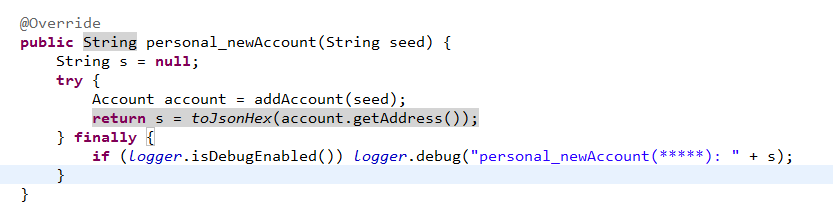
1. String cowAcct = jsonRpc.personal\_newAccount("cow");



获取账户地址，输入信息是用户名（作为种子），涉及到的类有Account

其中Account代表一个真实的账户或联系方式，

Account的主要属性：

1. 用于签名的公私钥ecKey（ECKey）
2. 账户地址（byte[]）
3. 待验证的交易集合（Set<Transaction>）

主要方法：

1. getNonce()，获取该账户的某一地址发送的交易数，或者合约账户模式下，该账户创建的合约数目
2. getBalance()，获取账户余额
3. getAddress()，获取账户地址
4. getPendingTransactions()/addPendingTransaction(Transaction)/clearAllPendingTransactions()，获取/添加/清空账户待验证的交易

注释：注入repository（Repository，接口），Account的行为库

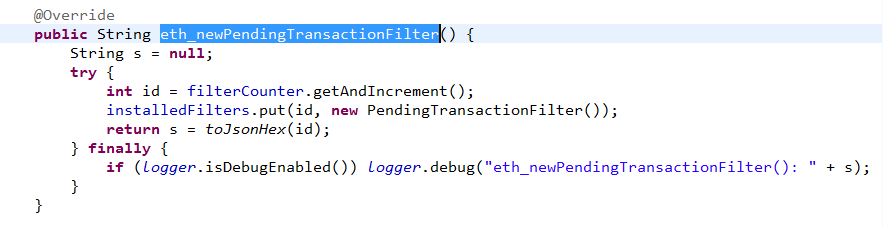
Repository的主要行为有：

1. createAccount(byte[] addr)，在数据库中创建一个账户，返回类型为AccountState
2. isExist(byte[] addr),判断账户是否存在
3. getAccountState(byte[] addr),检索账户
4. deleteZ(byte[] addr),删除账户
5. increaseNonce(byte[] addr)，增加账户的交易数，若addr的账户不存在，则创建账户，并使其交易数加1
6. setNonce(addr, nonce),设置账户的交易数
7. getNonce(addr),获取账户的交易数
8. getContractDetails(addr)，从数据库中检索账户的合约详情
9. hasContractDetails(addr), 判断账户是否有合约
10. saveCode(addr, code),存储账户的代码
11. getCode(addr),检索账户的代码

2. String bal0 = jsonRpc.eth\_getBalance(cowAcct);

获得账户的余额，bal0 is 0x100000000000000000000000000000000000000000000000000

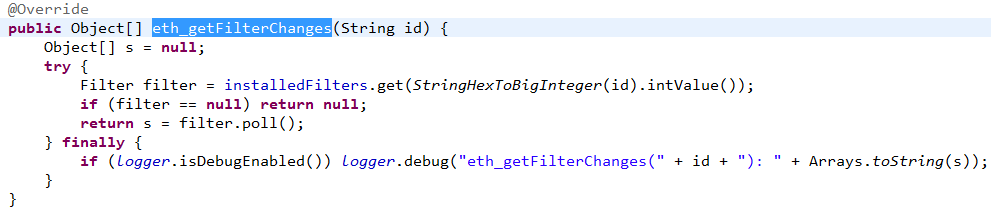
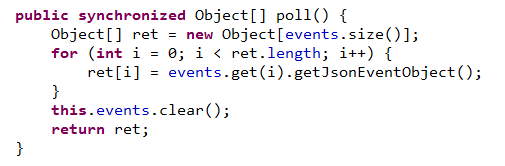
3. String pendingTxFilterId = jsonRpc.eth\_newPendingTransactionFilter();



