**华东师范大学数据科学与工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**： 区块链 | **年级**：19 | **上机实践成绩**： |
| **指导教师**：张召 | **姓名**：赵煜硕 |  |
| **上机实践名称**：第一次实验报告 | **学号**： 10195501415 | **上机实践日期**： |
| **上机实践编号**： | **组号**： | **上机实践时间**： |

1. **实验目的**

实现简单的区块链系统，并进一步理解且实践签名、验签以及UTXO，模拟spv简单支付验证

1. **实验任务**

**1.区块链系统的简单实现**

**Merkle树**

**2.签名验签以及UTXO的简单实现**

**账户类与钱包功能**

**UTXO类与解锁脚本功能**

**账户签名交易，旷工验签功能**

**3.SPV简单支付验证**

**Spv轻节点模块**

**Spv验证流程**

1. **使用环境**

**Windows10**

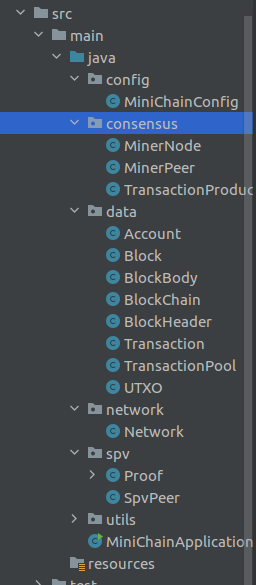
**VMware**

**Ubuntu1804**

**IDEA**

**Minichain**

1. **实验过程**
   * + 1. 文件理解说明



Config包中MiniChainConfig类存储简易区块链系统的几个关键配置

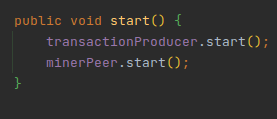


分别是挖矿难度，交易池大小，账户金额..

Consensus包中对矿工进行类抽象，minernode类继承自thread类，功能是从交易池中获取一批次的交易，构造新的区块体，区块头，组合一个新的区块，然后使用随机值nonce进行区块哈希值的计算，直到哈希值满足难度条件，即可将 区块加入区块链中，然后继续生成下一个区块。TransactionProducer类同样集成自thread类，功能是随机生成一批次交易放入交易池中，供旷工线程使用。

Data包中的类是对交易，区块体，区块头以及区块链的抽象(额外抽象了一个交易池类)交易池类主要用于存储多笔交易。旷工从交易池中获取一批次的交易，进行Merkle树根哈希值的计算和区块体的构造

Network包中的network类组合了上述的功能类，提供两个主要工作线程启动入口，实现整个系统运作



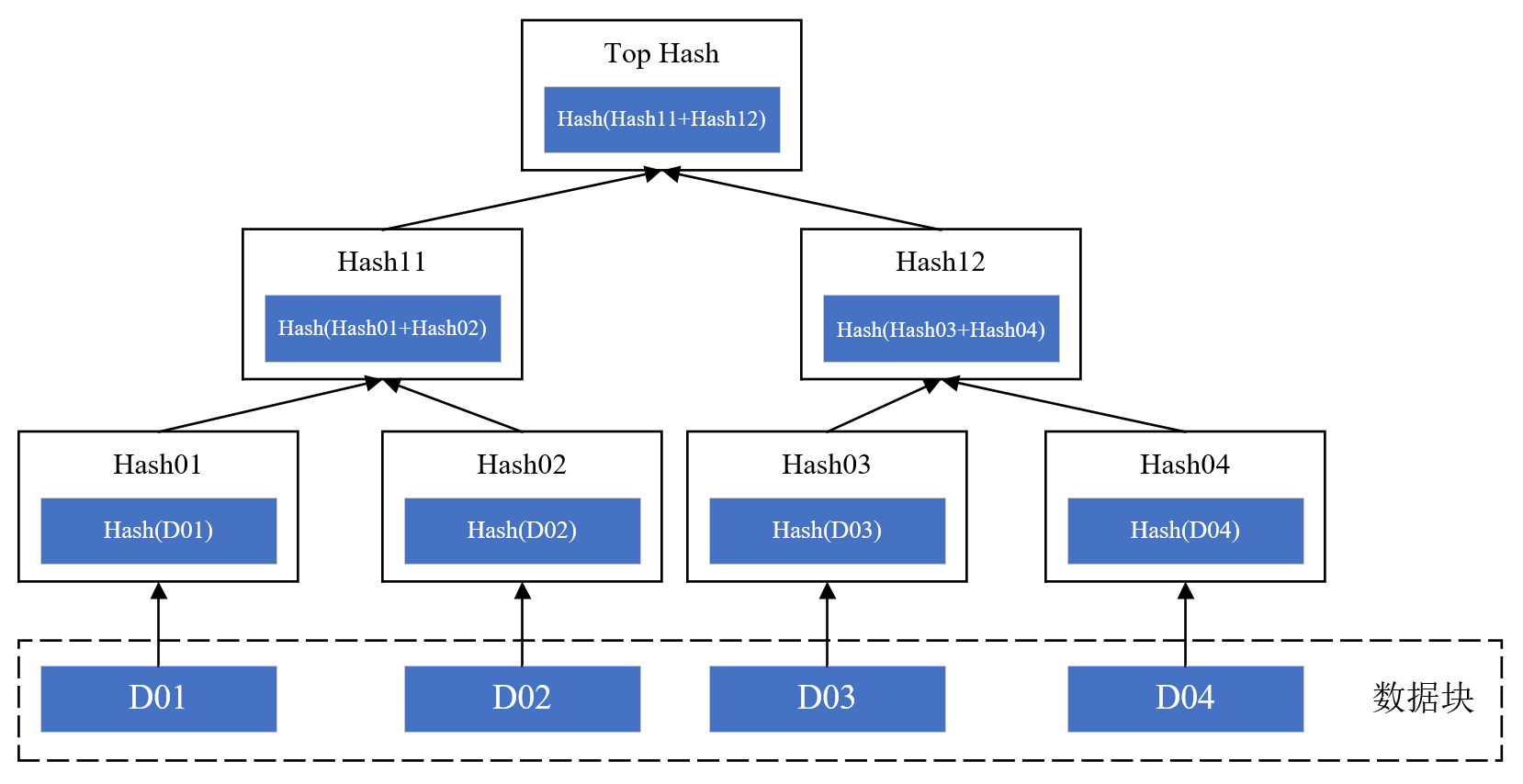
Utils包提供可直接调用的工具类，(SHA256算法哈希值计算，判断区块的哈希值是否满足难度条件)

