

# 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	大三	专业（方向）	软件工程
学号	17343092	姓名	潘鹏程
电话	13242867595	Email	949138604@qq.com
开始日期	2019/11/2	完成日期	2019/12/13

### 一、项目背景

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。

### 二、方案设计

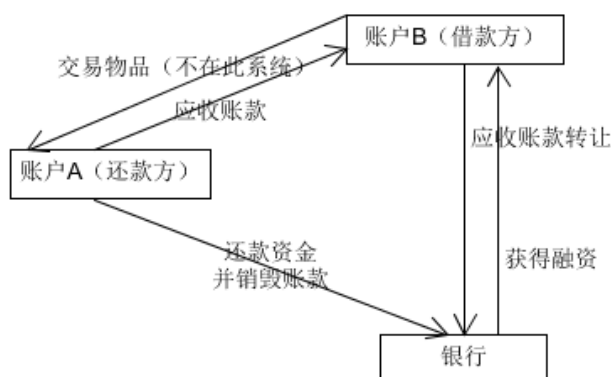
存储设计：

```
4 struct Bill{
5     address borrower;
6     uint amount;
7 }
8 address public bank;
9 mapping (address => uint) private balances;
10 mapping (address => Bill[]) private bill_list;
```

第三方可信机构（银行）的账号地址

每个用户的账号地址和对应的账户余额/应收账款的映射。

数据流图：



所有数据通过合约部署至区块链的形式来储存在区块链上。

核心功能介绍：

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。

```

50 //功能一：签订应收账款单据
51 function create_bill(address borrower, uint amount) public {
52     if (msg.sender == bank) return;
53     bill_transfer(borrower, msg.sender, amount);
54 }

```

通过调用账单转移函数来实现创建新账单功能。两个账户之间单向的多笔交易会合并。

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。

```

56 //功能二：转让单据
57 function borrow_funds(address borrower, uint amount) public {
58     if (msg.sender == bank) return;
59     uint i = bill_list[borrower].length;
60     do {
61         i--;
62         if (bill_list[borrower][i].amount > amount){
63             bill_list[borrower][i].amount -= amount;
64             bill_transfer(bill_list[borrower][i].borrower, msg.sender, amount);
65             return;
66         }
67         else{
68             amount -= bill_list[borrower][i].amount;
69             bill_transfer(bill_list[borrower][i].borrower, msg.sender, bill_list[borrower][i].amount);
70         }
71         bill_list[borrower].length--;
72         if (amount == 0) return;
73     } while (i > 0);
74     if (amount != 0)
75         create_bill(borrower, amount);
76 }

```

通过转让还款方手中有的应收账款给借款方，实现债务转移。若应收账款总额不足，则创建新的应收账款以补足差额。

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。

```

78 //功能三：转给银行
79 function bill_to_bank() public {
80     uint amount = 0;
81     uint i = bill_list[msg.sender].length;
82     do {
83         i--;
84         amount += bill_list[msg.sender][i].amount;
85         bill_transfer(bill_list[msg.sender][i].borrower, bank, bill_list[msg.sender][i].amount);
86     } while (i > 0);
87     bill_list[msg.sender].length = 0;
88     balances[msg.sender] += amount;
89 }

```

直接把应收账款转让给银行，从而从银行处获得融资。

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。

```

91 //功能四：结算单据
92 function remove_bill(address borrower) public {
93     for (uint i = 0; i < bill_list[msg.sender].length; i++){
94         if (bill_list[msg.sender][i].borrower == borrower){
95             balances[borrower] -= bill_list[msg.sender][i].amount;
96             if (msg.sender != bank)
97                 balances[msg.sender] += bill_list[msg.sender][i].amount;
98             clear_bill(msg.sender, i);
99             return;
100         }
101     }
102 }