获取并注册待验证交易过滤器，返回过滤器索引

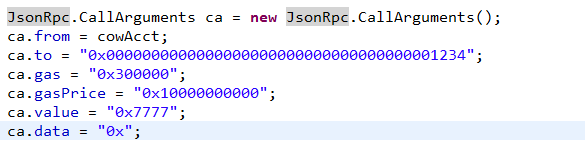
此时，pendingTxFilterId= 0x1

4. Object[] changes = jsonRpc.eth\_getFilterChanges(pendingTxFilterId);

获取索引为pendingTxFilterId的过滤器事件数组，此时changes为空

5. 定义JsonRpc的调用参数：



from-交易付款方，to-交易收款方，gas-交易燃料，gasPrice-交易燃料价格，value-交易价值,data-交易数据。

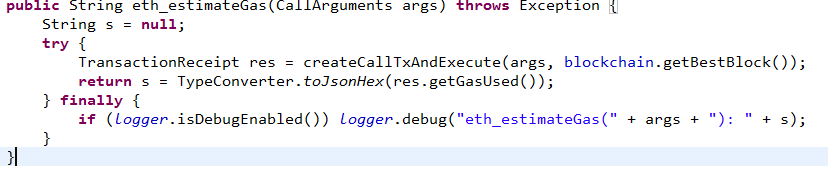
注：在以太坊运行程序需要付费，可以理解成比特币中的交易费用。以太坊的交易费用=gas数量 \* gas price ，gas price (gas 单价，以以太币计）可以由开发者设置。不同的计算类型所需的gas也不一样，计算越复杂，需要的gas越多。所以，如果想运行程序，先用以太币购买gas。每个区块提供的gas是有上限的，目前大约是310万个gas。如果你设置的gas price太低，是买不到gas的，这就是一个gas市场。 gas可以翻译为燃料，加了燃料后，以太坊共识引擎才能启动。就像汽车一样，先用人民币去买汽油。

6. 根据ca计算当前可用油耗的上限

long sGas = TypeConverter.StringHexToBigInteger(jsonRpc.eth\_estimateGas(ca)).longValue();

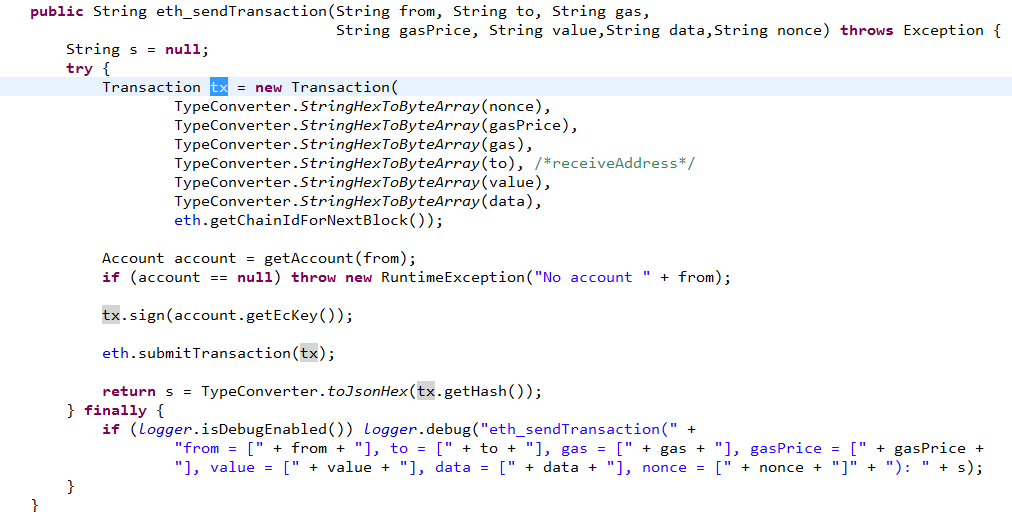
以太坊平台处理调用参数ca得到sGas,其中sGas表示可用的最大gas，这里涉及到许多的处理，具体分析在后面，主要是TransactionExecutor

此时，sGas = 21000



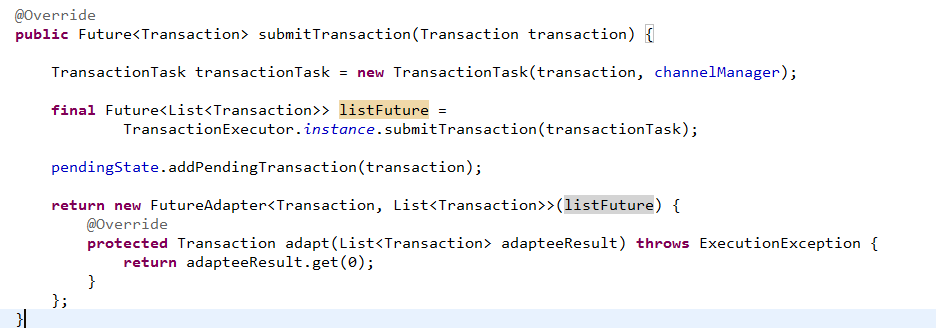
7. String txHash1 = jsonRpc.eth\_sendTransaction(cowAcct, "0x0000000000000000000000000000000000001234", "0x300000",

"0x10000000000", "0x7777", "0x", "0x00");



发送交易，返回交易的hash。实参与上文中设置的调用参数一致，nonce=0x00,表示交易的处理次数，确保每个交易只能被处理一次

处理过程主要包括创建交易，交易签名，发送交易submitTransaction(tx)

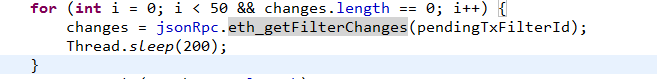


将交易发送给其他节点

返回的交易hash:

Tx hash: 0x5f3d6118867367add63a2a9316d05d9d1c5e649ee9e15eaa3b4b9c2d9e2c8bf3

8.



