

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)

年级	17	专业 (方向)	软件工程
学号	17343095	姓名	彭裕航
电话	15019598665	Email	785795086@qq.com
开始日期	2019/11/01	完成日期	2019/12/07

Github网址: <https://github.com/Yuhang-Follheart/blockchain-trading-system>

演示视频: <https://github.com/Yuhang-Follheart/blockchain-trading-system/blob/master/demo.mp4>

一、项目背景

从最开始的以物换物，到今天的数字货币时代，人类的货币文明在不断地向前发展。自21世纪初期以来，人们对去中心化的交易模式的探索也在不断地冲击着以信任为前提中心化交易。在中心化系统中，企业间的债券交易难以实现，特别是小型企业难以凭债券获得银行的贷款融资。本次项目就是为了解决这个难题而做出的小小的尝试，让企业间的借债关系上链，使得企业可以转让债凭关系来获得融资。

二、方案设计

程序用户分为两种：普通企业和银行企业。普通企业和银行企业均可在链上进行注册，并依此在链上进行交易。

交易的形式有四种：用余额交易、用已有应收账款交易、签发新的应收账款和向银行的贷款交易。

除了普通企业间的交易，企业还可以用应收账款向银行获取贷款。

具体功能如下：

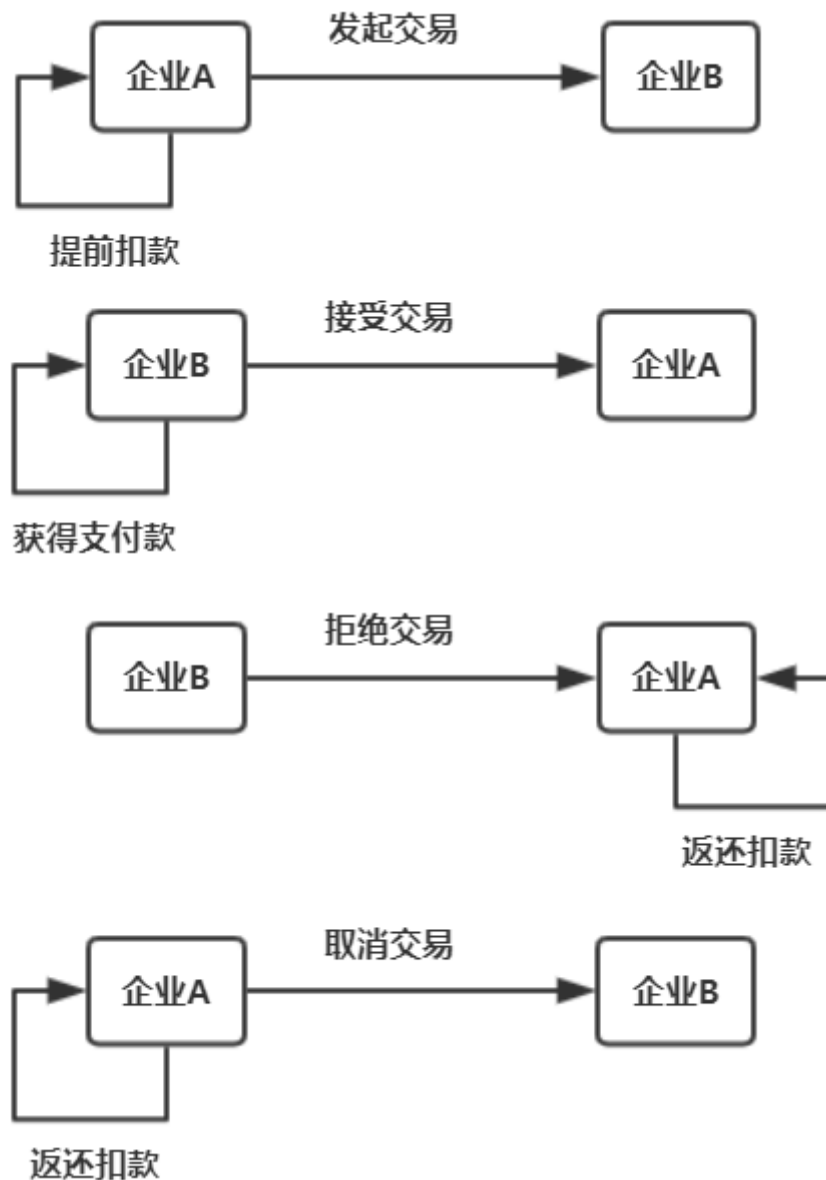
1. 用户账户管理

1. 账户注册：账户凭借地址在链上进行注册，注册时需指明类型(是银行还是普通企业)，注册资产量和唯一昵称。只有已注册的账户才能执行合约的其他操作。

2. 获取信息：

1. 查看余额：余额可以为负数，因为到期的应收欠款可以被债主强制执行支付
2. 查看欠款数及各笔欠款的单号
3. 查看应有账款数和各笔应收账款的单号
4. 查看未处理的交易数及各个交易单号

