项目设计与功能测试

18340188 严佳星 18340195 杨智博 18342016 范建聪

一、项目设计说明

1、变量设置

(1) 账单存储

使用列表存储账单，结构体账单内包括债权方 (string)、债务方 (string) 、金额 (float)、交易货物 (string) 、账单是否有效(bool)五个成员：

```
struct voucher
{
    bytes32 creditor; //债权方
    bytes32 debtor; //债务方
    int amount; //交易金额
    bytes32 goods; //交易货物
    bool valid; //有效性
}
voucher[] private _voucherList;
```

(2) 账户管理

使用结构体(record)存储公司账户信息，账户信息包括公司种类(int)、公司名称(string)、记录信息是否有效(bool)。使用名称到账户余额的映射(map)、哈希地址到信息的映射(map)、名称到哈希地址的映射(map)共同管理账户信息，便于后续的查找与访问。

```
struct record
{
    int kind; //kind表示类型，1为核心企业，2为供应链企业，3为第三方可信机构
    bytes32 name; //公司或机构名称
    bool valid; //记录是否有效
}
mapping( bytes32 => int ) private _balances; //名称到账户余额的映射
mapping( address => record ) private _enterprises; //哈希地址到信息的映射
mapping( bytes32 => address ) private _hashAddress; //名称到哈希地址的映射
```

(3) 权限管理

针对所有登录控制台的账户，将其分为四个身份：管理员（只有一个，负责其他三个身份的设定和改变，权限最高）、银行、核心企业、下游企业。设置变量(address)来存储管理员账户的哈希地址，将部署合约的账户设置为管理员，只有管理员可以创建公司信息、设置公司的账户余额等等管理性操作。

```
address private _administrator; //管理员哈希地址

constructor() public
{
    _administrator = msg.sender;
}
```

2、功能实现

(1) 功能一

- 功能需求：实现采购商品—签发应收账款交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。
- 参数：债务方(string) 、金额(int) 、交易货物(string)
- 返回值：true/false
- 实现思路：生成一张新账单用上述四个参数初始化，并且设置为valid。债权方的哈希地址就是msg.sender，我们可以直接通过映射来访问到对应的名称。需要注意的是，由于只有下游企业可以向核心企业供货，需要检查债务方是否是下游企业、msg.sender对应的是否为核心企业。不需要检查账户余额是否大于账单金额 。

```
//功能一
function createVoucher(bytes32 creditor,int amount,bytes32 goods) public
isCoreEnterprises returns(bool)
{
    require(_enterprises[msg.sender].name != creditor,"Creditor and Debtor
can't be the same!");
    require(_enterprises[_hashAddress[creditor]].valid,"Invalid Creditor!");
    require(_enterprises[_hashAddress[creditor]].kind == 2,"Creditor has to
be a SupplyEnterprise");
    _voucherList.push(voucher({creditor:creditor,
                               debtor:_enterprises[msg.sender].name,
                               amount:amount,
                               goods:goods,
                               valid:true}));
    emit
createVoucherEvent(creditor,_enterprises[msg.sender].name,amount,goods,"Core
-Supply Transaction");
    return true;
}
```