Spv包于实验三中实现

* + - 1. Merkle树的理解

如果有偶数个节点，每个节点先做一次Hash，结果两两相加后再Hash(第1个和第2个，第3个和第4个相加)，一直这样相加再Hash直到做到TopHash；

如果有奇数个节点，把最后一个节点复制一边变成偶数个节点。



* + - 1. 挖矿流程理解

在MinerNode类中有mine，getBlock和getBlockBody三个函数

Mine在循环中完成挖矿，其实就是通过不断变换区块中的nonce字段，直到区块的哈希值满足难度条件，即可将该区块假如区块链中。

Getblock供mine调用，功能为根据传入的区块体参数，构造一个区块对象返回

Getblockbody根据传入参数中的交易构造并且返回一个相应的区块体对象，并根据这些交易计算merkle树的根哈希值

* + - 1. 非对称加密算法和数字签名算法理解



发起人私钥签名，发起人公钥验签

发起方对包用自己的私钥签名，接收方收到包后用发送方的公钥验签

私钥：识别发送方是否是合法的

非对称加密

接收人公钥加密，接收人私钥解密

发起方使用接收方的公钥进行加密，接收方用自己的私钥进行解密

公钥：保证加密后接收方能够解密

* + - 1. UTXO模型和账户模型理解

UTXO(Unspent Transaction Output)

因为丢失私钥丢失账户控制权后，UTXO会一直保存账户的余额,因而没有输出.

