中山大学数据科学与计算机学院本科生实验报告

(2019 年秋季学期)

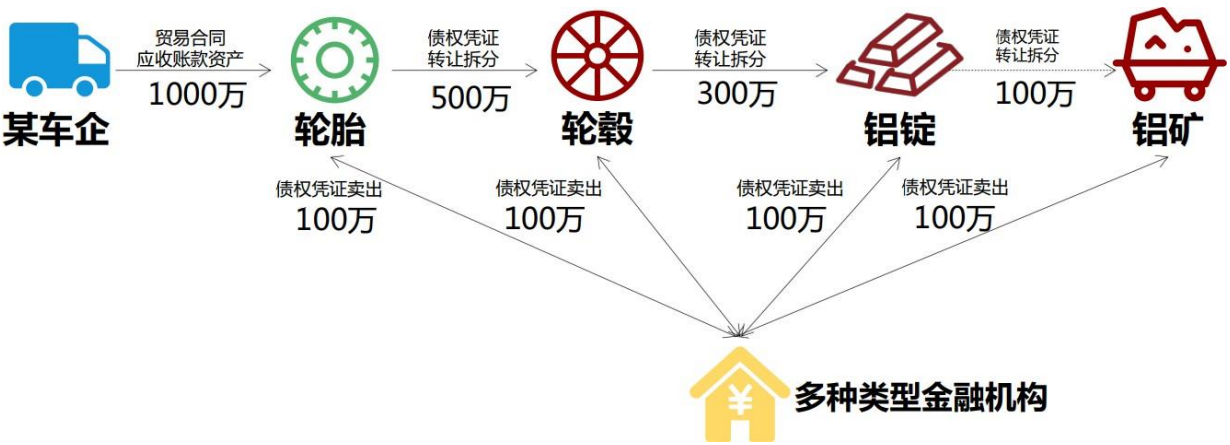
课程名称：区块链原理与技术　　任课教师：郑子彬

年级	17 级	专业（方向）	软件工程
学号	17343027	姓名	冯上清
电话	13570480625	Email	1218424633@qq.com
开始日期	2019/10/27	完成日期	2019/12/13

Github 项目地址：

https://github.com/Passenger0/BlockChainProject

一、 项目背景



传统供应链金融：

某车企（宝马）因为其造车技术特别牛，消费者口碑好，所以其在同行业中占据绝对优势地位。因此，在金融机构（银行）对该车企的信用评级将很

高，认为他有很大的风险承担的能力。在某次交易中，该车企从轮胎公司购买了一批轮胎，但由于资金暂时短缺向轮胎公司签订了 1000 万的应收账款单据，承诺 1 年后归还轮胎公司 1000 万。这个过程可以拉上金融机构例如银行来对这笔交易作见证，确认这笔交易的真实性。在接下里的几个月里，轮胎公司因为资金短缺需要融资，这个时候它可以凭借跟某车企签订的应收账款单据向金融结构借款，金融机构认可该车企（核心企业）的还款能力，因此愿意借款给轮胎公司。但是，这样的信任关系并不会往下游传递。在某个交易中，轮胎公司从轮毂公司购买了一批轮毂，但由于租金暂时短缺向轮胎公司签订了 500 万的应收账款单据，承诺 1 年后归还轮胎公司 500 万。当轮毂公司想利用这个应收账款单据向金融机构借款融资的时候，金融机构因为不认可轮胎公司的还款能力，需要对轮胎公司进行详细的信用分析以评估其还款能力同时验证应收账款单据的真实性，才能决定是否借款给轮毂公司。这个过程将增加很多经济成本，而这个问题主要是由于该车企的信用无法在整个供应链中传递以及交易信息不透明化所导致的。

区块链+供应链金融：

将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让， 融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

必要功能：

- 功能一：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。

- 功能二：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。
- 功能三：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。
- 功能四：应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠款。

二、 方案设计

实现功能：

- 注册账号。
- 发起交易，即采购商品—签发应收账款收据，交易上链。
- 转让收据账款。
- 利用收据账款进行融资。
- 支付双方的收据账款。
- 查询公司资产余额。
- 查询公司注册ID。
- 手动增加公司资产。
- 手动减少公司资产。
- 查询某方与另一方之间所有收据的账款总额。
- 通过收据ID查询收据信息

设计方案：

1. 存储设计

将公司信息，收据信息和融资信息各自存储在结构体中，以方便管理和访问。公司信息包括：公司ID，公司名，公司地址，公司总资产，公司角色（银行/其他企业）。

```
struct Company{
         uint256 id;//公司 ID
         string name;//姓名
         string address_;//地址
         uint256 money;//总资产
         uint256 role; // 0 for bank,1 for company
         uint256 registed;//是否注册
     }
```