(2) 功能二

- 功能需求：实现应收账款的转让上链，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还钱款。
- 参数：接受方(string)、转让金额(int) 、交易货物 (string)
- 返回：true/false
- 思路：遍历所有有效，且债权方为自己的账单，若找到一张金额大于转让金额的账单，则直接修改该账单的相关信息，返回true，当前账单金额小于转让金额的数目，则将其累加；若累加的金额大于等于了转让的金额，则向前销毁所有的符合要求的账单，然后把最近遍历到的自己的账单信息修改，返回true；若遍历到最后都没有找到符合要求的账单或者累加金额小于转让金额，则返回false。转让方的哈希地址就是msg.sender 直接映射得出索引，并且需要检查接收方是否是下游企业。

```
function transferVoucher(bytes32 receiver,int amount,bytes32 goods)
isSupplyEnterprises public returns(bool)
{
    require(_enterprises[_hashAddress[receiver]].valid,"The receiver doesn't
exist!");
    require(_enterprises[_hashAddress[receiver]].kind == 2,"The receiver has
to be a SupplyEnterprise!");
```

```
    int sum = 0;
    bool success = false;
    for(uint i = 0;i < _voucherList.length;i++)
    {
        if(_voucherList[i].valid && _voucherList[i].creditor ==
_enterprises[msg.sender].name)
        {
            if(_voucherList[i].amount > amount)
            {
                _voucherList[i].amount -= amount;
                _voucherList.push(voucher({creditor:receiver,
                                           debtor:_voucherList[i].debtor,
                                           amount:amount,
                                           goods:goods,
                                           valid:true}));
                success = true;
            }
            else if(_voucherList[i].amount == amount)
            {
                _voucherList[i].creditor = receiver;
                success = true;
            }
            else
            {
                sum += _voucherList[i].amount;
            }
            if(sum >= amount)
            {
                if(sum > amount)
                {
                    _voucherList[i].amount = sum - amount;
                }
                else
                {
                    _voucherList[i].valid = false;
                }
                for(uint j = i - 1;j >= 0;j--)
                {
                    if(_voucherList[j].valid && _voucherList[j].creditor ==
_enterprises[msg.sender].name)
                    {
                        _voucherList[j].valid = false;
                    }
                }
                _voucherList.push(voucher({creditor:receiver,
                                           debtor:_voucherList[i].debtor,
                                           amount:amount,
                                           goods:goods,
                                           valid:true}));
                success = true;
            }
        }
    }
    if(success)
    {
        emit
transferVoucherEvent(receiver,_enterprises[msg.sender].name,amount,goods,"Su
pply-Supply V-Transaction");
```

```
    }
    return success;
}
```

(3) 功能三

- 功能需求：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。
- 参数：接受方(string)、转让金额(int)
- 返回：true/false
- 思路：需要检查接收方是否是银行，遍历思路与功能二一致，但是在返回true之前将转让方的账户金额增加。转让方的哈希地址就是msg.sender直接找索引。

```
//功能三
function financingByVoucher(bytes32 receiver,int amount) isSupplyEnterprises public returns(bool)
{
    require(_enterprises[_hashAddress[receiver]].valid,"The receiver doesn't exist!");
    require(_enterprises[_hashAddress[receiver]].kind == 3,"The receiver has to be a ThirdParty!");
    require(_enterprises[msg.sender].kind == 2,"Only SupplyEnterprises can finance by voucher!");
    int sum = 0;
    bool success = false;
    for(uint i = 0;i < _voucherList.length;i++)
    {
        if(_voucherList[i].valid && _voucherList[i].creditor == _enterprises[msg.sender].name)
        {
            if(_voucherList[i].amount > amount)
            {
                _voucherList[i].amount -= amount;
                _voucherList.push(voucher({creditor:receiver,
                                           debtor:_voucherList[i].debtor,
                                           amount:amount,
                                           goods:"",
                                           valid:true}));
                _balances[_enterprises[msg.sender].name] += amount;
                success = true;
            }
            else if(_voucherList[i].amount == amount)
            {
                _voucherList[i].creditor = receiver;
                _balances[_enterprises[msg.sender].name] += amount;
                success = true;
            }
            else
            {
                sum += _voucherList[i].amount;
            }
            if(sum >= amount)
            {
                if(sum > amount)
                {
                    _voucherList[i].amount = sum - amount;
```

```
                }
                else
                {
                    _voucherList[i].valid = false;
                }
                for(uint j = i - 1;j >= 0;j--)
                {
                    if(_voucherList[j].valid && _voucherList[j].creditor == _enterprises[msg.sender].name)
                    {
                        _voucherList[j].valid = false;
                    }
                }
                _voucherList.push(voucher({creditor:receiver,
                                           debtor:_voucherList[i].debtor,
                                           amount:amount,
                                           goods:"",
                                           valid:true}));
                _balances[_enterprises[msg.sender].name] += amount;
                success = true;
            }
        }
    }
    if(success)
    {
        emit financingByVoucherEvent(receiver,_enterprises[msg.sender].name,amount,"Supply-ThirdParty Transaction");
    }
    return success;
}
```

