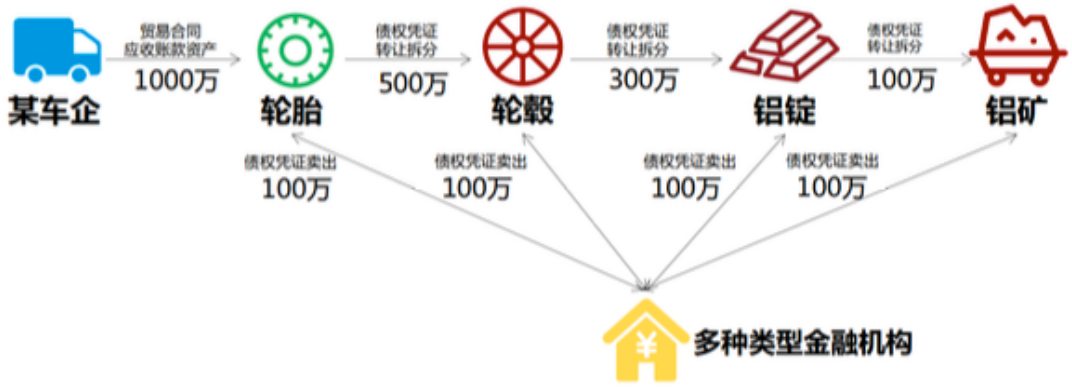
中山大学数据科学与计算机学院本科生实验报告

（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	17343147	姓名	张涵健
电话	13242883497	Email	969437072@qq.com
开始日期	2019.12.8	完成日期	2019.12.11

一、 项目背景



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。

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

大作业二中的需求分析：

针对四大功能进行面向对象抽象：

1. 确定的对象：应收款账单

2. 对象具有的属性：债主、欠债人、企业名称、开始日期、还款日期、应收款金额

3. 具有的行为（从功能抽象）: 签名应收款账单、应收款转让、评估企业风承、支付余款

需求分析深化：

向金融行业从业人士咨询相关需求的概念时发现，该供应链上的所有应收款都是建立在核心企业（车企）的信用上的，因此真实有效的资本只有核心企业首次签发的应收账款（1000万元），而后所有的应收款都是建立在此之上的变种。分析后即可简化上述需求分析。

变化：将“金额”的单位从基本货币（RMB），转为核心企业的应收款债权，即把核心企业的应收账款视作通货，合约的整体就只有账户、对象、金额三个主要属性

通过区块链的特性，产生 Event 则视作一笔交易，与例程 Asset 不同的是，把调用方法的主体属性输入去掉，改用 msg.sender 解决账户的权限问题

存储设计：

使用 FISCO 内置的 Table.sol 作为数据存储，(account， balance)

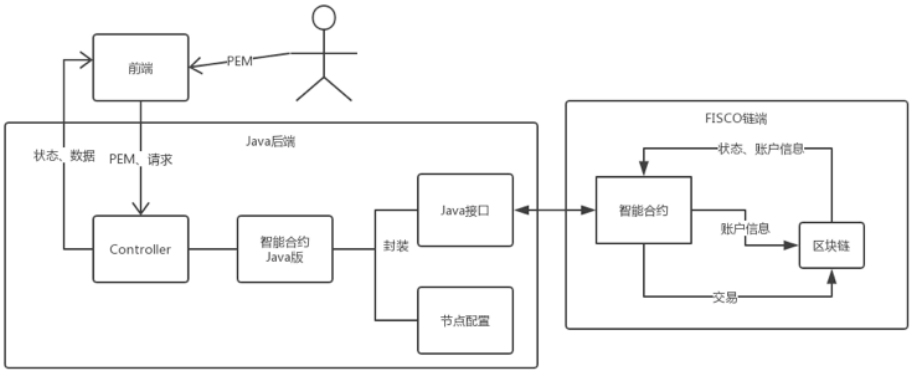
```
event RegisterEvent(int256 ret, string account, uint256 asset_value);
event TransferEvent(int256 ret, string from_account, string to_account, uint256 amount);
event PayEvent(int256 ret);

address constant master = 0xbAF9163Da71F944b2B26a8CBe6484563cbE3EB47;
string constant lastdate = "2019-12-31";

string[] accountList;
mapping(string=>address) name2addr;
mapping(address=>string) addr2name;
```