在10（200\*50/1000=10）秒钟内根据待验证交易过滤器ID捕捉过滤器事件。由于刚刚提交了一个待验证交易，所以会有一个触发事件被捕捉到，至于为什么是10秒的时间待确认

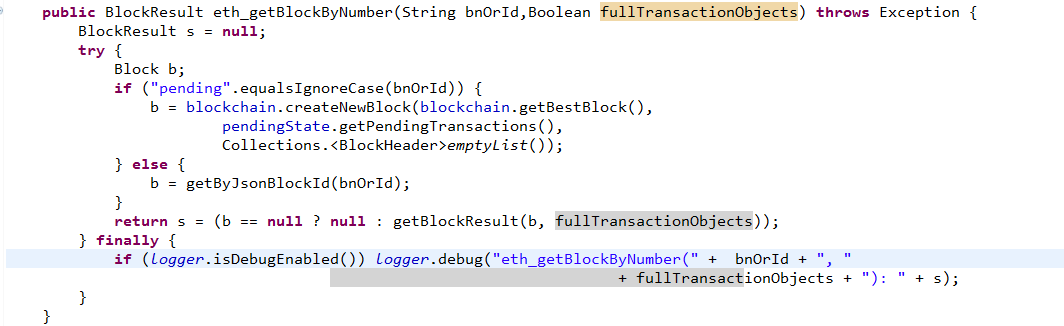
Changes[0] = 0x5f3d6118867367add63a2a9316d05d9d1c5e649ee9e15eaa3b4b9c2d9e2c8bf3

Changes[0] = txHash1

9. changes = jsonRpc.eth\_getFilterChanges(pendingTxFilterId);

此时再次捕捉过滤器事件，将不会捕捉到，因为没有发送新的交易

10. JsonRpc.BlockResult blockResult = jsonRpc.eth\_getBlockByNumber("pending", true);



新建块结果，这里将为待验证的交易创建一个块。在实现代码中，可以看到，根据其状态“pending”，将为其创建一个新块。

此时这个块中包含一个交易，返回blockResult：

BlockResult{

number='null', //块号，当块处于pending状态时，此值为null

hash='null', //块hash，当块处于pending状态时，为null

parentHash='0x2e9340bb94b8f1e37e08209cf7808111347911c9df432a1699589be75344f907', //从何处获得？，父块hash.第一块的hash是多少？

nonce='null', //工作量证明算法的计数器，对应块头属性nonce,当块处于pending状态时，此值为null

sha3Uncles='0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347', //SHA3 of the uncles data in the block.。当uncles=[]时，统一为这个值

logsBloom='null', //bloom过滤器的日志，当块处于pending状态时，为null

transactionsRoot='0x72664c15a8417307190c026dc5d71a100edf1cec77a1026a8c4641dbd6a5df32', //区块中交易的Merkle树根的数据

stateRoot='0x7731eafac894bb6b563c5301aa33c7aa21253a8128fb005c7c1f00c789d33dd9', //块中最终状态树的根数据

receiptsRoot='0xebaecc2f25da988b95ad7b3a4a7c86968c1cf8c743a50ab15658256a9c12481e', //块中收据树的根数据

miner='null', //获得挖矿奖励的账户

difficulty='0x10', //该块的难度

totalDifficulty='0x10', //区块链直到该块的总难度

extraData='0xcccccccccccccccccccc', //对应块属性‘extraData’，具体含义是什么

size='0x24d', //块数据的字节数

gasLimit='0x1000000000', //该块允许的最大油耗

gasUsed='0x5208', //该块内所有交易已使用的油耗总量

timestamp='0x5949526f', //验证块时的unix时间戳

transactions=[TransactionResultDTO{

hash='0x5f3d6118867367add63a2a9316d05d9d1c5e649ee9e15eaa3b4b9c2d9e2c8bf3',

nonce='0x00', //对应Transaction的nonce,确保每一笔交易只能被处理一次的计数器blockHash='0xd65e2d9d41fc2d855db44c92ed009a820113936da37673f8365ce935177d0b36', //该块的hash为null，这里的值从哪里来的，而且后面会更改为所属块hash

blockNumber='0x1', //是初始值吗？

transactionIndex='0x0',

from='0xcd2a3d9f938e13cd947ec05abc7fe734df8dd826', to='0x0000000000000000000000000000000000001234',

gas='0x300000', //对应Transaction 的gasLimit：本次计算允许的油耗量。Gas是计算、引擎的燃料，每个计算步骤和每个添加到状态或交易列表的字节都需要gas

gasPrice='0x010000000000', //对应Transaction 的gasPrice：以太坊支付给矿工每单位油耗的交易费价格

value='0x7777', //对应Transaction的value:消息调用的指定输入数据，类型为限制长度的字节数组

input='0x' //对应Transaction的Data

}],

uncles=[] //Array of uncle hashes

}

分析blockResult数据可知：

块处于pending状态时，number、hash、nonce、logsBloom值为null。

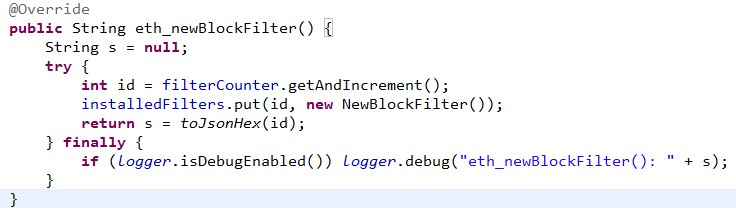
11. String hash1 = mineBlock();

