

中山大学数据科学与计算机学院本科生实验报告

课程名称：区块链原理与技术

任课教师：郑子彬

年级：17 级

专业：软件工程

学号：17343081

姓名：陆俞因

电话：13360584660

Email: 13360584660@163.com

开始日期：2019/11/16

完成日期：2019/12/10

一、项目背景

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。

功能需求

1. 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。
2. 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。
3. 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。
4. 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

开发环境

- Linux Ubuntu16.04
- Fisco-Bcos 2.0 区块链底层
- eclipse IDE
- JAVA GUI
- Gradle 6.0
- Spring 4.0.6 框架

项目结构

```
1  .
2  |— bin
3  |— build.gradle
4  |— gradle
```

```

5 | └─ gradlew
6 | └─ gradlew.bat
7 | └─ settings.gradle
8 | └─ solidity
9 |   └─ SupplyChain.sol
10 | └─ src
11 |   └─ main
12 |     └─ java
13 |       └─ ChainApp
14 |         └─ Library.java
15 |           └─ Runner.java
16 |       └─ gui
17 |         └─ Mainwindow.java
18 |           └─ Startwindow.java
19 |       └─ web3j
20 |         └─ SupplyChain.java
21 |     └─ resources
22 |       └─ 0x25c90b627ba385104b4a0200dce0b69f562129c0.pem
23 |       └─ 0x25c90b627ba385104b4a0200dce0b69f562129c0.public.pem
24 |       └─ ...
25 |       └─ applicationContext.xml
26 |       └─ ca.crt
27 |       └─ log4j2.xml
28 |       └─ sdk.crt
29 |       └─ sdk.key
30 |   └─ test
31 |     └─ java
32 |       └─ ChainApp
33 |         └─ LibraryTest.java
34 |     └─ resources

```

`./bin/` 可执行文件路径

`./build.gradle` gradle 配置文件

`./solidity/SupplyChain.sol` 智能合约源文件

`./src/main/` JAVA 源文件路径

`./src/main/java/ChainApp/Runner.java` 应用入口

`./src/main/java/gui/` 应用 GUI 源文件路径

`./src/main/java/web3j/` 智能合约 JAVA 源文件

`./src/resources/` 应用 JAVA 源文件依赖文件目录

`./src/resources/applicationContext.xml` Spring 配置文件

存储设计

公司注册信息与交易回执存储在区块链，在智能合约中定义，数据结构如下：

公司信息数据结构

```

2      *   公司结构体:
3      *       name    公司名称
4      *       addr    公司账户地址
5      *       balance  公司账户余额
6      *       type_t   公司类型("company" 普通公司/ "bank" 银行/ "arbitrator" 仲裁
机构)
7      *       receipts_num 拥有回执数量
8      *       receipts 拥有回执序号的映射
9      */
10     struct Company {
11         string name;
12         address addr;
13         int256 balance;
14         string type_t;
15         uint receipts_num;
16         mapping (uint => uint256) receipts;
17     }

```

交易回执数据结构

```

1  /*
2      *   回执结构体:
3      *       from    付款公司名称
4      *       to      收款公司名称
5      *       value   款项大小
6      *       return_time 付款时限, 格式 yyMMdd
7      *       status  回执状态(0 无效/ 1 有效/ 2 已转移/ 3 逾期未付款/ 4 已付款)
8      *       used    已用于申请融资/融资回执
9      *       signer  签名账户地址(signer[0] 收款公司地址/ signer[1] 仲裁机构地址)
10     *       sign    签名(sign[0] 收款公司签名/ sign[1] 仲裁机构签名)
11     */
12     struct Receipt {
13         string from;
14         string to;
15         int256 value;
16         uint return_time;
17         int status;
18         bool used;
19         address[] signer;
20         bool[] sign;
21     }

```

存储数据结构

使用两个动态数组分别存储应用的所有注册公司信息 and 交易回执信息。

```

1     uint256 companies_num; // 注册公司数量
2     Company[] companies;  // 注册公司动态数组
3     uint256 receipts_num;  // 回执数量
4     Receipt[] receipts;    // 回执动态数组

```

数据流图

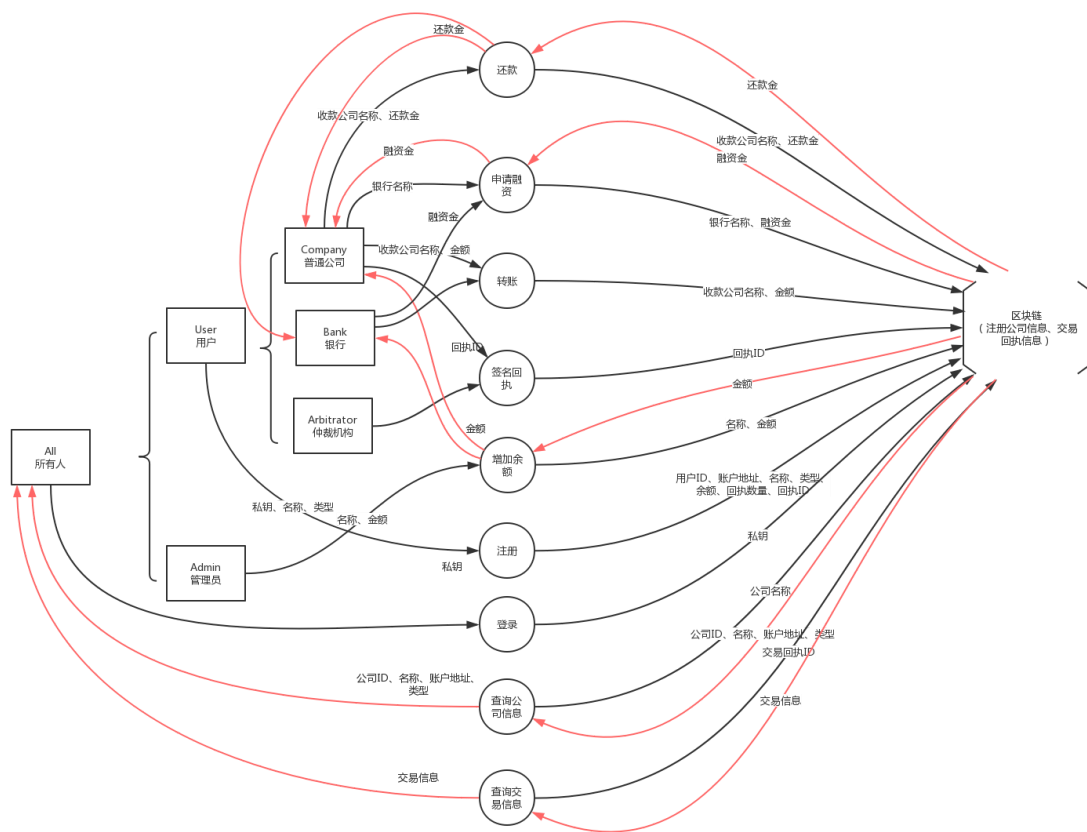


图 供应链金融应用数据流

角色介绍

每个外部账户对应唯一——一个角色，本应用为不同的角色提供不同的功能。

管理员(Admin)

部署智能合约的账户自动成为唯一的管理员，可以增加其它用户的余额。

Supply Chain App:Admin

Show & Sign Receipt

Show Company

Add Balance

Please enter the index of receipt:

OK

Receipt Index:
From:
To:
Value:
Return Time:
Status:
Used:
Signer:
Sign:

图 管理员界面

普通公司(Company)

供应链上的公司，可以发起交易、签名交易、转移应收账款、申请融资等。

The screenshot shows a web application window titled "Supply Chain App:Client". On the left is a sidebar menu with the following items: "Account" (highlighted), "Show & Sign Receipt", "Show Company", "Register Receipt", "Transfer Receipt", "Financing", "Return Debt", and "Transfer Balance". The main content area displays a form for a company account. The form has the following fields and values: "Name:" with value "company", "Type:" with value "company", "Balance:" with value "250", "Receipt Number:" with value "3", and "Receipt Index List:" with value "[0 1 3]". At the bottom of the form is a "Refresh" button.

Account	Name:	company
Show & Sign Receipt	Type:	company
Show Company	Balance:	250
Register Receipt	Receipt Number:	3
Transfer Receipt	Receipt Index List:	[0 1 3]
Financing		
Return Debt		
Transfer Balance		

Refresh

图 普通公司界面

银行(Bank)

主要功能是为供应链上的公司提供融资服务。

The screenshot shows the same "Supply Chain App:Client" window, but the sidebar menu has "Account" selected. The main content area displays a form for a bank account. The form has the following fields and values: "Name:" with value "bank", "Type:" with value "bank", "Balance:" with value "9950", "Receipt Number:" with value "1", and "Receipt Index List:" with value "[2]". At the bottom of the form is a "Refresh" button.

