# 中山大学数据科学与计算机学院本科生实验报告


- 年级：2017级
- 专业：软件工程
- 学号：17343153
- 姓名：张淇
- 手机：13826656099
- 邮箱：zhangq295@mail2.sysu.edu.cn

- 开始日期：12.11
- 结束日期：12.13

## 一、项目背景

依照作业需求【区块链大作业：基于区块链的供应链金融平台.pdf】

## 二、方案设计

- 因为本次作业的主要需求是**信用向链条下端的传递**，所以在设计联盟链时采用了【用户-信用额度】绑定。
- 接下来围绕**信用额度**来解释本项目的设计逻辑：
  - 和作业需求中图中的一样，上游企业（对应程序中的acc0）因为规模较大，所以其初始信用额度较大（10000）；而下游企业的初始信用额度比较少（1000）。
  - **交易：**上游企业与下游企业进行交易，理解为两者之间签订了一个订单【订单号-债主-欠款人-交易金额-未结清金额】（可视为欠条），并未产生资金上的转移，而是进行了信用额度之间的传递。（该行为会记录在链上）
  - **查询**：用户可以查询自身的信用额度以及根据订单号来查询相应的交易。
  - **融资/贷款：**任何企业均可以使用**自身的信用额度进行融资/向银行贷款**，获得一笔不超过自身信用额度的金额。在合约中将这种行为视为企业与银行之间发生了一笔交易，将企业的信用额度转移到银行下。这种做法在现实中也是合乎逻辑的，因为银行不会贷款给一个老赖。（该行为会记录在链上）
  - **还款：**在一笔交易中作为欠款人的企业可以通过还钱给债主来让订单中的**未结清金额**下降，并且从债主上返还同等数量的信用额度。当未结清金额为0时意味着该订单结束，但不会被销毁。（该行为会记录在链上）
  - **欠条拆分：**在一笔交易中，债主可以进行欠条拆分，可以将**不超过该欠条未结清金额**的债权转移给另外一个用户；与此同时，该用户也会从债主那获得同等金额的信用额度。（该行为会记录在链上）
  - **下面是一笔交易的例子：**
    - 车企（上游企业，信用额度为A）与轮胎厂（下游企业，信用额度为B）之间进行了一次交易，交易额为X，那么就会产生一个【id0 - 轮胎厂 - 车企 - X - X】的订单。与此同时，产生的信用额度的转移`Y = min(A, X)`。交易完成后，车企的信用额度`A' = A - Y`，轮胎厂的信用额度为`B' = B + Y`。
    - 之后车企对该订单进行支付（还款）M（其中`M∈(0,X]`），则该订单更新为【id0 - 轮胎厂 - 车企 - X - (X-M)】。还款完成后，车企的信用额度`A'' = A' + M`,轮胎厂的信用额度为`B'' = B' - M`。

    - 轮胎厂与轮毂厂（信用额度为C）进行了一次交易，产生了一个【id1-轮毂厂-轮胎厂-Z-Z】（`Z <= X-M`）的订单。此时轮胎厂可以将“id0”的欠条进行拆分，用来支付“id1”的欠条。经过拆分后，“id0”的欠条变成【id0 - 轮胎厂 - 车企 - X - (X - M - Z)】；“id1”的欠条变成【id1 - 轮毂厂 - 轮胎厂 - Z - 0】（已还清）；同时产生一个新的欠条【id2 - 轮毂厂 - 车企 - Z - Z】。同时，信用额度也会发生相应改变，轮胎厂的信用额度为`B''' = B'' - Z`，轮毂厂的信用额度为`C' = C + Z`。
    - 最后，轮毂厂可以使用自己的信用额度进行贷款，获得不超过`C'`的金额。
- 核心代码：
    - 合约：
        - ```
          /*
              Asset.sol
              描述 : 添加交易记录（打欠条）
              参数 :
                      id : 交易编号
                      acc1 : 债主
                      acc2 : 借债人
                      money: 初始金额 （欠条未还清的金额与初始金额一致）
              返回值：
                      0  交易添加成功
                      -1 交易ID已存在
                      -2 其他错误
                      -3 信用额度转让失败
              */
              function addTransaction(string id, string acc1, string acc2,
          int256 money) public returns(int256){
                  int256 ret_code = 0;
                  int256 ret = 0;
                  bytes32[] memory str_list = new bytes32[](2);
                  int256[] memory int_list = new int256[](3);