开始挖矿

挖矿的具体实现：

1. String blockFilterId = jsonRpc.eth\_newBlockFilter();

获取块过滤器索引ID



由eth\_newBlockFilter()的实现方法可知，该方法是注册一个新块过滤器，并返回块过滤器索引

1. jsonRpc.miner\_start();

开始挖矿，具体实现方式，有待分析

3)记录被挖出的区块

4) jsonRpc.miner\_stop();停止挖矿

5)Thread.sleep(100);等待0.1秒

6)Object[] blocks = jsonRpc.eth\_getFilterChanges(blockFilterId);根据过滤器索引查找已经挖出的区块，此时应该没有区块被找到，因为已经停止了挖矿，根据后面的输出语句可以验证在挖矿过程中只有一块区块被挖出

7) boolean b = jsonRpc.eth\_uninstallFilter(blockFilterId);注销过滤器

12. JsonRpc.BlockResult blockResult1 = jsonRpc.eth\_getBlockByHash(hash1, true);

根据新块hash1生成块结果，变量blockResult1与blockResult对应的block是同一个，从之后的程序看出，blockResult和blockResult1并没有被调用，那么BlockResult类的实际意义是什么？

BlockResult定义在JsonRpc.java中，仅为JsonRpcTest测试使用。JsonRpc.java这个接口内部定义类有：SyncingResult、CallArguments、BlockResult、CompilationResult、CompilationInfo、FilterRequest、LogFilterElement。由此可看出JsonRpc应该是一个客户端程序接口，用于远程程序调用以太坊平台的各项功能

这时的blockResult1：标红的字段值被填充或校正为真正的值

BlockResult{

number='0x1',

hash='0x3424c90cc36a531f04638841f7fb8635f846af6178ec990d72739574c5de93dd',

parentHash='0x2e9340bb94b8f1e37e08209cf7808111347911c9df432a1699589be75344f907',

nonce='0xaa0fa0378af353a8',

sha3Uncles='0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347',

logsBloom='0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',

transactionsRoot='0x72664c15a8417307190c026dc5d71a100edf1cec77a1026a8c4641dbd6a5df32',

stateRoot='0x7731eafac894bb6b563c5301aa33c7aa21253a8128fb005c7c1f00c789d33dd9',

receiptsRoot='0xebaecc2f25da988b95ad7b3a4a7c86968c1cf8c743a50ab15658256a9c12481e',

miner='0x0000000000000000000000000000000000000000',

difficulty='0x10',

totalDifficulty='0x20',

extraData='0xcccccccccccccccccccc',

size='0x275',

gasLimit='0x1000000000',

gasUsed='0x5208',

timestamp='0x594f3e53',

transactions=[TransactionResultDTO

hash='0x5f3d6118867367add63a2a9316d05d9d1c5e649ee9e15eaa3b4b9c2d9e

2c8bf3',

nonce='0x00',

blockHash='0x3424c90cc36a531f04638841f7fb8635f846af6178ec990d72739574c5de

93dd',

blockNumber='0x1',

transactionIndex='0x0',

from='0xcd2a3d9f938e13cd947ec05abc7fe734df8dd826',

to='0x0000000000000000000000000000000000001234',

gas='0x300000',

gasPrice='0x010000000000',

value='0x7777',

input='null'}

],

uncles=[]}

13. TransactionReceiptDTO receipt1 = jsonRpc.eth\_getTransactionReceipt(txHash1);

根据交易hash获取交易收据信息，TransactionReceiptDTO类中定义了transactionHash、transactionIndex、blockHash、blockNumber、cumulativeGasUsed、gasUsed、contractAddress和logs(交易生成过程中的日志信息)属性

receipt1:TransactionReceiptDTO{

transactionHash='0x5f3d6118867367add63a2a9316d05d9d1c5e649ee9e15eaa3b4b9c2d9e2c8bf3',

transactionIndex='0',

blockHash='0x3424c90cc36a531f04638841f7fb8635f846af6178ec990d72739574c5de93dd',

blockNumber='1',

cumulativeGasUsed='21000',

gasUsed='21000',

contractAddress='null'}

cumulativeGasUsed=gasUsed=sGas

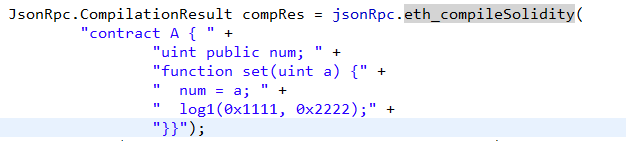
14. String bal1 = jsonRpc.eth\_getBalance(cowAcct); 账户cowAcct此时的余额

bal1: 0x100000000000000000000000000000000000000000000000000

bal1=bal0，挖矿后，账户的 balance并没有变化，为什么？

**根据合约定义交易2**

15.定义智能合约程式：



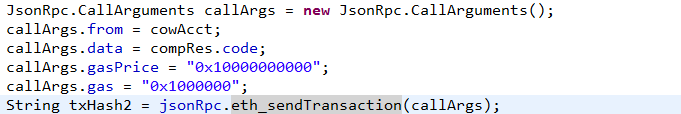
compRes.code:0x606060405234610000575b60a9806100186000396000f300606060405263ffffffff60e060020a6000350416634e70

b1dc8114602c57806360fe47b1146048575b6000565b3460005760366057565b60408051918252519081900360200190f35b346000576

