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	17343166	姓名	左杰文
电话	13242883420	Email	928468089@qq.com
开始日期	12 月 4 日	完成日期	12 月 11 日

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。

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信

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。

实现功能：

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

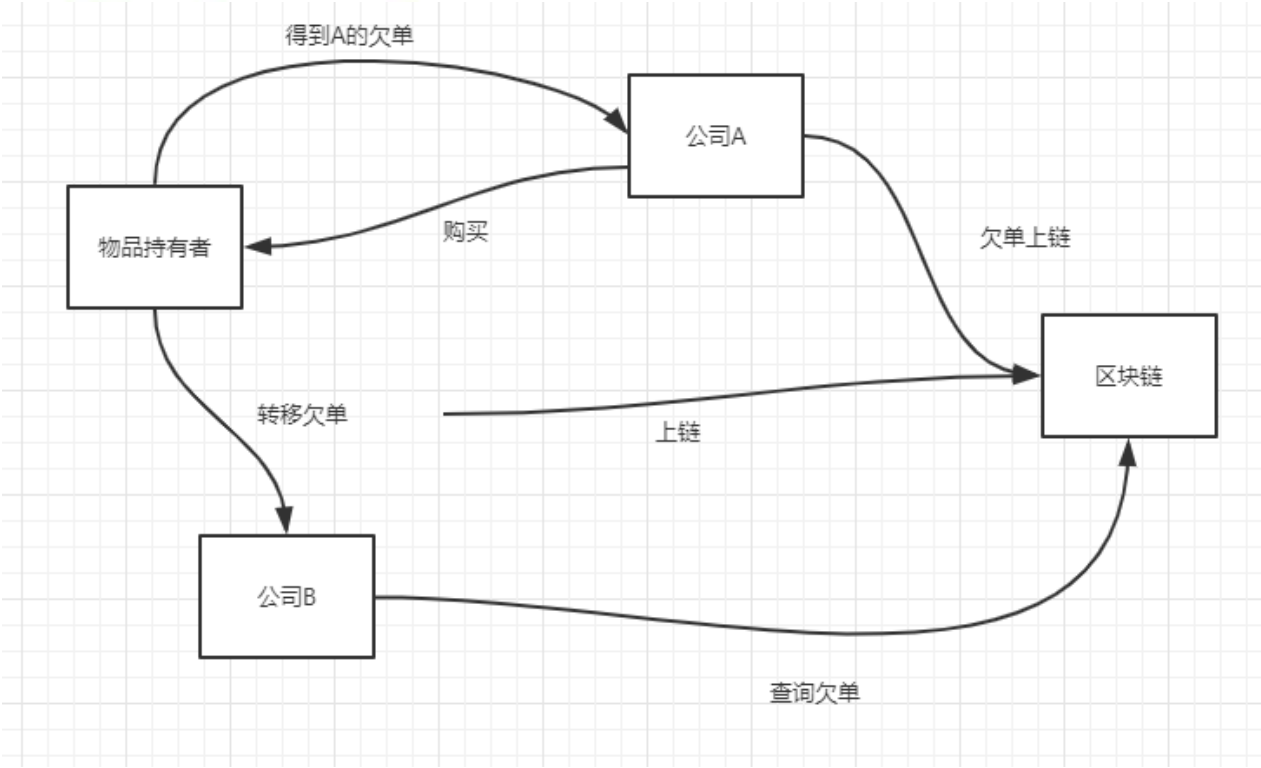
合约内部使用一个 bills 账本来记录每个人拥有的账单，每个账单为结构体 bill 储存有账单价值，欠款方姓名（唯一标识），账单是否有效以及对于欠款方记录其欠款金额。

```
struct bill{
    int money;
    int qian_m;
    string from_user;
    bool valid;
}
```

```
mapping(string=>bill) bills;
```
（由姓名查找账单）

除此之外还有一个 zhangdan 来记录某个人欠的金额的相关信息，有收款方姓名以及金额。使用姓名到 zhangdan 数组的映射来储存。

```
mapping(string=>zhangdan[]) z;
```



核心功能：

1. 查询账单

   查询返回的数据有欠款方姓名与欠款金额，分两个函数返回。

```
function get_bills_money(string user) public constant returns(int256,int256) {
    if(bills[user].valid==true) {
        if(bills[user].money<0) return (0,bills[user].qian_m);
        return (1,int256(bills[user].money));
    }
    return (0,0);
}

function get_bills_from(string user) returns(string) {
    if(bills[user].valid==false) {
        return "";
    }
    return bills[user].from_user;
}
```