验证一笔交易的余额是否足够需要向上追溯，除了BTC交易外每一笔交易的输出都是另一笔交易的输入

大量生成地址，每个地址只使用一次，一旦该地址付出过比特币，那么公钥就暴露了，也就不抗量子攻击了，所以找零不会回到付款地址，而是一个新地址

优点：

只有直到私钥才有账户的控制权

可以一定程度上避免双花攻击

通过找零时设置新地址增加了一定稳定性

在简单业务和跨链上，UTXO 有其非常独到和开创性的优点

缺点：

丢失账户控制权后，UTXO会一直保存账户的余额；随着比特币的分支和碎片化以及账户私钥丢失，UTXO模型会越来越膨胀

交易时效率可能受到限制，交易成本较高

没有一个相对稳定的身份，一旦出现交易失败或合约违约，无法断定是谁的责任

Account模型

优点：

具有一个相对稳定的身份，可以支持智能合约

合约以二进制形式保存在Account中，并且Account拥有自身状态。模型具有更好的可编程性

Account 模型在可编程性，灵活性等方面更有优势

批量交易成本低

缺点：

模型交易之间没有依赖性，需要解决重放问题

对于实现闪电网络等，用户需要更复杂的证明机制

是一种状态机。交易是事件本身，不包含结果。

* + - 1. Spv的理解

分为 全节点，轻节点和spv节点

通过简化的支付验证（SPV）的方式，可以使它们在不必存储完整区块链的情况下进行工作。这种类型的客端被称为轻节点和SPV节点。

全节点指的是维持包含全部交易信息的完整区块链的结点。

轻极点指的是节点本地只保存与其自身相关的交易数据，但并不保存完整区块链信息的技术。

SPV的目标是验证某个支付是否真实存在，并得到了多少个确认。

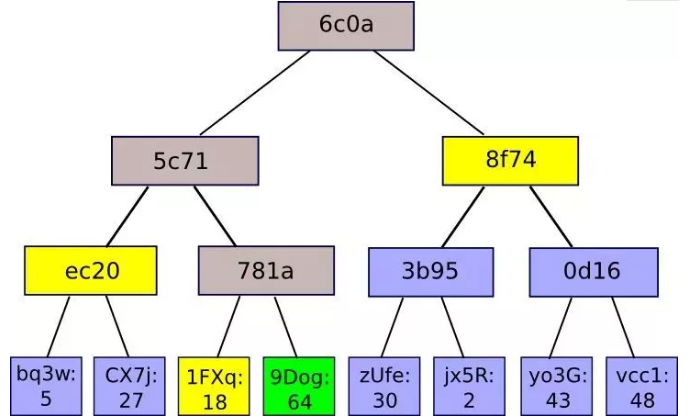
轻节点的目标不仅是支付验证，还有管理节点资深的资产收入支付等信息

轻节点和SPV的区别在于：

轻节点仍然需要下载每个新区块的全部数据并进行解析，获取并且在本地存储与资深相关的交易数据，只是不需要在本地保存

SPV节点不需要下载新区块的全部数据，只需要保存区块头部信息。

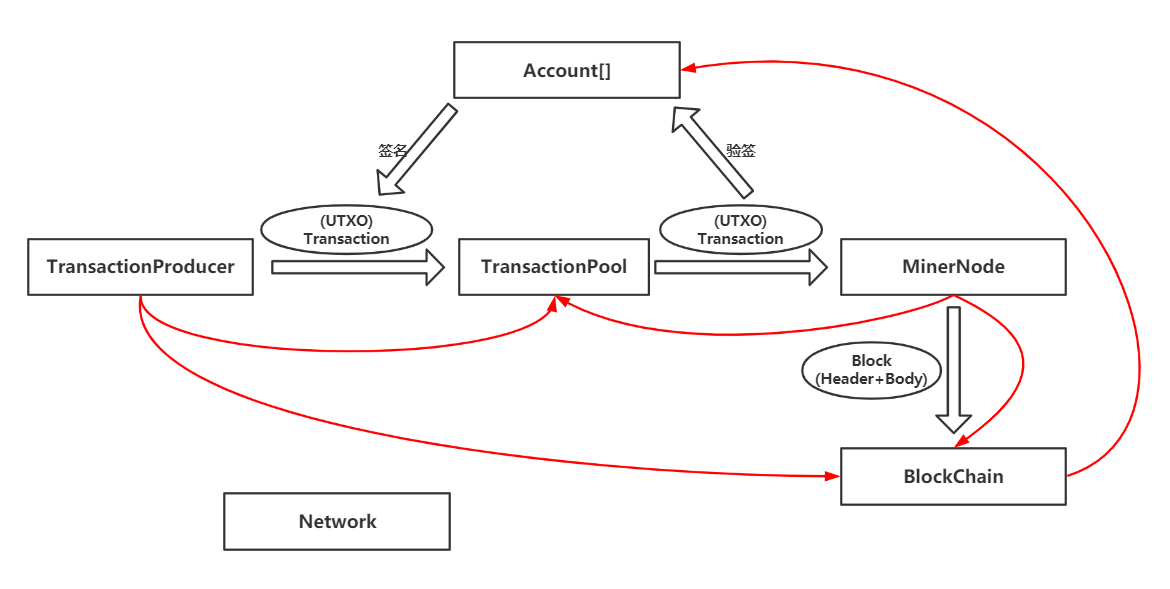
当SPV节点想验证某笔交易是否存在时，它将该交易的哈希发送给一个全节点，请求该交易的merkle树验证路径，当得到全节点发送过来的验证路径后（说明全节点反馈交易是存在的），SPV节点在本地使用它们重新计算包含该交易的区块的merkle根哈希，与本地区块头里存储的merkle根哈希作比较，判断是否相同，若相同，则该笔交易得到确认，否则说明对方在撒谎。