055600435605d565b005b60005481565b600081905560408051611111815290516122229181900360200190a15b505600a165627a7a72

30582083b02d561d58ed5dc714dcf5beb449032f1d7e8e2c539ca253b5482c17551d910029

16.生成并发送交易2，用于生成交易2的参数有from,data,gasPrice,gas.参数中没有to,这笔交易的目标账户是谁？



17. 计算此时的sGas



sGas:67043

18.对第二笔交易进行挖矿



19. 生成第2笔交易的块结果



blockResult2:BlockResult{

number='0x2',

hash='0xda218f4cd8cd4f16b14f6dd1c71928dcb6601ad55215b6a403e0ca98fa7ff4b8',

parentHash='0x3424c90cc36a531f04638841f7fb8635f846af6178ec990d72739574c5de93dd',

nonce='0x9735882a22395ac1',

sha3Uncles='0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347',

logsBloom='0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',

transactionsRoot='0x901abce22aa7e5ad4b88b5ed6856110a1b598d4641a988488c25c8fc3eea8ca5',

stateRoot='0xfa2935f7d8b0c2b8a31b11dedb9731a648cd81be3a26bbd18c0348a2c8c2ef1c',

receiptsRoot='0x1f24d984a9e8878abda44c8b6a5c6fb637ec3e5b3eb9c3c7b50694dba71ce569',

miner='0x0000000000000000000000000000000000000000',// 在何处赋值

difficulty='0x10',

totalDifficulty='0x30',

extraData='0xcccccccccccccccccccc',

size='0x325', gasLimit='0x1000000000',

gasUsed='0x105e3',

timestamp='0x594f3e5a',

transactions=[TransactionResultDTO{

hash='0x7348d48fbc3a4d6c74ac2138bd851bff851549ea8c604329efdfba59070137dd',

nonce='0x01',

blockHash='0xda218f4cd8cd4f16b14f6dd1c71928dcb6601ad55215b6a403e0ca98fa7ff4b8',

blockNumber='0x2',

transactionIndex='0x0',

from='0xcd2a3d9f938e13cd947ec05abc7fe734df8dd826',

to='null',

gas='0x01000000',

gasPrice='0x010000000000',

value='0x00',

input='0x606060405234610000575b60a9806100186000396000f300606060405263ffffff

ff60e060020a6000350416634e70b1dc8114602c57806360fe47b1146048575b6000565b346

0005760366057565b60408051918252519081900360200190f35b346000576055600435605d5

65b005b60005481565b600081905560408051611111815290516122229181900360200190a1

5b505600a165627a7a7230582083b02d561d58ed5dc714dcf5beb449032f1d7e8e2c539ca253

b5482c17551d910029'}],//输入交易，在哪里指定的

uncles=[]}

20.生成交易2的交易收据DTO（data transfer object）



receipt2:TransactionReceiptDTO{

transactionHash='0x7348d48fbc3a4d6c74ac2138bd851bff851549ea8c604329efdfba59070137dd',

transactionIndex='0',

blockHash='0xda218f4cd8cd4f16b14f6dd1c71928dcb6601ad55215b6a403e0ca98fa7ff4b8',

blockNumber='2',

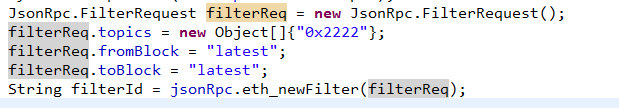
cumulativeGasUsed='67043',//如何计算出？

gasUsed='67043',

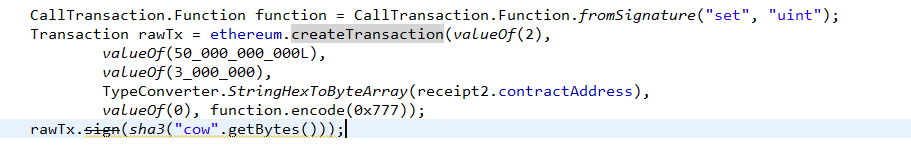
contractAddress='0xda7ce79725418f4f6e13bf5f520c89cec5f6a974'}

从上述分析可知，交易2没有经过账户签名,为什么呢？

21.定义过滤器请求信息，根据请求信息获得过滤器ID



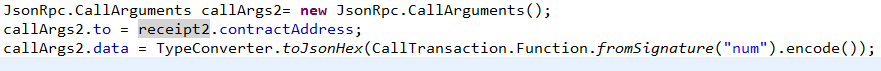
22.定义第三笔交易rawTx，function为交易调用智能合约函数，函数名为set,参数为uint。并将rawTx用账户名签名。