收据信息包括：收据ID，收据欠款方，收据收款方，收据金额，签发收据日期，收据到期日。

```
  struct Receipt{
        uint256 id;//收据 id
        string from;//收据欠款方
        string to;//收据收款方
        uint256 amount;//收据金额
       // bool status; // 是否用于融资
        uint256 begin;//收据开始日期
        uint256 end; //收据到期日
        //bool finished;
   }
```

融资信息包括：融资ID，融资企业名，融资所用收据ID，融资金额。

```
struct Finance{
         uint256 finanID;//融资 ID
         string company;//融资企业名，
         uint256 rcptID;//融资所用收据 ID
         uint256 amount;//融资金额。
     }
```

所有的功能都围绕着这些结构体实现。

2. 数据流图

3. 核心功能介绍

1) 注册账号。

通过提供：公司名，公司地址，公司总资产，公司角色（银行/企业）注册一个属于公司的账号。如果公司名和地址**同时**被另一个账户使用则不可注册（考虑了**分公司**的存在）。智能合约代码如下：

```
[1] function registerCompany(string name_,string address_,uint256 money,uint256 role_) public returns(int256){
[2]         int256 ret_code = 0;
[3]         int256 ret= 0;
[4]         uint256 temp_asset_value = 0;
[5]
[6]         // 查询账户是否存在
[7]         (ret, temp_asset_value) = findCompany(name_,address_);
[8]
[9]         if(ret != 0){
[10]            Company memory newCompany = Company(++companyID,name_,address_,money,role_,uint256(1));
[11]            uint256 temp = companyCount;
[12]            companyCount = CompanyList.push(newCompany);
[13]            if(companyCount > temp){
[14]                //result = 0;
[15]                ret_code = 0;//details = "Company addition success!";
[16]            }else{
[17]                //result = false;
[18]                ret_code = 1;//details = "Company addition failed!";
[19]            }
[20]        }else {
[21]            ret_code = -1;//details = "Company already exists!";
[22]        }
[23]
[24]        emit RegisterEvent(ret_code,name_,address_,money,role_);
[25]        return ret_code;
[26]    }
```

2) 发起交易。

发起交易需要提供交易双方账户，交易金额，以及用于签发收据的开始时间和结束时间。交易双方必须在系统中，交易完成后会签发收据。智能合约代码如下：

```
[1] function sendTxs(string from ,string to,uint256 amount,uint256 begin,uint256 en
    d) public returns(int256){
[2]         int256 ret_code = 0;
[3]         int256 ret= 0;
[4]         uint256 temp_asset_value = 0;
[5]
[6]
[7]         // 查询账户是否存在
[8]         (ret, temp_asset_value) = findCompany(from);
[9]
[10]        if(ret == 0){
[11]            (ret, temp_asset_value) = findCompany(to);
[12]            if(ret == 0){
[13]                Receipt memory newReceipt = Receipt(++receiptID,from,to,amount,
    begin,end);
[14]                uint256 temp = receiptCount;
[15]                receiptCount = ReceiptList.push(newReceipt);
[16]
[17]                if(temp < receiptCount){
[18]                    //result = true;
[19]                    ret_code = 0;//details = "Sendtransaction success!";
[20]                }else {
[21]                    //result = true;
[22]                    ret_code = 1;//details = "Sendtransaction failed!";
[23]                }
[24]            }else {
[25]                ret_code = -
    2;//"The receiver is not a member of the system!";
[26]            }
[27]        } else {
[28]            ret_code = -
    1;//details = "The debtor is not a member of the system!";
[29]        }
[30]
[31]        emit SendTxEvent(ret_code,from,to,amount,begin,end);
[32]        return ret_code;
```

```
[33]}
```