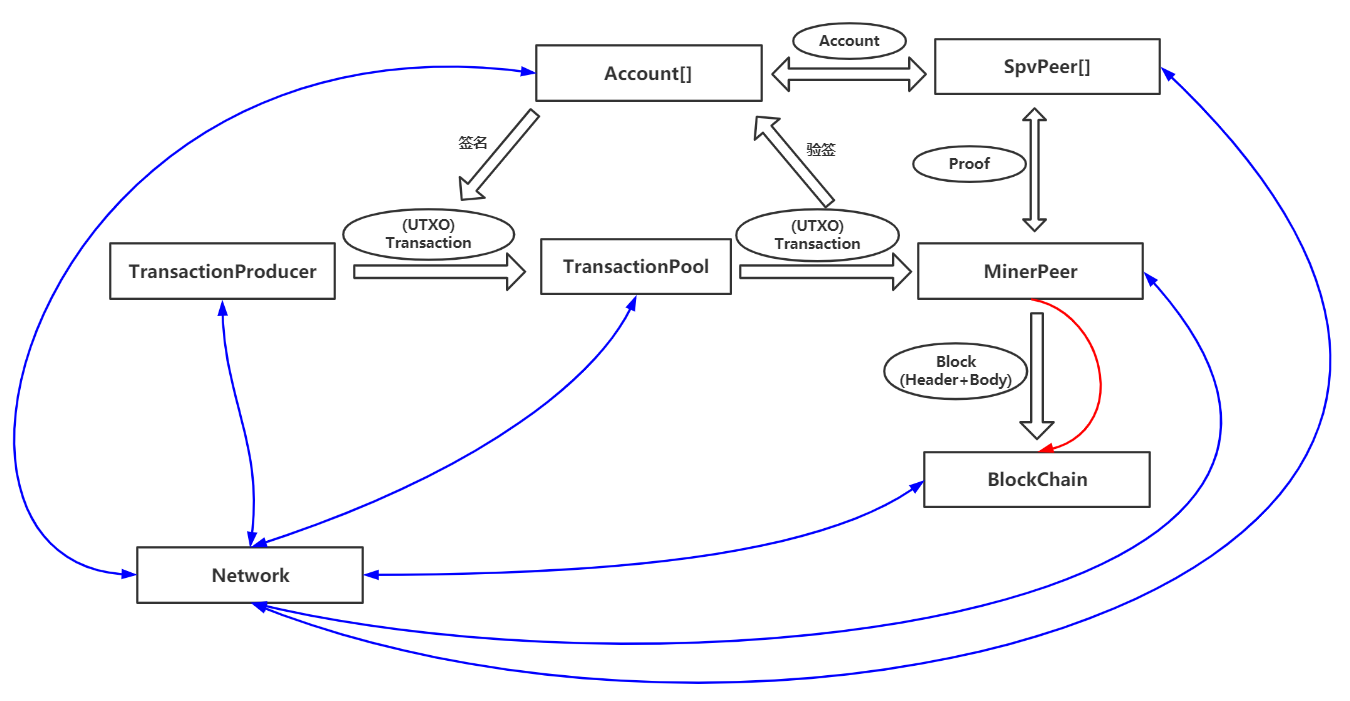


* + - 1. Spv的准确性

SPV因为没有保存全部区块的节点信息，需要和其他节点配合才能进行验证，所以SPV节点存在被诱导连入了一个虚假的网络中的情况，存在被恶意攻击的可能。

* + - 1. 新旧框架业务交互

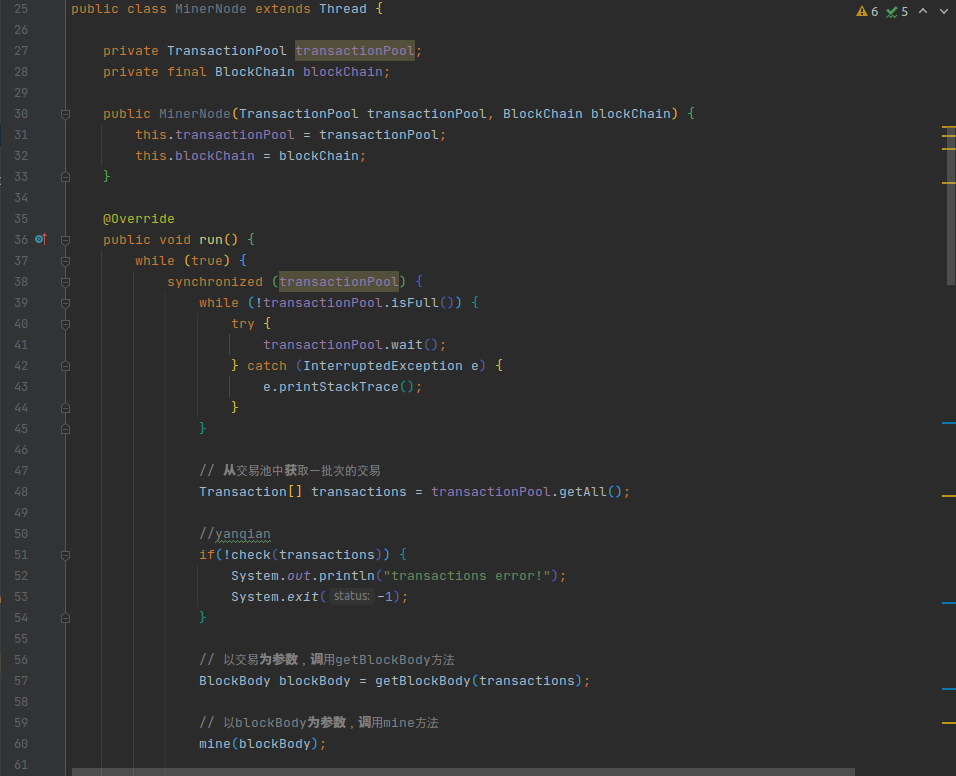