(4) 功能四

- 功能需求：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。
- 参数：付款方(string)
- 返回：收到的款项总数(int)
- 思路：需要检查接收方是否是核心企业。遍历所有债权人为自己，债务人为付款方的所有账单，若余额足够，则销毁该账单，并记录还款额，直至遍历结束或可还余额不足。若余额不足，则访问的最后一张账单只进行修改，而不销毁。

```
//功能四
function payVoucher(bytes32 debtor) isSupplyEnterprises public returns(int)
{
    require(_enterprises[_hashAddress[debtor]].kind == 1,"The debtor has to be a CoreEnterprise!");
    require(_enterprises[msg.sender].kind != 1,"CoreEnterprises can't have vouchers!");
    int pays = 0;
    for(uint i = 0;i < _voucherList.length;i++)
    {
        if(_voucherList[i].valid && _voucherList[i].creditor == _enterprises[msg.sender].name && _voucherList[i].debtor == debtor)
        {
            if (_balances[debtor] - pays >= _voucherList[i].amount)
```

```
            {
                pays += _voucherList[i].amount;
                _voucherList[i].valid = false;
            }
            else
            {
                int rem = _balances[debtor] - pays;
                pays += rem;
                _voucherList[i].amount -= rem;
                break;
            }
        }
    }
    emit payVoucherEvent(_enterprises[msg.sender].name,debtor,pays,"Voucher-to-cash Transaction");
    _balances[debtor] -= pays;
    _balances[_enterprises[msg.sender].name] += pays;
    return pays;
}
```

二、功能测试

1.以管理员身份创建一家核心企业，两家供应链企业和一家银行。

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 create "0x99e5689d596626a5534c74264f593fb9c320f308" "ce000000000000000000000000000000" 1
transaction hash: 0x129773e3e368bdabc5da802d1231ddf98445b35c7614cc222012d6486c8c78b6
-------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
-------------------------------------------------------------------------------------------
Event logs
Event: {"createEnterpriseEvent":[["0x99e5689d596626a5534c74264f593fb9c320f308","q\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000", 1, "Enterprise Created!\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"]]}

[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 create "0x1c7bae8efea058cf51b504c00919daa778b65dd7" "se100000000000000000000000000000" 2
transaction hash: 0xe43baac1e23f8311ffb34c6d54e0ca9062aa00bb77e0cbf44a2b989382f1df04
-------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
-------------------------------------------------------------------------------------------
Event logs
Event: {"createEnterpriseEvent":[["0x1c7bae8efea058cf51b504c00919daa778b65dd7","\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000", 2, "Enterprise Created!\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"]]}

[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 create "0x99e86a885a0a4fd40ebb70174289120110d28bb5" "se200000000000000000000000000000" 2
transaction hash: 0xe0166d2e21c2200f665e9264a7d73f53dfbb3218def6dc554b1a38f1e1bac3bd
-------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
-------------------------------------------------------------------------------------------
Event logs
Event: {"createEnterpriseEvent":[["0x99e86a885a0a4fd40ebb70174289120110d28bb5","\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000", 2, "Enterprise Created!\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"]]}

[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 create "0x5c6dce41f2f64c6e1ed70022bcbd78451736867c" "bank000000000000000000000000000000" 3
transaction hash: 0x306747d19c9355cfa98be98f7ef9ba668f2fc87364c11fd380dabfee4fba7194
-------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
-------------------------------------------------------------------------------------------
Event logs
Event: {"createEnterpriseEvent":[["0x5c6dce41f2f64c6e1ed70022bcbd78451736867c","m\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000", 3, "Enterprise Created!\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"]]}
```

2.管理员给核心企业设定初始余额为10000000

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 setBalance "0x99e5689d596626a5534c74264f593fb9c320f308" 10000000
transaction hash: 0x5312b68c2874d6136301d1abaaa421d3241663ec7f7394ca3c77fb252f9b4239d
---------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
---------------------------------------------------------------------------------------------
Event logs
Event: {"setBalanceEvent":[["0x99e5689d596626a5534c74264f593fb9c320f308",10000000,"Set Balance\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"]]}
```