3) 转让收据账款。

转让收据账款需要提供收据欠款方（永远不会变），收据收款方，收据账款转入方三方企业账户名，以及需要转账的金额。三方企业账户必须在系统中，且欠款方与收款方之间的收据总额必须大于待转账金额，否则转账失败。转账时，对于金额小于剩余待转账金额的单个收据，系统会将收据的收款方改为账款转入方（避免产生新的收据）；而对于金额大于剩余待转账金额的单个收据A，系统会产生一个欠款方与账款转入方之间的收据B，金额为剩余待转账金额C，而原收据A金额减少C。智能合约代码如下：

```
1.  //2. 收据账款转移
2.      function transfer(string source,string from,string to,uint256 amount) pu
    blic returns(int256){
3.
4.          int256 ret_code;
5.          int256 ret= 0;
6.          uint256 temp_asset_value = 0;
7.
8.          uint256 allMoney;
9.          // 查询账户是否存在及收据总款
10.
11.
12.         (ret, allMoney) = allReceiptsMoney(source,from);
13.         if(ret == 0){
14.             (ret, temp_asset_value) = findCompany(to);
15.             if(ret == 0){
16.                 if(allMoney >= amount){
17.                 uint256 amount_to_transfer = amount;
18.                 for (uint256 i = uint256(0)  ; i < receiptCount ; i++){
19.                     if(hashCompareInternal(ReceiptList[i].from ,source)
20.                          && hashCompareInternal(ReceiptList[i].to,from)){
21.                         //将收款人改为 to
22.                         if(ReceiptList[i].amount < amount_to_transfer){
23.                             amount_to_transfer -= ReceiptList[i].amount;
24.                             ReceiptList[i].to = to;
25.                             //delete ReceiptList[i];
```

```
                        }else {
                            //新增收款人为 to 的收据，金额为 amount_to_transfer
                            uint256 begin = ReceiptList[i].begin;
                            uint256 end = ReceiptList[i].end;
                            Receipt memory newReceipt = Receipt(++receiptID,source,to,amount_to_transfer,begin,end);
                            //不可少的一步：将新增收据信息加入收据总表
                            receiptCount = ReceiptList.push(newReceipt);

                            ReceiptList[i].amount -= amount_to_transfer;
                            amount_to_transfer = 0;
                        }
                        if(amount_to_transfer == 0){
                            ret_code = 0;
                            break;
                        }
                    }
                    }
                }else{
                    ret_code = 1;//details = "Money in the receipts not enough to transfer!";
                }
            }
            else {
                ret_code = -3;//to_account not found
            }
        }else {
            ret_code = ret;//-1 or -2 (source / from not exists)
        }

        emit TransferEvent(ret_code,source,from,to,amount);
        return ret_code;
}
```

4) 利用收据账款进行融资。

融资需要提供融资企业名与用于融资的收据ID。系统会要求融资企业必须为系统账户；用于融资的收据必须存在且收据收款方必须为该融资企业；收据必须为有效收据（金额不为）。否则不可融资。满足条件则该收据金额

即为融资金额并返回。

```
function findReceiptByID(uint256 id)view public returns(int256,string,string,uint256){
    int256 ret_code;
    if(id > receiptID){
        ret_code = -1;
    }
    else {
        for (uint256 i = uint256(0); i < receiptCount ; i++){
            if(ReceiptList[i].id == id){
                /*newReceipt.id = ReceiptList[i].id;
                newReceipt.from = ReceiptList[i].from;
                newReceipt.to = ReceiptList[i].to;
                newReceipt.amount = ReceiptList[i].amount;
                newReceipt.begin = ReceiptList[i].begin;
                newReceipt.end = ReceiptList[i].end;*/
                ret_code = 0;
                return (ret_code,ReceiptList[i].from,ReceiptList[i].to,ReceiptList[i].amount);
            }
        }
        return (ret_code,"","",uint256(0));
    }
    //return newReceipt;
}
//3. 融资
function financing(string company,uint256 rcptID) public  returns(int256,uint256){
    int256 ret_code;
    int256 ret= 0;
    uint256 temp_asset_value = 0;

    uint256 amount;


    (ret, temp_asset_value) = findCompany(company);
    if(ret == 0){
        string memory from;
        string memory to;
        (ret,from,to,amount) = findReceiptByID(rcptID);
        if(ret == 0){
            if(hashCompareInternal(to,company)){
                if(amount > uint256(0)){
```

```
                    Finance memory finance = Finance(++financeID,company,rcptID,amount);
                    financeCount = FinanceList.push(finance);

                    if(financeCount == financeID){
                        ret_code = 0;//details = "Receipt financing success!";
                    }else {
                        ret_code = 1;//details = "Receipt financing failed!"

                    }
                }else {
                    ret_code = -4;//details = "Disabled receipt!"
                }
            }else {
                ret_code = -3;//details = "The receiver of the receipt is not the company!"
            }
        }else {
            ret_code = -2;//"Receipt not found!!";
        }
    }else {
        ret_code = -1;//details ="company not found!"
    }

    emit FinanceEvent(ret_code,company,rcptID);

    return (ret_code,amount);
}
```

5) 支付双方的收据账款&&手动增加/减少账户资产。

用户需指定待支付收据双方账户名，并且指定支付金额，收据双方必须在系统中。系统会根据支付金额依次支付双方之间的收据直至还够支付金额或者所有有效收据皆被还清。还款时欠款方资产减少，收款方资金增加，收据减少相应金额（余额为0则为失效收据；单独考虑手动减少账户资产时，减少的金额必须小于账户剩余资产金额）。成功还款时返回指定还款金额-总还款金额（即如果指定金额超出了所有的有效收据金额，则返回剩余的金

