Author:

Name: Austin Simpson
Email: asimpson@chapman.edu
Date: 12/6/2023

Solutions Contract:

1. **Create account**
   - Service contract: Creates a user account in the blockchain
   - Method: `createAccount(String accountAddress)` which uses `addAccount(String address, Account account)` as a helper method.
   - These methods create a new account with the address passed to the function with an balance of 0 and adds the account to the next uncommitted block's accountBalanceMap.
   - This throws a LedgerException if the address passed in already exists.
2. **Process Transaction**
   - Service Contract: Processes a transaction between two accounts and adds the transaction to the blockchain
   - Method: `processTransaction(Transaction transaction)'
   - This method takes in a transaction object, and uses the information of the transaction to adjust the balances of the relevant accounts. It then adds the transaction to the next uncommitted block and persists the updated account balances to the database. Every 10th transaction will commit the block to the blockchain and create a new uncommitted block to replicate the accounts to and to put future transactions in.
   - This throws a LedgerException if any of the following happen:
     - Transaction amount is less than 0
     - Transaction fee is less than 10
     - Transaction note's length is longer than 1024 characters
     - Transaction id has been used before
     - Transaction's payer has insufficient funds
3. **Get Account Balance**
   - Service Contract: Obtain an account given the address for the account.
   - Method: `getAccountBalance(String accountAddress)`

- This method takes in the address of the account and returns the balance of the account.
- This method throws LedgerException if the account has not yet been committed to a block, or if the account does not exist

4. Get Transaction

- Service Contract: Obtain a transaction given its id.
- Method: `Transaction getTransaction(String transactionID)`
- This method takes a string of the transactionID and will return the transaction with the given id.
- This method does not throw an exception, but will instead return null if a transaction with the given id is not found.

5. Get Block

- Service Contract: Obtain a block given the block number.
- Method `Block getBlock(Integer blockNumber)`
- This method takes in a block number and returns the block if it exists.
- This method throws a LedgerException if the block with the given block number does not exist

API

#	End point	Api Method	Request Parameters	Response Parameters	Response Code
1	/transactions	POST	http://localhost:8080/transactions { "transactionId": "string", "amount": 0, "fee": 0, "note": "string", "payer": { "address": "string", "balance": 0 }, "receiver": { "address": "string", "balance": 0 }	transactionID	200 - Transaction processed successfully 400 - Invalid transaction data 500 - Internal Server Error

			}		

2	/transactions/{transactionID}	GET	http://localhost:8080/transactions/{transactionId}
			
			{transactionID} is replaced with the a string that is the transactionID once one is added
				Transaction in the following format:
				
				{
				 "transactionId": "string",
				  "amount": 0,
				  "fee": 0,
				  "note": "string",
				  "payer": {
				    "address": "string",
				    "balance": 0
				  },
				  "receiver": {
				    "address": "string",
				    "balance": 0
				  }
				}
					200 - OK
					
					404 - Transaction not found
					
					500 - Internal Server Error

3	/accounts	POST	http://localhost:8080/accounts
			
			{
			  "address": "string",
			  "balance": 0
			}
				Account which was just created in the following format
				
				{
				  "address": "string",
				  "balance": 0
				}
					200 - OK
					
					500 - Internal Server Error

4	/accounts/{address}	GET	http://localhost:8080/accounts/{address}
			
			Where {address} is replaced with the address of the desired account
				Account with the desired address in the following format:
				
				{
				  "address": "string",
				  "balance": 0
				}
					200 - OK
					
					204 - There is no account with such address
					
					500 - Internal Server Error

QA

#	Test Description	Assumptions and Pre-Conditions	Test data	Steps to be executed	Expected result
1	As an admin, I want to be able to get get the balance of someone's account	The account with the desired address already exists	{ "address": "address1", "balance": 0 }	Execute: testGetAccountBalances	Account balances are successfully acquired
2	As an admin, I want to be able to create a new account	An account with the desired address does not already exist	{ "address": "newAccount", "balance": 0 }	Execute: testCreateAccount	Account named newAccount is created with balance 0
3	As an admin, I want to be able to make transactions between user accounts	The two accounts already exist and the payer account has a sufficient balance	{ "transactionId": "blockTest1", "amount": 100, "fee": 10, "note": "testNote", "payer": { "address": "master", "balance": 2147483647 }, "receiver": { "address": "testAccount", "balance": 0 } }	Execute: testMultipleAccountTransactions()	4 transactions are created and processed
4	As an admin, I want to be able to view a block committed to the block chain	A full block has been committed to the block chain	blockID, in this case: 1	Execute: testGetBlock()	11 transactions are created and then the first block is acquired
5	As an admin, I want to be able to set the notes for transactions	The transaction has already been created	String containing the new note, in this case: "Test Note"	Execute: testSetNote()	A transaction is created, and then its note is set to "Test Note".