记录核心企业（master）的用户地址，
该核心企业的应收账款到期日期（lastdate），
账号名称的列表（accountList）
账户名称到账户地址的映射（name2addr）
账户地址到账户名称的映射（addr2name）

数据流图：



核心功能介绍：

（1）用户私钥登录、注册。

用户登录时，通过上传 PEM 私钥文件进行登录验证，后端接收并处理 login 请求，通过私钥解密出的地址作为账户地址，调用智能合约 login 方法，并返回状态码。

```
    返回值:
            3:已注册的、核心企业master身份
            2:未注册的、核心企业master身份
            1:已注册的、普通企业身份
            0:未注册的、普通企业身份
    */

    function login() public view returns(int){
        if (name2addr[addr2name[msg.sender]] != address(0)){
            if (msg.sender == master){
                return 3;
            }
            return 1;
        }
        if (msg.sender == master){
            return 2;
        }
        return 0;
    }
```

用户注册时，根据登录返回的参数确定身份，注册用户名，若为 master 则可注册签发的应收款总额，普通企业为 0。

```
/*
描述 : 资产注册
参数 :
        account : 资产账户
        amount : : 资产金额
返回值:
        0 资产注册成功
        1 管理员资产注册成功
        -1 资产账户已存在
        -2 其他错误
*/
function register(string memory account, uint256 asset_value) public returns(int256){
    int256 ret_code = 0;
    int256 ret = 0;
    uint256 temp_asset_value = 0;
    // 查询账户是否存在
    (ret, temp_asset_value) = select(account);
    if(ret != 0) {
        name2addr[account] = msg.sender;
        addr2name[msg.sender] = account;
        accountList.push(account);
        Table table = openTable();

        Entry entry = table.newEntry();
        entry.set("account", account);
        if(msg.sender == master){
            entry.set("asset_value", int256(asset_value));
        }else{
            entry.set("asset_value", int256(0));
        }
        // 插入
        int count = table.insert(account, entry);
        if (count == 1) {
            // 成功
            ret_code = 0;
            if (msg.sender == master){
                ret_code = 1;
            }
        } else {
            // 失败? 无权限或者其他错误
            ret_code = -2;
        }
    } else {
        // 账户已存在
        ret_code = -1;
    }

    emit RegisterEvent(ret_code, account, asset_value);

    return ret_code;
}
```

（2）用户资产查询。（无权限限制）

```
/*
描述 : 根据资产账户查询资产金额
参数 :
        account : 资产账户

返回值:
        参数一: 成功返回0, 账户不存在返回-1
        参数二: 第一个参数为0时有效, 资产金额
*/


function select(string memory account) public view returns(int256, uint256) {
    // 打开表
    Table table = openTable();
    // 查询
    Entries entries = table.select(account, table.newCondition());
    uint256 asset_value = 0;
    if (0 == uint256(entries.size())) {
        return (-1, asset_value);
    } else {
        Entry entry = entries.get(0);
        return (0, uint256(entry.getInt("asset_value")));
    }
}
```

（3）用户应收款签发、转让。使用了权限控制机制，即操作者不能操作别人的账户。

```
/*
描述 : 资产转移
参数 :
        from_account : 转移资产账户
        to_account : 接收资产账户
        amount : 转移金额
返回值:
        0 资产转移成功
        -1 转移资产账户不存在
        -2 接收资产账户不存在
        -3 金额不足
        -4 金额溢出
        -5 其他错误
*/
function transfer(string memory to_account, uint256 amount) public returns(int256)
```

（4）应收账款支付功能。因没有给出具体的支付实现，因此这里设计的是只有 master 才能进行支付，即把该链上的所有应收款（金额）清零。

```
function pay() public returns(int256){
    Table table = openTable();
    for(uint i = 0; i < accountList.length; i++){
        Entry entry = table.newEntry();
        entry.set("account", accountList[i]);
        entry.set("asset_value", int256(0));
        table.update(accountList[i], entry, table.newCondition());
    }

    emit PayEvent(0);
    return 0;
}
```