额）。

```
function incCompanyMoney(string company,uint256 amount) public returns(int256){
        int256 ret_code;
        int256 ret= 0;
        uint256 temp_asset_value = 0;

        // 查询账户是否存在
        (ret, temp_asset_value) = findCompany(company);
        if(ret == 0){
            for (uint256 i = 0 ; i < companyCount ; i++){
                if(hashCompareInternal(CompanyList[i].name,company)){
                    uint256 temp = CompanyList[i].money;

                    CompanyList[i].money += amount;
                    //return (0,CompanyList[i].money);

                    if(temp >= CompanyList[i].money){
                        //result = false;
                        ret_code = 1;//details = "Operation failed!";
                    }
                    else {
                        //result = true;
                        ret_code = 0;//details = "Operation success!";
                        temp_asset_value = CompanyList[i].money;
                    }
                }
                //return false;
            }

        }else {
            ret_code = -1;//details = "Company not found!"
        }

        emit IncMoneyEvent(ret_code,company,temp_asset_value);
        return ret_code;
    }
    function decCompanyMoney(string company,uint256 amount) public returns(int256){
        int256 ret_code;
        int256 ret= 0;
        uint256 temp_asset_value = 0;

```

```
        // 查询账户是否存在
        (ret, temp_asset_value) = findCompany(company);

        if(ret == 0){
            if(temp_asset_value >= amount){
                for (uint256 i = 0 ; i < companyCount ; i++){
                    if(hashCompareInternal(CompanyList[i].name,company)){

                        uint256 temp = CompanyList[i].money;

                        CompanyList[i].money -= amount;
                        //return (0,CompanyList[i].money);

                        if(temp <= CompanyList[i].money){
                            //result = false;
                            ret_code = 1;//details = "Operation failed!";
                        }
                        else {
                            //result = true;
                            ret_code = 0;//details = "Operation success!";
                            temp_asset_value = CompanyList[i].money;
                        }
                    }
                    //return false;
                }

            }else {
                ret_code = -2;//details = "Money not enough!"
            }
        }else {
            ret_code = -1;//details = "Company not found!"
        }

        emit DecMoneyEvent(ret_code,company,temp_asset_value);//final balanc
    e
        return ret_code;
}

//4.支付收据款项
   function pay(string from,string to,uint256 amount)public returns(int256,u
    int256){

       int256 ret_code;
       int256 ret= 0;
```

```
83.         uint256 temp_asset_value = 0;
84.
85.         int256 inc;
86.         int256 dec;
87.
88.         uint256 leftAmount = amount;
89.
90.         (ret, temp_asset_value) = findCompany(from);
91.         if(ret == 0){
92.             (ret, temp_asset_value) = findCompany(to);
93.             if(ret == 0){
94.                 for (uint256 i = 0; i < receiptCount ; i++){
95.                     if(hashCompareInternal(ReceiptList[i].from,from)
96.                         && hashCompareInternal(ReceiptList[i].to,to)
97.                         && ReceiptList[i].amount > uint256(0)){
98.
99.                         if(ReceiptList[i].amount < leftAmount){
100.
101.                             leftAmount -= ReceiptList[i].amount;
102.                             dec = decCompanyMoney(from,ReceiptList[i].amount
    );
103.                             inc = incCompanyMoney(to,ReceiptList[i].amount);

104.                             if(dec != 0 || inc != 0){
105.
106.                             }
107.                             ReceiptList[i].amount = 0;
108.                         }
109.                         else {
110.
111.                             ReceiptList[i].amount -= leftAmount;
112.                             decCompanyMoney(from,leftAmount);
113.                             incCompanyMoney(to,leftAmount);
114.                             leftAmount = 0;
115.                             ret_code = 0;//details = "All receipt paid!No Mo
    ney Left"
116.                             break;
117.                         }
118.                     }
119.                 }
120.                 if(leftAmount != 0 ){
121.                     ret_code = 1;//details = "All receipt paid!Money Left"
122.                 }
123.
```

```
            }else {
                ret_code = -2;//details = "to_account not found!"
            }
        }else {
            ret_code = -1;//details = "from_account not found!"
        }

        emit PayEvent(ret_code,from,to,amount - leftAmount);
        return (ret_code,leftAmount);
}
```