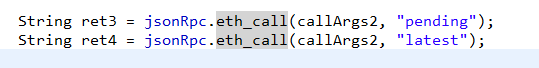
24.发送交易rawTx

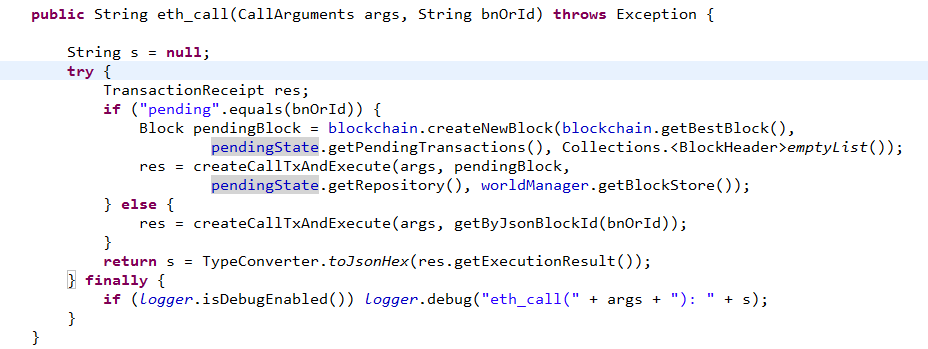


25.定义调用参数callArgs2：目标地址为交易2收据的合约地址，数据为智能合约函数num的代码



26.ret3和ret4位根据调用参数callArgs2根据状态“pending”和“latest”返回的交易收据执行结果





27.挖矿



28.生成块结果blockResult3



blockResult3:BlockResult{

number='0x3',

hash='0xe53703d3bab3ec2e9d35bbfb18b562ac5f5e35d2409a18cb81c4dc1e3272b42a',

parentHash='0xda218f4cd8cd4f16b14f6dd1c71928dcb6601ad55215b6a403e0ca98fa7ff4b8',

nonce='0x1a871cdd5f6837f3',

sha3Uncles='0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347',

logsBloom='0x00000000000000000000000000000000000000000000000000000000000000000000000000080000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000008000000000000200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000040000000000000000000040000',

transactionsRoot='0xcbbf99c3bb430da474b0a8758ed14e1199985ee65c5270e6b6bf9693b0ed16cc',

stateRoot='0x7e248da61ad8e8fbb4031f47eec1b73fa2d9d3fc1d57554b7fea4487e68d0ac3',

receiptsRoot='0xfb0d3559ed561d813b6a0802e00023117befe3e34439b9b3def321517a45ffb2',

miner='0x0000000000000000000000000000000000000000',

difficulty='0x10',

totalDifficulty='0x50',

extraData='0xcccccccccccccccccccc',

size='0x296',

gasLimit='0x1000000000',

gasUsed='0xa6fc',

timestamp='0x594f3e5b',

transactions=[TransactionResultDTO{

hash='0xfcc7625bfcb1392738482b80afd003433e8b55a9f7a697cbbe99d12d985f9e02',

nonce='0x02',//这里的nonce为什么是2，在Transaction中nonce:a counter used to make sure each transaction can only be processed once

blockHash='0xe53703d3bab3ec2e9d35bbfb18b562ac5f5e35d2409a18cb81c4dc1e3272b42a',

blockNumber='0x3',

transactionIndex='0x0',

from='0xcd2a3d9f938e13cd947ec05abc7fe734df8dd826',

to='0xda7ce79725418f4f6e13bf5f520c89cec5f6a974',

gas='0x2dc6c0',

gasPrice='0x0ba43b7400',

value='0x00',

input='0x60fe47b10000000000000000000000000000000000000000000000000000000000000777'}],

uncles=[]}

29.生成交易数据DTO



receipt3:TransactionReceiptDTO{

transactionHash='0xfcc7625bfcb1392738482b80afd003433e8b55a9f7a697cbbe99d12d985f9e02',

transactionIndex='0',

blockHash='0xe53703d3bab3ec2e9d35bbfb18b562ac5f5e35d2409a18cb81c4dc1e3272b42a',

blockNumber='3',

cumulativeGasUsed='42748',

gasUsed='42748',

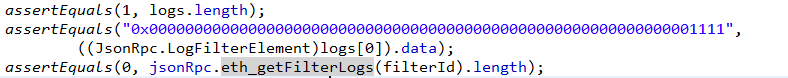
contractAddress='null'}

30. 根据filterId生成过滤器记录



filterId = blockResult.number

31. 两次断言：一次为1，一次为0.为什么？



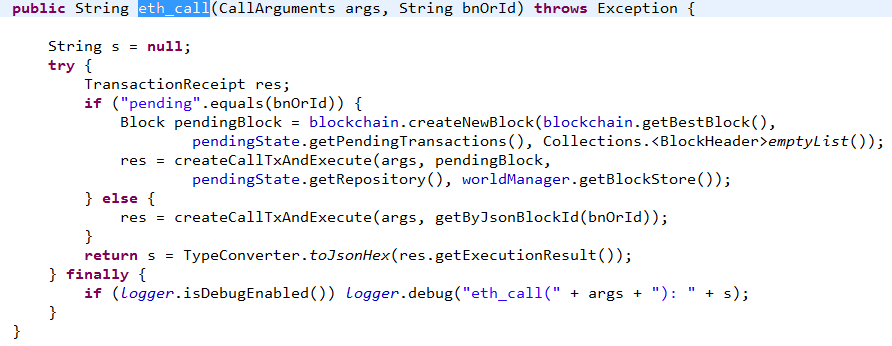
运行结果证明断言是正确的，为什么

32.最后断言了ret1,ret2,ret3,ret4

String ret1 = jsonRpc.eth\_call(callArgs2, blockResult2.number);