   在查询金额的时候会根据这个账单是记录欠款金额还是拥有金额来进行返回，第一个参数就是用来区分返回的类型。

2. 转移账单

   通过输入转账人的姓名和接收账单的姓名以及金额来进行转账，通过 bool 值判断转账是否成功。

```
function tran_bills(string f_u,string t_u,int m) returns(bool){
        if(bills[f_u].valid==false) {
            bills[f_u].money-=m;
            bills[f_u].valid=true;
            bills[f_u].qian_m+=m;
            bills[t_u].valid=true;
            bills[t_u].money+=m;
            bills[t_u].from_user=f_u;
            zhangdan memory temp=zhangdan(t_u,m);
            z[f_u].push(temp);
            return true;
        }
        if(bills[f_u].money<m) {
            if(bills[f_u].money>0){
                return false;
            }
            //return false;
        }
        if(bills[f_u].money<0){
            bills[f_u].qian_m+=m;
            bills[t_u].from_user=f_u;
            zhangdan memory temp1=zhangdan(t_u,m);
            z[f_u].push(temp1);
        }
        else {
            bills[t_u].from_user=bills[f_u].from_user;
            string origin=bills[f_u].from_user;
            for(uint i=0;i<z[origin].length;i++) {
                string memory t=z[origin][i].user;
                if(keccak256(t)==keccak256(f_u)) {
                    z[origin][i].money-=m;
                    break;
                }
            }
            zhangdan memory temp2=zhangdan(t_u,m);
            z[origin].push(temp2);
        }
        bills[f_u].money-=m;
        bills[t_u].valid=true;
        bills[t_u].money+=m;

        return true;
    }
```

这里在转移账单的时候也会在 zhangdan 结构体中更新最初欠款方的账单，使其添加新的债权人，也更新金额。这里在银行的操作的时候也是一样的，将银行也看成是一个个体用户。