3.以核心企业账户登录，查看自己的信息

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 getBalance
---------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------------------------
Return values:
[
    10000000
]
---------------------------------------------------------------------------------------------

[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 getRecord
---------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------------------------
Return values:
[
    "q\u0000\u0000\u0000\u0000\u0000\u0000\u0000",
    1
]
---------------------------------------------------------------------------------------------
```

可以看到，其类型值为1，表示核心企业，其余额为初始设置的10000000。

4.核心企业向供应链企业1发起购买，提供单据（**功能1**）

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 createVoucher "se10000000000000000000000000000" 2000000 "wheels00000000000000000000000000"
transaction hash: 0xbea86fcef9b62eb128d6a04f1b48d0d0c05f34b6de685914c931cab58612e47e
---------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
---------------------------------------------------------------------------------------------
Event logs
Event: {"createVoucherEvent":[["\u0000\u0000\u0000\u0000\u0000\u0000\u0000","q\u0000\u0000\u0000\u0000\u0000\u0000\u0000",2000000,"\u0017\u0000\u0000\u0000\u0000\u0000\u0000","Core-Supply Transaction\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"]]}
```

这里我们没有设置查看单据的函数，故只有返回值和event作为成功依据，但后面的融资和还款操作均基于此步，其正确性可以通过后面的操作辅助验证。

5.以供应链企业1登录，向供应链企业2使用持有的单据进行购买，并发生单据转移（**功能2**）

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 transferVoucher "se20000000000000000000000000000" 1000000 "gulugulu00000000000000000000000"
transaction hash: 0x659d3e05735d7c0c6c332af55ae642c9db48cb1b3239ed3976b331469579c1ee
---------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
---------------------------------------------------------------------------------------------
Event logs
Event: {"transferVoucherEvent":[["\u0000\u0000\u0000\u0000\u0000\u0000\u0000","\u0000\u0000\u0000\u0000\u0000\u0000\u0000",1000000,"\u0000\u0000\u0000\u0000\u0000\u0000\u0000","Supply-Supply V-Transaction\u0000\u0000\u0000\u0000\u0000"]]}
```

同上，这里可以证明1确实持有金额大于等于1000000的单据，才能向2购买成功。而单据的转移，需要通过后面的融资和还款来证明。

6.以供应链企业2登录，使用单据向银行发起融资（**功能3**）

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 financingByVoucher "bank00000000000000000000000000" 800000
transaction hash: 0x51dc5d3feba4f7a1b854a0a5b20d879c79cbe55659d4344b4cc4de06dd8f0dc7
---------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 1
---------------------------------------------------------------------------------------------
Event logs
Event: {"financingByVoucherEvent":[["m\u0000\u0000\u0000\u0000\u0000\u0000\u0000","\u0000\u0000\u0000\u0000\u0000\u0000\u0000",800000,"Supply-ThirdParty Transaction\u0000\u0000\u0000"]]}
```

融资成功，证明供应链企业2持有大于800000的单据，效果见下一步的查看余额。

7.查看融资后的余额

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 getBalance
---------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------------------------
Return values:
[
    800000
]
---------------------------------------------------------------------------------------------
```

查看余额发现，余额从默认的0变成了800000，证明融资成功了。

8.通过供应链企业2，向车企发出还款要求（**功能4**）

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 payVoucher "ce000000000000000000000000000000"
transaction hash: 0x72aceccf134cf11b1828559d8fb7cacbc4e9916ebc648c838fbf05acbcc6a119
---------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 200000
---------------------------------------------------------------------------------------------
Event logs
Event: {"payVoucherEvent":[["◆◆◆4◆4◆4◆4◆4◆4◆4◆4◆4\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000","q◆4◆4◆4◆4◆4◆4◆4◆4◆4\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000",200000,"Voucher-to-cash Transaction\u0000\u0000\u0000\u0000\u0000"]]}
```