Account	Name:	bank
Show & Sign Receipt	Type:	bank
Show Company	Balance:	9950
Transfer Balance	Receipt Number:	1
	Receipt Index List:	[2]

Refresh

图 银行界面

仲裁机构(Arbitrator)

主要功能是为交易回执签名。

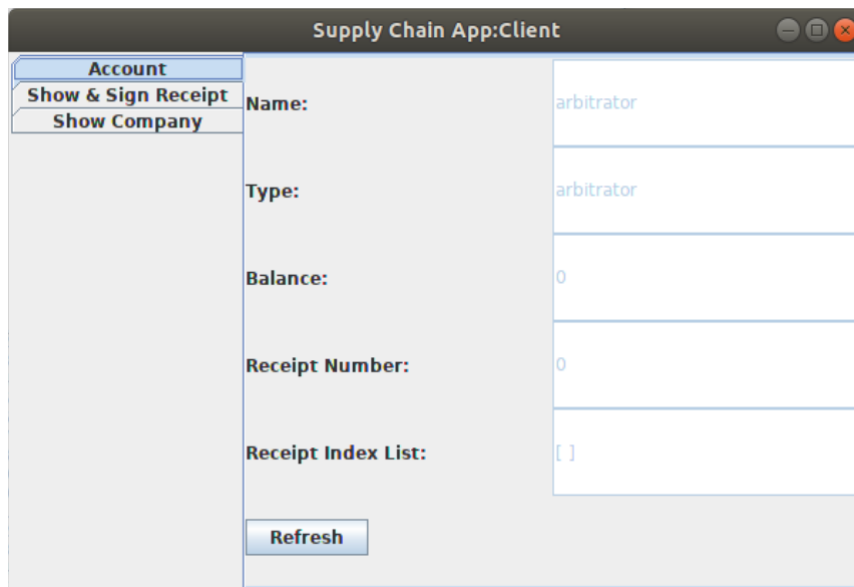


图 仲裁机构界面

核心功能介绍

1. 登录(附加功能)

每个应用账户对应区块链上一个有公私钥对的外部账户，通过向应用提供其私钥，登录账户。

创建外部账户

使用 Fisco-Bcos 提供的 `get_account.sh` 脚本，可以生成 PEM 格式或 PKCS12 格式的公私钥文件。此处选择使用 PEM 格式。

```
fisco-bcos@fiscobcos-VirtualBox: ~/fisco/console
File Edit View Search Terminal Help
fisco-bcos@fiscobcos-VirtualBox:~/fisco/console$ bash get_a
ccount.sh
[INFO] Account Address : 0xcaf3325bdb9c96b70dd904b5ea01cc
ff2716f443
[INFO] Private Key (pem) : accounts/0xcaf3325bdb9c96b70dd90
4b5ea01ccff2716f443.pem
[INFO] Public Key (pem) : accounts/0xcaf3325bdb9c96b70dd90
4b5ea01ccff2716f443.public.pem
```

图 创建外部账户

web3sdk 使用账户

将私钥文件 `.pem` 和公钥文件 `.public.pem` 导入工程的 `./src/main/resources` 文件夹下。

在 spring 配置文件 `applicationContext.xml` 中引入私钥文件。

```
1 <bean id="pemid" class="org.fisco.bcos.channel.client.PEMManager" init-
method="load" >
2 <property name="pemFile" value="classpath:address.pem" />
3 </bean>
```

通过 `PEMManager` 取得公私钥，并用对应账户连接区块链。

```
1 PEMManager pem = context.getBean("pemid", PEMManager.class);
2 EKeyPair pemKeyPair = pem.getEKeyPair();
3 credentials =
GenCredential.create(pemKeyPair.getPrivateKey().toString(16));
```

智能合约实现

`SupplyChain::getMyInfo` 方法根据传入的账户地址参数取得对应公司信息。若未注册，返回地址信息为 0。

```
1  /*
2     * 描述:
3     *      输出公司信息
4     */
5  function getMyInfo() public
6      returns(string memory out_name, address out_addr, int256 out_balance,
7              string memory out_type, uint out_r_num, uint256[] out_r) {
8      for(uint256 i = 0; i < companies_num; i ++) {
9          if(companies[i].addr == msg.sender) {
10             out_name = companies[i].name;
11             out_addr = companies[i].addr;
12             out_balance = companies[i].balance;
13             out_type = companies[i].type_t;
14             out_r_num = companies[i].receipts_num;
15             out_r = new uint256[](out_r_num);
16             for(uint j = 0; j < out_r_num; j ++) {
17                 out_r[j] = companies[i].receipts[j];
18             }
19             break;
20         }
21     }
```

JAVA 代码实现

凭借私钥，使用对应的外部账号连接区块链，并根据已部署的智能合约 `SupplyChain.sol` 的合约地址加载合约。JAVA 代码实现如下：

```
1  /*
2     * fisco-bcos
3     */
4  private ApplicationContext context;
5  private Service service;
6  private ChannelEthereumService channelEthereumService;
7  private Web3j myWeb3j;
8  private Credentials credentials;
9  private SupplyChain supplyChain;
10
11  private static String contractAddr = "...";
12
13  /*
14     * apply web3sdk
15     */
16  context = new
17  ClassPathXmlApplicationContext("classpath:applicationContext.xml");
18  service = context.getBean(Service.class);
19  service.run();
20  ChannelEthereumService channelEthereumService = new
21  ChannelEthereumService();
22  channelEthereumService.setChannelService(service);
23  channelEthereumService.setTimeout(10000);
```

```

22     myWeb3j = web3j.build(channelEthereumService,
    service.getGroupId());
23
24     BigInteger gasPrice = new BigInteger("300000000");
25     BigInteger gasLimit = new BigInteger("300000000");
26     /*
27      * load up external account with private key
28      */
29     PEMManager pem = context.getBean("pem"+pk_str, PEMManager.class);
30     ECKeypair pemKeyPair = pem.getECKeypair();
31     credentials =
    GenCredential.create(pemKeyPair.getPrivateKey().toString(16));
32     /*
33      * load deployed smart contract
34      */
35     supplyChain = SupplyChain.load(contractAddr, myWeb3j, credentials,
    new StaticGasProvider(gasPrice, gasLimit));

```

调用 `SupplyChain::getMyInfo` 方法取得公司信息。

[web3sdk 调用合约方法详情](#)

```

1 TransactionReceipt getMyInfoTx = supplyChain.getMyInfo().send();

```

调用 web3sdk 提供的 `TransactionDecoder` 类解析调用合约方法的返回值和事件，以下功能介绍时不再赘述。此处使用 `TransactionDecoder.decodeOutputReturnJson` 方法，取得 JSON 格式的输出值字符串，并调用 `fastJSON` 库解析输出。

[web3sdk 交易解析详情](#)

```

1 static String abi = "...";
2 static String bin = "";
3 private TransactionDecoder txDecoder = new TransactionDecoder(abi, bin);
4
5 String strJson = txDecoder.decodeOutputReturnJson(getMyInfoTx.getInput(),
    getMyInfoTx.getOutput());

```

2.注册(附加功能)

注册需要输入私钥、名称、公司类型，要求：

- 一个外部账户(即一个私钥)只能注册一个应用账户；
- 名称不能重复。

智能合约实现

`SupplyChain::registerCompany` 方法实现注册功能。

```

1  /*
2      * 描述：
3      *      注册公司
4      * 参数：
5      *      name    公司名称
6      *      addr    公司账户地址
7      *      type_t  公司类型

```



```

8      *   返回值:
9      *       >=0       公司的数组下标
10     *       -1        公司姓名或地址已被注册
11     *       -2        调用账户地址不是注册公司地址
12     */
13     function registerCompany(string memory name, address addr, string memory
type_t) public returns(uint256){
14         if(addr != msg.sender)
15             return (uint256)(-2);
16
17         for(uint256 i = 0;i < companies.length; i ++) {
18             if(utilCompareInternal(companies[i].name, name)
19                 || companies[i].addr == addr) {
20                 return (uint256)(-1);
21             }
22         }
23         // 余额初始化为 0
24         addCompany(name, addr, 0, type_t);
25
26         return companies_num-1;
27     }

```

JAVA 代码实现

调用 `SupplyChain::registerCompany` 方法。

```

1 TransactionReceipt tx;
2 myAddr = new BigInteger(credentials.getAddress().substring(2),16);
3 tx = supplyChain.registerCompany(name, credentials.getAddress(),
type_t).send();

```

3.发起应收账款交易(基本功能)

发起采购商品签发应收账款交易，生成回执单据，要求：

- 交易双方公司都已注册。

若成功，返回回执 ID。

智能合约实现

`registerReceipt` 方法。

```

1  /*
2      *   描述:
3      *       付款公司发起交易，生成应收账款回执
4      *   参数:
5      *       from       付款公司名称
6      *       to         收款公司名称
7      *       value      款项大小
8      *       return_time 付款时限
9      *   返回值:
10     *       >=0        回执的数组下标
11     *       -1         公司未注册
12     *       -2         调用者不是付款公司
13     */

```

```

14     function registerReceipt(string memory from, string memory to, uint256
value, uint return_time) public returns(uint256) {
15         uint256 from_idx = getCompanyIdx(from);
16         if(from_idx == (uint256)(-1))
17             return (uint256)(-1);
18
19         if(companies[from_idx].addr != msg.sender)
20             return (uint256)(-2);
21
22         uint256 to_idx = getCompanyIdx(to);
23         if(to_idx == (uint256)(-1))
24             return (uint256)(-1);
25
26         addReceipt(from_idx, to_idx, from, to, value, return_time);
27
28         return (receipts_num-1);
29     }

```

JAVA 代码实现

以 GUI 输入内容为参数，调用 `registerReceipt` 方法。

```

1 receiptTx = supplyChain.registerReceipt(rfrom_text.getText(),
rto_text.getText(), new BigInteger(rval_text.getText()), new
BigInteger(rreturn_time_text.getText())).send();

```

4. 查看回执信息(附加功能)