```

通过资金转移来移除借款人手中的应收账款。

以上为作业二的合约介绍，最终成品使用了 fisco 官网提供的工程项目

<https://github.com/FISCO-BCOS/LargeFiles/raw/master/tools/asset-app.tar.gz>

在此框架的基础上进行后端的搭建，框架结构如下：

```
-- build.gradle // gradle 配置文件
-- gradle
| -- wrapper
| | -- gradle-wrapper.jar // 用于下载 Gradle 的相关代码实现
| | -- gradle-wrapper.properties // wrapper 所使用的配置信息，比如 gradle 的版本等信息
-- gradlew // Linux 或者 Unix 下用于执行 wrapper 命令的 Shell 脚本
-- gradlew.bat // Windows 下用于执行 wrapper 命令的批处理脚本
-- src
| -- main
| | -- java
| | | -- org
| | | | -- fisco
| | | | | -- bcos
| | | | | | -- asset
| | | | | | | -- client // 放置客户端调用类
| | | | | | | | -- AssetClient.java
| | | | | | | -- contract // 放置 Java 合约类
| | | | | | | | -- Asset.java
| -- test
| | -- resources // 存放代码资源文件
| | | -- applicationContext.xml // 项目配置文件
| | | -- contract.properties // 存储部署合约地址的文件
| | | -- log4j.properties // 日志配置文件
| | | -- contract // 存放 solidity 约文件
| | | | -- Asset.sol
| | | | -- Table.sol
-- tool
| -- asset_run.sh // 项目运行脚本
```

大作业则是重写了 AssetClient.java、Asset.java、asset\_run.sh 和相关的 sol 文件，为方便查看已放在主目录的代码文件夹下。