                  // 查询交易是否存在
                  (int_list, str_list) = select_transaction(id);
                  if(int_list[0] != int256(0)) {
                      Table table = openTransactionTable();

                      Entry entry0 = table.newEntry();
                      entry0.set("id", id);
                      entry0.set("acc1", acc1);
                      entry0.set("acc2", acc2);
                      entry0.set("money", int256(money));
                      entry0.set("status", int256(money));
                      // 插入
                      int count = table.insert(id, entry0);
                      if (count == 1) {
                          // 将欠款人的信用额度转移一部分给债主
                          ret = transfer(acc2,acc1,money);
                          // 信用额度转让失败
                          if(ret != 0) {
                              ret_code = -3;
                          } else {
                              ret_code = 0;
                          }
                      } else {
                          // 失败? 无权限或者其他错误
                          ret_code = -2;
          ```
          ```

```
            }
        } else {
            // 交易ID已存在
            ret_code = -1;
        }

        emit AddTransactionEvent(ret_code, id, acc1, acc2, money);

        return ret_code;
    }
```

```
/*
    Asset.sol
    描述 : 更新交易记录(支付欠条)
    参数 :
            id : 交易编号
            money: 金额
    返回值:
             0  交易更新成功
            -1 交易ID不存在
            -2 还债金额大于债款
            -3 其他错误
            -4 信用返还有问题
    */
    function updateTransaction(string id, int256 money) public
returns(int256, string[]){
        int256 ret_code = 0;
        // int256 ret = 0;
        bytes32[] memory str_list = new bytes32[](2);
        int256[] memory int_list = new int256[](3);
        string[] memory acc_list = new string[](2);
        // 查询该欠条是否存在
        (int_list, str_list) = select_transaction(id);
        acc_list[0] = byte32ToString(str_list[0]);
        acc_list[1] = byte32ToString(str_list[1]);

        if(int_list[0] == 0) { // 交易ID存在

            // 还款金额大于欠款金额
            if(int_list[2] < money){
                ret_code = -2;
                emit UpdateTransactionEvent(ret_code, id, money);
                return (ret_code, acc_list);
            }

            // 余额不足
            // uint256 acc2_balance;
            // (ret, acc2_balance) =
select(byte32ToString(str_list[1]));
            // if(int256(acc2_balance) < money) {
            //     ret_code = -2;
            //     emit UpdateTransactionEvent(ret_code, id,
money);
            //     return ret_code;
            // }

            // 更新交易状态
```

```
            Table table = openTransactionTable();

            Entry entry0 = table.newEntry();
            entry0.set("id", id);
            entry0.set("acc1", byte32ToString(str_list[0]));
            entry0.set("acc2", byte32ToString(str_list[1]));
            entry0.set("money", int_list[1]);
            entry0.set("status", (int_list[2] - money));

            // 更新欠条
            int count = table.update(id, entry0,
table.newCondition());
            if(count != 1) {
                ret_code = -3;
                // 失败? 无权限或者其他错误?
                emit UpdateTransactionEvent(ret_code, id, money);
                return (ret_code,acc_list);
            }

            // 信用额度返还
            int256 temp =
transfer(byte32ToString(str_list[0]),byte32ToString(str_list[1]),mo
ney);
            if(temp != 0){
                ret_code = -4 * 10 + temp;
                emit UpdateTransactionEvent(ret_code, id, money);
                return (ret_code,acc_list);
            }

            ret_code = 0;

        } else { // 交易ID不存在
            ret_code = -1;
        }
        emit UpdateTransactionEvent(ret_code, id, money);

        return (ret_code,acc_list);
    }
```

- ```
    /*
    Asset.sol
    描述 : 信用额度转移
    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 //pass
            -4 金额溢出 //pass
            -5 其他错误
    */
    function transfer(string from_account, string to_account,
int256 amount) public returns(int256) {
        // 查询转移资产账户信息
```
```

