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	17343103	姓名	孙浥尘
电话	15626415128	Email	1368018481@qq.com
开始日期	2019/12/8	完成日期	2019/12/13

一、 项目背景

基于已有的开源区块链系统 FISCO-BCOS（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。**

场景介绍：

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

存储设计、数据流图、核心功能介绍（文字+代码）

学习过程介绍：

根据 https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/tutorial/sdk_application.html 来开始进行区块链应用的学习。首先是合约编译，我们得到了合约 Asset.sol 的存储与接口，给出了完整实现，但是 Java 程序无法直接调用 Solidity 合约，需要先将 Solidity 合约文件编译为 Java 文件。控制台提供了编译工具，可以将 Asset.sol 合约文件存放在 console/contracts/solidity 目录。利用 console 目录下提供的 sol2java.sh 脚本进行编译：

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/console$ ./sol2java.sh org.fisco.bcos.as
set.contract

Compile solidity contract files to java contract files successfully!
```

接着是 SDK 的配置，与教程中相同，我使用的是 JAVASDK，首先获取教程中的项目并解压缩：

```
fisco-bcos@fiscobcos-VirtualBox:~$ curl -LO https://github.com/FISCO-BCOS/LargeF
iles/raw/master/tools/asset-app.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   151  100   151    0     0    193      0 --:--:-- --:--:-- --:--:--   193
  0     0    0     0    0     0      0      0 --:--:--  0:00:21 --:--:--     0cu
rl: (7) Failed to connect to raw.githubusercontent.com port 443: Connection refu
sed
fisco-bcos@fiscobcos-VirtualBox:~$ tar -zxf asset-app.tar.gz
```

然后就是区块链节点证书配置，拷贝区块链节点对应的 SDK 证书。

接下来可以运行项目，测试功能是否正常。

首先是编译：

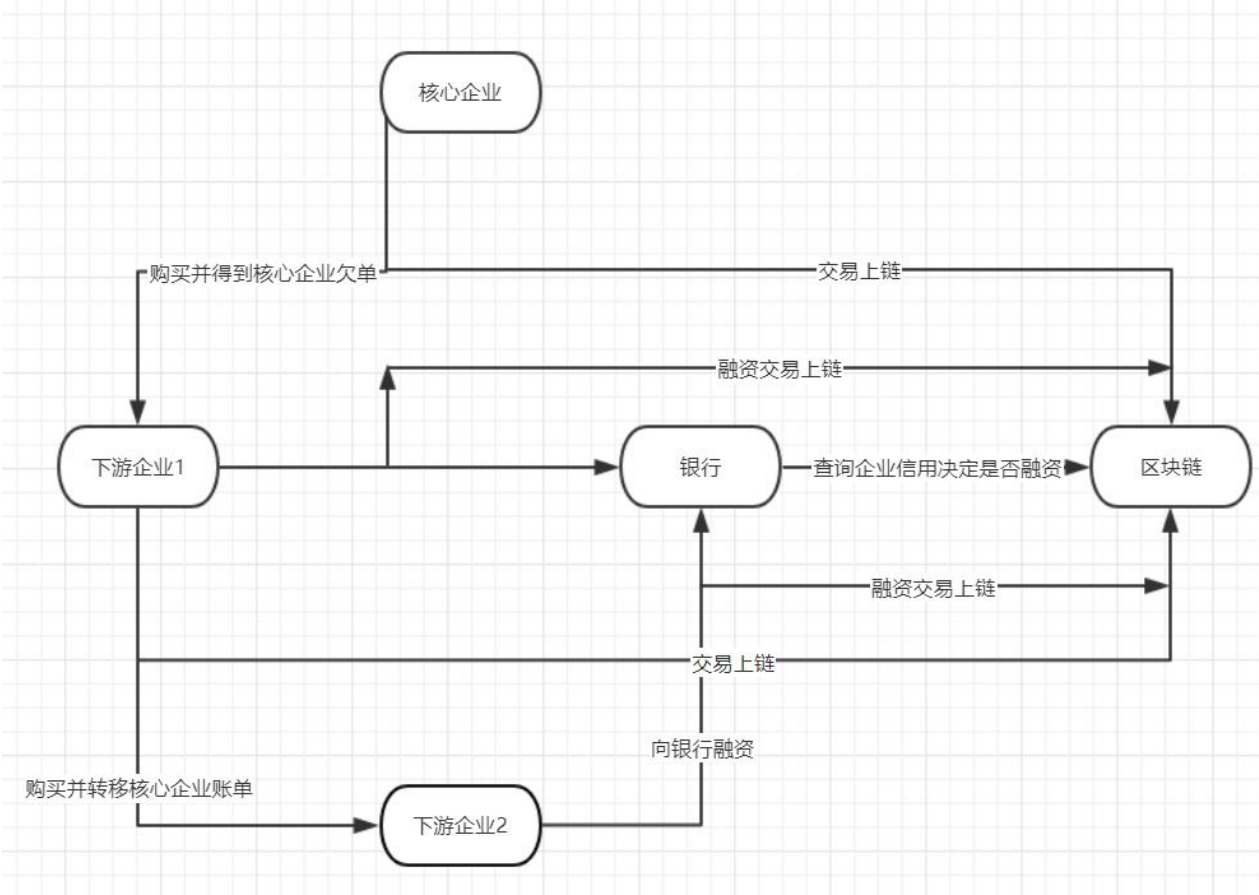
```
fisco-bcos@fiscobcos-VirtualBox:~$  cd ~/asset-app
fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ ./gradlew build
Downloading https://services.gradle.org/distributions/gradle-5.6.2-bin.zip
..........................................................................
.........