6) 查询公司资产余额&&查询公司注册ID。

查询公司资产和注册ID需要提供企业账户名（查询公司资产也可同时提供企业地址），仍然要求账户必须在系统中。

```
function findCompanyID(string name)public constant returns(int256,uint256){

        /*if(companyCount == uint256(0)){
            return false;
        }*/
        for (uint256 i = 0 ; i < companyCount ; i++){
            if(hashCompareInternal(CompanyList[i].name,name)){
                return (0,CompanyList[i].id);
            }

        }
         return (-1,uint256(0));
    }
    function findCompany(string name)public constant returns(int256,uint256)
{
        /*if(companyCount == uint256(0)){
            return false;
        }*/
        for (uint256 i = 0 ; i < companyCount ; i++){
            if(hashCompareInternal(CompanyList[i].name,name)){

                return (0,CompanyList[i].money);
            }
            //return false;
        }
        return (-1,uint256(0));
    }
```

```
    function findCompany(string name,string address_)public constant returns (int256,uint256){
        /*if(companyCount == uint256(0)){
            return false;
        }*/
        for (uint256 i = 0 ; i < companyCount ; i++){
            if(hashCompareInternal(CompanyList[i].name,name)
                 && hashCompareInternal(CompanyList[i].address_ ,address_)){

                return (0,CompanyList[i].money);
            }
            //return false;
        }
        return (-1,uint256(0));
    }
```

7) 查询某方与另一方之间所有单向收据的账款总额。

提供双方账户名即可查询双方之间所有单向收据的账款总额。

```
function allReceiptsMoney(string from,string to)public constant returns(int256,uint256){
        int256 ret_code;
        int256 ret= 0;
        uint256 temp_asset_value = 0;

        uint256 amount = uint256(0);
        // 查询账户是否存在
        (ret, temp_asset_value) = findCompany(from);

        if(ret == 0){
            (ret, temp_asset_value) = findCompany(to);
            if(ret == 0){
                //uint256 amount = uint256(0);
                for (uint256 i = uint256(0) ; i < receiptCount ; i++){
                    if(hashCompareInternal(ReceiptList[i].from ,from)
                     && hashCompareInternal(ReceiptList[i].to ,to)){
                        amount += ReceiptList[i].amount;
                    }
                }
                ret_code = 0;
            }else {
                ret_code = -2;//"The receiver is not a member of the system!";
```

```
            }
        } else {
            ret_code = -
    1;//details = "The debtor is not a member of the system!";
        }
        return (ret_code,amount);

    }
```

8) 通过收据ID查询收据信息

提供收据ID查询收据欠款方，收款方，收据金额等信息。

```
function findReceiptByID(uint256 id)view public returns(int256,string,string
,uint256){
        int256 ret_code;
        if(id > receiptID){
            ret_code = -1;
        }
        else {
            for (uint256 i = uint256(0); i < receiptCount ; i++){
                if(ReceiptList[i].id == id){
                    /*newReceipt.id = ReceiptList[i].id;
                    newReceipt.from = ReceiptList[i].from;
                    newReceipt.to = ReceiptList[i].to;
                    newReceipt.amount = ReceiptList[i].amount;
                    newReceipt.begin = ReceiptList[i].begin;
                    newReceipt.end = ReceiptList[i].end;*/
                    ret_code = 0;
                    return (ret_code,ReceiptList[i].from,ReceiptList[i].to,R
    eceiptList[i].amount);
                }
            }
            return (ret_code,"","",uint256(0));
        }
        //return newReceipt;
    }
```

三、 功能测试

此处展示的是智能合约在 webase-front 的测试。**所有函数返回**

的第一个数为 0 则表示功能正常执行。首先部署合约获取合约地址（合约命名为 Test.sol）

```
[group:1]> deploy Test
contract address: 0xe573452985074cf06f4d794bf08c2110ead8e03d
```

- 查询公司资产余额。

  合约在部署时会初始化生成一个账户“bank”，可直接使用

```
constructor() public{
    receiptID = uint256(0);
    companyID = uint256(0);
    financeID = uint256(0);
    receiptCount = companyCount = financeCount = uint256(0);
    registerCompany("bank","bankAddress",12345678,0);
}
```

findCompany查询其余额，第一个返回的数表示函数是否顺利执行，第二个为公司资产：

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d findCompany "bank"
[0, 12345678]
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d findCompany "bank" "bankAddress
"
[0, 12345678]
```

第一个数为0，正常执行

地址可有可无

资产

假如查询一个不在系统中的账户则会返回-1：

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d findCompany "fsq"
[-1, 0]
```

- 注册账号。

  分别提供账户名，公司地址，公司资产，公司角色代码（0为银行，1为其他企业）：

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d registerCompany "fsq" "sysusdcs
" 12345678 1
transaction hash: 0x4e5807c23f6ccfdcb1d4a14b471f4eea084b6efcac2827b5ece94ede61fec54c
----------------------------------------------------------------------------------------
Output
function: registerCompany(string,string,uint256,uint256)
return type: (int256)
return value: (0)
----------------------------------------------------------------------------------------
Event logs
event signature: RegisterEvent(int256,string,string,uint256,uint256) index: 0
event value: (0, fsq, sysusdcs, 12345678, 1)
----------------------------------------------------------------------------------------
```

注册成功

此时可查询刚刚注册的账户余额

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d findCompany "fsq"
[0, 12345678]
```