根据回执 ID 查询回执信息。

智能合约实现

`showReceipt` 方法。

```

1  /*
2      * 描述:
3      *      输出回执信息
4      * 参数:
5      *      r_idx    回执的数组下标
6      */
7      function showReceipt(uint256 r_idx) public
8          returns(uint256 out_idx, string memory out_from, string memory out_to,
int256 out_value, uint out_return_time, int out_status, bool out_used,
address[2] out_signer, bool[2] out_sign){
9          // emit showReceiptEvent(r_idx ,receipts[r_idx].from,
receipts[r_idx].to, receipts[r_idx].value, receipts[r_idx].return_time,
receipts[r_idx].status, receipts[r_idx].used);
10         out_idx = r_idx;
11         out_from = receipts[r_idx].from;
12         out_to = receipts[r_idx].to;
13         out_value = receipts[r_idx].value;
14         out_return_time = receipts[r_idx].return_time;
15         out_status = receipts[r_idx].status;
16         out_used = receipts[r_idx].used;
17         for(uint256 i = 0; i < 2; i ++){

```

```

18         out_signer[i] = receipts[r_idx].signer[i];
19         out_sign[i] = receipts[r_idx].sign[i];
20     }
21 }

```

JAVA 代码实现

调用 `showReceipt` 方法。

```

1 showReceiptTx = supplyChain.showReceipt(new
  BigInteger(ridx_text.getText())).send();

```

5. 回执签名(附加功能)

收款公司或仲裁机构对回执签名，只有在收款公司签名后，回执才有效，仲裁机构对回执签名后增加回执的可信度。要求：

- 签名账户是收款公司或仲裁机构。

智能合约实现

`signReceipt` 方法。

```

1  /*
2      *   描述:
3      *       收款公司/仲裁机构对回执签名
4      *   参数:
5      *       r_idx   回执的数组下标
6      *   返回值:
7      *       0       成功
8      *       -1      不是收款公司/仲裁机构, 签名失败
9      */
10 function signReceipt(uint256 r_idx) public returns(int) {
11     uint256 c_idx = getCompanyId(receipts[r_idx].to);
12     if(companies[c_idx].addr == msg.sender) {
13         receipts[r_idx].sign[0] = true;
14         // 收款公司签名, 回执生效
15         receipts[r_idx].status = 1;
16         return 0;
17     }
18
19     c_idx = getCompanyIdByAddr(msg.sender);
20     if(utilCompareInternal(companies[c_idx].type_t, "arbitrator")) {
21         receipts[r_idx].signer[1] = msg.sender;
22         receipts[r_idx].sign[1] = true;
23         return 0;
24     }
25     return -1;
26 }

```

JAVA 代码实现

调用 `signReceipt` 方法。

```
1 | signTx = supplyChain.signReceipt(new BigInteger(ridx_text.getText())).send();
```

6. 查看公司信息(附加功能)

根据公司名称查看公司信息。

智能合约实现

`getCompany` 方法。

```
1  /*
2      *   描述:
3      *       输出公司信息
4      *   参数:
5      *       c_name   公司名称
6      */
7      function getCompany(string memory c_name) public returns(uint256,
string, address, string){
8          uint256 c_idx = getCompanyId(c_name);
9          if(c_idx != (uint256)(-1)){
10             emit showCompanyEvent(companies[c_idx].name,
companies[c_idx].addr, companies[c_idx].balance, companies[c_idx].type_t,
companies[c_idx].receipts_num);
11             return (c_idx, companies[c_idx].name, companies[c_idx].addr,
companies[c_idx].type_t);
12         }
13     }
```

JAVA 代码实现

调用 `getCompany` 方法。

```
1 | companyTx = supplyChain.getCompany(cname_text.getText()).send();
```

7. 转移应收账款(基本功能)

转移指定金额的应收账款，要求：

- 交易双方公司都已注册；
- 付款公司能够转移的应收账款的金额大于等于指定金额；
- 调用账户地址与付款公司地址一致。

智能合约实现

`transferReceipt` 方法。

```
1  /*
2      *   描述:
3      *       转移应收账款
4      *   参数:
5      *       from   付款公司名称
6      *       to     收款公司名称
7      *       value  款项大小
```

```

8      * 返回值:
9      *      >=0      成功
10     *      -1      公司未注册
11     *      -2      付款公司应收账款的金额不足
12     *      -3      调用者不是付款公司
13     */
14     function transferReceipt(string memory from, string memory to, int256
value) public returns(uint256){
15         uint256 from_idx = getCompanyIdx(from);
16         if(from_idx == (uint256)(-1))
17             return (uint256)(-1);
18
19         if(companies[from_idx].addr != msg.sender)
20             return (uint256)(-3);
21
22         uint256 r_idx = (uint256)(-1);
23         for(uint256 i = 0; i < companies[from_idx].receipts_num; i ++){
24             uint256 idx = companies[from_idx].receipts[i];
25             // 付款公司是应收账款的收款人, 回执有效且金额足够
26             if(utilCompareInternal(receipts[idx].to, from) &&
receipts[idx].status == 1
27                 && receipts[idx].value >= value){
28                 r_idx = idx;
29                 break;
30             }
31         }
32         if(r_idx == (uint256)(-1))
33             return (uint256)(-2);
34
35         uint256 to_idx = getCompanyIdx(to);
36         if(to_idx == (uint256)(-1))
37             return (uint256)(-1);
38
39         if(receipts[r_idx].value == value){
40             receipts[r_idx].status = 2;
41         }
42         else {
43             receipts[r_idx].value -= value;
44         }
45         emit showReceiptEvent(r_idx ,receipts[r_idx].from,
receipts[r_idx].to, receipts[r_idx].value, receipts[r_idx].return_time,
receipts[r_idx].status, receipts[r_idx].used);
46         uint256 f_idx = getCompanyIdx(receipts[r_idx].from);
47         addReceipt(f_idx, to_idx, receipts[r_idx].from, to, value,
receipts[r_idx].return_time);
48
49         return (receipts_num-1);
50     }

```

JAVA 代码实现

调用 `transferReceipt` 方法。

```

1  transferTx = supplyChain.transferReceipt(tfrom_text.getText(),
    tto_text.getText(), new BigInteger(tval_text.getText()));

```

8. 申请融资(基本功能)

公司凭借应收账款向银行申请融资，要求：

- 申请公司已注册；
- 融资对象是银行，不能是普通公司；
- 申请公司拥有可用于融资凭证的应收账款。

智能合约实现

`applyFinancing` 方法。

```
1  /*
2      *   描述：
3      *       根据应收账款向银行申请融资
4      *   参数：
5      *       from    申请公司名称
6      *       to      银行名称
7      *   返回值：
8      *       参数 1:
9      *           0      成功
10      *          -1     公司未注册
11      *          -2     接收申请者不是银行
12      *          -3     调用者不是申请公司
13      *          -4     申请公司不拥有可融资的回执
14      *       参数 2:
15      *           0      融资失败
16      *          >0     融资金额
17      */
18  function applyFinancing(string memory from, string memory to) public
returns(int, int256){
19      int ret = -4;
20      int val = 0;
21
22      uint256 from_idx = getCompanyIdx(from);
23      if(from_idx == (uint256)(-1))
24          return (-1, val);
25
26      if(companies[from_idx].addr != msg.sender)
27          return (-3, val);
28
29      uint256 to_idx = getCompanyIdx(to);
30      if(to_idx == (uint256)(-1))
31          return (-1, val);
32
33      if(!utilCompareInternal(companies[to_idx].type_t, "bank"))
34          return (-2, val);
35
36      uint256 r_num = companies[from_idx].receipts_num;
37      for(uint256 i = 0; i < r_num; i ++){
38          uint256 r_idx = companies[from_idx].receipts[i];
39          // 申请公司是应收账款的收款人，回执有效，未曾用于申请融资且不是融资回执
40          if(utilCompareInternal(receipts[r_idx].to, from) &&
receipts[r_idx].status == 1
41              && receipts[r_idx].used == false){
42              addReceipt(from_idx, to_idx, from, to,
receipts[r_idx].value, receipts[r_idx].return_time);
43              receipts[receipts_num-1].used = true;
```

```

44         receipts[receipts_num-1].status = 1;
45
46         receipts[r_idx].used = true;
47
48         transferBalancePrivate(to, from, receipts[r_idx].value);
49
50         ret = 0;
51         val += receipts[r_idx].value;
52
53         emit showCompanyEvent(from, companies[from_idx].addr,
companies[from_idx].balance,
companies[from_idx].type_t, companies[from_idx].receipts_num);
54     }
55 }
56
57     return (ret, val);
58 }

```

JAVA 代码实现

调用 `applyFinancing` 方法。

```

1  financingTx = supplyChain.applyFinancing(ffrom_text.getText(),
    fto_text.getText()).send();

```

9. 还款(附加功能)

付款公司支付应收账款，要求：

- 付收款公司都已注册；
- 还款公司余额足够；
- 有付收款公司符合申请，有效且未还款的应收账款。

智能合约实现

`returnDebtCompany` 方法。

```

1  /*
2      *   描述：
3      *       付款公司支付应收账款
4      *   参数：
5      *       from   付款公司名称
6      *       to     收款公司名称
7      *   返回值：
8      *       参数 1:
9      *           0      成功
10     *           -1     公司未注册
11     *           -2     还款公司余额不足
12     *           -3     没有符合应还账款
13     *       参数 2:
14     *           0      还款失败
15     *           >0     还款金额
16     */
17     function returnDebtCompany(string memory from, string memory to) public
returns(int, int256) {