2. 交易功能



1. 发起交易：

1. 发起余额交易：要求余额大于等于待支付金额，发起交易后余额会立即被扣除。若对方接受交易则交易完成，若对方拒绝交易或己方及时取消交易，余额会退款。
2. 发起已有应收账款的交易：要求应收账款总额大于等于待支付金额，发出交易后应收账款总额会立即扣去，但实际应收账款账单数据暂未改变。若对方接受交易则交易完成，会转移相应应收账款给对方，若对方拒绝交易或己方及时取消交易，应收账款总额会返回。
3. 签发新的应收账款：发出交易后暂无影响，需等待对方的确认。
4. 向银行的贷款交易：形式上和第二种交易一样，向银行支付已有的应收账款，银行接受交易后会获得相应金额的现金。

2. 接受交易：

在交易的到期时间之前，接受方可以选择接受交易。

1. 接受余额交易：接受方的余额会增加，发起方扣除的余额将不会返回
2. 接受已有应收账款交易：收集发起方的若干应收账款直到达到目标金额，将这些应收账款的接受权转移给交易的接受方。
3. 接受新的应收账款交易：发起方和接受方形成新的债务关系
4. (银行)接受贷款交易：收集发起方的若干应收账款直到达到目标金额，将这些应收账款的接受权转移给银行，银行支付一定的余额。

3. 拒绝交易/取消交易：

交易的接受方可以拒绝交易，发起方可以选择取消交易，造成的影响是一致的，因此放在一起讲

1. 拒绝余额交易：返回发起方已扣除的余额
2. 拒绝已有应收账款交易：返回发起方已扣除的应收欠款

3. 拒绝新的应收账款交易：无连带影响
4. (银行)拒绝贷款：返回发起方已扣除的应收欠款

3. 债务相关功能

1. 偿还债务：债务(应收账款的付款方)可以选择支付自己的账款取消债务关系
2. 追要债务：本着人文关怀的态度，债主在债务付款期限内不能追讨债务但是过了期限债主可以强行让背负人支付(所以允许企业余额出现负数，这时企业很多行为将受到限制)

数据存储方面采用链上数据存储，因为fisco-bcos提供的数据库其实也是solidity写的，同样也是部署到链上，没有必要多曲折一圈。

Gui用Java的swing，后端也是用java写的，放在一个工程里。

三、功能测试

1. 使用私钥登录



2. 注册企业用户

比如我注册一个叫Alice的普通企业，资产为1000

区块链金融系统

企业注册

名称：

资产：

类型：☒ 普通企业 ☐ 银行


注册成功：

区块链金融系统

企业注册

名称：

注册成功

 注册成功!

3. 进入主页面

区块链金融系统

企业信息 | 待处理交易 | 债务信息 | 应收账款信息

企业信息

名称: Alice

余额: 1000

欠款总额: 0

应收账款: 0

发起交易

交易对象:

交易类型: 余额交易

交易金额:

4. 再另开三个程序。用两个私钥注册两个普通企业Bob, Coco, 资产都为1000。然后创建一个银行企业Bob企业:

区块链金融系统

企业信息 待处理交易 债务信息 应收账款信息

企业信息

名称: Bob
余额: 1000
欠款总额: 0
应收账款: 0

发起交易

发起交易

交易对象: Alice

交易类型: 余额交易

交易金额:

发起交易

Coco企业:

区块链金融系统

企业信息 待处理交易 债务信息 应收账款信息

企业信息

名称: Coco
余额: 1000
欠款总额: 0
应收账款: 0

发起交易

发起交易

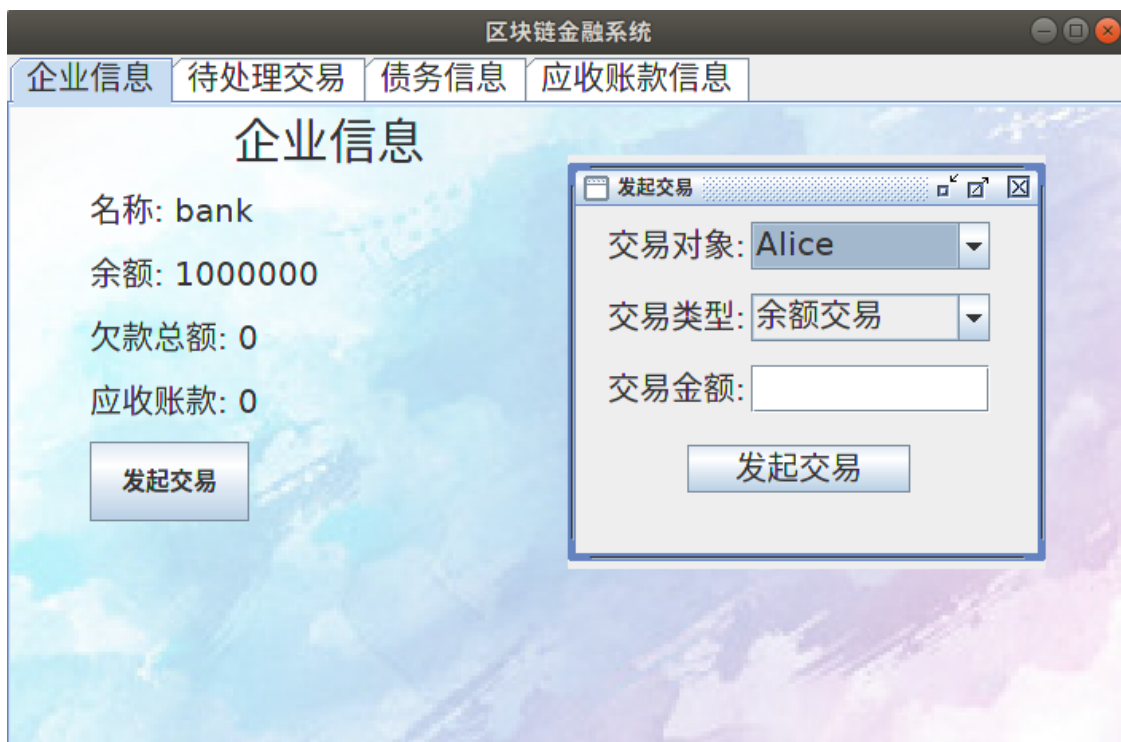
交易对象: Alice

交易类型: 余额交易

交易金额:

发起交易

银行企业:



5. Alice与Bob完成余额交易

1. Alice发起余额交易



此时Alice的钱已经被预先划扣了:

名称: Alice
余额: 900
欠款总额: 0
应收账款: 0

Alice的待处理交易中出现刚刚发起的交易:

企业信息 待处理交易 债务信息 应收账款信息

选择要处理的交易编号: 0 接受交易 取消/拒绝交易

交易发起人	交易接受者	现金交易	金额	交易代号
Alice	Bob	是	100	0

Alice可以在Bob接受交易前取消交易，我们为了演示就不这么做了。

2. Bob接受交易

Bob在“待处理交易”面板接受Alice的交易:

企业信息 待处理交易 债务信息 应收账款信息

选择要处理的交易编号: 0 接受交易 取消/拒绝交易

交易发起人	交易接受者	现金交易	金额	交易代号
Alice	Bob	是	100	0

接受交易成功

接受交易成功!

OK

此时Alice的钱已经打到Bob上了:

名称: Bob
余额: 1100
欠款总额: 0
应收账款: 0

6. Alice签发应收凭证给Bob

1. 用Alice发起欠条交易

区块链金融系统

企业信息 待处理交易 债务信息 应收账款信息

企业信息

名称: Alice
余额: 900
欠款总额: 0
应收账款: 0

发起交易

发起交易

交易对象: Bob

交易类型: 欠条交易

金额: 100

发起交易

发起交易成功

签发欠条交易成功!

OK

2. 让Bob接受这个交易，然后查看Alice的企业信息和债务信息

Alice企业信息:

名称: Alice
余额: 900
欠款总额: 100
应收账款: 0

Alice债务信息:

选择要处理的债务编号: 0 ▾ 偿还债务

债主	欠款金额	债务编号
Bob	100	0

可以看到Alice多了100元的债务

3. 查看Bob的企业信息和应收账款信息

Bob企业信息:

企业信息

名称: Bob
余额: 1100
欠款总额: 0
应收账款: 100

Bob的应收账款信息:

选择要处理的债务编号: 0 ▾ 讨债

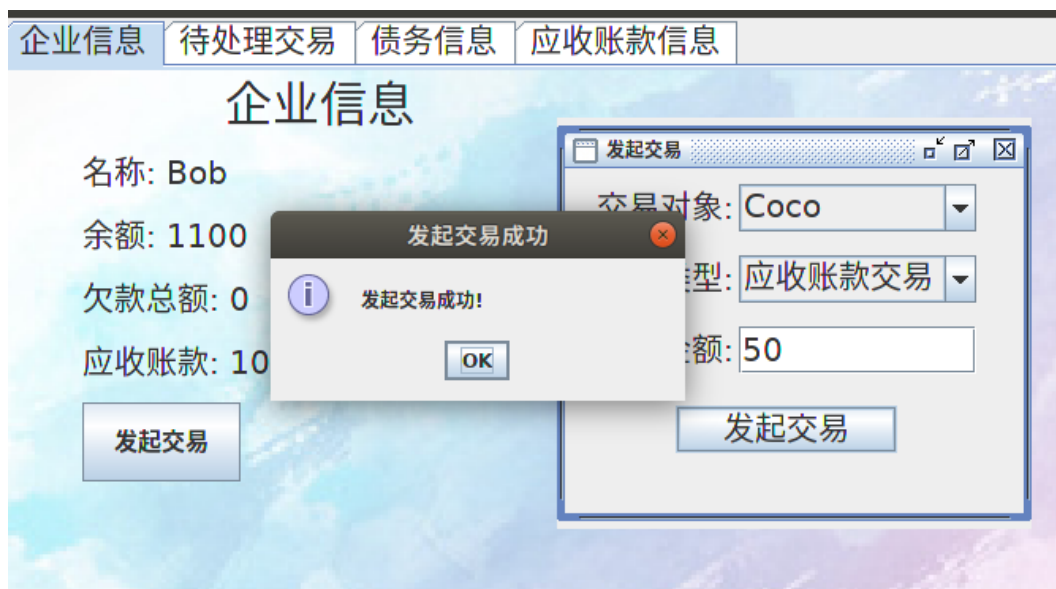
欠款人	欠款金额	债务编号
Alice	100	0

可以看到Bob知道Alice欠自己100元

7. Bob用已有的应收账款向Coco发起50元交易

上面的交易使得Bob拥有100元的Alice的应收账款, 现在让Bob拿出其中的50向Coco交易

1. Bob发起应收账款交易



2. 让Coco接受这个交易，然后查看三人的相关信息

Coco的应收账款:

企业信息	待处理交易	债务信息	应收账款信息
选择要处理的债务编号: 1 讨债			
欠款人	欠款金额	债务编号	
Alice	50	1	

可以看到Coco得到了Alice背债的50元

Bob的应收账款:

选择要处理的债务编号:

0

讨债

欠款人	欠款金额	债务编号
Alice	50	0

可以看到Bob原本拥有的Alice的应收账款减少了50

查看Alice的债务信息:

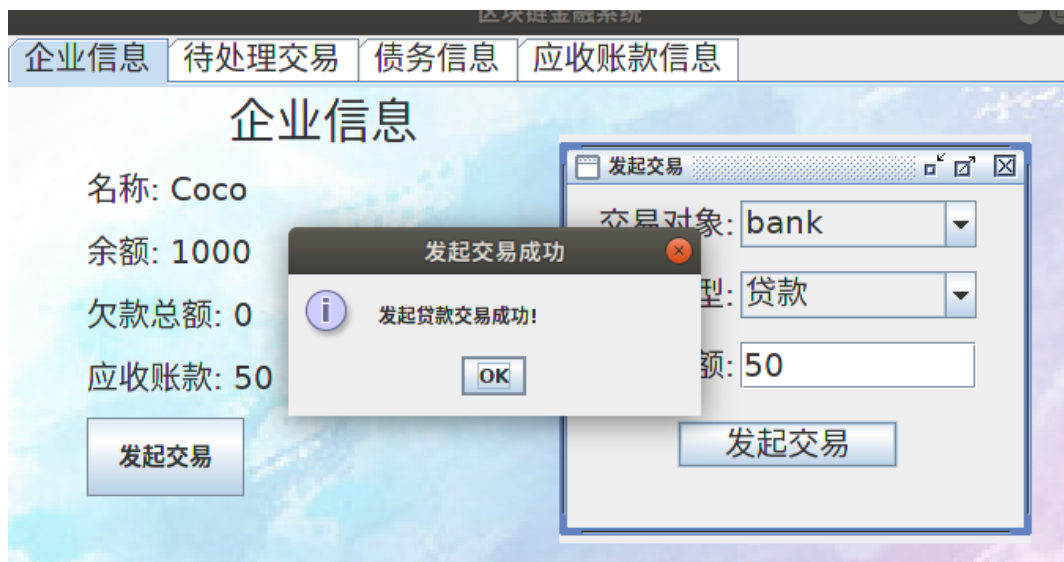
选择要处理的债务编号: 0 ▾ 偿还债务

债主	欠款金额	债务编号
Bob	50	0
Coco	50	1

可以看到Alice多了一条欠债信息，现在欠Coco50元了，但是欠Bob的减少了50元。

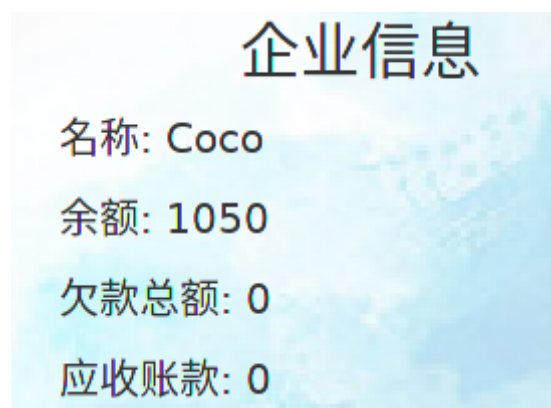
8. 让Coco用已有的50应收账款向银行贷款

1. Coco发起贷款交易:



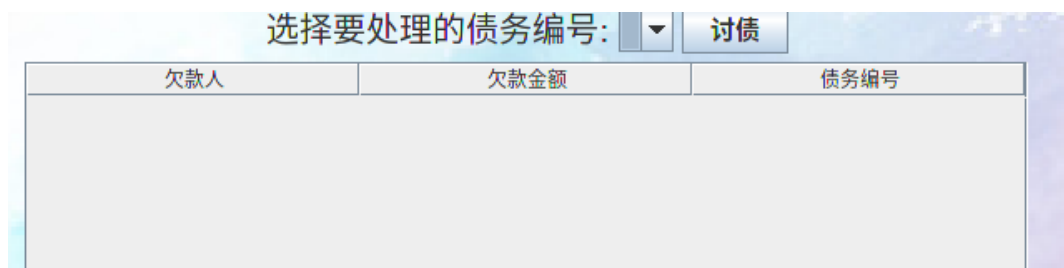
2. 用银行接受这个贷款交易，然后查看Coco的余额，应收账款信息。

Coco的余额:



可以看到Coco获得了50元融资

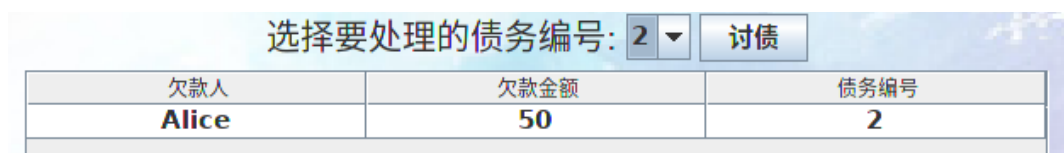
查看Coco的应收账款信息:



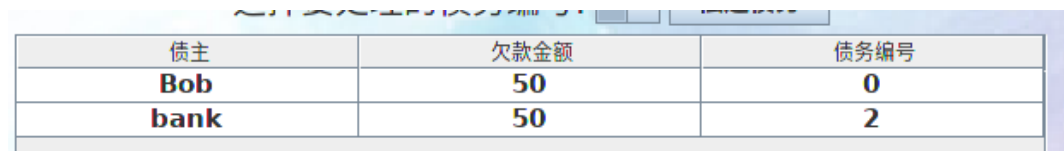
为空，说明Coco与Alice不再拥有债务关系

3. 查看银行和Alice的信息

查看银行的应收账款信息:



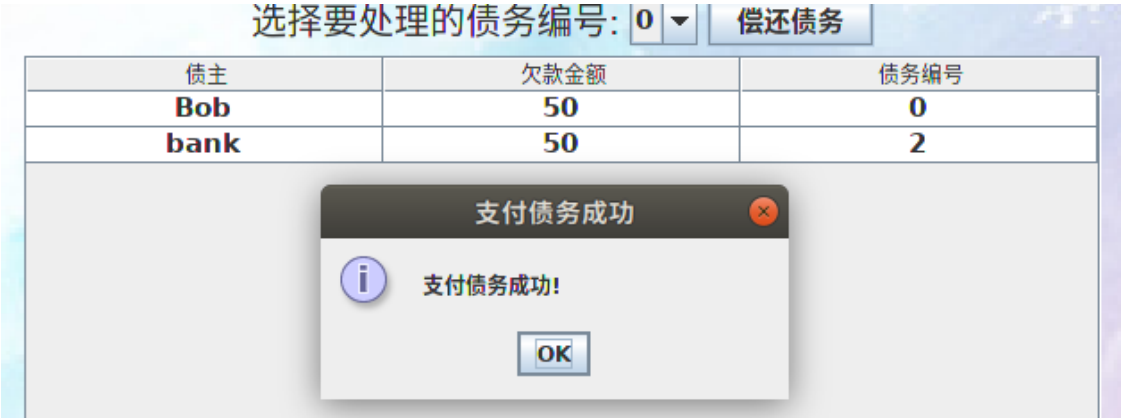
查看Alice的债务关系:



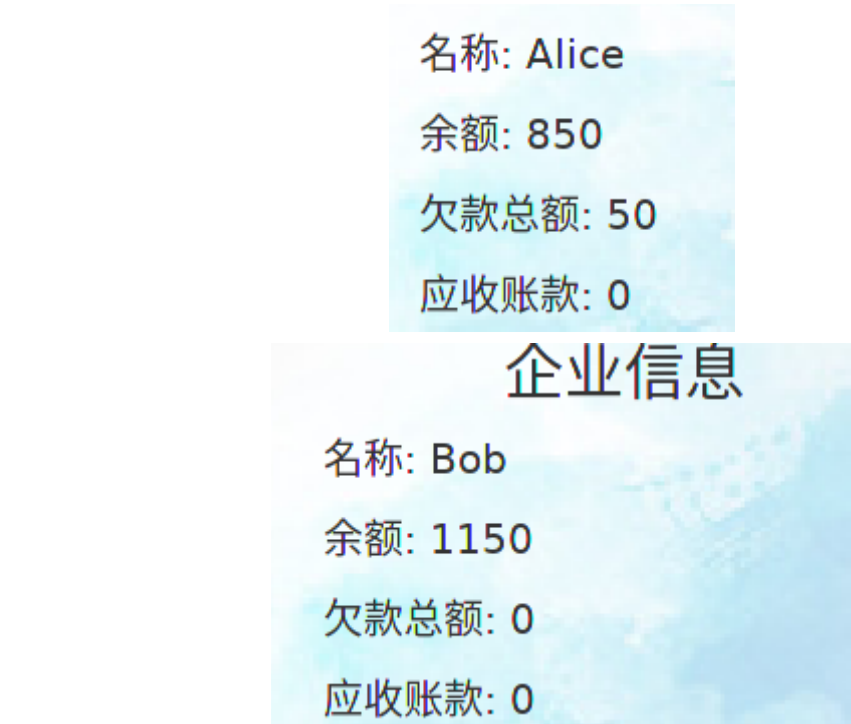
可以看到Alice与Coco的债务关系转移到Alice与银行之间了

9. Alice支付欠款

我们用Alice支付欠Bob的50块钱



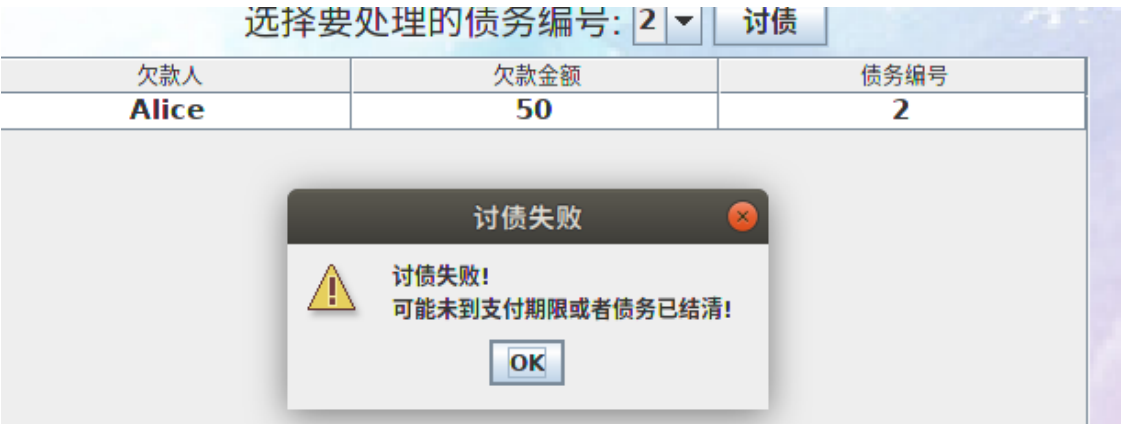
查看Alice和Bob的余额



可以看到Alice成功用余额支付了欠款

10. 银行讨要欠款

使用银行讨要Alice的欠款:



因为未到付清期限(一个月)所以不能讨债

四、界面展示

1. 登录界面



2. 注册界面



3. 企业信息界面

区块链金融系统

企业信息待处理交易债务信息应收账款信息

企业信息

名称: Alice
余额: 750
欠款总额: 50
应收账款: 50

发起交易

发起交易

交易对象: Bob

交易类型: 余额交易

交易金额: 100

发起交易

4. 待处理交易界面

区块链金融系统

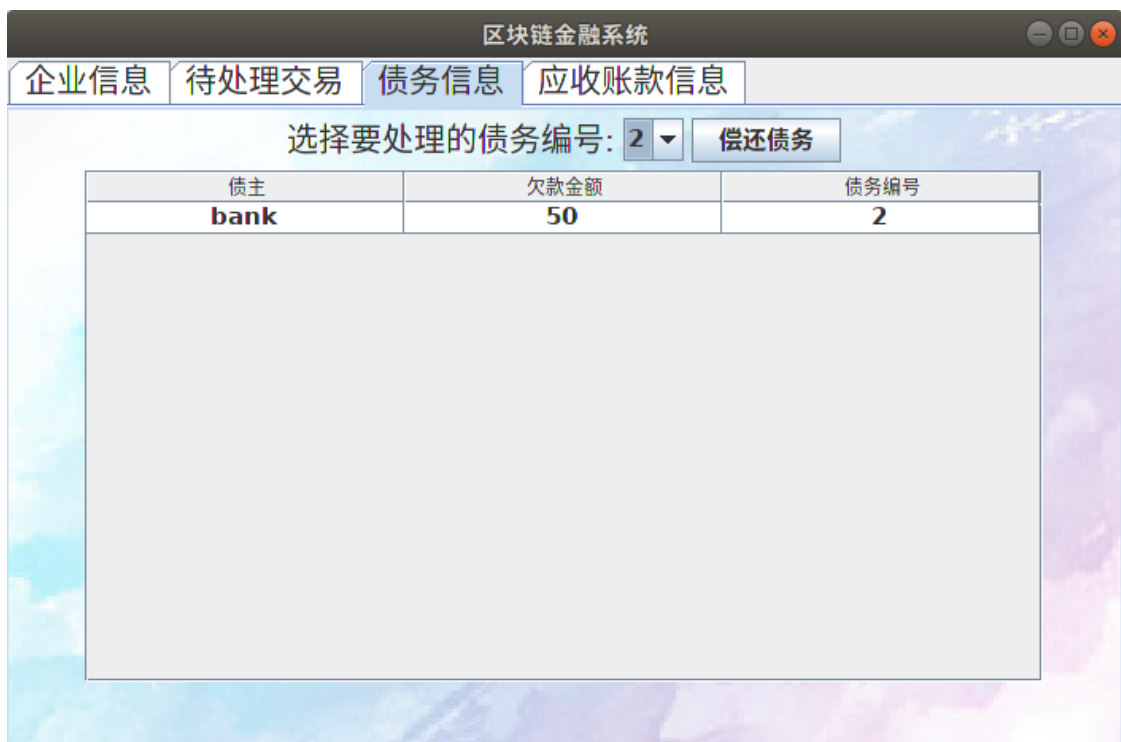
企业信息待处理交易债务信息应收账款信息

选择要处理的交易编号: 4

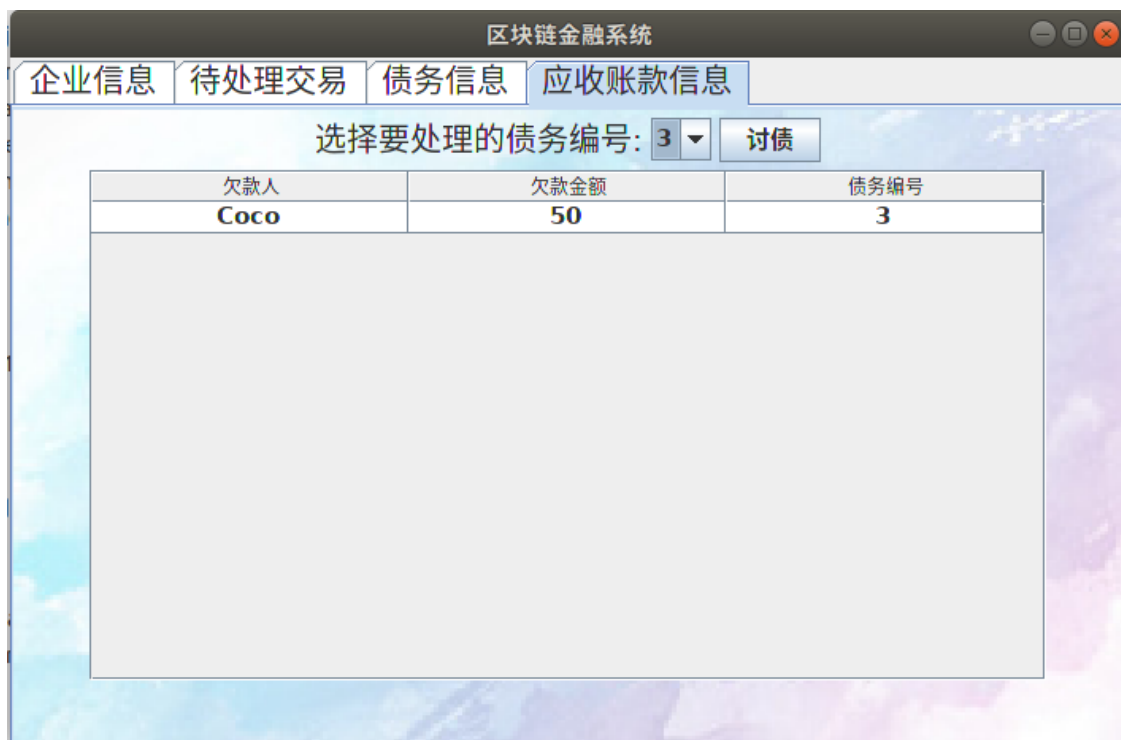
接受交易取消/拒绝交易

交易发起人	交易接受者	现金交易	金额	交易代号
Alice	Bob	是	100	4

5. 债务信息界面



6. 应收账款界面



五、心得体会

经过这次的区块链项目设计与完成，我对区块链的知识有了更深入的理解，同时也感觉编程能力有所提高。

此次项目是基于fisco-bcos的区块链底层实现的，如果直接用以太坊的官方链的话应该会麻烦不少。我对fisco-bcos的理解就是其建立了一套与区块链打交道的方法，包括合约命名，节点分组等，最重要的就是提供了solidity合约的编译工具，这使得我们可以使用其他高级语言与区块链进行交互，能完成这次作业可以说是非常感谢fisco-bcos这样巨人般工程的存在。