```
        int ret_code = 0;
        int256 ret = 0;
        int256 from_asset_value = 0;
        int256 to_asset_value = 0;

        // 转移账户是否存在?
        (ret, from_asset_value) = select(from_account);
        if(ret != 0) {
            ret_code = -1;
            // 转移账户不存在
            emit TransferEvent(ret_code, from_account, to_account,
amount);
            return ret_code;

        }

        // 接受账户是否存在?
        (ret, to_asset_value) = select(to_account);
        if(ret != 0) {
            ret_code = -2;
            // 接收资产的账户不存在
            emit TransferEvent(ret_code, from_account, to_account,
amount);
            return ret_code;
        }

        if(from_asset_value < amount) {
            ret_code = -3;
            // 转移资产的账户金额不足
            emit TransferEvent(ret_code, from_account, to_account,
amount);
            return ret_code;
        }

        if (to_asset_value + amount < to_asset_value) {
            ret_code = -4;
            // 接收账户金额溢出
            emit TransferEvent(ret_code, from_account, to_account,
amount);
            return ret_code;
        }

        Table table = openAssetTable();

        Entry entry0 = table.newEntry();
        entry0.set("account", from_account);
        entry0.set("asset_value", int256(from_asset_value -
amount));
        // 更新转账账户
        int count = table.update(from_account, entry0,
table.newCondition());
        if(count != 1) {
            ret_code = -5;
            // 失败? 无权限或者其他错误?
            emit TransferEvent(ret_code, from_account, to_account,
amount);
            return ret_code;
        }
```

```
        Entry entry1 = table.newEntry();
        entry1.set("account", to_account);
        entry1.set("asset_value", int256(to_asset_value + amount));
        // 更新接收账户
        table.update(to_account, entry1, table.newCondition());

        emit TransferEvent(ret_code, from_account, to_account,
amount);

        return ret_code;
    }
```

- ```
    /*
        Asset.sol
        描述 ： 欠条拆分
        参数 ：
                old_id: 需要拆分的欠条id
                new_id: 新创建的欠条的id
                acc: 新创建欠条的债主
                money: 欠条拆分的金额
        返回值 ：
                 0 欠条拆分成功
                -1 拆分的欠条id不存在
                -2 需要拆分的金额大于欠条金额（余额）
                -3 新欠条创建不成功
                -4 其他错误
                -5 用户acc不存在
        */
        function splitTransaction(string old_id, string new_id, string
    acc, int256 money) public returns(int256) {
            int256 ret_code = 0;
            int256 ret = 0;
            int temp = 0;
            bytes32[] memory str_list = new bytes32[](2);
            int256[] memory int_list = new int256[](3);
            string[] memory acc_list = new string[](2);
            // 查询该欠条是否存在
            (int_list, str_list) = select_transaction(old_id);

            if(int_list[0] == 0) {
                // acc不存在
                (ret, temp) = select(acc);
                if(ret != 0) {
                    ret_code = -5;
                    emit SplitTransactionEvent(ret_code, old_id,
    new_id, acc, money);
                    return ret_code;
                }

                if(int_list[2] < money){    // 拆分的金额大于欠条余额
                    ret_code = -2;
                    emit SplitTransactionEvent(ret_code, old_id,
    new_id, acc, money);
                    return ret_code;
                }
    ```
    ```

```
            // acc1先“还钱”给acc2，然后acc2再“借钱”给acc
            (ret,acc_list) = updateTransaction(old_id, money);
            if (ret != 0) {
                ret_code = -4;
                emit SplitTransactionEvent(ret_code, old_id,
new_id, acc, money);
                return ret_code;
            }
            ret = addTransaction(new_id, acc,
byte32ToString(str_list[1]), money);
            if (ret != 0) {
                ret_code = -3;
                emit SplitTransactionEvent(ret_code, old_id,
new_id, acc, money);
                return ret_code;
            }

        } else {    // 拆分的欠条id不存在
            ret_code = -1;
        }

        emit SplitTransactionEvent(ret_code, old_id, new_id, acc,
money);
        return ret_code;
    }
```