```

```

18         int ret = -3;
19         uint256 val = 0;
20         uint256 from_idx = getCompanyIdx(from);
21         if(from_idx == (uint256)(-1))
22             return (-1, val);
23
24         uint256 to_idx = getCompanyIdx(to);
25         if(to_idx == (uint256)(-1))
26             return (-1, val);
27
28         for(uint256 i = 0; i < companies[from_idx].receipts_num; i ++) {
29             uint256 r_idx = companies[from_idx].receipts[i];
30             if(utilCompareInternal(receipts[r_idx].from, from)
31                 && utilCompareInternal(receipts[r_idx].to, to)
32                 && (receipts[r_idx].status == 1 || receipts[r_idx].status
33 == 3)) {
34                 int ret_t = transferBalancePrivate(from, to,
35 receipts[r_idx].value);
36                 if(ret_t == -2)
37                     return (ret_t, val);
38                 ret = 0;
39                 val += receipts[r_idx].value;
40                 receipts[r_idx].status = 4;
41                 emit showReceiptEvent(r_idx ,receipts[r_idx].from,
42 receipts[r_idx].to, receipts[r_idx].value, receipts[r_idx].return_time,
43 receipts[r_idx].status, receipts[r_idx].used);
44             }
45         }
46
47         return (ret, val);
48     }

```

JAVA 代码实现

调用 `returnDebtCompany` 方法。

```

1 debtTx = supplyChain.returnDebtCompany(dfrom_text.getText(),
    dto_text.getText()).send();

```

10. 转账(附加功能)

简单的转账功能，要求：

- 交易双方公司已注册；
- 付款公司余额足够。

智能合约实现

`transferBalance` 方法。

```

1  /*
2      *   描述:
3      *       转账
4      *   参数:
5      *       from       付款公司

```



```

6      *      to      收款公司
7      *      value   款项大小
8      *      返回值:
9      *      0       成功
10     *      -1      公司不存在
11     *      -2      付款公司余额不足
12     *      -3      调用者不是付款公司
13     */
14     function transferBalance(string memory from, string memory to, int256
value) public returns(int) {
15         uint256 from_idx = getCompanyIdx(from);
16         if(from_idx == (uint256)(-1))
17             return -1;
18
19         if(companies[from_idx].addr != msg.sender)
20             return -3;
21
22         uint256 to_idx = getCompanyIdx(to);
23         if(to_idx == (uint256)(-1))
24             return -1;
25
26         if(companies[from_idx].balance < value)
27             return -2;
28
29         companies[from_idx].balance -= value;
30         companies[to_idx].balance += value;
31
32         return 0;
33     }

```

JAVA 代码实现

调用 `transferBalance` 方法。

```

1  tbTx = supplyChain.transferBalance(tbfrom_text.getText(),
    tbto_text.getText(), new BigInteger(tbval_text.getText())).send();

```

11. 增加余额(附加功能)

管理员账户修改注册公司的余额，要求：

- 公司已注册；
- 调用者为管理员。

智能合约实现

`addBalance` 方法。

```

1  /*
2      *      描述:
3      *      管理员增加公司的余额
4      *      参数:
5      *      to      收款公司
6      *      value   款项大小
7      *      返回值:

```

```

8      *      0      成功
9      *      -1     调用者不是管理员
10     *      -2     公司不存在
11     */
12     function addBalance(string memory to, int256 value) public returns(int)
13     {
14         if(msg.sender != admin_addr)
15             return -1;
16
17         uint256 to_idx = getCompanyIdx(to);
18         if(to_idx == (uint256)(-1))
19             return -2;
20
21         companies[to_idx].balance += value;
22         return 0;
23     }

```

JAVA 代码实现

调用 `addBalance` 方法。

```

1  addbTx = supplyChain.addBalance(acname_text.getText(), new
   BigInteger(aval_text.getText())).send();

```

三、功能测试

1. 登录(附加功能)

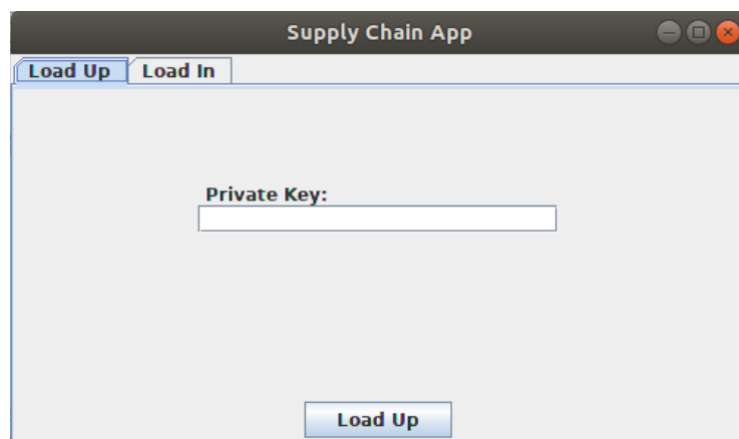


图 登录界面

异常处理

(1) 账户未注册

登录失败，提示必须先注册。

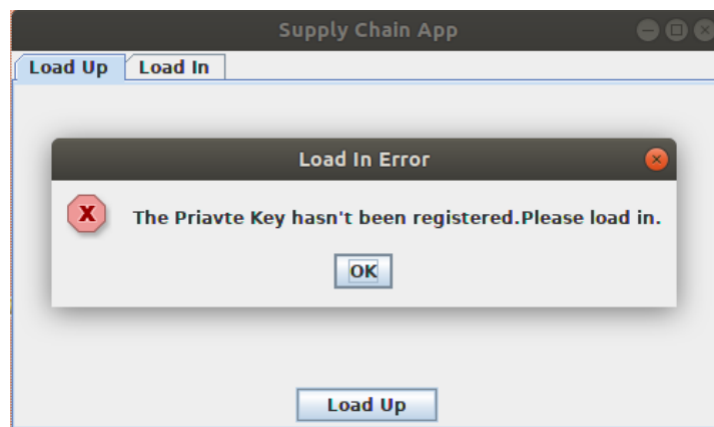


图 账户未注册，登陆失败

2.注册(附加功能)

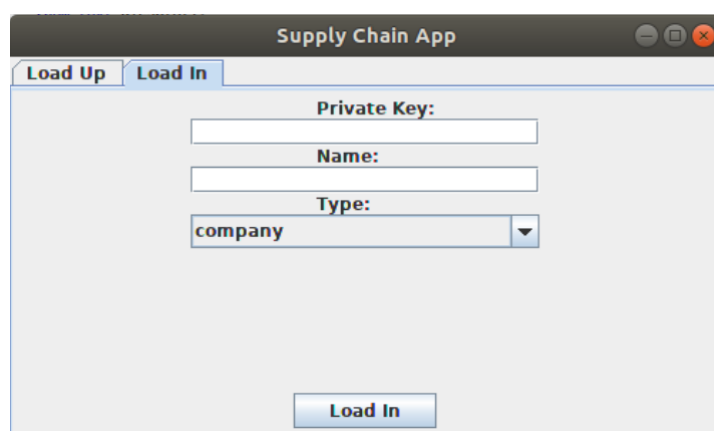


图 注册界面

异常处理

(1) 私钥已注册

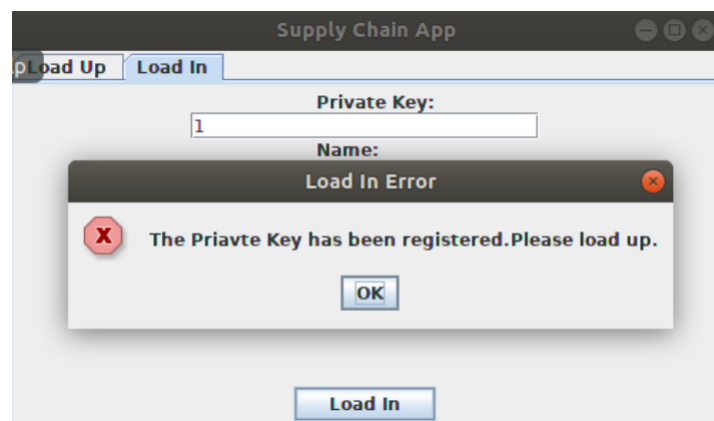


图 私钥已注册，注册失败界面

(2) 名称已注册

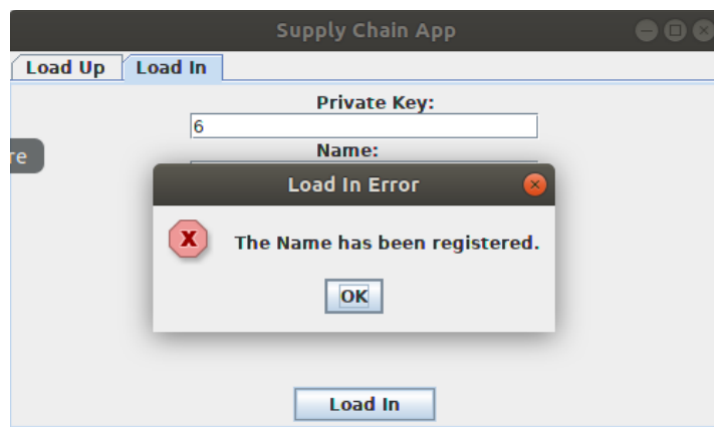


图 名称已注册，注册失败界面

3.发起应收账款交易(基本功能)