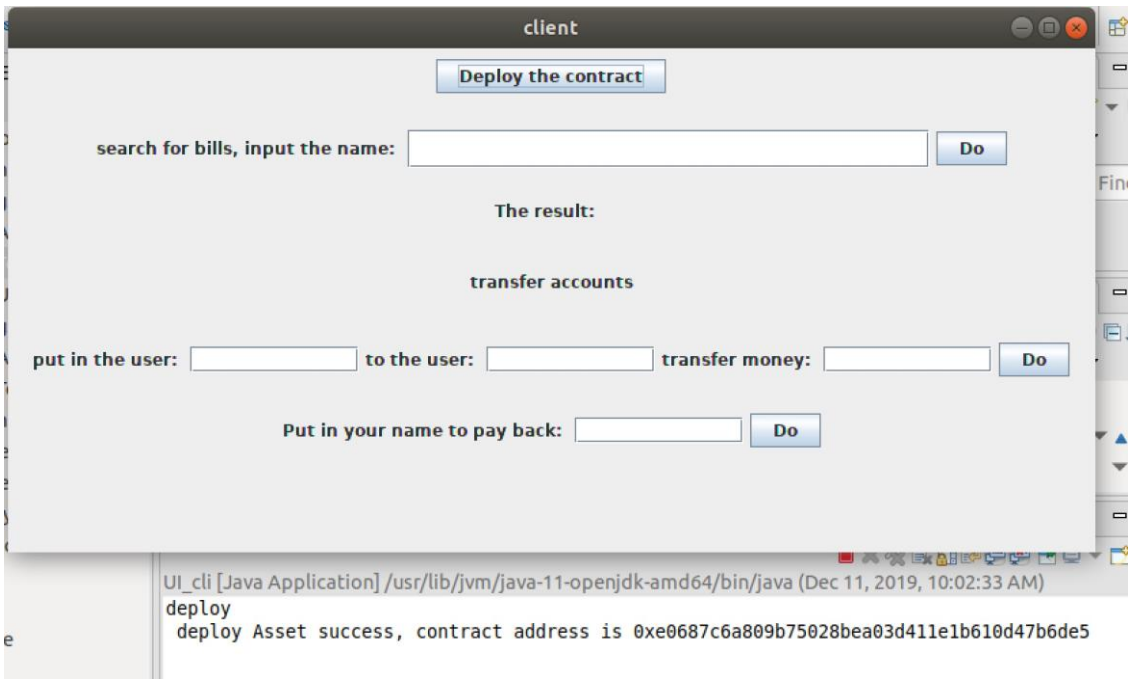
3. 偿还账单

通过一个函数实现，输入用户姓名查找其所有的债权人并对其所欠金额进行偿还，更新与注销债权人的相关账单。

```
function huanqian(string com){
    bills[com].qian_m=0;
    bills[com].money=0;
    bills[com].valid=false;
    for(uint i=0;i<z[com].length;i++) {
        bills[z[com][i].user].money=0;
        bills[z[com][i].user].valid=false;
        bills[z[com][i].user].from_user="";
    }
}
```