### 三、 功能测试

首先，在 webase-web 上的私钥管理中添加四个用户：普通企业用户 A、B、C 和第三方权威机构 Bank。

用户查看

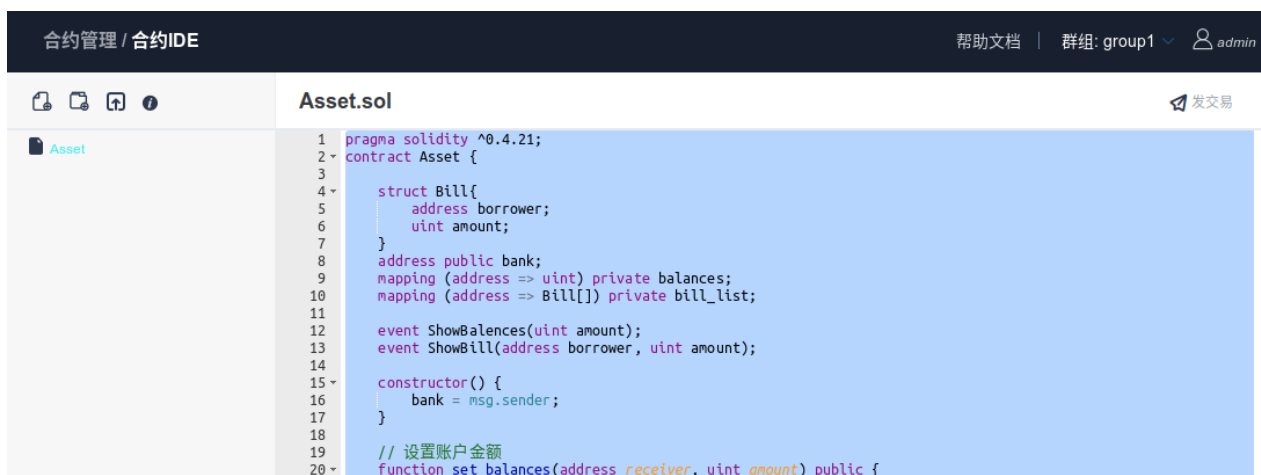
帮助文档 | 群组: group1 admin

新增用户

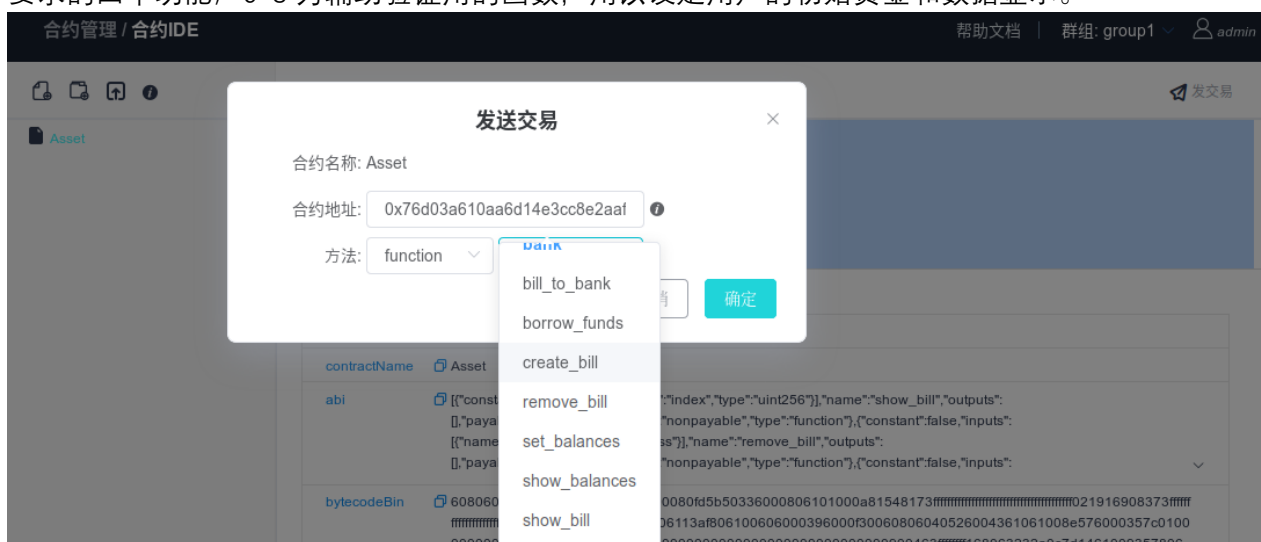
请输入用户名或公钥地址

用户名称	用户ID	用户描述	用户公钥地址信息	用户状态	操作
C	700005		0xd5db56696f34e2...	正常	修改
B	700004		0x0c4ecca7648af86...	正常	修改
A	700003		0xd57427fca6543f...	正常	修改
Bank	700002		0x3ab012c3fa543a...	正常	修改

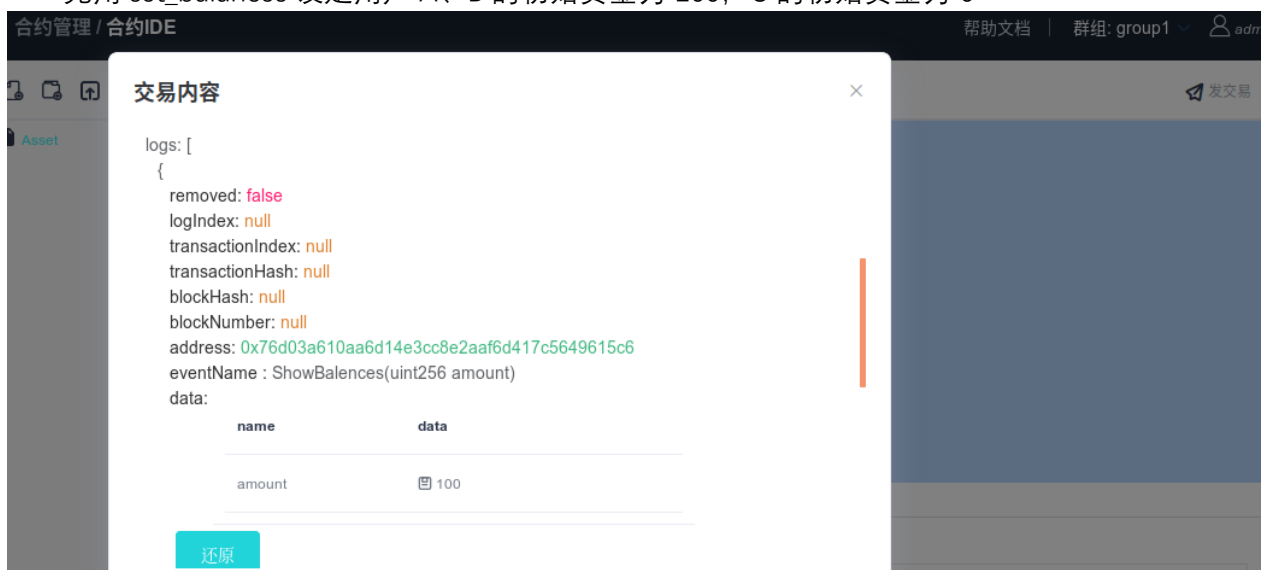
然后上传前文的合约，编译成功后部署到用户 Bank。接下来就可以发送交易验证功能了。



