the voter is registered.
- signature: A byte array representing the digital signature of the transaction.

Constructor:

The VoterVerificationTransaction constructor initializes a new instance with the verifier's public key, the voter's public key, the city, and the timestamp.

Methods:

- calculateTransactionHash(): Calculates the SHA-256 hash of the transaction data.
- signTransaction(): Signs the transaction data using the verifier's private key, using the SHA256withRSA algorithm.
- verifySignature(): Verifies the digital signature of the transaction using the verifier's public key.
- createDataString(): Creates a string representation of the transaction data for signing, verifying the signature, and hashing.
- isTransactionValid(): Determines if the transaction is valid by verifying the signature.
- isVerified(): Checks if the transaction is valid and if the provided voter public key matches the one stored in the transaction.
- getCity(): Returns the city where the voter is registered if the transaction is verified, otherwise, returns "na".
- toString(): Returns a string representation of the VoterVerificationTransaction object.

The VoterVerificationTransaction class is designed to store and manage voter verification transactions, ensuring the integrity of the transaction data by signing and verifying digital signatures and voters' identity.

VoteTransaction class

Represents a vote transaction within the blockchain voting system.

Class Variables:

- transactionId: A unique identifier for the transaction.
- voter: The public key of the voter.
- votingEventId: The unique identifier for the voting event.
- votingChoice: The voter's choice in the voting event.
- signature: A byte array representing the digital signature of the transaction.
- timestamp: The timestamp at which the transaction is created.

Constructor:

The VoteTransaction constructor initializes a new instance with the voter's public key, the voting event ID, the voting choice, and the timestamp.

Methods:

- calculateTransactionHash(): Calculates the SHA-256 hash of the transaction data.
- getEventId(): Returns the unique identifier for the voting event.
- getChoice(): Returns the voter's choice in the voting event.
- getBytes(): Serializes the transaction data into a byte array.
- signTransaction(): Signs the transaction data using the voter's private key, using the SHA256withRSA algorithm.
- verifySignature(): Verifies the digital signature of the transaction using the voter's public key.
- createDataString(): Creates a string representation of the transaction data for signing, verifying the signature, and hashing.
- getVoter(): Returns the public key of the voter.
- isTransactionValid(): Determines if the transaction is valid by verifying the signature, checking if the voter has already voted, if the voter is known by at least three people, and if the voter is voting in their city.
- toString(): Returns a string representation of the VoteTransaction object.

The VoteTransaction class is designed to store and manage vote transactions, ensuring the integrity of the transaction data by signing and verifying digital signatures, and verifying the identity of voters and the validity of their votes.

Wallet class

Represents a wallet within the blockchain voting system.

Class Variables:

- privatekey: The private key of the wallet owner.
- publickey: The public key of the wallet owner.
- keypairholder: An instance of the keyPairHolder class to hold the private and public keys.

Constructors:

The Wallet constructor with parameters PrivateKey and PublicKey initializes a new instance with the provided private and public keys.

The Wallet constructor without parameters generates a new key pair for the wallet owner using the RSA algorithm with a key size of 2048 bits.

Methods:

- getKeypair(): Returns the keyPairHolder instance containing the wallet's public and private keys.
- verifyAWallet(): Creates a VoterVerificationTransaction to verify another wallet with the given public key and city.
- vote(): Creates a VoteTransaction instance for the wallet owner to vote in a specific event with a given voting choice.
- voteEvent(): Creates a VoteEvent instance to represent a voting event in a specific city with a specified duration.

The Wallet class is designed to manage the cryptographic keys for a user in the blockchain-based voting system, verify other users, and create vote transactions and vote events.

Node class

Represents a node within the blockchain voting system.

Class Variables:

- name: The name of the node.
- nodesink: A list of NodeInterface objects representing the nodes this node receives information from.
- nodesource: A list of NodeInterface objects representing the nodes this node sends information to.
- id: The public key of the node.
- visited: A boolean flag indicating if the node has been visited.
- isMiner: A boolean flag indicating if the node is a miner.
- visitedNodes: A set of NodeInterface objects representing the visited nodes.
- semaphore: A Semaphore object used for synchronization.
- blockChain: A Blockchain object representing the blockchain used by the node.
- pending_vt_pool: A list of VoteTransaction objects representing unprocessed vote transactions.
- pending_vvt_pool: A list of VoterVerificationTransaction objects representing unprocessed voter verification transactions.
- pending_ve_pool: A list of VoteEvent objects representing unprocessed vote events.

Constructor:

The Node constructor initializes a new instance with the provided name, public key, semaphore, and miner status.

Methods:

- getBlockChainFromNetwork(): Acquires the semaphore and sends the node's blockchain to all connected nodes.
- sendChain(): Sends the node's blockchain to connected nodes using propagateChain().
- receiveChain(): Receives a blockchain from another node and updates the node's blockchain if the received chain is valid and longer.
- propagateChain(): Propagates the received blockchain to unvisited nodes and updates the visited nodes list.

- sendBlockChain(): Returns the node's blockchain.
- addNodesink(): Adds a NodeInterface object to the nodesink list.
- addNodesource(): Adds a NodeInterface object to the nodesource list.
- send(), receive(), propagate(): Sends, receives, and propagates a message to all nodes in the network.
- sendVt(), receiveVt(), propagateVt(): Sends, receives, and propagates a vote transaction to all nodes in the network.
- sendVvt(), receiveVvt(), propagateVvt(): Sends, receives, and propagates a voter verification transaction to all nodes in the network.
- sendBlock(), receiveBlock(), propagateBlock(): Sends, receives, and propagates a block to all nodes in the network.
- createBlock(): Creates a new block with pending transactions and vote events, mines the block, adds it to the blockchain, sends the block to other nodes, and clears the pending transaction pools.
- getName(): Returns the name of the node.
- setBlockchain(): Sets the node's blockchain if the provided chain is valid.
- getBlockChain(): Returns the node's blockchain.
- sendChain(): Sends the node's blockchain to connected nodes using propagateChain().
- sendVE(), receiveVE(), propagateVE(): Sends, receives, and propagates a vote event to all nodes in the network.

The main purpose of this class is to manage communication between nodes in the P2P network for the blockchain-based voting system. It sends and receives various types of messages, such as vote transactions and voter verification transactions, and propagates them to other nodes in the network. When a miner node collects enough transactions, it creates a new block and adds it to the blockchain.