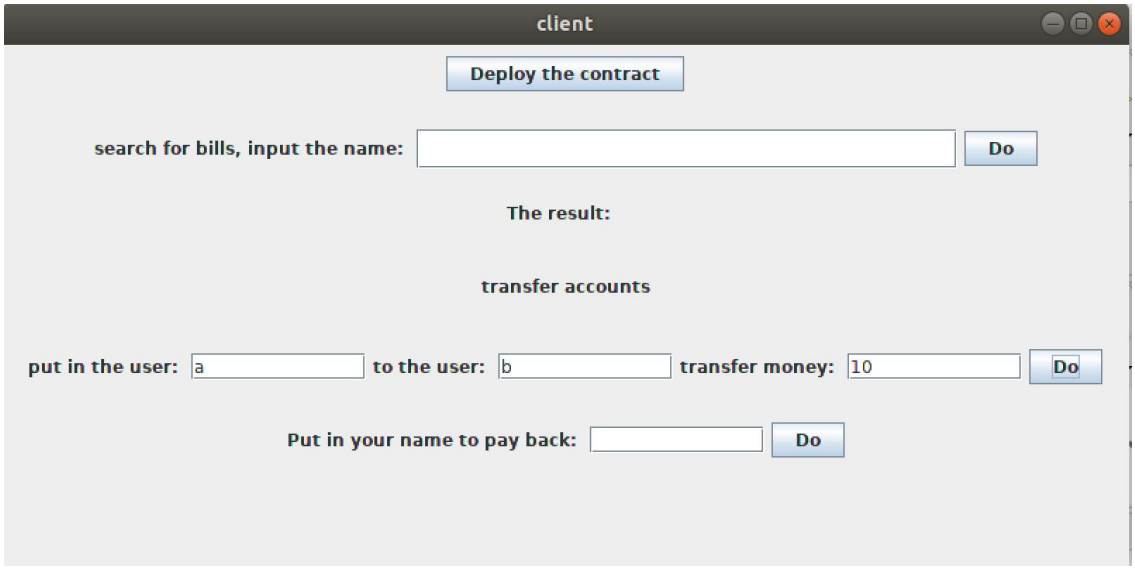
三、 功能测试

1. 合约部署

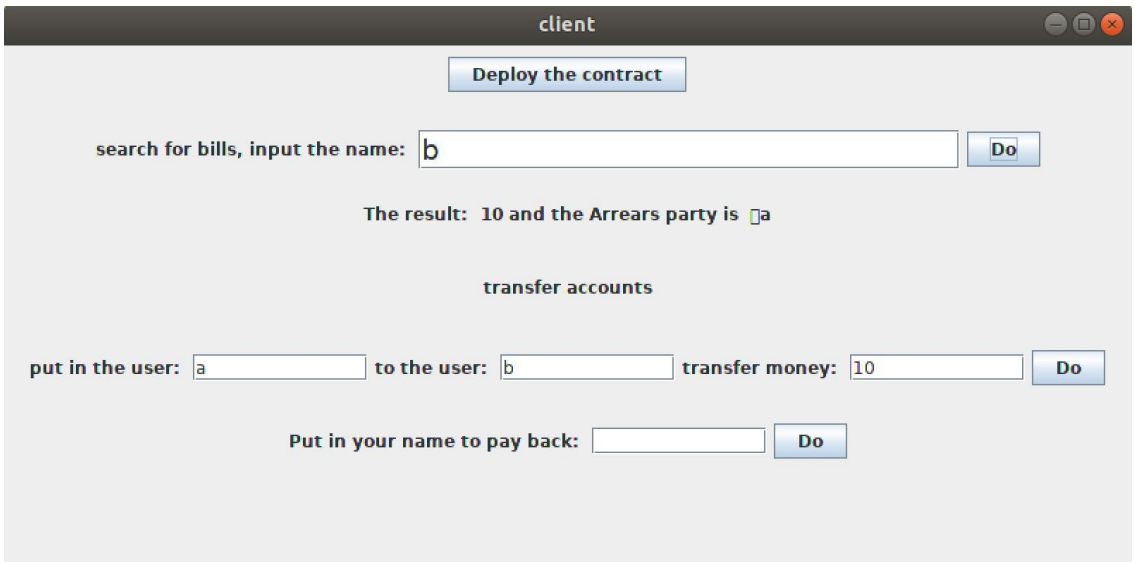


2. 转移账单

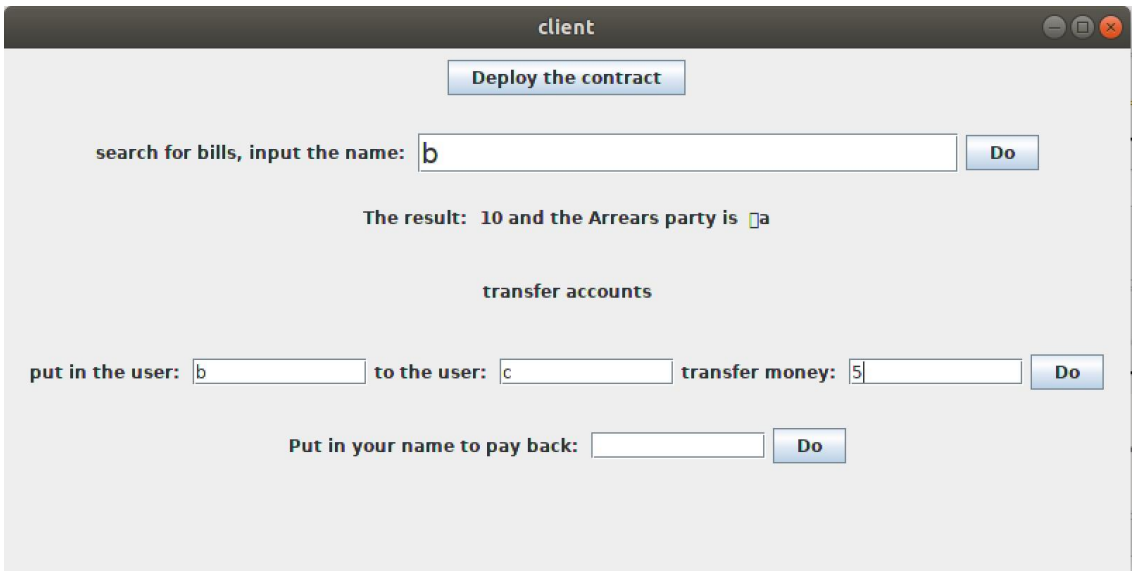
a 向 b 转账 10:



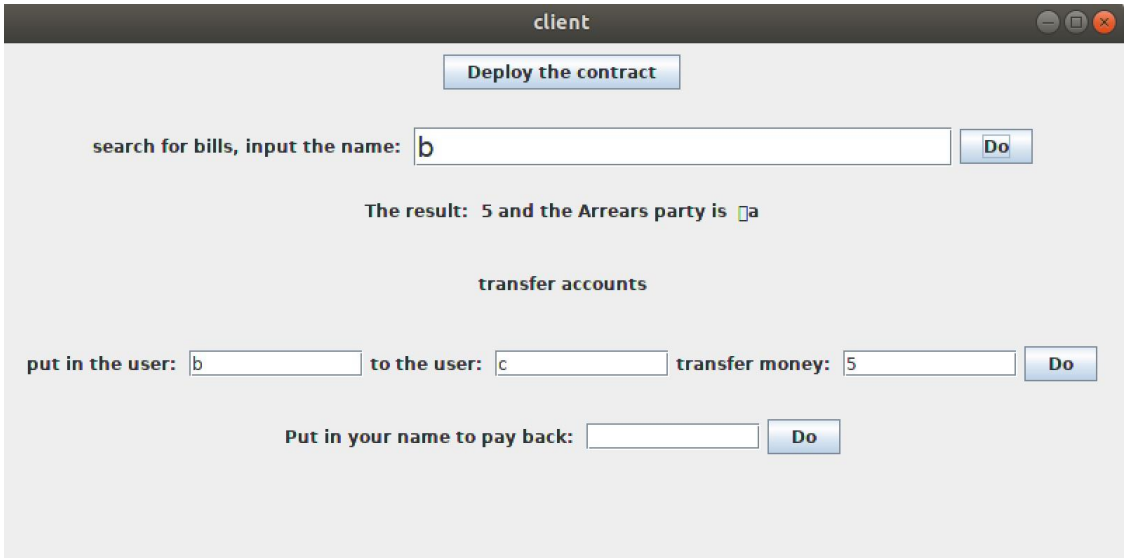
查询 b 此时拥有的账单：



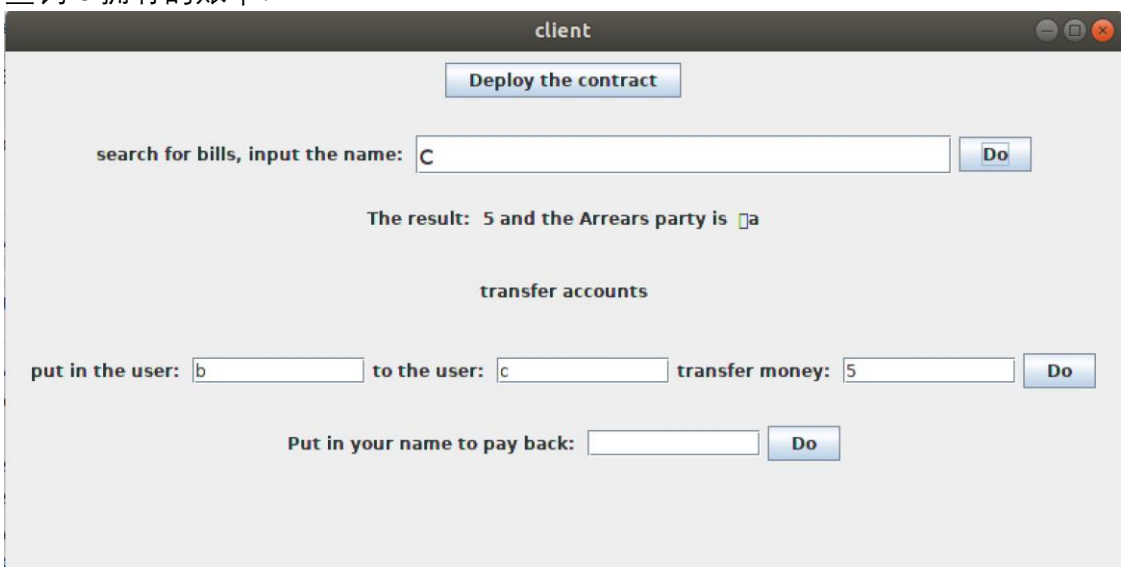
b 再向 c 转账 5（将 b 拥有的 a 的账单转移一部分出去）：



然后查询 b 拥有的账单：

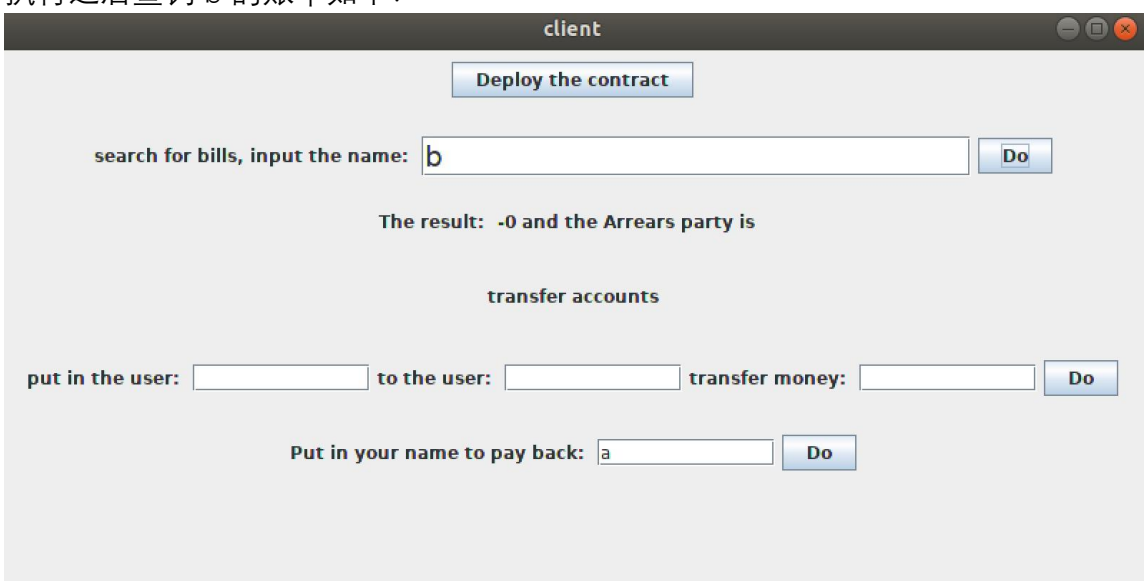


查询 c 拥有的账单：

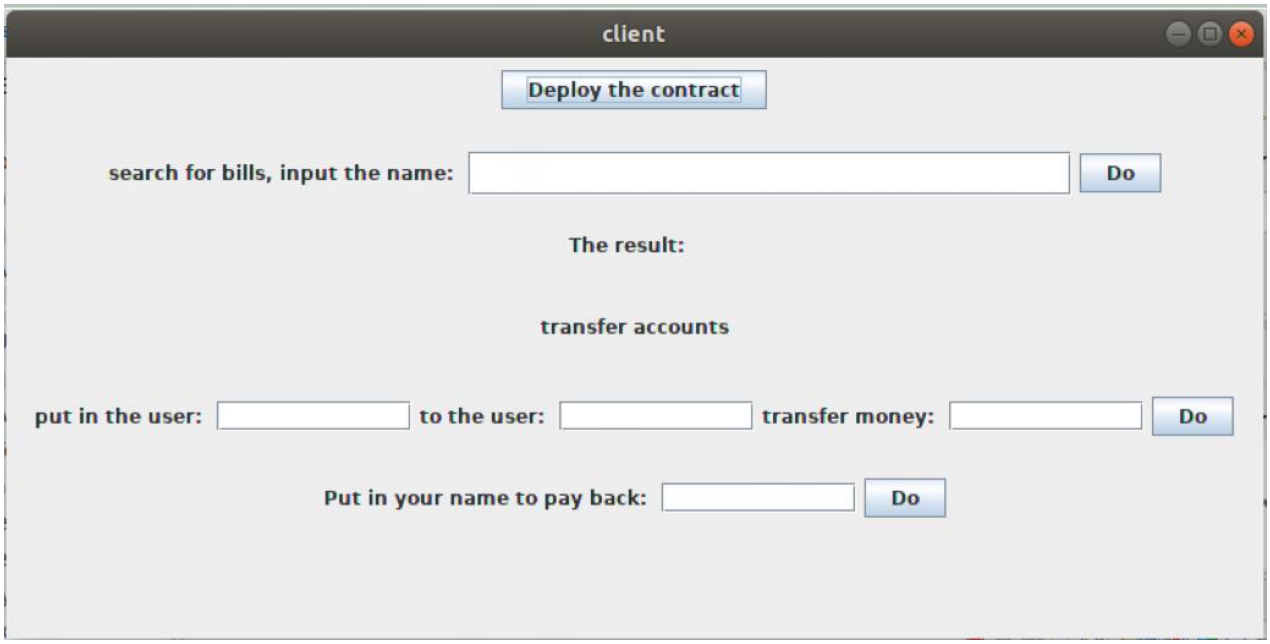


3. a 付清所有账单

执行之后查询 b 的账单如下：



四、 界面展示



客户端与服务端是写在一起的，点击“Deploy the contract”对合约进行部署，有查询账单与转移账单的功能，转账给银行同样可以使用转账功能完成。

五、 心得体会

本次是第一次开发区块链的相关应用，对所有的配置都是第一次学习，收获到了很多，包括怎么去将部署的合约调用起来，通过 java 来对函数进行调用，解析返回值，这里其实并不是很复杂，主要是因为其相关的 API 都已经写好了，只需要将相关的参数输入然后调用即可，大大减少了工作量，也使得编码工作变得非常的简单。在官方文档上也有详细的例子来说明应该怎么操作，通过阅读这些文档与例子就可以很好的去掌握这些 API 然后加以运用。总的来说通过这次的实验学习到了很多的东西，但是由于事件仓促所以之前有很多的构想没有实现，比如加一个用户登陆验证等功能，完善一下安全方面的函数。这次知实现了最基础的转移账单等功能，如果还有时间应该可以做的更好。