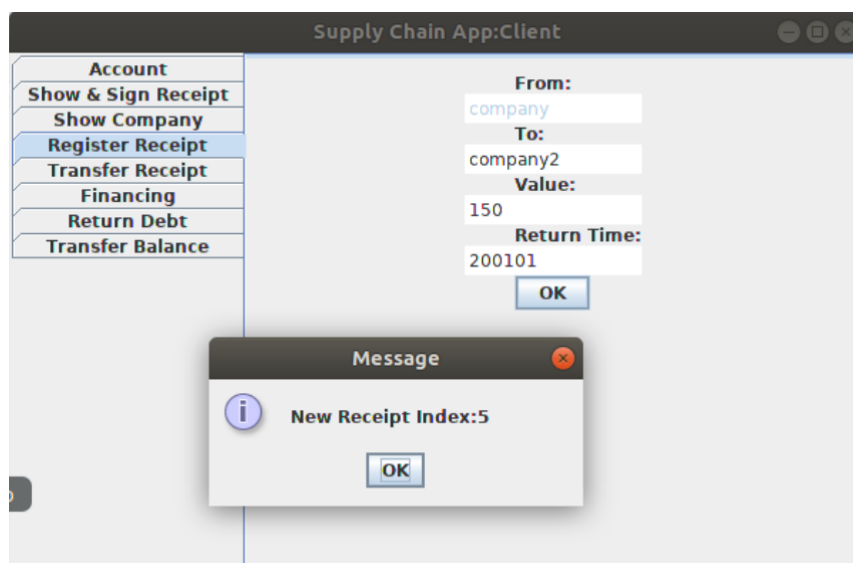


图 发起应收账款交易界面

异常处理

(1) 收款公司未注册

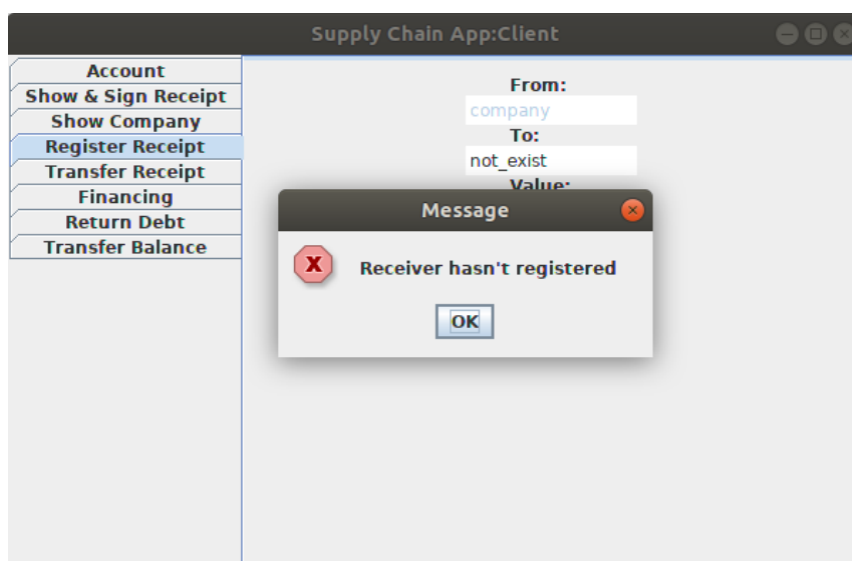


图 收款公司未注册，发起应收账款交易失败

4. 查看回执信息(附加功能)

Supply Chain App:Client

Account	
Show & Sign Receipt	Please enter the index of receipt: 0 OK
Show Company	
Register Receipt	
Transfer Receipt	
Financing	
Return Debt	
Transfer Balance	

Receipt Index:	0
From:	company
To:	company2
Value:	50
Return Time:	200101
Status:	4
Used:	true
Signer:	00000000000000000000000000000000]
Sign:	[true, false]
Sign	

图 查看回执界面

5. 回执签名(附加功能)

Supply Chain App:Client

Account	
Show & Sign Receipt	Please enter the index of receipt: 5 OK
Show Company	
Register Receipt	
Transfer Receipt	
Financing	
Return Debt	
Transfer Balance	

Message

Sign Success!

OK

Receipt Index:	5
From:	company
To:	company2
Value:	150
Return Time:	200101
Status:	0
Used:	false
Signer:	00000000000000000000000000000000]
Sign:	[false, false]
Sign	

图 回执签名界面

Supply Chain App:Client

Account	
Show & Sign Receipt	Please enter the index of receipt: 5 OK
Show Company	
Register Receipt	
Transfer Receipt	
Financing	
Return Debt	
Transfer Balance	

Receipt Index:	5
From:	company
To:	company2
Value:	150
Return Time:	200101
Status:	1
Used:	false
Signer:	00000000000000000000000000000000]
Sign:	[true, false]
Sign	

图 回执签名后，sign 的值变为 true

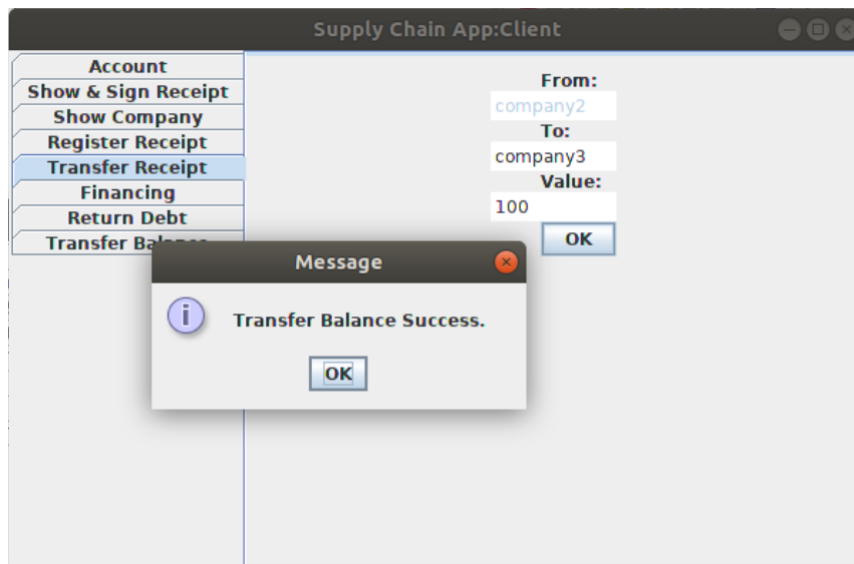


图 转移应收账款界面

异常处理

(1) 收款方未注册

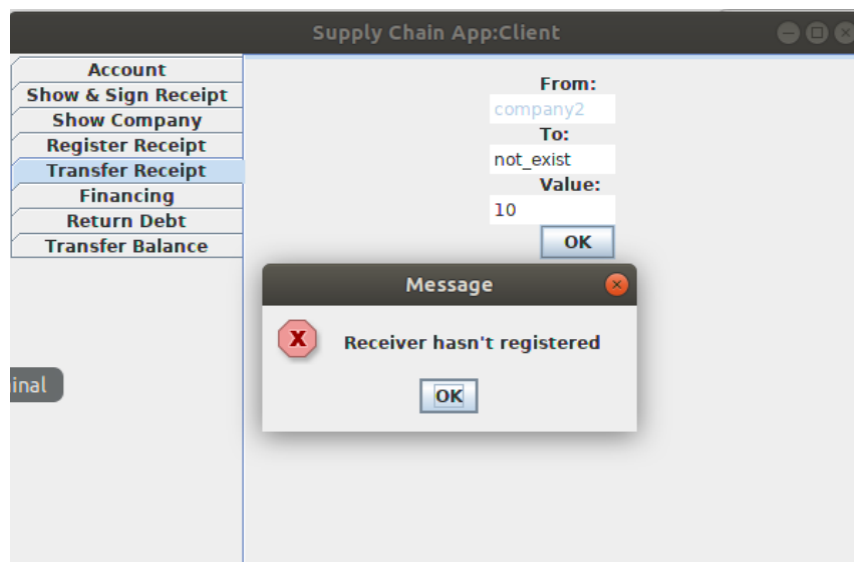


图 收款方未注册，转移应收账款失败

(2) 付款方金额不足

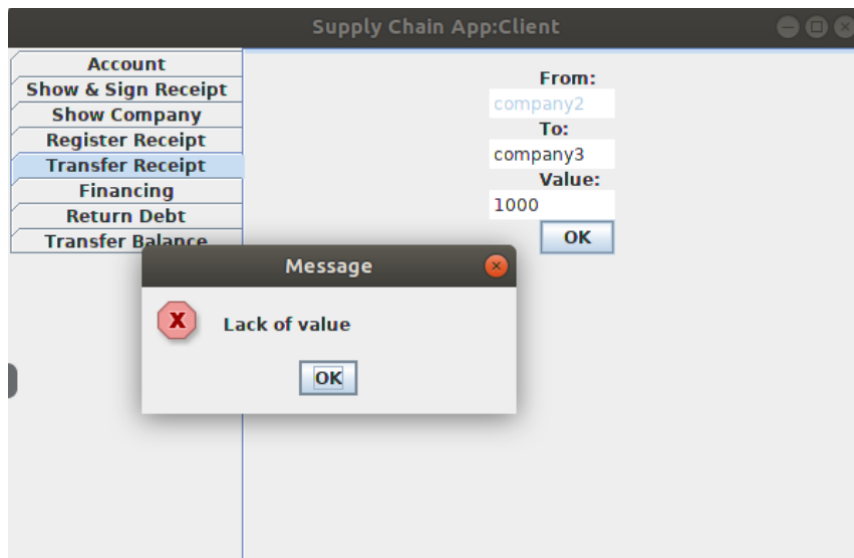


图 付款方应收账款金额不足，转移应收账款失败

8. 申请融资(基本功能)