- 后端：
  - ```
    // AssetClient.java
    // 初始化
        public void initialize() throws Exception {

            // init the Service
            @SuppressWarnings("resource")
            ApplicationContext context = new
    ClassPathXmlApplicationContext("classpath:applicationContext.xml");
            Service service = context.getBean(Service.class);
            service.run();

            ChannelEthereumService channelEthereumService = new
    ChannelEthereumService();
            channelEthereumService.setChannelService(service);
            Web3j web3j = Web3j.build(channelEthereumService, 1);

            // init Credentials
            Credentials credentials =
    Credentials.create(Keys.createEcKeyPair());

            setCredentials(credentials);
            setWeb3j(web3j);

            logger.debug(" web3j is " + web3j + " ,credentials is " +
    credentials);
        }
    ```
  - ```
    // AssetClient.java
    // 部署合约
    ```

```
public void deployAssetAndRecordAddr() {

        try {
            Asset asset = Asset.deploy(web3j, credentials, new
StaticGasProvider(gasPrice, gasLimit)).send();
            System.out.println(" deploy Asset success, contract
address is " + asset.getContractAddress());

            recordAssetAddr(asset.getContractAddress());
        } catch (Exception e) {
            // TODO Auto-generated catch block
            // e.printStackTrace();
            System.out.println(" deploy Asset contract failed,
error message is  " + e.getMessage());
        }
    }
```

- 

```
// AssetClient.java
// 查询信用额度、查找交易
    public boolean queryAssetAmount(String assetAccount) {
        try {
            String contractAddress = loadAssetAddr();

            Asset asset = Asset.load(contractAddress, web3j,
credentials, new StaticGasProvider(gasPrice, gasLimit));
            Tuple2<BigInteger, BigInteger> result =
asset.select(assetAccount).send();
            if (result.getValue1().compareTo(new BigInteger("0"))
== 0) {
                System.out.printf(" 您的信用额度为: %s \n",
result.getValue2());
                return true;
            } else {
                System.out.printf(" %s asset account is not exist
\n", assetAccount);
                return false;
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            // e.printStackTrace();
            logger.error(" queryAssetAmount exception, error
message is {}", e.getMessage());

            System.out.printf(" query asset account failed, error
message is %s\n", e.getMessage());
        }
        return false;
    }

    public void queryAssetTransaction(String t_id) {
        try {
            String contractAddress = loadAssetAddr();

            Asset asset = Asset.load(contractAddress, web3j,
credentials, new StaticGasProvider(gasPrice, gasLimit));
            Tuple2<List<BigInteger>, List<byte[]>> result =
asset.select_transaction(t_id).send();
```

```
            if (result.getValue1().get(0).compareTo(new
BigInteger("0")) == 0) {
                String temp1 = new
String(result.getValue2().get(0));
                String temp2 = new
String(result.getValue2().get(1));
                System.out.printf("Transaction\n ID: " + t_id + ";
Acc1: " + temp1 + "; Acc2: " + temp2 + "; Money: " +
result.getValue1().get(1) + "; Current: " +
result.getValue1().get(2) + "\n");
            } else {
                System.out.printf("Transaction %s is not exist \n",
t_id);
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            // e.printStackTrace();
            logger.error(" queryAssetAmount exception, error
message is {}", e.getMessage());

            System.out.printf(" query asset account failed, error
message is %s\n", e.getMessage());
        }
    }
```

- ```
  // AssetClient.java
  // 产生交易
      public void addAssetTransaction(String t_id, String acc1,
  String acc2, BigInteger money) {
          try {
              String contractAddress = loadAssetAddr();

              Asset asset = Asset.load(contractAddress, web3j,
  credentials, new StaticGasProvider(gasPrice, gasLimit));
              TransactionReceipt receipt = asset.addTransaction(t_id,
  acc1, acc2, money).send();
              List<AddTransactionEventEventResponse> response =
  asset.getAddTransactionEventEvents(receipt);
              if (!response.isEmpty()) {
                  if (response.get(0).ret.compareTo(new
  BigInteger("0")) == 0) {
                      System.out.printf(" Add transaction success!
  id:" + t_id+"\n");
                  } else {
                      System.out.printf(" Add transaction failed, ret
  code is %s \n",
                              response.get(0).ret.toString());
                  }
              } else {
                  System.out.println(" event log not found, maybe
  transaction not exec. ");
              }
          } catch (Exception e) {
              // TODO Auto-generated catch block
              // e.printStackTrace();
  ```
  ```