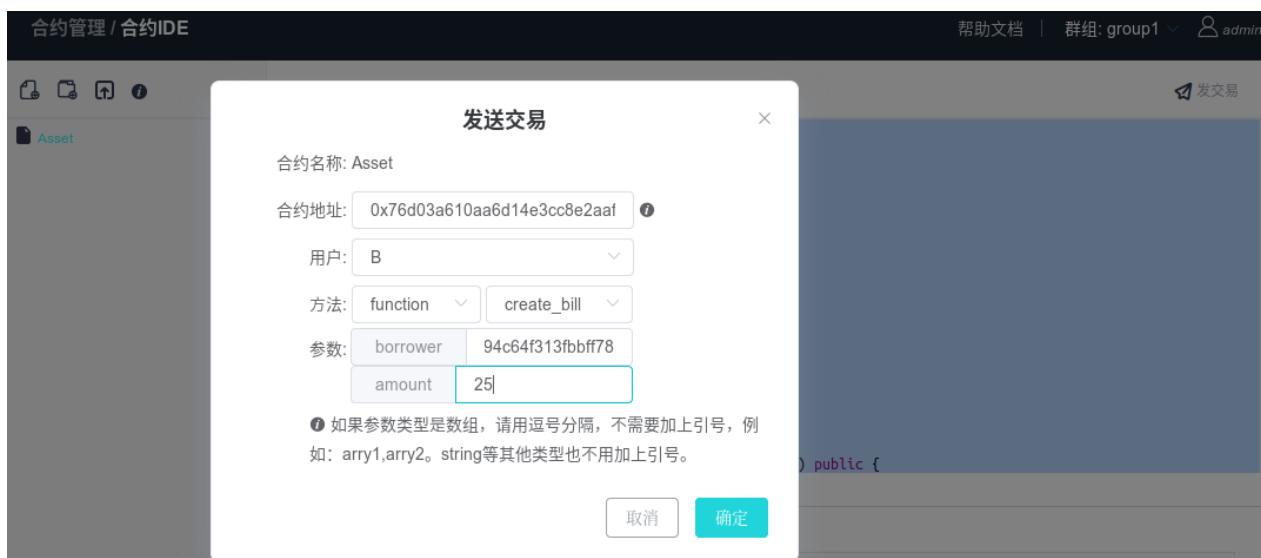
下图中的方法里有 8 个选项；第一个选项为构造函数得到的 bank 地址；2-5 对应着大作业 s 要求的四个功能；6-8 为辅助验证用的函数，用以设定用户的初始资金和数据显示。



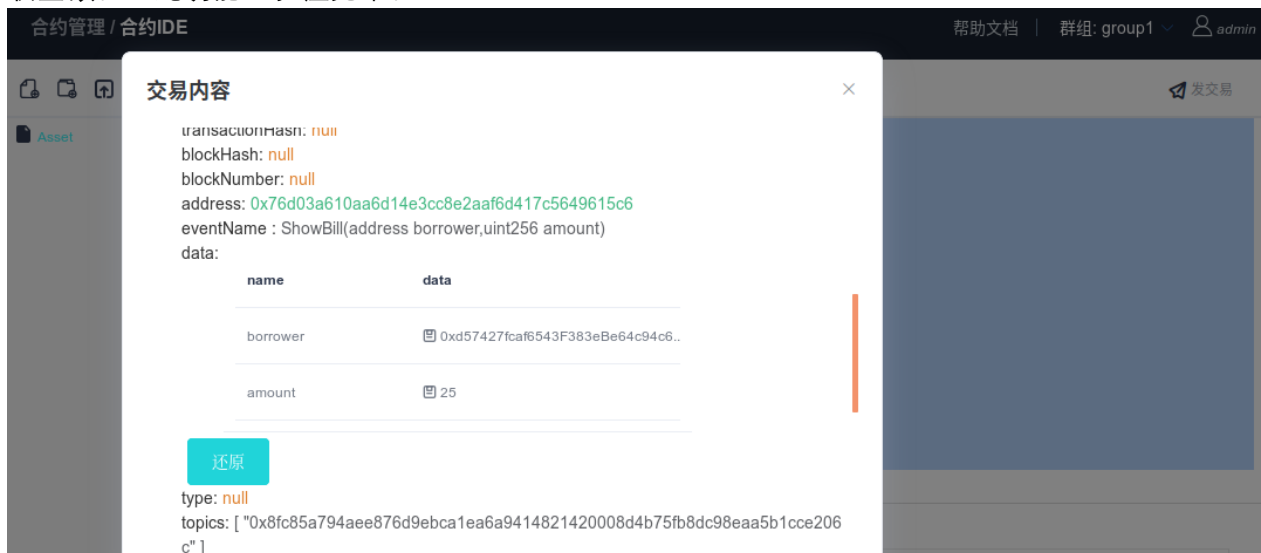
先用 set\_balances 设定用户 A、B 的初始资金为 100，C 的初始资金为 0