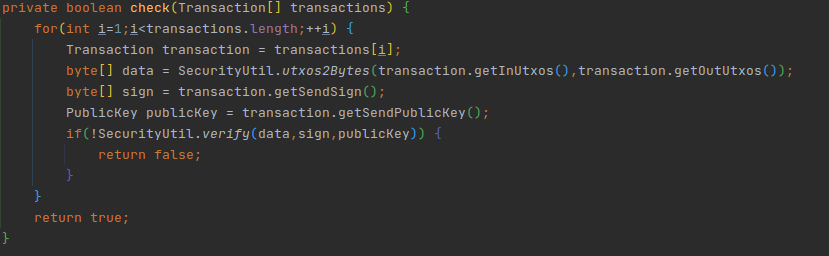


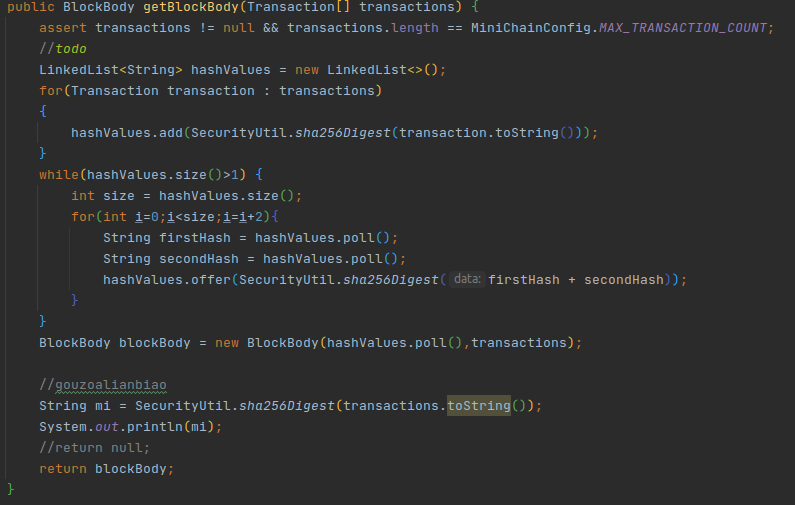
新框架仅仅保留旷工节点对区块链的引用，切断了其他的引用联系，各模块和网络模块有一个双向引用，这样模块与模块都通过网络来交互，从而降低系统的耦合度。