作业的完成过程其实是很曲折的，我觉得最难的是合约设计部分。项目的要求懂了，大概知道也要实现什么功能，但是我就是只能对着电脑发呆，因为感觉对solidity无从下手。从大一到现在几乎都是用C++，没怎么用过别的语言，所以感到无所适从吧。后来周末泡了两天图书馆，看了各种solidity合约的例子，才对这种语言有了更多的把握，最后才跌跌撞撞地把代码写了出来。

到了第三阶段，这阶段其实就是写一个GUI,通过SDK去访问链上的合约方法。在思索了很久后我决定使用Java来写这部分，因为刚好实训也要用到Java。于是又开始疯狂学习Java语言，还要Java语言真的比Solidity好写多了，所以花的精力要少一些。fisco-bos的SDK文档其实已经很详细了，但是在环境配置方面还是有些问题。不知为何使用甲骨文官网的JDK的话是不能与区块链通信的，一定要用OpenJDK，解决这个问题用了我好多时间，因为也没有报错输出什么的，只能一个一个试，还好装了OpenJDK后的好了。

第一次尝试用swing写程序界面也是感觉很神奇吧。以前一直写的都是那种命令行窗口的黑框程序，这次终于写了个有界面的，自豪感满满吧感觉。Swing最难的部分我个人感觉在于各组件的嵌套和大小关系难以理解，组件的大小设置也较难调，费了好大功夫才调的自己觉得好看(可能是直男审美吧)

总的来说通过这次项目我学到了很多，也充分意识到了自己的不足，比如基础知识薄弱，学习能力差，对环境的配置和Linux的使用也不熟练。我会针对这些缺点进行改进，希望能成为更好的自己吧，也在这里谢谢TA和老师这段时间的教导。