```
            logger.error(" registerAssetAccount exception, error
message is {}", e.getMessage());
            System.out.printf(" register asset account failed,
error message is %s\n", e.getMessage());
        }
    }
```

- ```
// AssetClient.java
// 支付欠条
    public void updateAssetTransaction(String t_id, BigInteger
money) {
        try {
            String contractAddress = loadAssetAddr();
            Asset asset = Asset.load(contractAddress, web3j,
credentials, new StaticGasProvider(gasPrice, gasLimit));
            TransactionReceipt receipt =
asset.updateTransaction(t_id, money).send();
            List<UpdateTransactionEventEventResponse> response =
asset.getUpdateTransactionEventEvents(receipt);
            // Tuple2<BigInteger, List<String>> result =
asset.getUpdateTransactionOutput(asset.updateTransaction(t_id,
money).send());

            if (!response.isEmpty()) {
                if (response.get(0).ret.compareTo(new
BigInteger("0")) == 0) {
                    System.out.printf(" Update transaction
success.\n" );
                } else {
                    System.out.printf(" Update transaction failed,
ret code is %s \n",
                            response.get(0).ret.toString());
                }
            } else {
                System.out.println(" event log not found, maybe
transaction not exec. ");
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            // e.printStackTrace();

            logger.error(" registerAssetAccount exception, error
message is {}", e.getMessage());
            System.out.printf(" register asset account failed,
error message is %s\n", e.getMessage());
        }
    }
```

- ```
// AssetClient.java
// 欠条拆分
public void splitAssetTransaction(String old_id, String new_id,
String acc, BigInteger money) {
        try {
            String contractAddress = loadAssetAddr();
            Asset asset = Asset.load(contractAddress, web3j,
credentials, new StaticGasProvider(gasPrice, gasLimit));
```

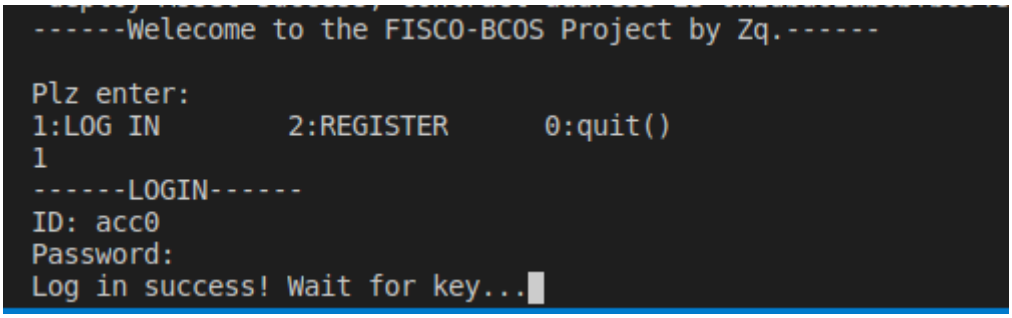
```
            TransactionReceipt receipt =
asset.splitTransaction(old_id, new_id, acc, money).send();
            List<SplitTransactionEventEventResponse> response =
asset.getSplitTransactionEventEvents(receipt);
            if (!response.isEmpty()) {
                if (response.get(0).ret.compareTo(new
BigInteger("0")) == 0) {
                    System.out.printf(" Split transaction success!
old_id: "+ old_id +" new_id: "+new_id+"\n");
                } else {
                    System.out.printf(" Split transaction failed,
ret code is %s \n",
                            response.get(0).ret.toString());
                }
            } else {
                System.out.println(" event log not found, maybe
transaction not exec. ");
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            // e.printStackTrace();