接下来是功能一的验证：调用 create\_bill 来创建 A 欠下 B 25 资金的应收账款（在实际应用中，即 A/B 的资金无任何变化，A 则通过创建这一应收账款，即可交换到 B 的物品）。



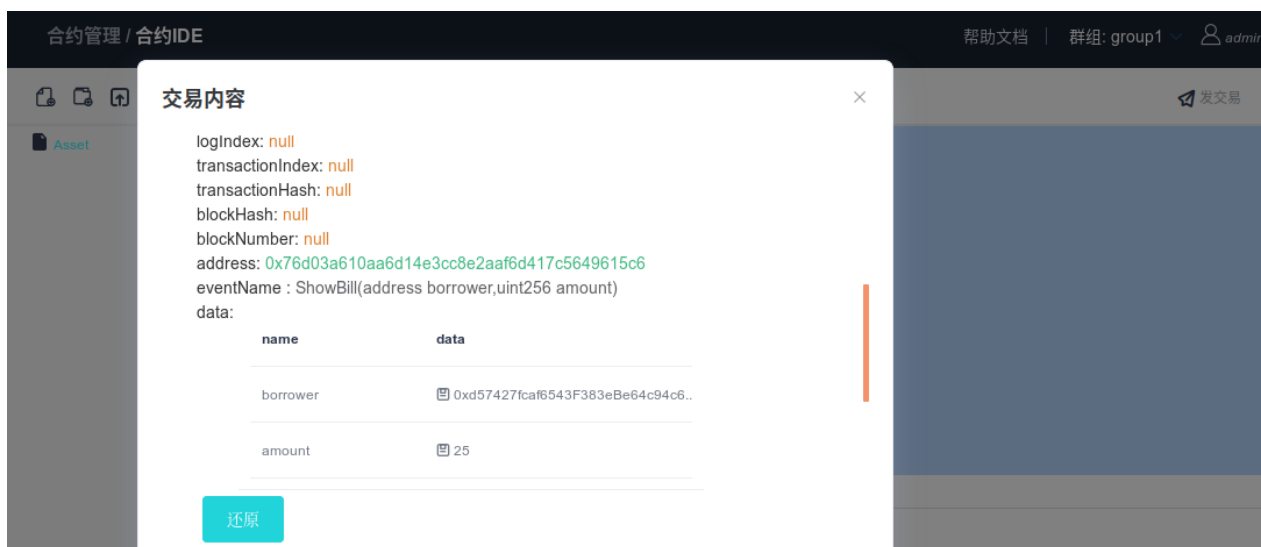
调用 `show_bill` 来查看 B 的应收账款库，可以看到 borrower 一栏为 A 的公钥，amount 为应收金额。至此功能一验证完毕。