此处还款成功，还款金额为200000，证明供应链企业2在此操作前还持有200000的核心企业的单据，与前面的操作的理论结果相同。（还款设定的是将持有的所有单据进行还款，直到债务人余额不足，而债务人前面设置的余额为10000000，是保证充足的）

9.还款后查看余额

```
[group:1]> call SupplyChainFinance 0xca2f113a904c55a878b7bd984503b68be8ef41c6 getBalance
---------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------------------------
Return values:
[
    1000000
]
---------------------------------------------------------------------------------------------
```

还款后查看，余额从800000变为了1000000，证明了还款成功。

附录：完整代码

```
pragma solidity >=0.4.25 <=0.7.0;

contract SupplyChainFinance
{
    struct voucher
    {
        //债权方
        bytes32 creditor;
        //债务方
        bytes32 debtor;
        //交易金额
        int amount;
        //交易货物
        bytes32 goods;
        //有效性
        bool valid;
    }
    struct record
    {
        //kind表示类型，1为核心企业，2为供应链企业，3为第三方可信机构
        int kind;
        //公司或机构名称
```

```
23          bytes32 name;
24          //记录是否有效
25          bool valid;
26      }
27      //账单列表
28      voucher[] private _voucherList;
29      //名称到账户的映射
30      mapping( bytes32 => int ) private _balances;
31      //哈希地址到信息的映射
32      mapping( address => record ) private _enterprises;
33      //名称到哈希地址的映射
34      mapping(bytes32 => address) private _hashAddress;
35      //管理员哈希地址
36      address private _administrator;
37
38      constructor() public
39      {
40          _administrator = msg.sender;
41      }
42      //事件
43      event createEnterpriseEvent(address addr,bytes32 name,int kind,bytes32
    message);
44      event removeEnterpriseEvent(address addr,bytes32 message);
45      event setBalanceEvent(address addr,int balance,bytes32 message);
46      event createVoucherEvent(bytes32 creditor,bytes32 debtor,int
    amount,bytes32 goods,bytes32 message);
47      event transferVoucherEvent(bytes32 creditor,bytes32 debtor,int
    amount,bytes32 goods,bytes32 message);
48      event financingByVoucherEvent(bytes32 creditor,bytes32 debtor,int
    amount,bytes32 message);
49      event payVoucherEvent(bytes32 creditor,bytes32 debtor,int pays,bytes32
    message);
50      //修饰器
51      modifier isAdministrator
52      {
53          require(_administrator==msg.sender,"Auth: administrator required");
54          _;
55      }
56      modifier isCoreEnterprises
57      {
58
     require(_enterprises[msg.sender].valid&&_enterprises[msg.sender].kind==1,"
    Auth: core enterprises required");
59          _;
60      }
61      modifier isSupplyEnterprises
62      {
63
     require(_enterprises[msg.sender].valid&&_enterprises[msg.sender].kind==2,"
    Auth: supply enterprises required");
64          _;
65      }
66      modifier isThirdParties
67      {
68
     require(_enterprises[msg.sender].valid&&_enterprises[msg.sender].kind==3,"
    Auth: third parties required");
69          _;
```

```
70      }
71
72      //获取节点对应的公司或机构名与类型
73      function getRecord() public view returns(bytes32,int)
74      {
75          require(_administrator!=msg.sender,"Auth: administrator does not
    have a enterprise");
76          require(_enterprises[msg.sender].valid,"Auth: this address did not
    create a enterprise");
77          return
    (_enterprises[msg.sender].name,_enterprises[msg.sender].kind);
78      }
79      //获取节点对应的公司或机构的余额
80      function getBalance() public view returns(int)
81      {
82          require(_administrator!=msg.sender,"Auth: administrator does not
    have a enterprise");
83          require(_enterprises[msg.sender].valid,"Auth: this address did not
    create a enterprise");
84          return _balances[_enterprises[msg.sender].name];
85      }
86      //添加新的企业或机构
87
88      function create(address addr,bytes32 name,int kind) public
    isAdministrator returns(bool)
89      {
90          //一个节点只能占据一个企业或机构
91          require(!_enterprises[addr].valid,"The address is occupied!");
92          _enterprises[addr].kind = kind;
93          _enterprises[addr].name = name;
94          _enterprises[addr].valid = true;
95          _balances[name] = 0;
96          _hashAddress[name] = addr;
97          emit createEnterpriseEvent(addr,name,kind,"Enterprise Created!");
98          return true;
99      }
100     //删除地址对应的企业或机构