Welcome to Gradle 5.6.2!

Here are the highlights of this release:
 - Incremental Groovy compilation
 - Groovy compile avoidance
 - Test fixtures for Java projects
 - Manage plugin versions via settings script

For more details see https://docs.gradle.org/5.6.2/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 3m 27s
4 actionable tasks: 4 executed
```

编译成功之后，将在项目根目录下生成 dist 目录。dist 目录下有一个 asset_run.sh 脚本，简化项目运行。接着部署 Asset.sol 合约，首先启动 fisco 的节点，然后进行部署就可以成功：

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ bash start_all.sh
try to start node0
try to start node1
try to start node2
try to start node3
try to start node5
 node3 start successfully
 node0 start successfully
 node1 start successfully
 node2 start successfully
 node5 start successfully
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ cd ../../../
fisco-bcos@fiscobcos-VirtualBox:~$ cd asset-app/
fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ ls
build  build.gradle  dist  gradle  gradlew  gradlew.bat  src  tool
fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ cd dist/
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ ls
apps  asset_run.sh  conf  contract  lib  log
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh deploy
 deploy Asset success, contract address is 0x56bbf894427517e77f4ae233673d41fbd50
96cad
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$
```

这样合约就部署成功了，我就继续在这个项目的基础上进行操作。

数据流图示例:



存储设计以及核心功能介绍:

实现中分别对应这四个函数。首先在合约中声明两个结构体 Receipt 表示收据，其中包括借款人 c，收款人 to，收据金额以及可信度。Company 表示公司，其中包括公司名称，公司地址以及属性，即是银行或核心公司或下游公司。全局声明一个地址->公司的映射来存储公司，地址->金额的映射存储地址上的余额，一个数组 rec 来存储收据信息，以及 bank 存储银行的地址。

```
struct Receipt {
    address c;
    address to;
    uint amount;
    uint status;
}
struct Company {
    string company;
    string adr;
    uint t;
}
mapping(address => Company) public adr2Com;
mapping(address => uint) public balances;
```