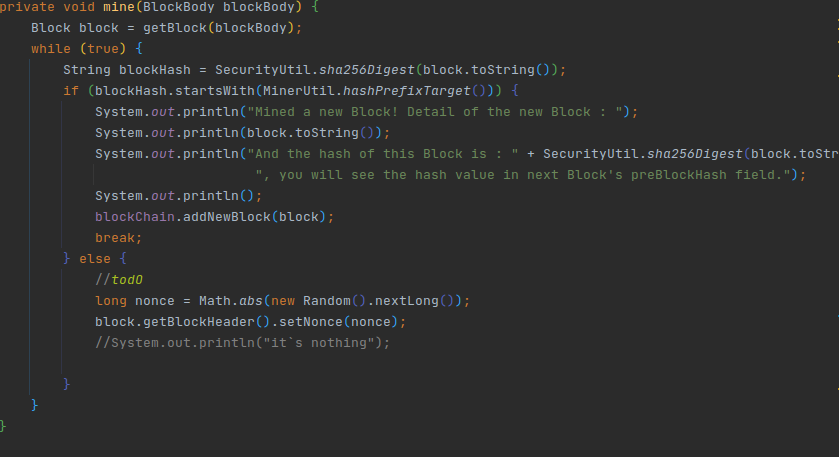
* + - 1. 实验过程

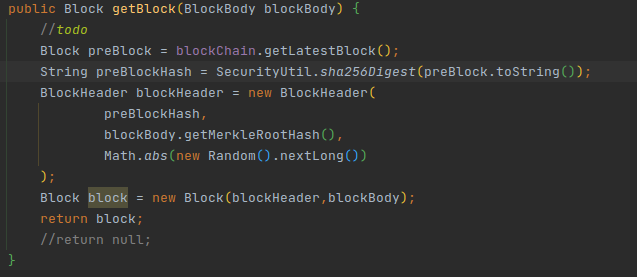
MinerNode的实现





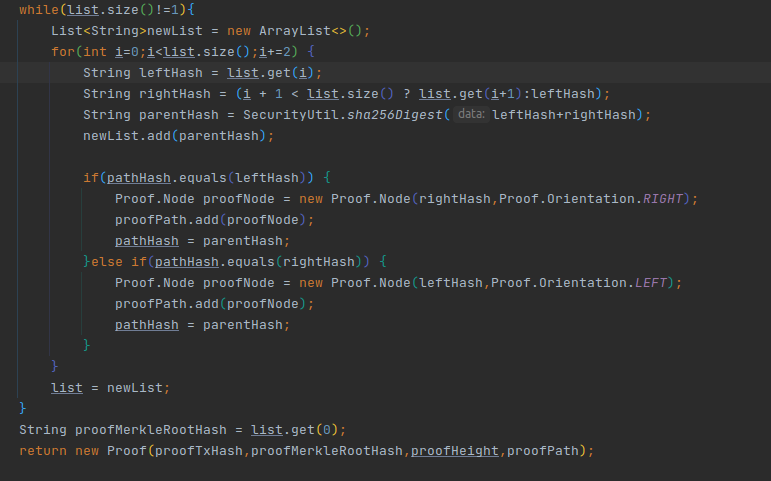




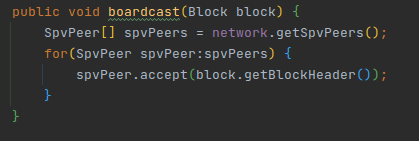


MinerPeer中getProof的实现

