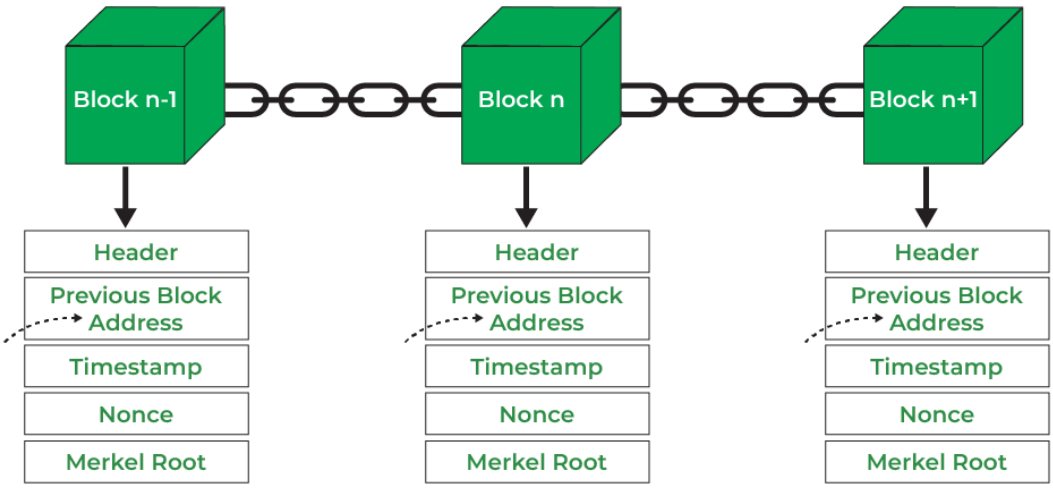
pyChain



Creating a UML (Unified Modeling Language) diagram for a simple cryptocurrency involves depicting the classes, their attributes, methods, and the relationships between these classes. Below is a basic UML class diagram for a simple cryptocurrency system.

Classes:

1. Cryptocurrency
2. Wallet
3. Transaction
4. Blockchain
5. Block
6. User

Attributes and Methods:

Cryptocurrency

- Attributes:
  - name: String
  - symbol: String
  - totalSupply: Double
- Methods:
  - createWallet(user: User): Wallet
  - getBlockchain(): Blockchain

Wallet

- Attributes:
  - address: String
  - balance: Double
  - owner: User
- Methods:
  - sendTransaction(recipient: Wallet, amount: Double): Transaction
  - receiveTransaction(transaction: Transaction)

Transaction

- Attributes:
  - transactionId: String
  - sender: Wallet
  - recipient: Wallet
  - amount: Double

- timestamp: Date

- Methods:

- validate(): Boolean

- execute()

Blockchain

- Attributes:

- blocks: List<Block>

- difficulty: Int

- Methods:

- addBlock(block: Block): Boolean

- validateChain(): Boolean

Block

- Attributes:

- blockId: String

- previousHash: String

- transactions: List<Transaction>

- timestamp: Date

- nonce: Int

- Methods:

- calculateHash(): String

- mineBlock(difficulty: Int)

User

- Attributes:

- userId: String

- name: String

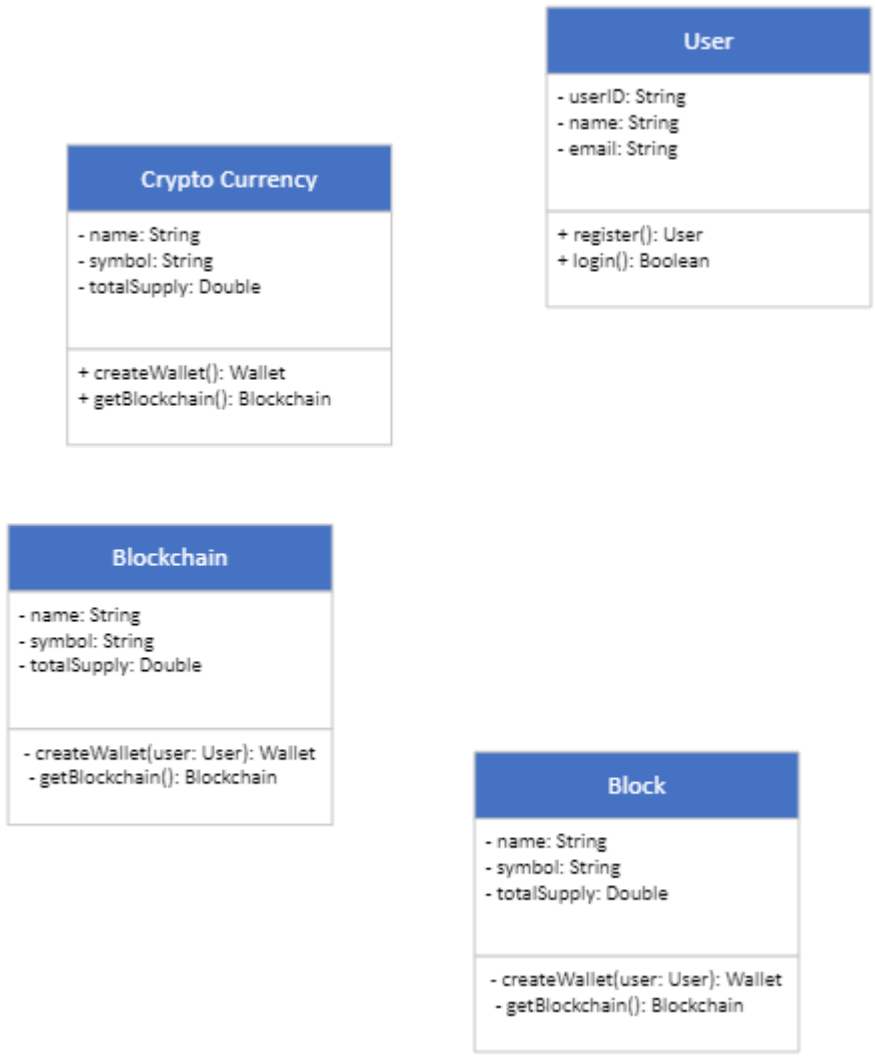
- email: String

- Methods:

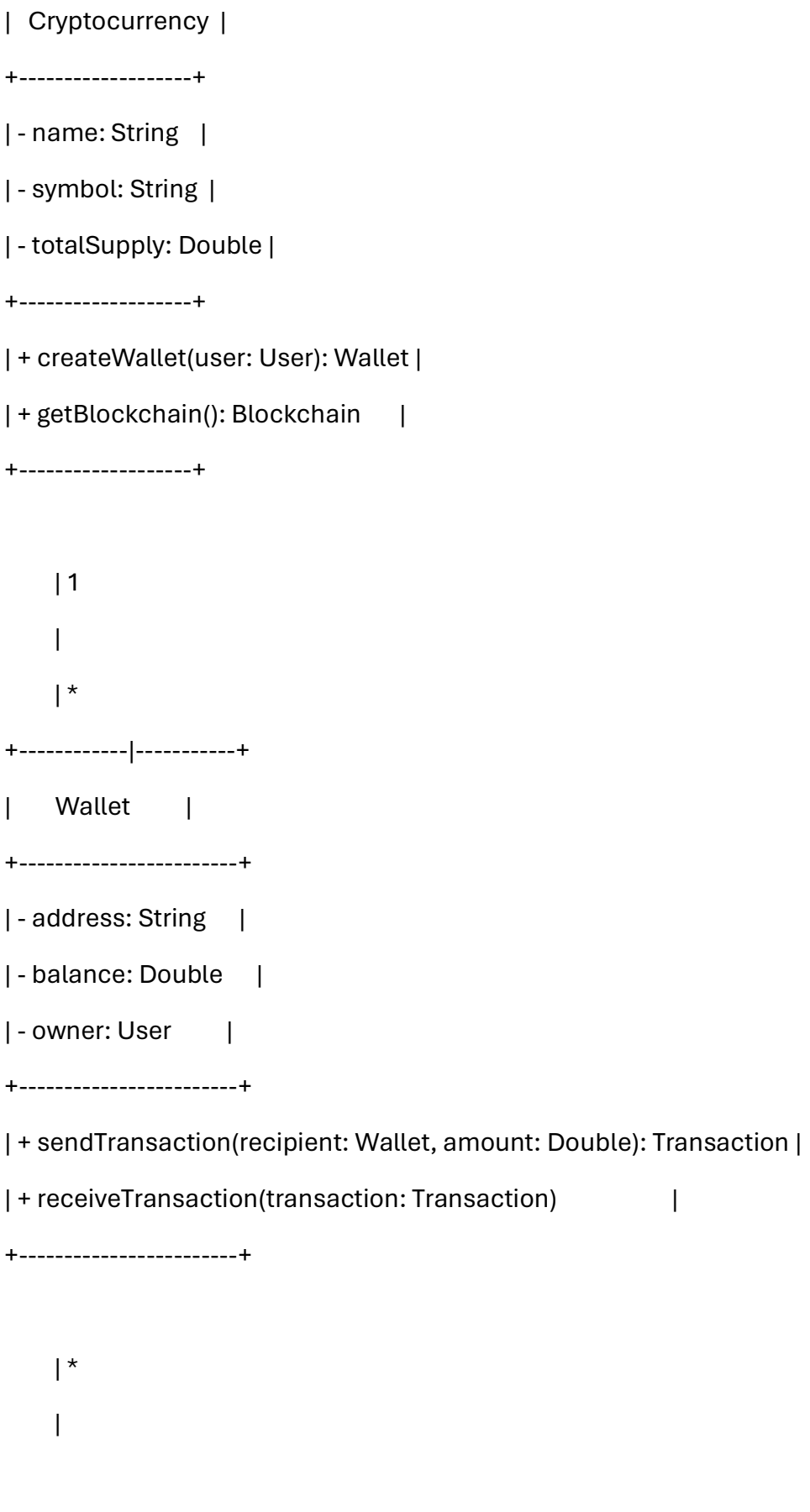
- register(name: String, email: String): User