            logger.error(" registerAssetAccount exception, error
message is {}", e.getMessage());
            System.out.printf(" register asset account failed,
error message is %s\n", e.getMessage());
        }
    }
```

- 交互界面：

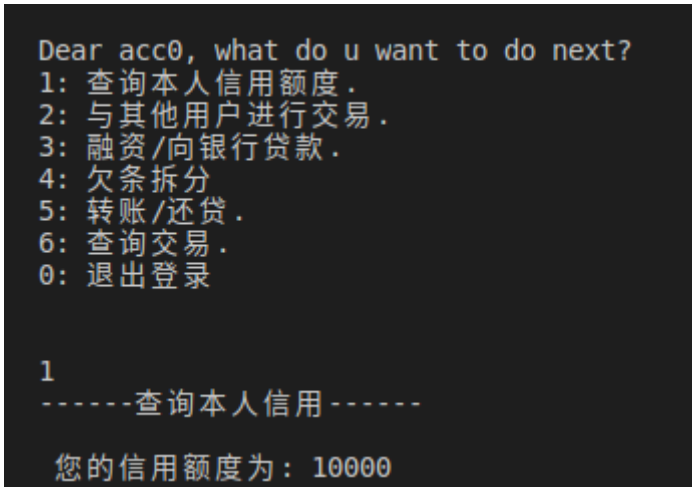
```
// CLI.java
public class CLI{
    ...
}
```

# 三、实验结果截图

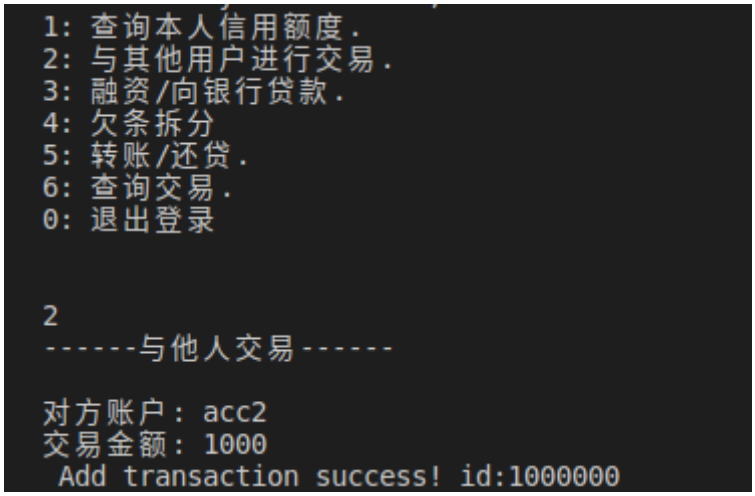
- 登陆界面



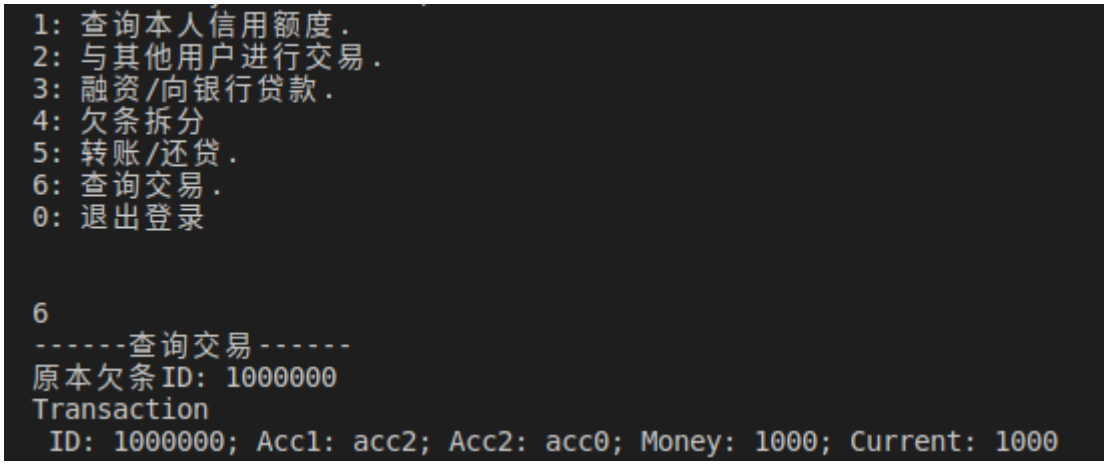

- 用户acc0：登陆成功后，查询自身信用额度




- acc0：与acc2进行一笔1000的交易




- acc0：查询这笔交易




- acc0：向银行贷款1000后查询该交易