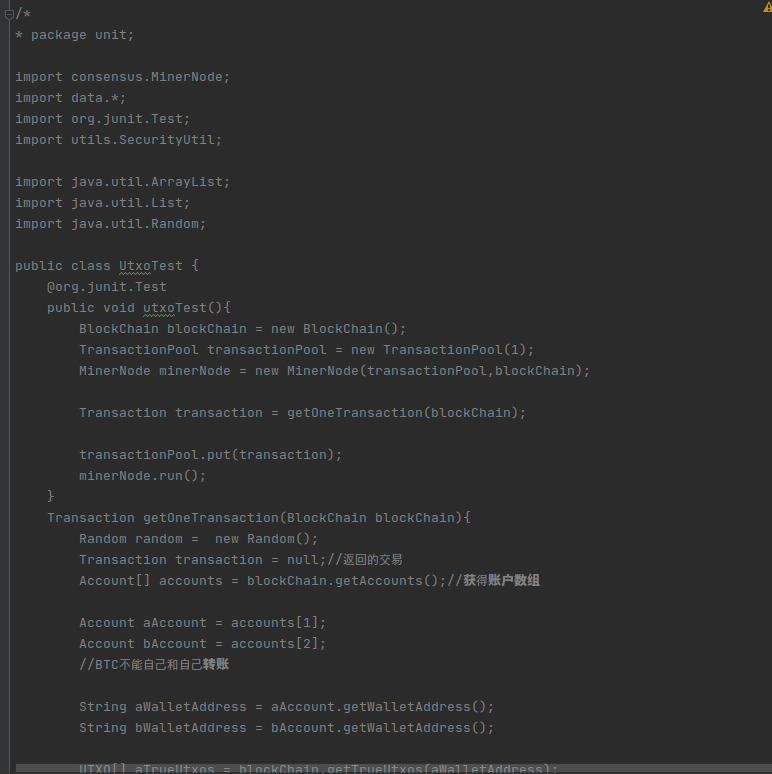


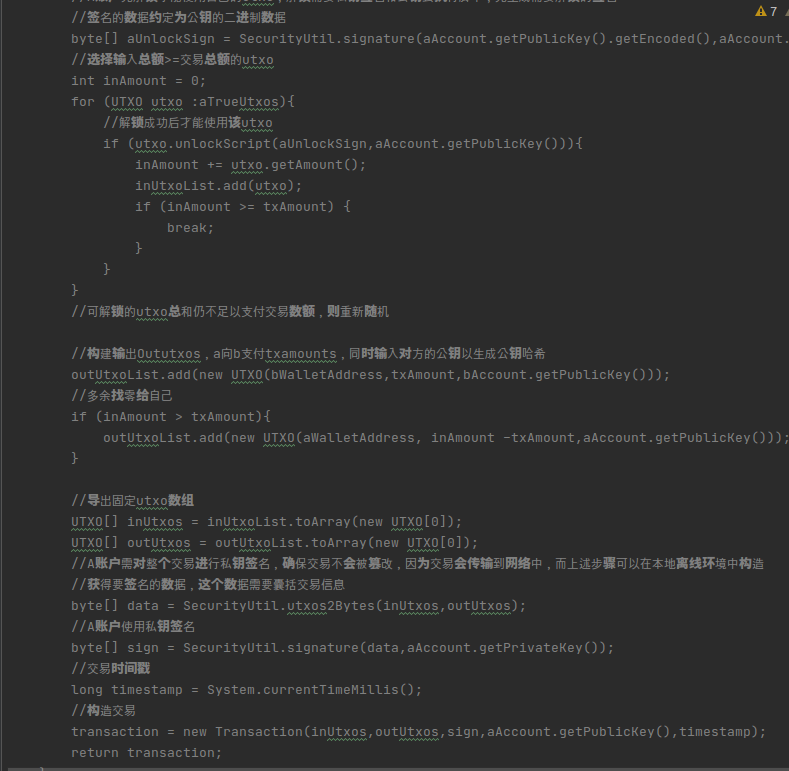


Boardcast的实现

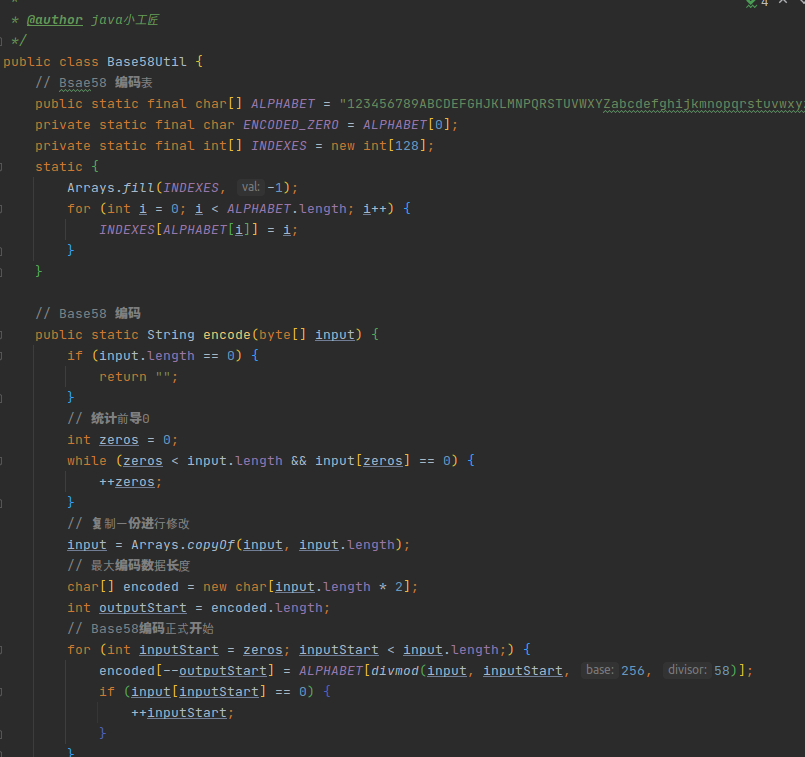


UtxoTest的实现

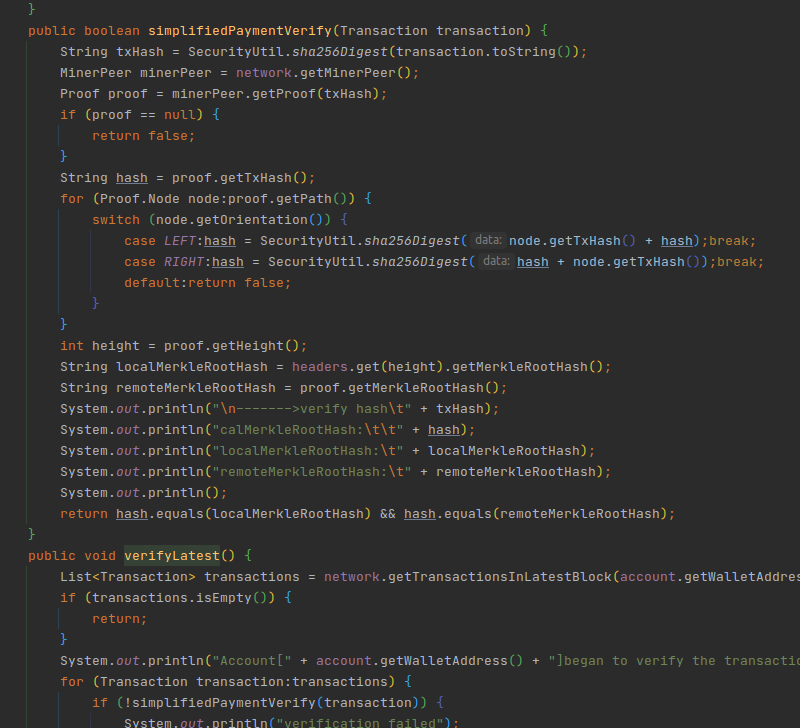




Base58Util实现