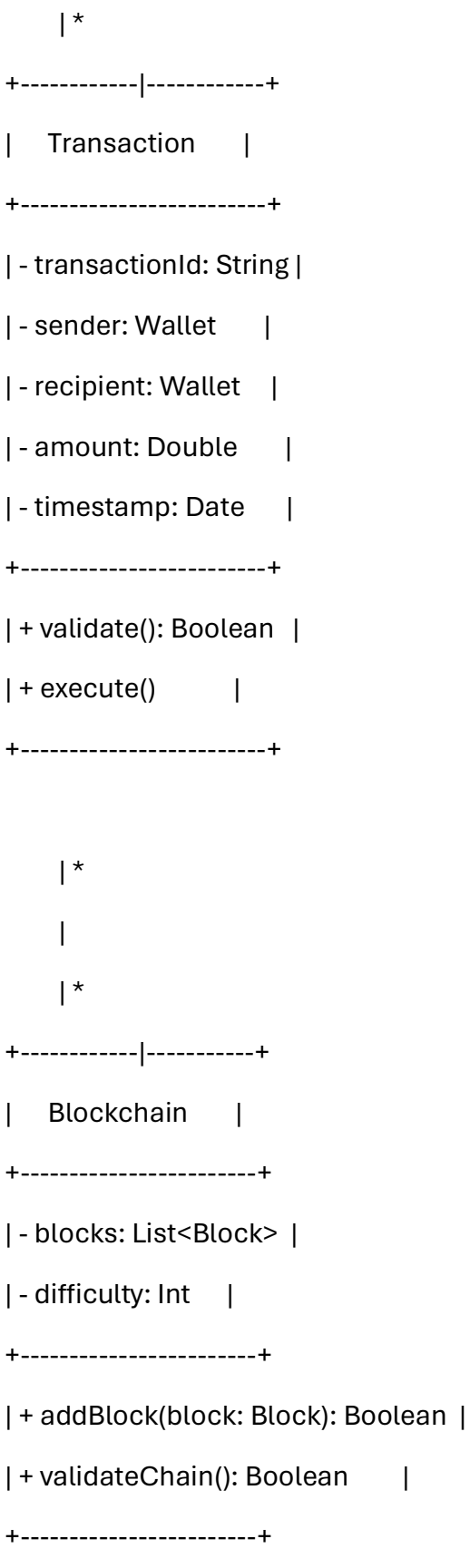
- login(email: String, password: String): Boolean