接着就是功能实现：

首先是用户通过私钥进行私钥注册。具体实现如下：

```
function addCompany(string company_,string adr_,uint t_){
    Company memory tem = Company(company_,adr_,t_);
    adr2Com[msg.sender] = tem;
}
```

第一个功能实现的函数是 accountsReceivable2Chain 函数，通过输入核心企业的姓名和下游企业的姓名以及金额来进行欠单的签发，实现购买并签发欠单的操作，并将此次交易上链。具体实现如下：

```
function accountsReceivable2Chain(address to_,uint amount_){
    Receipt memory r = Receipt(msg.sender,to_,amount_,1);
    rec.push(r);
}
```

第二个功能实现的函数是 transferOfAccountsReceivable2Chain 函数,通过输入下游企业 1 的姓名和下游企业 2 的姓名以及金额来进行交易欠单的转让，并将此次交易上链。

```
function transferOfAccountsReceivable2Chain(address to_,uint amount_){
    for(uint i=0;i<rec.length;++i){
        if(rec[i].to == msg.sender&& rec[i].status == 1){
            rec[i].amount-=amount_;
            Receipt memory r = Receipt(rec[i].c,to_,amount_,1);
            rec.push(r);
            return;
        }
    }
}
```

第三个功能实现的函数是 financing2Chain 函数，是下游企业 1 和下游企业 2 的操作，其可以向银行进行融资，通过输入融资金额，然后银行判断企业信誉度来决定是否进行融资，融资后将此次交易上链。具体实现如下：

```
function financing2Chain(uint amount_){
    uint sum = 0;
    for(uint i=0;i<rec.length;++i){
        if(rec[i].to == msg.sender&& rec[i].status == 1){
            sum+=rec[i].amount;
        }
    }
    if(amount_<=sum) {
        balances[msg.sender]+=amount_;
        rec.push(Receipt(msg.sender,bank,amount_,0));
    }
    else {
        balances[msg.sender]+=sum;
        rec.push(Receipt(msg.sender,bank,sum,0));
    }
}
```

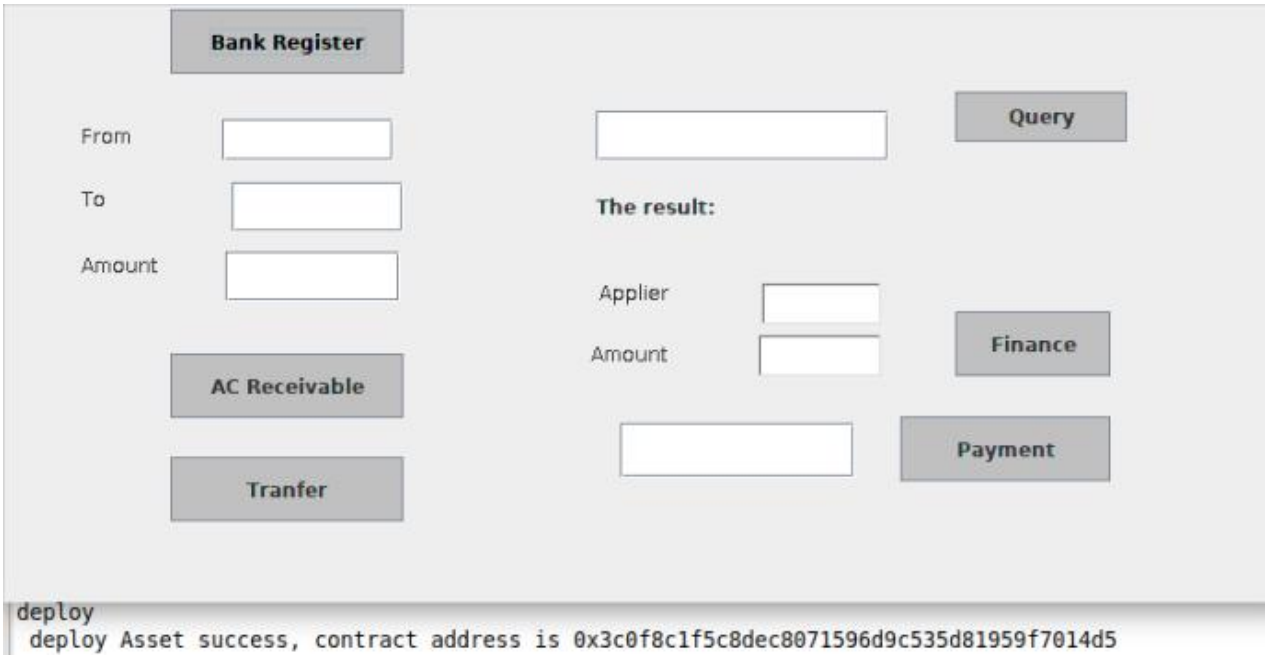
第四个功能实现的函数是 paymentSettlement2Chain 函数，应收账款支付功能。通过输入姓名并查询欠单将核心企业应支付的欠单全部进行归还并更新相关账单。

```
function paymentSettlement2Chain(){
    for(uint i=0;i<rec.length;++i){
        if(rec[i].c==msg.sender){
            balances[msg.sender]-=rec[i].amount;
            balances[rec[i].to]+=rec[i].amount;
            removeAtIndex(i);
    i--;
        }
    }
}
```