String ret2 = jsonRpc.eth\_call(callArgs2, "latest");

callArgs2的定义参见第25条



问题1：如何查看块链中块信息

交易执行过程中涉及类的解释：

WorldManager：是一个单例模式，包含了系统中各不同部分的引用

主要有：

logger Logger类型 ，打印general类的日志

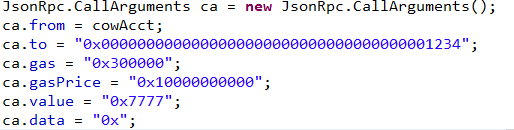
activePeer PeerClient类型 Netty框架下与远程账户的远程链接

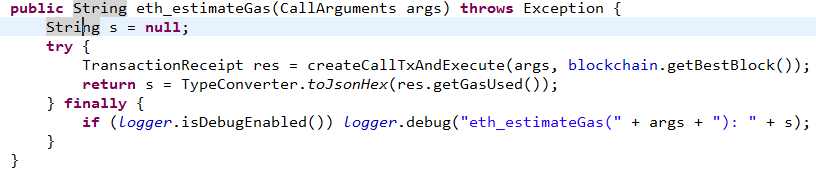
channelManager ChannelManager类型，管道管理器

adminInfo AdminInfo

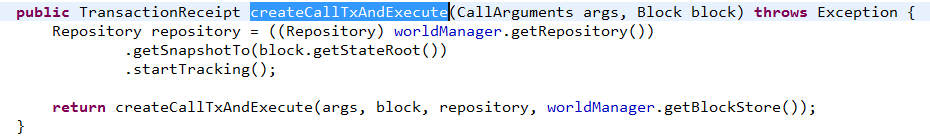
32. eth\_estimateGas(CallArguments args)方法详解：

此时args的定义为





createCallTxAndExecute(args, block)实现：



此时**block**：

BlockData [

hash=2e9340bb94b8f1e37e08209cf7808111347911c9df432a1699589be75344f907//blockhash

下面几项都是blockheader信息

hash=2e9340bb94b8f1e37e08209cf7808111347911c9df432a1699589be75344f907

parentHash=0000000000000000000000000000000000000000000000000000000000000000

unclesHash=1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347

coinbase=0000000000000000000000000000000000000000

stateRoot=95d5a2696ca75d0c4ec7e0aa4b474e514a8a71dd54d0bc9999c4bbc1f995d6e6

txTrieHash=56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421

receiptsTrieHash=56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421

difficulty=10

number=0

gasLimit=1000000000

gasUsed=0

timestamp=0 (1970.01.01 08:00:00)

extraData=11bbe8db4e347b4e8c937c1c8370e4b5ed33adb3db69cbdb7a38e1e50b1b82fa

mixHash=0000000000000000000000000000000000000000000000000000000000000000

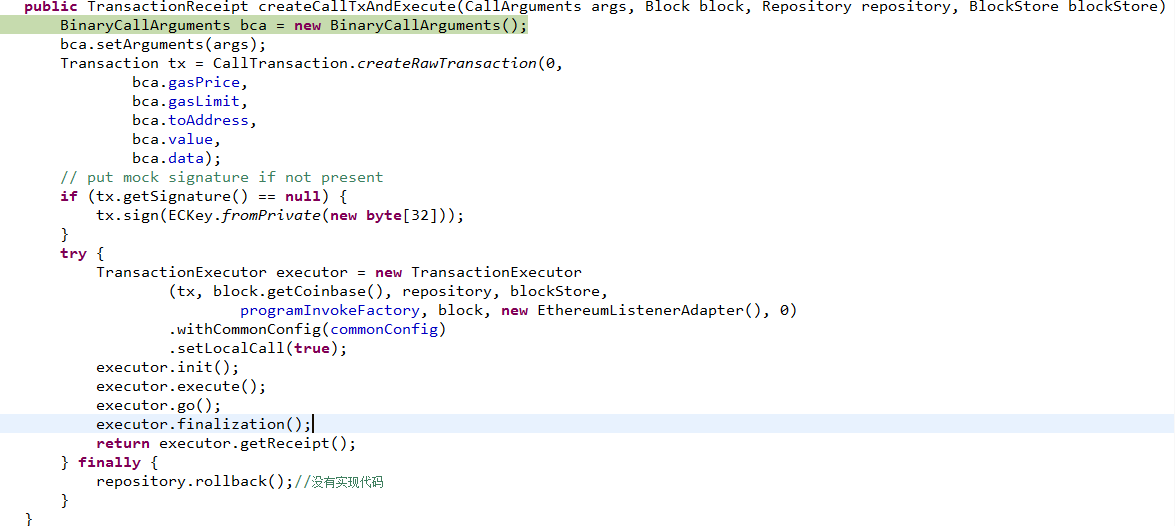
nonce=0000000000000000

Uncles []