101
102     function remove(address addr) public isAdministrator returns(bool)
103     {
104         require(_enterprises[addr].valid,"The address represents no
    enterprises");
105         _enterprises[addr].valid = false;
106         emit removeEnterpriseEvent(addr,"Node's Enterprise Removed!");
107         return true;
108     }
109     //给特定账户设置余额
110
111     function setBalance(address addr,int balance) public isAdministrator
    returns(bool)
112     {
113         require(_enterprises[addr].valid,"The address represents no
    enterprises");
114         _balances[_enterprises[addr].name] = balance;
115         emit setBalanceEvent(addr,balance,"Set Balance");
116         return true;
117     }
118     //功能一
```

```
119
120     function createVoucher(bytes32 creditor,int amount,bytes32 goods)
    public isCoreEnterprises returns(bool)
121     {
122         require(_enterprises[msg.sender].name != creditor,"Creditor and
    Debtor can't be the same!");
123         require(_enterprises[_hashAddress[creditor]].valid,"Invalid
    Creditor!");
124         require(_enterprises[_hashAddress[creditor]].kind == 2,"Creditor
    has to be a SupplyEnterprise");
125         _voucherList.push(voucher({creditor:creditor,
126                                    debtor:_enterprises[msg.sender].name,
127                                    amount:amount,
128                                    goods:goods,
129                                    valid:true}));
130         emit
    createVoucherEvent(creditor,_enterprises[msg.sender].name,amount,goods,"Cor
    e-Supply Transaction");
131         return true;
132     }
133     //功能二
134
135     function transferVoucher(bytes32 receiver,int amount,bytes32 goods)
    isSupplyEnterprises public returns(bool)
136     {
137         require(_enterprises[_hashAddress[receiver]].valid,"The receiver
    doesn't exist!");
138         require(_enterprises[_hashAddress[receiver]].kind == 2,"The
    receiver has to be a SupplyEnterprise!");
139         int sum = 0;
140         bool success = false;
141         for(uint i = 0;i < _voucherList.length;i++)
142         {
143             if(_voucherList[i].valid && _voucherList[i].creditor ==
    _enterprises[msg.sender].name)
144             {
145                 if(_voucherList[i].amount > amount)
146                 {
147                     _voucherList[i].amount -= amount;
148                     _voucherList.push(voucher({creditor:receiver,
149                                    debtor:_voucherList[i].debtor,
150                                    amount:amount,
151                                    goods:goods,
152                                    valid:true}));
153                     success = true;
154                 }
155                 else if(_voucherList[i].amount == amount)
156                 {
157                     _voucherList[i].creditor = receiver;
158                     success = true;
159                 }
160                 else
161                 {
162                     sum += _voucherList[i].amount;
163                 }
164                 if(sum >= amount)
165                 {
166                     if(sum > amount)
```

```
                    {
                        _voucherList[i].amount = sum - amount;
                    }
                    else
                    {
                        _voucherList[i].valid = false;
                    }
                    for(uint j = i - 1;j >= 0;j--)
                    {
                        if(_voucherList[j].valid &&
_voucherList[j].creditor == _enterprises[msg.sender].name)
                        {
                            _voucherList[j].valid = false;
                        }
                    }
                    _voucherList.push(voucher({creditor:receiver,
                                    debtor:_voucherList[i].debtor,
                                    amount:amount,
                                    goods:goods,
                                    valid:true}));
                    success = true;
                }
            }
        }
        if(success)
        {
            emit
