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	2017 级	专业（方向）	软件工程
学号	17343150	姓名	张寰宇
电话	132 4286 4431	Email	2854703630@qq.com
开始日期	2019/11/16	完成日期	2019/12/13

一、 项目背景

基于已有的开源区块链系统 FISCO-BCOS （https://github.com/FISCO-BCOS/FISCO-BCOS），以联盟链为主，开发基于区块链 或区块链智能合约的供应链金融平台，实现供应链应收账款资产的溯源、流转。

二、 方案设计

1. 在供应链金融系统中，有信誉机构和公司两种参与者。信誉机构选择银行，由合约的发起者担任。公司由银行进行注册，注册之后可以进行相应的操作。

2. 信誉机构在部署合约时创建。

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

3. 公司包括四个内容，名字，余额，信誉等级，账单。账单包括已完成的账单，未支付的账单，别的公司未向自己支付的账单。

```
struct Company{
    string name;
    uint balance;
    uint rank;
    uint i;
    mapping (uint => Receipt) finish;
    mapping (address => lendReceipt) lend;
    mapping (address => oweReceipt) owe;
}
```

4. 账单包括账单发起的双方，金额，状态（已支付，未支付等），标识（用于识别一个账单）。

```
struct Receipt{
    uint tag;
    address from;
    address to;
    uint amount;
    string status;
}
```

5. 银行可以注册公司，并评测信用等级，信用等级可以在融资时判断是否可信。

```
function addCompany(address c, string name,uint balance,uint r) public{
    require(msg.sender == bank);
    Company memory com;
    com.name = name;
    com.balance = balance;
    com.rank = r;
    com.i = 0;
    company[c] = com;
}
```

6. 两个公司之间直接支付，无需贷款。

```
function sent(address receiver, uint amount)public{
    //require(company[msg.sender].balance == amount , "Your balance is not abundent");
    require(company[msg.sender].balance >= amount);
    company[msg.sender].balance -= amount;
    company[receiver].balance += amount;

    Receipt memory r;
    r.from = msg.sender;
    r.to = receiver;
    r.amount = amount;
    addToFinishReceipt(msg.sender,receiver,r);
    addToFinishReceipt(receiver,msg.sender,r);
}
```

7. 需要向银行融资，此时要判断信用等级是否足够。

```
function financingBank(address receiver, uint amount) public{
    require(company[msg.sender].rank * 1000 >= amount);

    Receipt memory r;
    r.from = msg.sender;
    r.to = receiver;
    r.amount = amount;
    addToOweReceipt(msg.sender,receiver,r);
    addToLendReceipt(receiver,msg.sender,r);
}
```

8. 使用账单进行融资，此时需要判断账单的金额是否大于融资的金额。再进行融资时，使用的是账单拆分的方法，将一个账单拆分成两个账单，账单的总金额不变。

9. 支付未支付的账单

```
function pay(address from, address to)public{
    require(company[from].owe[to].i > 0);
    Receipt r = company[from].owe[to].owe[0];
    company[from].balance -= r.amount;
    company[to].balance += r.amount;
    removeFromOweReceipt(from,to,r);
    removeFromLendReceipt(to,from,r);
}
```

三、 功能测试

```
[group:1]> deploy Final
contract address: 0x16f14e8796ccfeec55b3bd974332929f575604da
```

可以编译部署成功，但是在后端测试时出错。

四、 界面展示

前端未实现

五、 心得体会

在做本次的大作业过程中，一直想将链端做的更好些，但是在实际的写的过程中，又遇到了很多的问题。开始想要调用数据库进行操作，但是在使用数据库后发现不会报错，但是在部署后实际的运行中是不会向数据库中写入任何的内容的内容的，就改用没用数据库的。在开始的时候使用可变长的数组存储账单，但是感觉这样的话，每次查找账单都得遍历一次，很费时间；使用 Mapping 来记录，可能会出现一个 key 值却对应几个目标的情况，最终使用 Mapping 和数组的结合使用。

前端和后端的开发，在写完链端之后发现时间不够，而且还不知道如何下手做，感觉很急躁效率就很低。后端写了很久但是仍然有问题，前端的开发看了很久，但是还是没弄明白怎么将底层的链与上层的界面联系在一起。