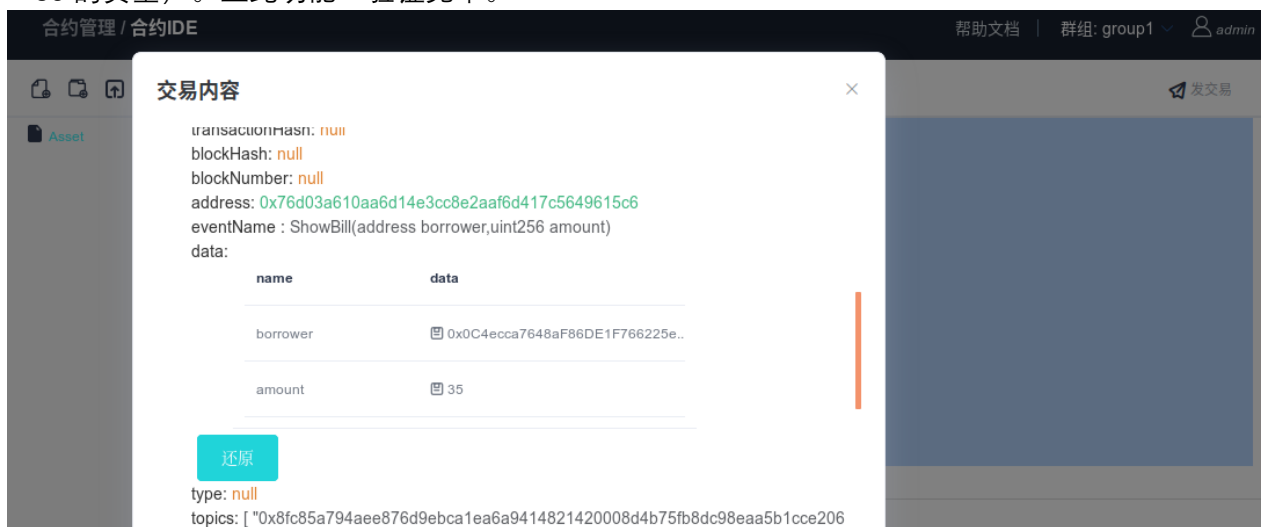
接下来是功能二的验证：调用 `borrow_funds`，让用户 B 向用户 C“借”60 资金（“借”的含义在功能一中有叙述）；由于用户 B 在前面已有了一条 25 资金的应收账款，用户 B 可通过转让这笔应收账款，从而减少 B 欠下 C 的应收账款。