transferVoucherEvent(receiver,_enterprises[msg.sender].name,amount,goods,"S
upply-Supply V-Transaction");
        }
        return success;
    }
    //功能三

    function financingByVoucher(bytes32 receiver,int amount)
isSupplyEnterprises public returns(bool)
    {
        require(_enterprises[_hashAddress[receiver]].valid,"The receiver
doesn't exist!");
        require(_enterprises[_hashAddress[receiver]].kind == 3,"The
receiver has to be a ThirdParty!");
        require(_enterprises[msg.sender].kind == 2,"Only SupplyEnterprises
can finance by voucher!");
        int sum = 0;
        bool success = false;
        for(uint i = 0;i < _voucherList.length;i++)
        {
            if(_voucherList[i].valid && _voucherList[i].creditor ==
_enterprises[msg.sender].name)
            {
                if(_voucherList[i].amount > amount)
                {
                    _voucherList[i].amount -= amount;
                    _voucherList.push(voucher({creditor:receiver,
                                    debtor:_voucherList[i].debtor,
                                    amount:amount,
                                    goods:"",
                                    valid:true}));
```

```
                    _balances[_enterprises[msg.sender].name] += amount;
                    success = true;
                }
                else if(_voucherList[i].amount == amount)
                {
                    _voucherList[i].creditor = receiver;
                    _balances[_enterprises[msg.sender].name] += amount;
                    success = true;
                }
                else
                {
                    sum += _voucherList[i].amount;
                }
                if(sum >= amount)
                {
                    if(sum > amount)
                    {
                        _voucherList[i].amount = sum - amount;
                    }
                    else
                    {
                        _voucherList[i].valid = false;
                    }
                    for(uint j = i - 1;j >= 0;j--)
                    {
                        if(_voucherList[j].valid &&
_voucherList[j].creditor == _enterprises[msg.sender].name)
                        {
                            _voucherList[j].valid = false;
                        }
                    }
                     _voucherList.push(voucher({creditor:receiver,
                                    debtor:_voucherList[i].debtor,
                                    amount:amount,
                                    goods:"",
                                    valid:true}));
                    _balances[_enterprises[msg.sender].name] += amount;
                    success = true;
                }
            }
        }
        if(success)
        {
            emit
financingByVoucherEvent(receiver,_enterprises[msg.sender].name,amount,"Supp
ly-ThirdParty Transaction");
        }
        return success;
    }
    //功能四

    function payVoucher(bytes32 debtor) isSupplyEnterprises public
returns(int)
    {
        require(_enterprises[_hashAddress[debtor]].kind == 1,"The debtor
has to be a CoreEnterprise!");
        require(_enterprises[msg.sender].kind != 1,"CoreEnterprises can't
have vouchers!");
```

```
        int pays = 0;
        for(uint i = 0;i < _voucherList.length;i++)
        {
            if(_voucherList[i].valid && _voucherList[i].creditor ==
_enterprises[msg.sender].name && _voucherList[i].debtor == debtor)
            {
                if (_balances[debtor] - pays >= _voucherList[i].amount)
                {
                    pays += _voucherList[i].amount;
                    _voucherList[i].valid = false;
                }
                else
                {
                    int rem = _balances[debtor] - pays;
                    pays += rem;
                    _voucherList[i].amount -= rem;
                    break;
                }
            }
        }
        emit
payVoucherEvent(_enterprises[msg.sender].name,debtor,pays,"Voucher-to-cash
Transaction");
        _balances[debtor] -= pays;
        _balances[_enterprises[msg.sender].name] += pays;
        return pays;
    }
}
```