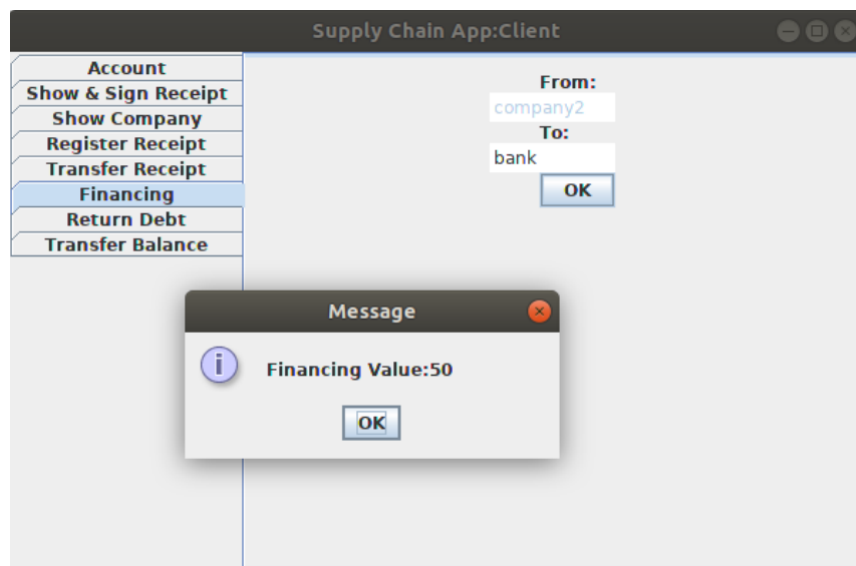


图 申请融资界面

异常处理

(1) 接收申请者未注册

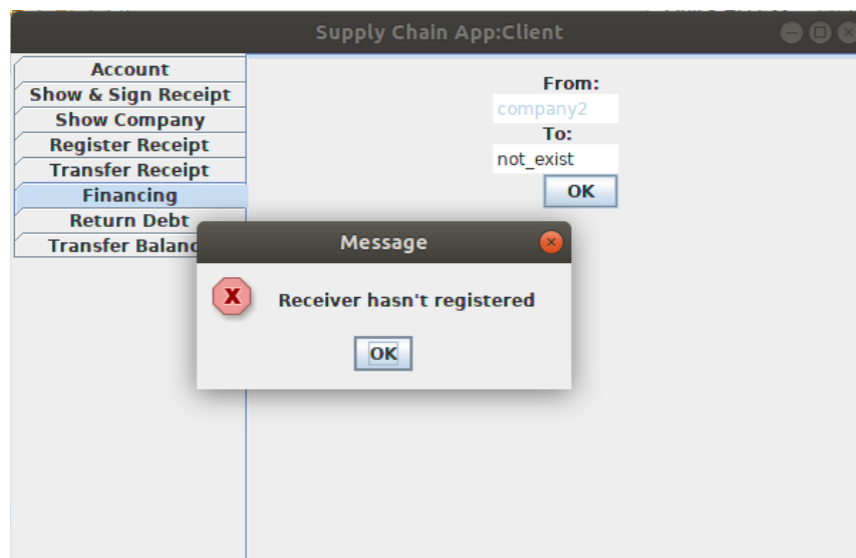


图 接收申请者未注册，融资失败

(2) 接收申请者不是银行

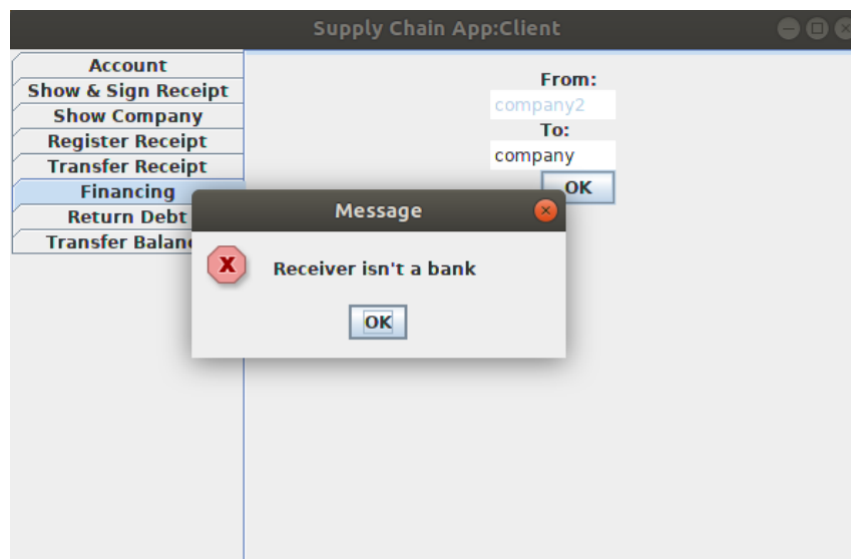


图 接收申请者不是银行，融资失败

(3) 申请者不拥有应收账款

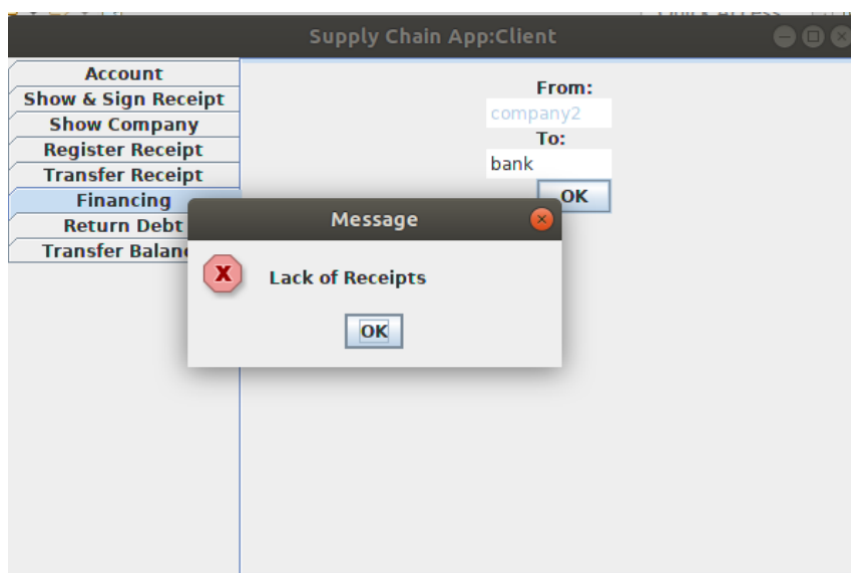


图 申请者不拥有应收账款，融资失败

9. 还款(附加功能)