- 发起交易，即采购商品—签发应收账款收据，交易上链。

  发起一场fsq（买家）到bank（卖家）的交易（可视为“无息贷款”），交易金额5678，签订的收据开始时间为2019/12/13，收据到期时间为2020/01/01。

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d sendTxs "fsq" "bank" 5678 20191
213 20200101
transaction hash: 0x6f545ce1b3ce3c19d15af9a5ff5c4d50f9a9d3dd5c9d3e3517b99746d668c84e
----------------------------------------------------------------------------------------
Output
function: sendTxs(string,string,uint256,uint256,uint256)
return type: (int256)
return value: (0)
----------------------------------------------------------------------------------------
Event logs
event signature: SendTxEvent(int256,string,string,uint256,uint256,uint256) index: 0
event value: (0, fsq, bank, 5678, 20191213, 20200101)
----------------------------------------------------------------------------------------
```

交易成功

- 通过收据ID查询收据信息

  此为第一张收据，所以id为1，可直接查询。

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d findReceiptByID 1
[0, fsq, bank, 5678]
```

- 查询某方与另一方之间所有收据的账款总额。

再次发起一场fsq（买家）到bank（卖家）的交易，金额为1000.

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d sendTxs "fsq" "bank" 1000 20191
213 20200101
transaction hash: 0xc4926e59090931301018f8cb3d4c75bdad9440e3fbfbbe8fd527f9e48b92d51d
-------------------------------------------------------------------------------------------
Output
function: sendTxs(string,string,uint256,uint256,uint256)
return type: (int256)
return value: (0)
-------------------------------------------------------------------------------------------
Event logs
event signature: SendTxEvent(int256,string,string,uint256,uint256,uint256) index: 0
event value: (0, fsq, bank, 1000, 20191213, 20200101)
-------------------------------------------------------------------------------------------
```

查询双方之间的收据总额（5678+1000=6678）

```
[group:1]> call Test 0x05137d4251a7758e7270d179b1288f6f91a25cec allReceiptsMoney "fsq" "bank"
[0, 6678]
```

- 查询公司注册ID。

  第一个注册的公司为“bank”,ID为1,所以“fsq”企业ID为2：

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d findCompanyID "bank"
[0, 1]

[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d findCompanyID "fsq"
[0, 2]
```

- 转让收据账款。

  另外注册一个账户fsq2，资产为12344678

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d registerCompany "fsq2" "sysusdc
s" 12344678 1
transaction hash: 0x1205f79f409ba936ad537aa98586d5ebdee1150cb1a44657ec585f1475b942e9
-------------------------------------------------------------------------------------------
Output
function: registerCompany(string,string,uint256,uint256)
return type: (int256)
return value: (0)
-------------------------------------------------------------------------------------------
Event logs
event signature: RegisterEvent(int256,string,string,uint256,uint256) index: 0
event value: (0, fsq2, sysusdcs, 12344678, 1)
-------------------------------------------------------------------------------------------
```

转让bank所拥有的fsq欠下的1000收据账款转给fsq2

```
[group:1]> call Test 0xe573452985074cf06f4d794bf08c2110ead8e03d transfer "fsq" "bank" "fsq2" 10
00
transaction hash: 0xa8c72aec90edfa01e2de2ed482b8dc51d4b03d5f9564bddd89f5e577c51ab540
--------------------------------------------------------------------------------------------
Output
function: transfer(string,string,string,uint256)
return type: (int256)
return value: (0)
--------------------------------------------------------------------------------------------
Event logs
event signature: TransferEvent(int256,string,string,string,uint256) index: 0
event value: (0, fsq, bank, fsq2, 1000)
--------------------------------------------------------------------------------------------
```

转让成功

（下一步可证明转让成功）

- 支付双方之间的收据账款。

（注：因为这个功能会调用手动增加账户金额和手动减少账户金额的函数，所以此功能正常则表明另外两个函数的功能也正常，所以不再展示另外两个函数的功能测试。）

根据以上的操作，fsq 对 bank 的收据欠款应为 5678（6678 – 1000 = 5678），尝试 fsq 对 bank 还款 6678

```
[group:1]> call Test 0x05137d4251a7758e7270d179b1288f6f91a25cec allReceiptsMoney "fsq" "bank"
[0, 5678]