Txs []

]

createCallTxAndExecute(CallArguments args, Block block, Repository repository, BlockStore blockStore)



参数格式的转换：

bca.setArguments(args):

在CallArguments中参数类型都是String类型

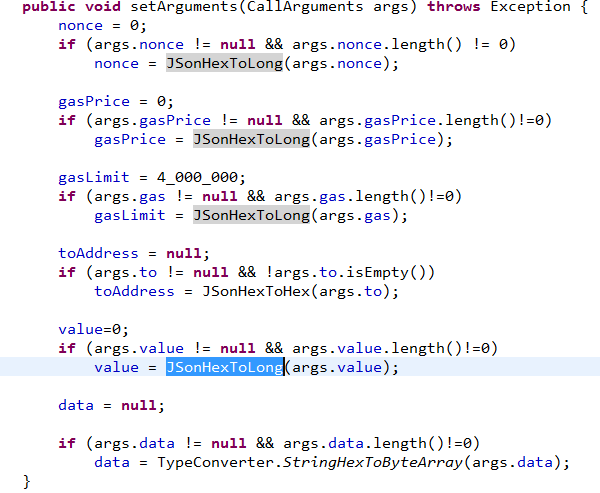
转换后的数据：

data=[]

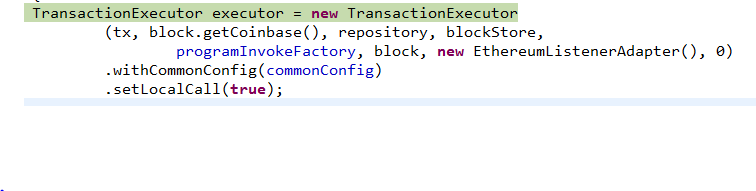
gasLimit = 3145728

gasPrice = 1099511627776

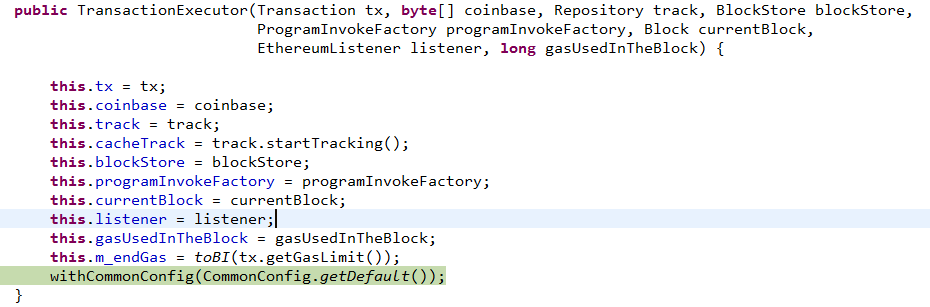
value=30583



根据bca创建交易，交易的参数是byte[]类型



实现代码：



**tx:**

TransactionData [hash= nonce=,

gasPrice=010000000000,

gas=300000,

receiveAddress=0000000000000000000000000000000000001234,

sendAddress=dcc703c0e500b653ca82273b7bfad8045d85a470,

value=7777,

data=,

signatureV=27,

signatureR=9d66d45eee3fc252f00b62c04dc6cd379a41649280dc8fffcf3d55665b194da6,

signatureS=7ab3a328b74e50ad379f0101a4b6e0d46ae2e52790100b68cd22d199f48ddea1]

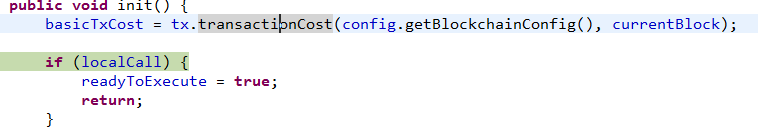
**coinbase**: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

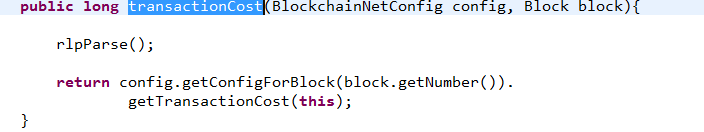
**currentBlock=block**

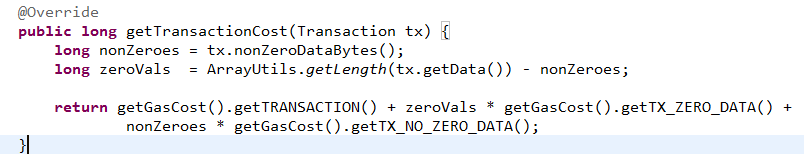
**gasUsedInTheBlock**=0

**m\_endGas**=3145728=**gasLimit**(bca中的参数)

executor.init()



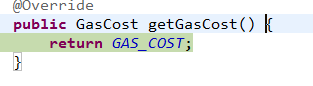




tx中data=[]

nonZeroes=0

zeroVals=0



GasCost中定义了

所以basicTxCost = 21000

接下来分析：

**public** TransactionExecutionSummary finalization()