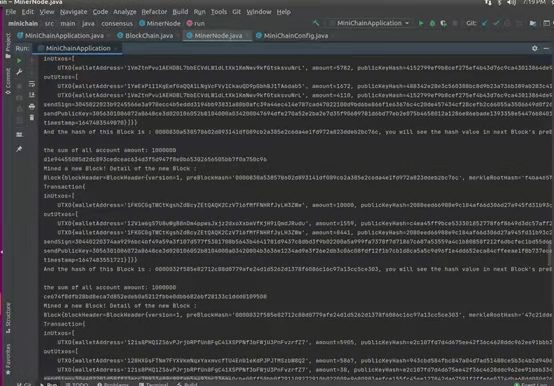


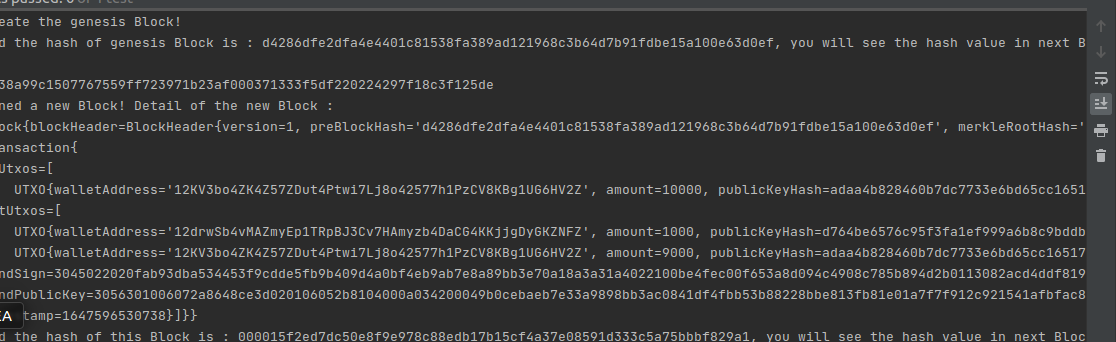
Spvpeer实现



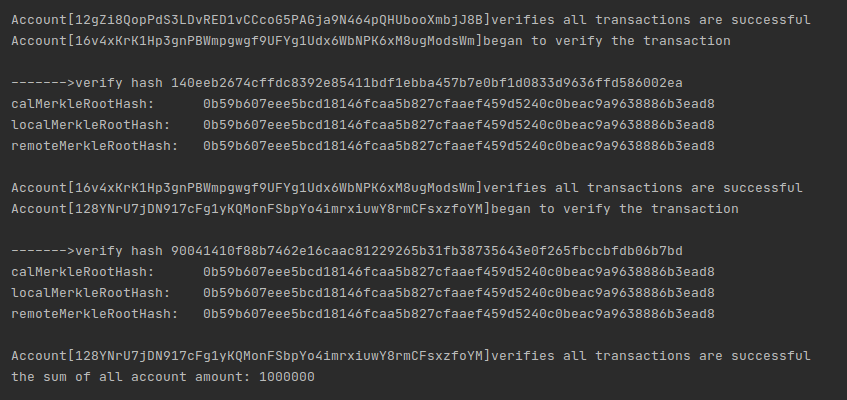
* + - 1. 实验结果

实验2





实验3



1. **总结**

通过简易区块链相关代码，逻辑说明和实验结果图，对merkle根哈希，签名、验签以及UTXO模型和SPV交易等进行了学习了解和深入理解。