[group:1]> call Test 0x05137d4251a7758e7270d179b1288f6f91a25cec pay "fsq" "bank" 6678
transaction hash: 0xe98b49b62423b4b399f335ffd66f40b84e8400335619aef47c5a641b573520d9
--------------------------------------------------------------------------------------------
Output
function: pay(string,string,uint256)
return type: (int256, uint256)
return value: (1, 1000)
--------------------------------------------------------------------------------------------
Event logs
event signature: DecMoneyEvent(int256,string,uint256) index: 0
event value: (0, fsq, 12341000)
event signature: DecMoneyEvent(int256,string,uint256) index: 1
event value: (0, fsq, 12340000)
event signature: IncMoneyEvent(int256,string,uint256) index: 0
event value: (0, bank, 12350356)
event signature: IncMoneyEvent(int256,string,uint256) index: 1
event value: (0, bank, 12351356)
event signature: PayEvent(int256,string,string,uint256) index: 0
event value: (1, fsq, bank, 5678)
--------------------------------------------------------------------------------------------
```

还款成功但多出1000，实际还款6678-1000=5678

fsq资产先减少4678，再减少1000

bank资产先增加4678，再增加1000

还款成功，实际还款5678

可知还款成功，实际还款 5678。观察可知系统先减少 fsq 的 4678（12345678-12341000 = 4678）资产，再减少 1000 资产；bank 先增加

4678（12345678+4678 = 12350356）,再增加 1000 资产。这是因为上面 bank 将 fsq 的 1000 收据欠款转让给了 fsq2，而系统时根据顺序转让欠款。Fsq 对 bank 的第一张欠收据欠款为 5678，大于 1000，所以第一张收据金额变为 5678-1000=4678，新增一张 fsq 对 fsq2 的收据，金额为 1000。所以此时 fsq 对 bank 的收据有两张，一张为 4678，一张为 1000。所以逐次还款时第一次还 4678，第二次还 1000。这也证明了上一步的转让操作成功。

此时 fsq 对 bank 的欠款清 0。

```
[group:1]> call Test 0x05137d4251a7758e7270d179b1288f6f91a25cec allReceiptsMoney "fsq" "bank"
[0, 0]
```

- 利用收据账款进行融资。

  此时 fsq2 还有 fsq 的 1000 欠款。

```
[group:1]> call Test 0x05137d4251a7758e7270d179b1288f6f91a25cec allReceiptsMoney "fsq" "fsq2"
[0, 1000]
```

这应该是第 3 张收据，所以 id 为 3

```
[group:1]> call Test 0x05137d4251a7758e7270d179b1288f6f91a25cec findReceiptByID 3
[0, fsq, fsq2, 1000]
```

利用此收据进行融资

```
[group:1]> call Test 0x05137d4251a7758e7270d179b1288f6f91a25cec financing "fsq2" 3
transaction hash: 0x42e1fa6ef3bd99d5ddcd5235e6d1f6033f7864b82e8fd11107bfac4e67de46be
----------------------------------------------------------------------------------------
Output
function: financing(string,uint256)
return type: (int256, uint256)
return value: (0, 1000)        融资成功，金额1000
----------------------------------------------------------------------------------------
Event logs
event signature: FinanceEvent(int256,string,uint256) index: 0
event value: (0, fsq2, 3)
----------------------------------------------------------------------------------------
```

四、 界面展示

最终制品为由智能合约制成的一个 java 应用——test-app，编译运行方式

为：

1. 进入 test-app 目录，运行./gradlew build（提交的文件已经编译运行过了，重新编译时最好将 build 和 dist 文件夹删除）。

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/console$ cd ~/test-app
fisco-bcos@fiscobcos-VirtualBox:~/test-app$ ./gradlew build
Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 19s
4 actionable tasks: 4 up-to-date
fisco-bcos@fiscobcos-VirtualBox:~/test-app$
```

编译成功之后，将在项目根目录下生成 `dist` 目录。dist 目录下有一个 `test_run.sh` 脚本，简化项目运行。

2. 进入 dist 目录，部署 Test.sol 合约：bash test_run.sh deploy

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh deploy
 deploy Test success, contract address is 0x61067d46ad2b54b60924da325d6d899923be5b15
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$
```

3. 用法

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh
    bash test_run.sh deploy
    bash test_run.sh register [account] [address] [amount] [role]
    bash test_run.sh incMoney [account] [amount]
    bash test_run.sh decMoney [account] [amount]
    bash test_run.sh ifAccountExists [account] ([address])
    bash test_run.sh accountID [account]
    bash test_run.sh sendTx [fromAccount] [toAccount] [amount] [beginTime] [endTime]
    bash test_run.sh findReceipt [receiptID]
    bash test_run.sh transfer [sourceAccount] [fromAccount] [toAccount] [amount]

    bash test_run.sh allReceiptsMoney [fromAccount] [toAccount]
    bash test_run.sh finance [account] [receiptID]
    bash test_run.sh pay [FromAccount] [toAccount] [amount]

examples :
    bash test_run.sh deploy
```

```
bash test_run.sh register  Account0  account0Address 10000000 1
bash test_run.sh incMoney Account0 10000
bash test_run.sh decMoney Account0 10000
bash test_run.sh ifAccountExists  Account1
bash test_run.sh ifAccountExists  Account0 account0Address
bash test_run.sh accountID Account0
bash test_run.sh sendTx Account0 Account1 100000 20191213 20200101
bash test_run.sh findReceipt 0000001
bash test_run.sh transfer  Account0 Account1 Account2  10000
bash test_run.sh allReceiptsMoney Account0 Account1
bash test_run.sh finance Account1 0000001
bash test_run.sh pay Account0 Account1 100000
```