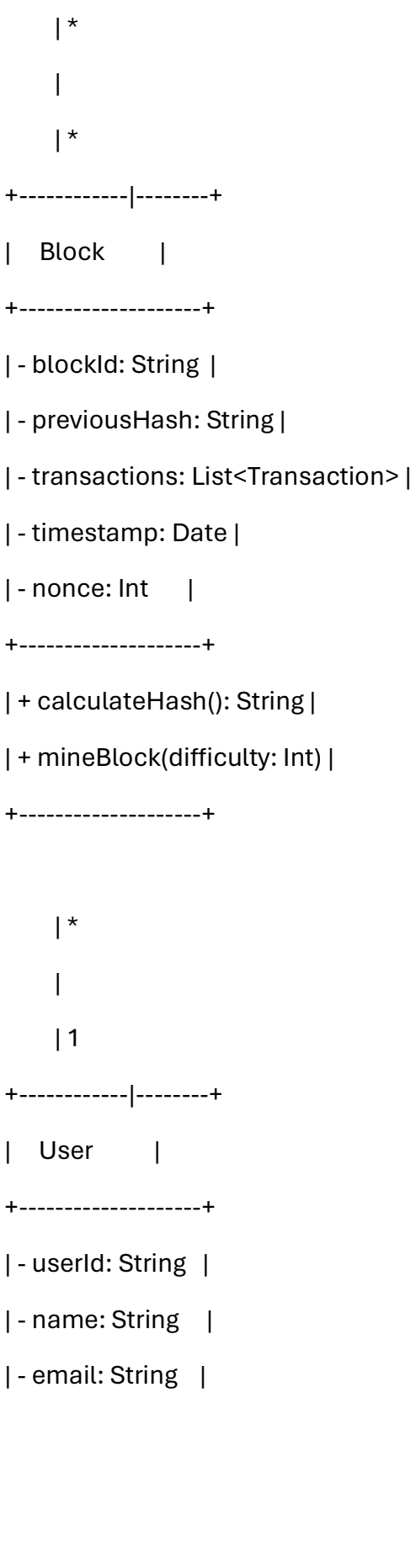
UML Class Diagram



+------------------+


| Cryptocurrency |
+-------------------+
| - name: String |
| - symbol: String |
| - totalSupply: Double |
+-------------------+
| + createWallet(user: User): Wallet |
| + getBlockchain(): Blockchain |
+-------------------+
| 1
|
| *
+------------|-----------+
| Wallet |
+-----------------------+
| - address: String |
| - balance: Double |
| - owner: User |
+-----------------------+
| + sendTransaction(recipient: Wallet, amount: Double): Transaction |
| + receiveTransaction(transaction: Transaction) |
+-----------------------+
| *
|


| *
+------------|------------+
| Transaction |
+-------------------------+
| - transactionId: String |
| - sender: Wallet |
| - recipient: Wallet |
| - amount: Double |
| - timestamp: Date |
+-------------------------+
| + validate(): Boolean |
| + execute() |
+-------------------------+
| *
|
| *
+------------|-----------+
| Blockchain |
+-----------------------+
| - blocks: List<Block> |
| - difficulty: Int |
+-----------------------+
| + addBlock(block: Block): Boolean |
| + validateChain(): Boolean |
+-----------------------+


| *
|
| *
+------------|--------+
| Block |
+--------------------+
| - blockId: String |
| - previousHash: String |
| - transactions: List<Transaction> |
| - timestamp: Date |
| - nonce: Int |
+--------------------+
| + calculateHash(): String |
| + mineBlock(difficulty: Int) |
+--------------------+
| *
|
| 1
+------------|--------+
| User |
+--------------------+
| - userId: String |
| - name: String |
| - email: String |

+--------------------+

| + register(name: String, email: String): User |

| + login(email: String, password: String): Boolean |

+--------------------+

```

Explanation:

- Cryptocurrency : Represents the main class controlling the overall cryptocurrency system, capable of creating wallets and accessing the blockchain.

- Wallet : Represents a user's wallet, storing their address, balance, and owner information. Capable of sending and receiving transactions.

- Transaction : Represents a transaction between wallets, storing transaction details and validating/executing transactions.

- Blockchain : Represents the blockchain, maintaining a list of blocks and ensuring the validity of the chain.

- Block : Represents a block in the blockchain, containing a list of transactions, timestamp, and methods for calculating hashes and mining.

- User : Represents a user in the system, capable of registering and logging in.

This UML diagram provides a high-level overview of a simple cryptocurrency system. Each class and its relationships are depicted, showing the interactions within the system.

## PyChain

Flask App

1. Pip install flask
2. Pip install flask-mysqldb
```

3. Pip install passlib
4. Mysql.server start

Virtual Environments

Windows:

venv\Scripts\activate