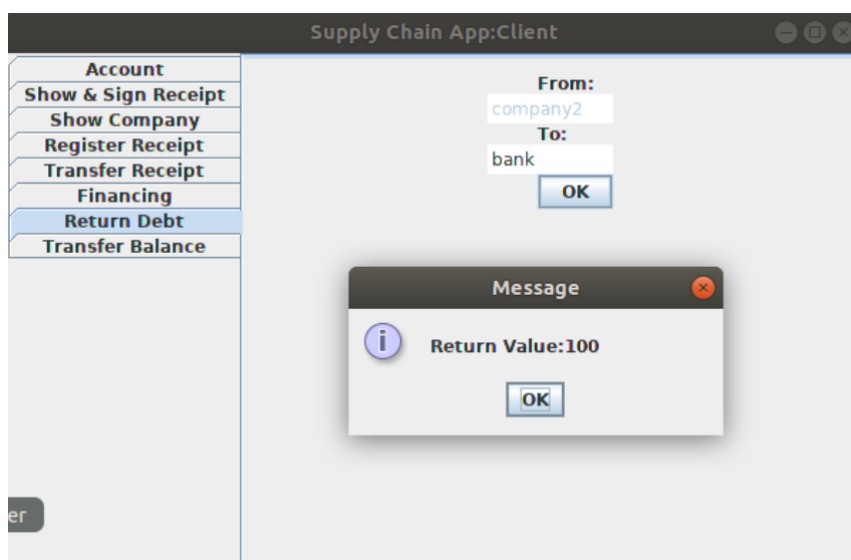


图 还款界面

异常处理

(1) 收款方未注册

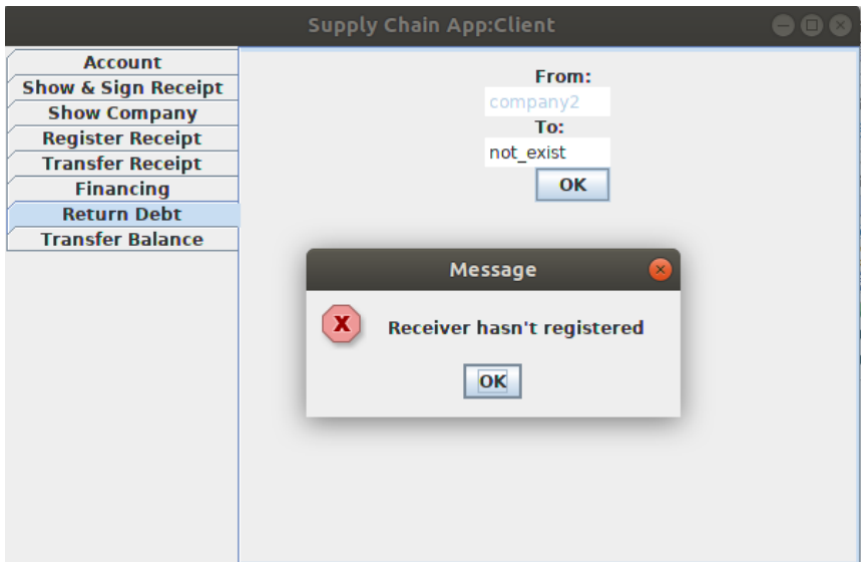


图 收款方未注册，还款失败

(2) 还款方余额不足

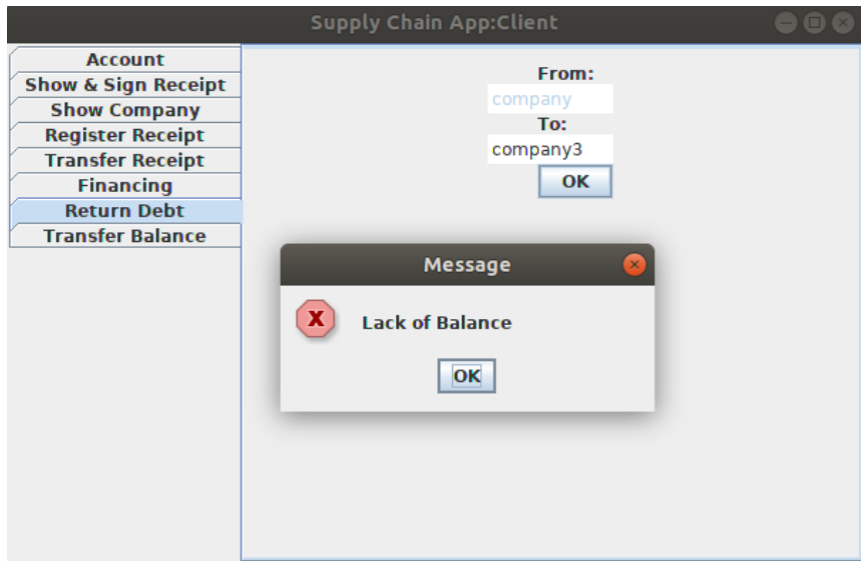


图 还款方余额不足，还款失败

(3) 没有待还款的账款

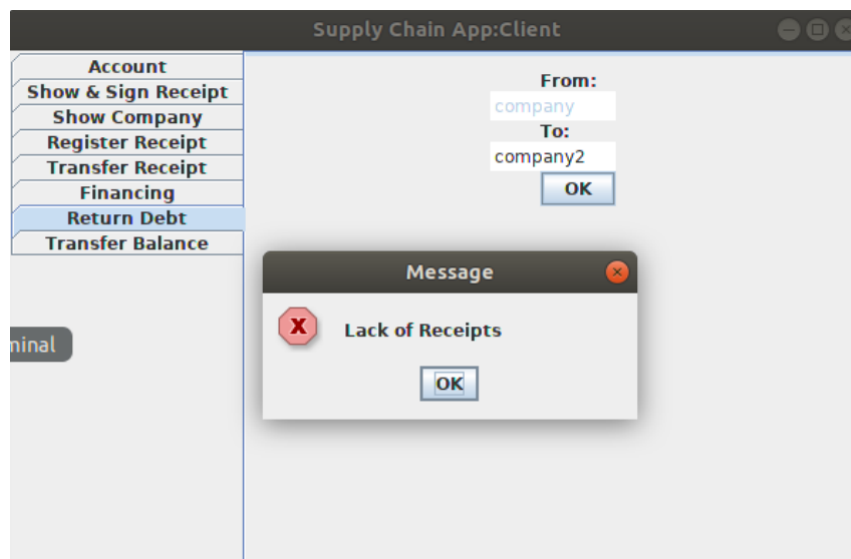
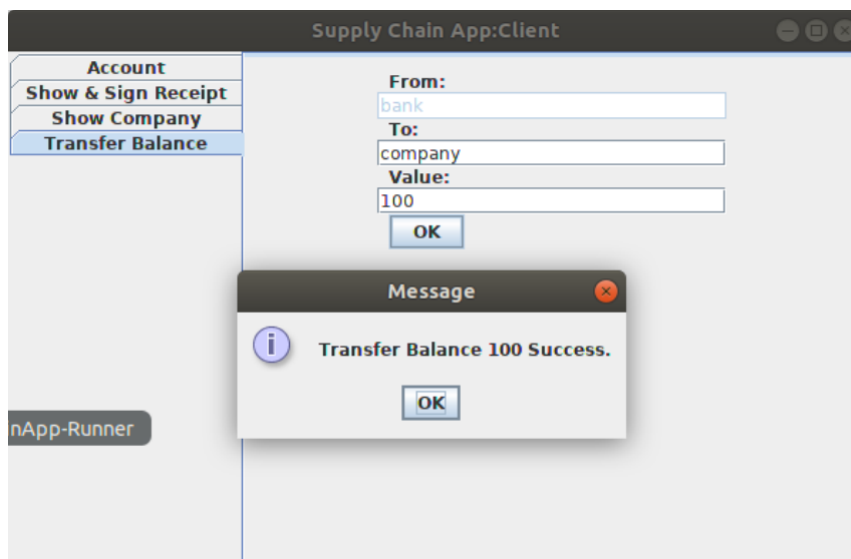


图 接没有待还款的账款，还款失败

10. 转账(附加功能)



转账界面

异常处理

(1) 收款方未注册

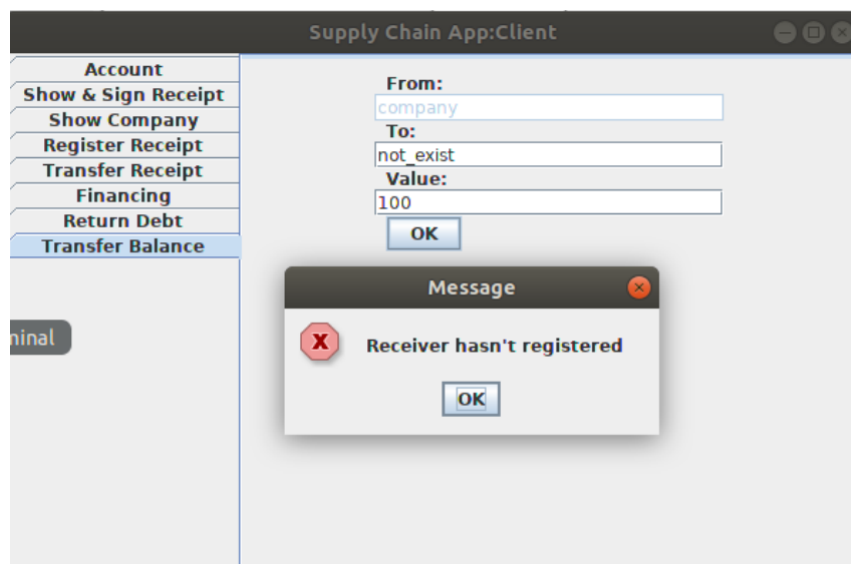


图 收款方未注册，转账失败

(2) 付款方余额不足

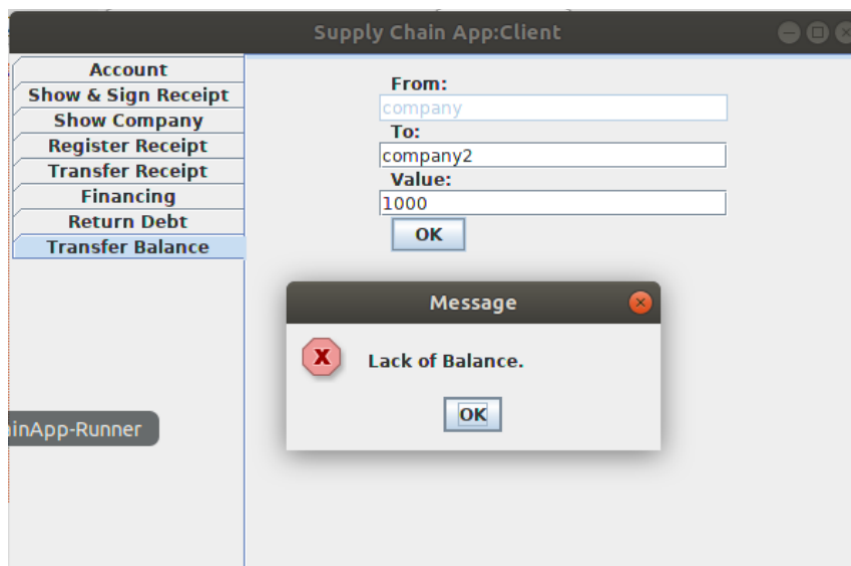


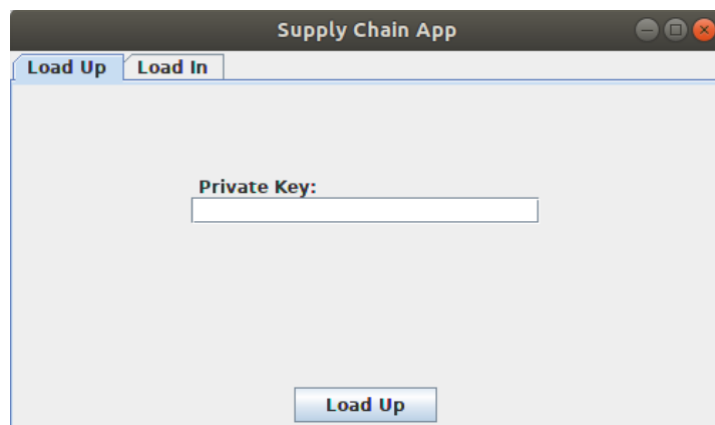
图 付款方余额不足，还款失败

四、界面展示

开始界面

上端选项卡切换登录界面与注册界面。

登录界面



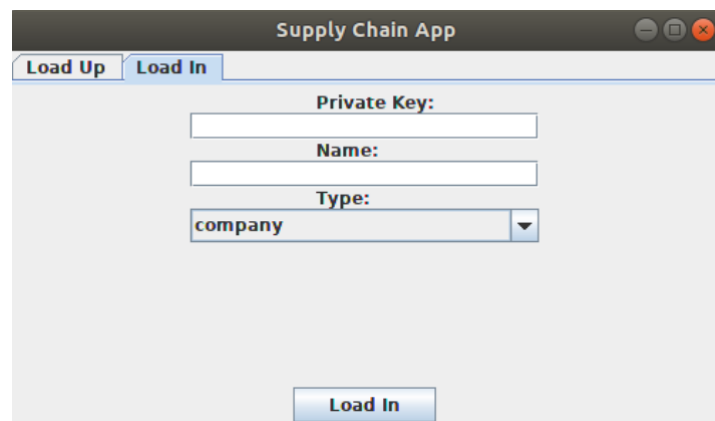
Supply Chain App

Load Up Load In

Private Key:

Load Up

注册界面



Supply Chain App

Load Up Load In

Private Key:

Name:

Type:

company

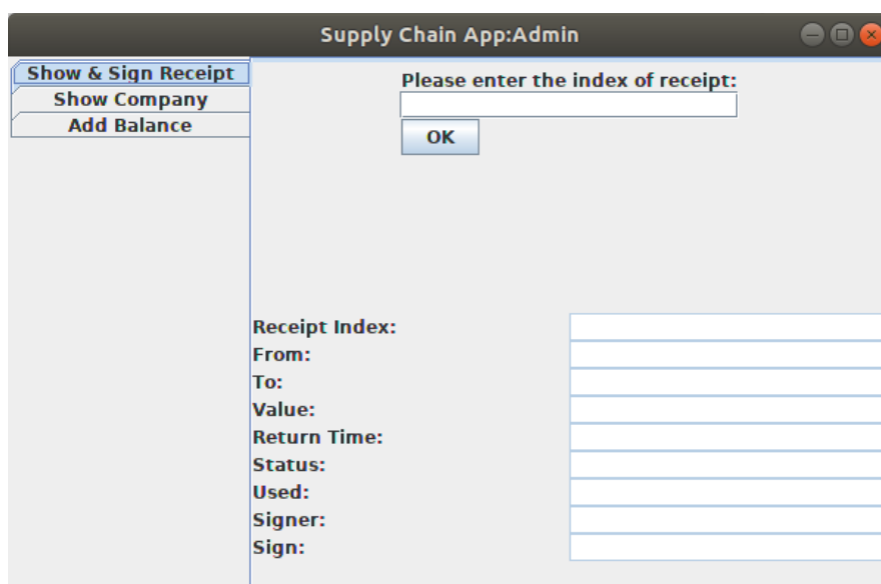
Load In

主界面

左端选项卡切换功能，右端为功能的输入和输出面板。

按照角色

管理员界面



Supply Chain App:Admin

Show & Sign Receipt

Show Company

Add Balance

Please enter the index of receipt:

OK

Receipt Index:

From:

To:

Value:

Return Time:

Status:

Used:

Signer:

Sign:

普通公司界面

Supply Chain App:Client

Account	Name:	company
Show & Sign Receipt	Type:	company
Show Company	Balance:	250
Register Receipt	Receipt Number:	3
Transfer Receipt	Receipt Index List:	[0 1 3]
Financing		
Return Debt		
Transfer Balance		
	Refresh	

银行界面

Supply Chain App:Client

Account	Name:	bank
Show & Sign Receipt	Type:	bank
Show Company	Balance:	9950
Transfer Balance	Receipt Number:	1
	Receipt Index List:	[2]
	Refresh	

仲裁机构界面

Supply Chain App:Client

Account	Name:	arbitrator
Show & Sign Receipt	Type:	arbitrator
Show Company	Balance:	0
	Receipt Number:	0
	Receipt Index List:	[]
	Refresh	

按照功能

登录账户信息界面(Account)

Refresh 按键刷新账户信息。

Supply Chain App:Client

Account

Show & Sign Receipt

Show Company

Name: arbitrator

Type: arbitrator

Balance: 0

Receipt Number: 0

Receipt Index List: []

Refresh

回执信息界面(Show & Sign Receipt)

输入需要查询或签名的回执 ID 后，点击 **OK** 按钮进行查询，查询结果显示在下方；点击 **sign** 按钮签名回执。

The screenshot shows the 'Supply Chain App:Client' window. On the left, a vertical menu lists several options: 'Account', 'Show & Sign Receipt' (highlighted), 'Show Company', 'Register Receipt', 'Transfer Receipt', 'Financing', 'Return Debt', and 'Transfer Balance'. The main window area contains a form titled 'Please enter the index of receipt:' with a text input field and an 'OK' button. Below this, there are labels for various receipt details: 'Receipt Index:', 'From:', 'To:', 'Value:', 'Return Time:', 'Status:', 'Used:', 'Signer:', and 'Sign:'. Each label is followed by a corresponding text input field. At the bottom of the form, there is a 'Sign' button.

公司信息界面(Show Company)

输入需要查询的公司名称后，点击 **OK** 按键进行查询，查询结果显示在下方。

Supply Chain App:Client

Account	
Show & Sign Receipt	
Show Company	
Register Receipt	
Transfer Receipt	
Financing	
Return Debt	
Transfer Balance	

Please enter the name of company:

Company Index:

Name:

Address:

Type:

生成应收账款界面(Register Receipt)

Supply Chain App:Client

Account	
Show & Sign Receipt	
Show Company	
Register Receipt	
Transfer Receipt	
Financing	
Return Debt	
Transfer Balance	

From:

To:

Value:

Return Time:

转移应收账款界面(Transfer Receipt)

Supply Chain App:Client

Account	
Show & Sign Receipt	
Show Company	
Register Receipt	
Transfer Receipt	
Financing	
Return Debt	
Transfer Balance	

From:

To:

Value:

申请融资界面(Financing)

Supply Chain App:Client

Account
Show & Sign Receipt
Show Company
Register Receipt
Transfer Receipt
Financing
Return Debt
Transfer Balance

From:
company

To:

OK

还款界面(Return Debt)

Supply Chain App:Client

Account
Show & Sign Receipt
Show Company
Register Receipt
Transfer Receipt
Financing
Return Debt
Transfer Balance

From:
company

To:

OK

p-Runner

转账界面(Transfer Balance)

Supply Chain App:Client

Account
Show & Sign Receipt
Show Company
Register Receipt
Transfer Receipt
Financing
Return Debt
Transfer Balance

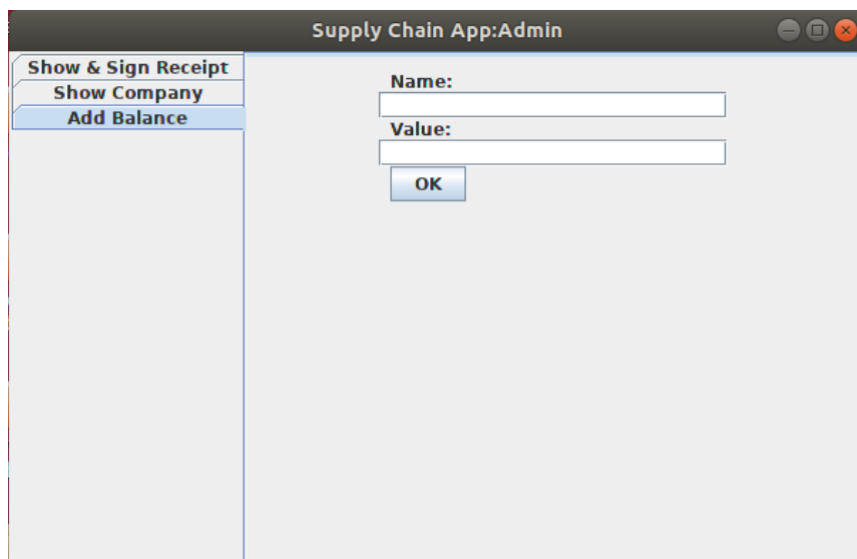
From:
company

To:

Value:

OK

增加余额界面(Add Balance)



五、加分项

- 实现供应链金融相关附加功能：
 - 使用公私钥加密系统，唯一私钥登录账户；
 - 交易签名功能，收款方和仲裁机构审核并签名回执；
 - 转账和还款的基本交易功能。
- 考虑并处理了较多的异常情况。
- 简洁美观、用户友好的图形界面。

六、心得体会

本次基于 FISCO-BCOS 区块链底层的供应链金融区块链应用开发项目，经过前中后期三个阶段。

- 前期
学习区块链的基本原理，包括比特币白皮书、共识机制、非对称密钥体系，以及联盟链等。重点学习的实践了使用 Solidity 语言编写智能合约，并在以 Remix 为例在线编译平台试编译运行。
- 中期
熟悉 FISCO-BCOS 区块链底层开发平台，在 FISCO-BCOS 上搭建区块链，并编译、部署、调用智能合约。
设计和编写供应链金融区块链应用开发项目的智能合约，并且初步部署、测试。
- 后期
搭建 JAVA SDK 环境，在 Eclipse 中集成 Spring 框架和 Gradle 工具，编写前端和后端，并连接区块链进行测试。
至此基于 FISCO-BCOS 区块链底层的供应链金融区块链应用开发项目基本完成。

经过本次项目，我对区块链基本原理以及区块链应用开发有了深刻的认识，也获得了一个包含前后端的完整项目开发的宝贵经验。