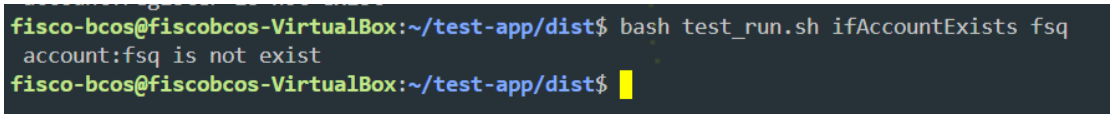
4. 示例。

上面已经展示了智能合约的测试，所以这里就只展示几个功能。

- 查询账户是否存在及其余额

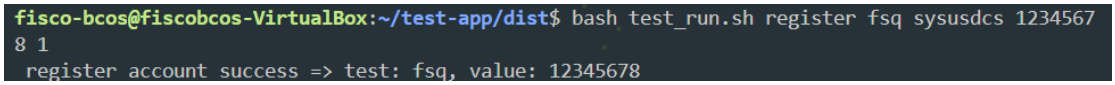
bash test_run.sh　ifAccountExists　企业账户名　(企业地址)

如果不存在：



- 注册账户

bash test_run.sh register　账户名　地址 资产 企业性质代码（0：银行，1：其他企业）



- 发起交易

bash test_run.sh sendTx 买家账户 卖家账户 金额 交易时间 收据到期时间

与上面实例一样，发起两笔从 fsq 到 bank 的交易，金额分别为 5678 和 1000

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh sendTx fsq bank 5678 20191213 20200101
 send transaction success => from_account: fsq, to_account: bank, amount: 5678
Receipt beginTime: 20191213  endTime: 20200101
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh sendTx fsq bank 1000 20191213 20200101
 send transaction success => from_account: fsq, to_account: bank, amount: 1000
Receipt beginTime: 20191213  endTime: 20200101
```

- 转让收据金额

bash test_run.sh transfer 欠款方 收款方 转款接受方 转款金额

先注册 fsq2（资金为 12344678），再对其转账收据金额 1000，查询 fsq 对 bank 的欠款（应为 5678）

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh register fsq2 sysusdcs 12344678 1
 register account success => account: fsq2, value: 12344678
```

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh transfer fsq bank fsq2 1000
 transfer success => from_test: bank, to_test: fsq2, amount: 1000
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh allReceiptsMoney fsq bank
 from_account: fsq, to_account: bank, total value of the receipts: 5678
```

- 还款

bash test_run.sh pay 欠款方 收款方 还款金额

尝试 fsq 对 bank 还款 6678（实际只会还 5678）

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh pay fsq bank 6678
 pay receipts success => All receipts paid! Total Amount: 5678
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$
```

此时 fsq 资产减少 5678（12345678-5678=12340000）

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh ifAccountExists fsq
 account: fsq, value: 12340000
```

尝试 fsq 对 fsq2 还款 2000（实际只会还 1000），且 fs2 资产变为 12345678（12344678+1000=12345678）

```
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh pay fsq fsq2 2000
 pay receipts success => All receipts paid! Total Amount: 1000
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$ bash test_run.sh ifAccountExists fsq2
 account: fsq2, value: 12345678
fisco-bcos@fiscobcos-VirtualBox:~/test-app/dist$
```

五、 心得体会

说实话，个人感觉课堂上讲的内容很多都是相互重叠而且是仅对于底层理论的讲解，对其技术的实现没有太详细的说明，而且对大作业几乎没有什么帮

助。

这是第一次做全栈项目，没有什么经验，做的时候也很茫然，甚至不知道做出来的东西应该是什么样。最后做出来的东西感觉也有所欠缺。这说明自己的确是缺乏锻炼，还需要进一步的学习。不过在做大作业的过程中，还是激起了自己对智能合约的兴趣，我会继续学习，弥补一下自己的区块链全栈开发能力，这次做出来的项目感觉的确是很烂，我会尝试继续完善的。