三、 功能测试

（1）测试流程（演示视频流程）

0、查看初始区块链状态

1、部署合约

2、配置合约地址

3、get_accouts.sh 准备 3 个用户

4、启动 JavaApp

5、浏览器打开 localhost:8081

6、使用核心企业 pem 登录

7、注册应收账款 1000

8、用两个下游企业用户登录注册

9、用核心企业向（下游企业 1）签发应收账款 500、并查询

10、用（下游企业 1）向（下游企业 2）转让 296、并查询（1、2）

11、核心企业支付后，查询（1、2）

12、查看链端变化

（2）使用 UI 界面进行测试（演示视频截图）

1、 权限控制（使用核心企业的私钥第一次登录）

使用普通企业的私钥第一次登录

2、核心企业向下游企业 1 签发应收账款（并查询检验）

3、下游企业 1 向下游企业 2 签发并转让应收账款（并查询检验）

4、核心企业支付所有应收账款（查询下游企业 1、2 的应收款额度）

5、初始区块与测试后区块状态比较（演示视频的测试流程）


FISCO BCOS
test
配置
区块链信息
帮助
当前块高
52
交易总量
52
正在处理的交易数
0
PBFT 当前视图
7
最近15天的交易量
节点Id
当前块高
pbftView
节点状态
409c1554c2fd7c463da971bd...
52
4
正常
83227c42698504276cfdc553...
52
5
正常
b22e920bd0a70a96c32043c...
52
7
正常
9b2f9bc09661dd014c6bc258...
52
6
正常

四、 界面展示


供应链金融平台
欢迎使用供应链金融平台
使用PEM私钥文件登录
应收账款（债权）转让
账户名称
额度
SUBMIT
（债权）查询
SUBMIT


供应链金融平台
注册
账户名称
初始账单额度
取消
确认
应收账款（债权）签发
账户名称
额度
SUBMIT
应收账款（债权）查询
名称搜索
SUBMIT


供应链金融平台
应收账款（债权）签发
账户名称
额度
SUBMIT
支付
SUBMIT
应收账款（债权）查询
名称搜索
SUBMIT


供应链金融平台
开发信息
版本：1.0.0
作者：张涵健
日期：2019-12-11
应收账款（债权）查询

五、 心得体会

本次大作业了解到了 FISCO BOCOS 的区块链平台，作为一款新出的区块链平台，有优点、也有不足。作为一个非区块链底层开发者，我认为该区块链平台还是有较大的提升空间的。首先是 API 问题，在讨论群里也可以看到，除了 Java API 外，其他的如 python、node.js 的接口甚至连使用文档都不全，对于一些 0 Java 基础的同学就不太友好。其次是关于 solidity 智能合约在 FISCO 中的堆栈深度问题，致使合约并不能写得太长，而且处处都得考虑这个问题，使得在本次课程设计中，合约的设计经过多次大改，最终还是不得不采用回例程 Asset 的写法。

对于区块链本身的认识，通过这次任务驱动的学习，从 FISCO 的官网中较为全面地了解到了其系统设计。节点的概念可以视为一个通信模块和合约处理计算模块，区块链是一个类似于数据库的存储模块，在互相连接的节点中，只要有节点广播出合约得到的区块，那么所有节点都会在所属的链上添加区块。在本次实验中，4 个节点都为本地节点，都连接在同一条链上，所以 4 个节点的区块信息是同态的。而部署到 AMOP 网络上时，通过节点间的通信，可以同步所有连接节点的区块链。

在和同学讨论时，我们一致得出，在本次实验中，我们可以把 FISCO 区块链当作一个特殊的数据库来对待，因此我也相仿数据库型系统的开发，写好一个 Java 类封装好与 FISCO 链端的交互，剩下的前后端部分即可与正常的中心化系统开发一样了。考虑到区块链的特性，于是使用了私钥登录的方法，是的 transaction 和系统的“交易”行为一致，符合本身的设计需求。

总的来说，通过本次实验更加清楚地了解到了区块链的概念及其应用场景，但仍然需要深入学习。

GitHub 地址：
https://github.com/CoderAT13/FISCO-FINAL