```
Wait for key...Dear acc0, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


6
------查询交易------
原本欠条ID: 1000001
Transaction
 ID: 1000001; Acc1: bank; Acc2: acc0; Money: 1000; Current: 1000
```

- acc0：查询本人信用额度

```
Dear acc0, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


1
------查询本人信用------

 您的信用额度为: 8000
```

- acc2：查询本人信用额度（初始值为1000）

```
Dear acc2, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


1
------查询本人信用------

 您的信用额度为: 2000
```

- acc2：对id = 1000000的欠条进行拆分

```
Wait for key...Dear acc2, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


4
------欠条拆分------

受益人: acc3
原本欠条ID: 1000000
转让金额: 500
 Split transaction success! old_id: 1000000 new_id: 1000002
```

- acc2：查询本人信用额度

```
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


1
------查询本人信用------

 您的信用额度为: 1500
```

- acc2：查询欠条拆分产生的新交易

```
Dear acc2, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


6
------查询交易------
原本欠条ID: 1000002
Transaction
 ID: 1000002; Acc1: acc3; Acc2: acc0; Money: 500; Current: 500
```

- acc2：查询欠条id = 100000此时的状态

```
Wait for key...Dear acc2, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


6
------查询交易------
原本欠条ID: 1000000
Transaction
 ID: 1000000; Acc1: acc2; Acc2: acc0; Money: 1000; Current: 500
```

- acc0：支付银行贷款的订单

```
Wait for key...Dear acc0, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


5
------转账/还贷------

PS: 若为还贷，则对方账户为【bank】
交易ID: 1000001
转账金额：1000
 Update transaction success.
```

- acc0：查询本人信用额度

```
Dear acc0, what do u want to do next?
1: 查询本人信用额度.
2: 与其他用户进行交易.
3: 融资/向银行贷款.
4: 欠条拆分
5: 转账/还贷.
6: 查询交易.
0: 退出登录


1
------查询本人信用------

 您的信用额度为：9000
```

# 四、加分项

- 设置了用户注册、登录的功能
- 使用了FISCO-BCOS的国密功能

# 五、实验心得

> 这个项目我本来预计用DDL前大概一周的时间来完成，但是因为恰好在这个时候得了一场重感冒，于是DDL前几天一直处于头昏脑胀的一种难受的状态，所以进度慢的感人。
>
> 可以认为我基本是在最后两天才能够勉强打起精神来做的（一面擤鼻涕一边敲代码也是一言难尽），实现的思路是链端→后端→前端，但是当我把后端与链端之间的功能调好之后发现已经距离DDL只剩十来个小时了。在学习了一下Spring + Gradle进行前后端链接的教程后发现事情远没有那么简单，于是最后我就只好选择使用命令行界面来完成。
>
> 不过仔细想一想，这个作业挺早之前就已经放出来给我们了，不过因为感觉给的期限比较宽松所以一直没有把他提上日程，导致最后拼命赶DDL的后果，也是怪自己的拖延症太严重吧。
>
> 这次作业是基于FISCO-BCOS的相关组件的，在Debug过程中，Webase的那一块给我提供了挺大的帮助，sol2java这个脚本的功能也提供了一个合约到java文件的一个转换，从而能够让程序更快的跑通。不过成也萧何败也萧何，在DDL前最后进行调试的过程中，因为FISCO-BCOS的Nodes出现了一些“莫名其妙”的bug导致程序Run不起来，差点让我崩溃。
>
> 总体来说，这个项目让我对区块链（联盟链）有了更深的认识，了解到了其实际生活中的应用场景，并且通过程序的方式来实现它的基本功能，让我明白还有很远的路要走。