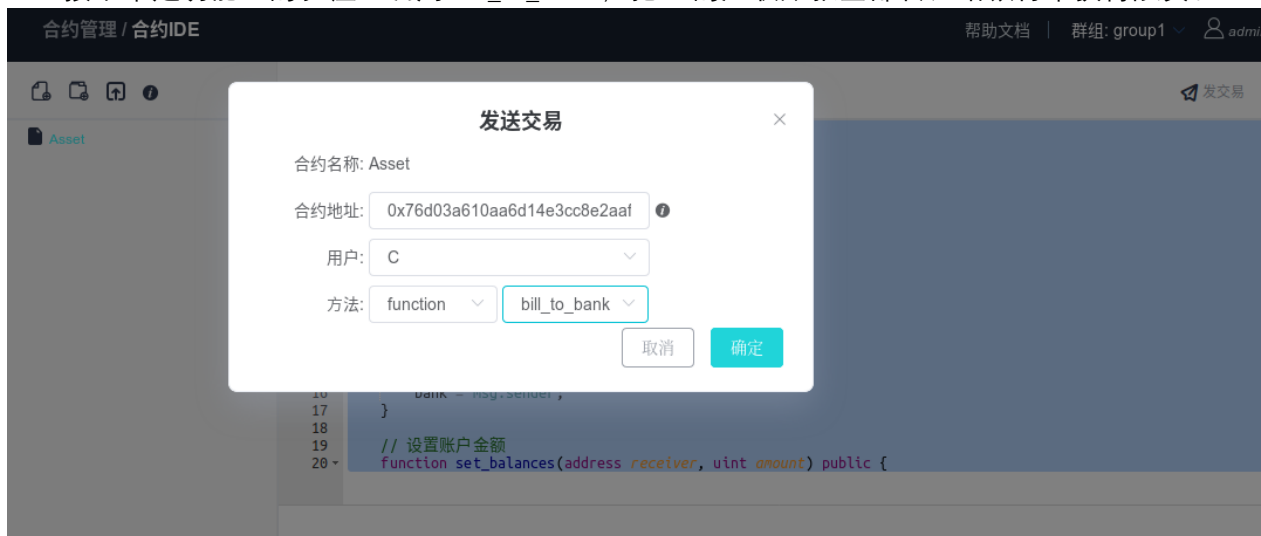
查看 C 的应收账款库，第一条记录如下所示，为还款人 A 的应收账款。



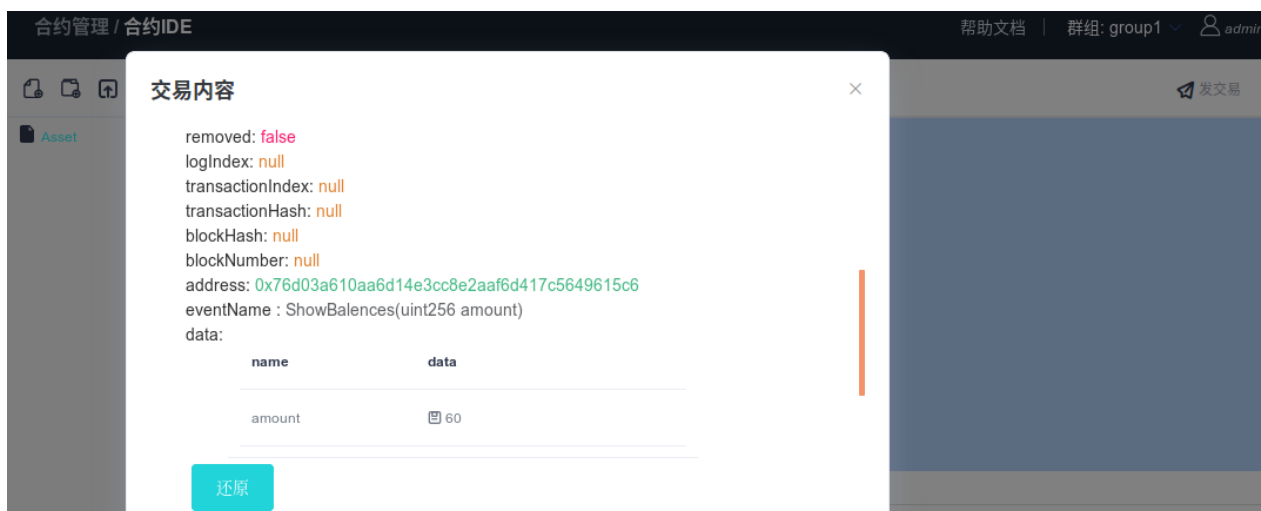
而第二条记录则为还款人 B 的应收账款（转让的应收账款抵消了一部分，因此 B 应还  $60 - 25 = 35$  的资金）。至此功能二验证完毕。



接下来是功能三的验证：调用 bill\_to\_bank，把 C 的应收账款全部转让给银行来获得融资。



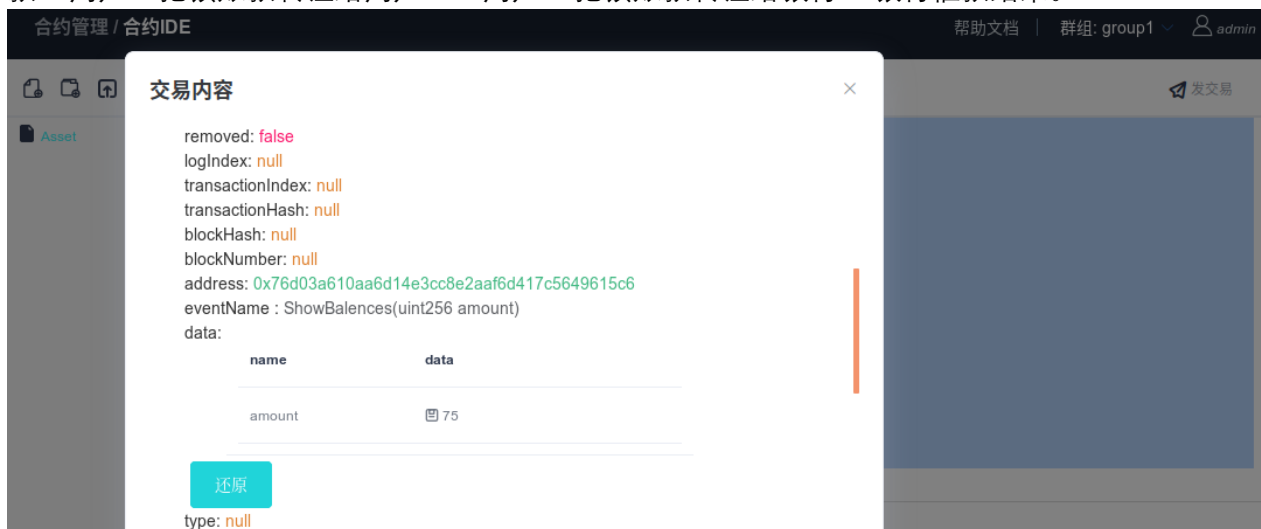
然后查看 C 的资金，如下图所示，C 获得了 60 资金的融资。至此功能三验证完毕。



最后是功能四的验证：调用 remove\_bill，银行对用户 A 催款。催款成功后移除该应收账款。

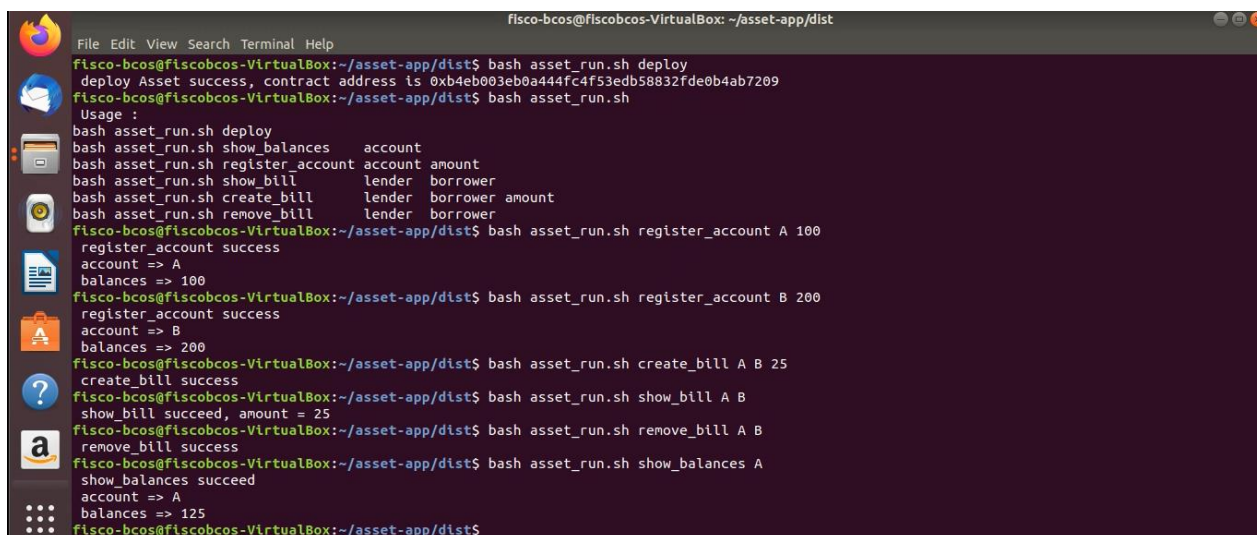


然后查看 A 的资金，如下图所示，A 的资金为 75，少的 25 资金经由从用户 B 创建的应收账款->用户 B 将该账款转让给用户 C->用户 C 将该账款转让给银行->银行催款结束。



#### 四、 界面展示

界面展示如下：



```
fisco-bcos@fiscobcos-VirtualBox: ~/asset-app/dist
File Edit View Search Terminal Help
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh deploy
deploy Asset success, contract address is 0xb4eb003eb0a444fc4f53edb58832fde0b4ab7209
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh
Usage :
bash asset_run.sh deploy
bash asset_run.sh show_balances account
bash asset_run.sh register_account account amount
bash asset_run.sh show_bill lender borrower
bash asset_run.sh create_bill lender borrower amount
bash asset_run.sh remove_bill lender borrower
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh register_account A 100
register_account success
account => A
balances => 100
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh register_account B 200
register_account success
account => B
balances => 200
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh create_bill A B 25
create_bill success
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh show_bill A B
show_bill succeed, amount = 25
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh remove_bill A B
remove_bill success
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh show_balances A
show_balances succeed
account => A
balances => 125
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$
```

由于本人能力有限，没能弄出前端，于是在终端下运行，界面如上所示。

## 五、心得体会

总的来说，完成度还不算太高（但本人之前也没做过前端…实在是太难了）

修改作业二的合约代码就花了不少的时间（sol 转 java 的工具是很方便，但有些数据结构用不了就很麻烦…再加上要同时修改 AssetClient 后才能发现问题…于是腰斩了不少的内容）

官网提供的工程项目还不错，在搭建后端时省了不少力气，但看“造好的轮子的说明书”也花了不少的时间…代码能力还需提高不少。

总之，这个大作业还是让我学到许多东西的。