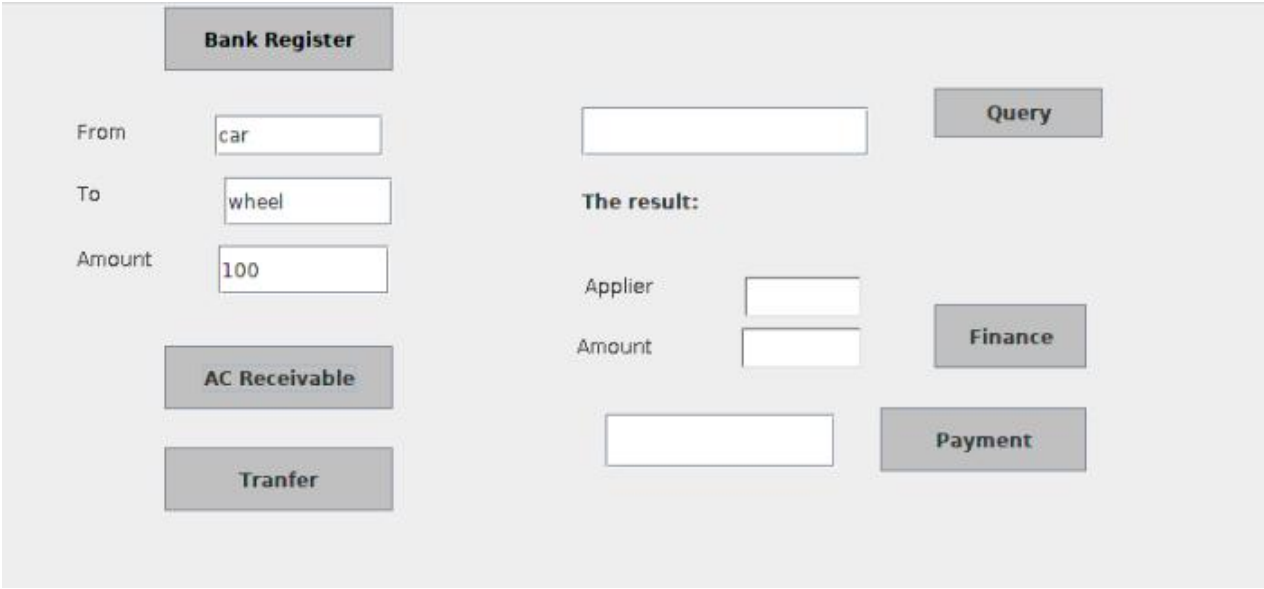
三、 功能测试

实验截图:

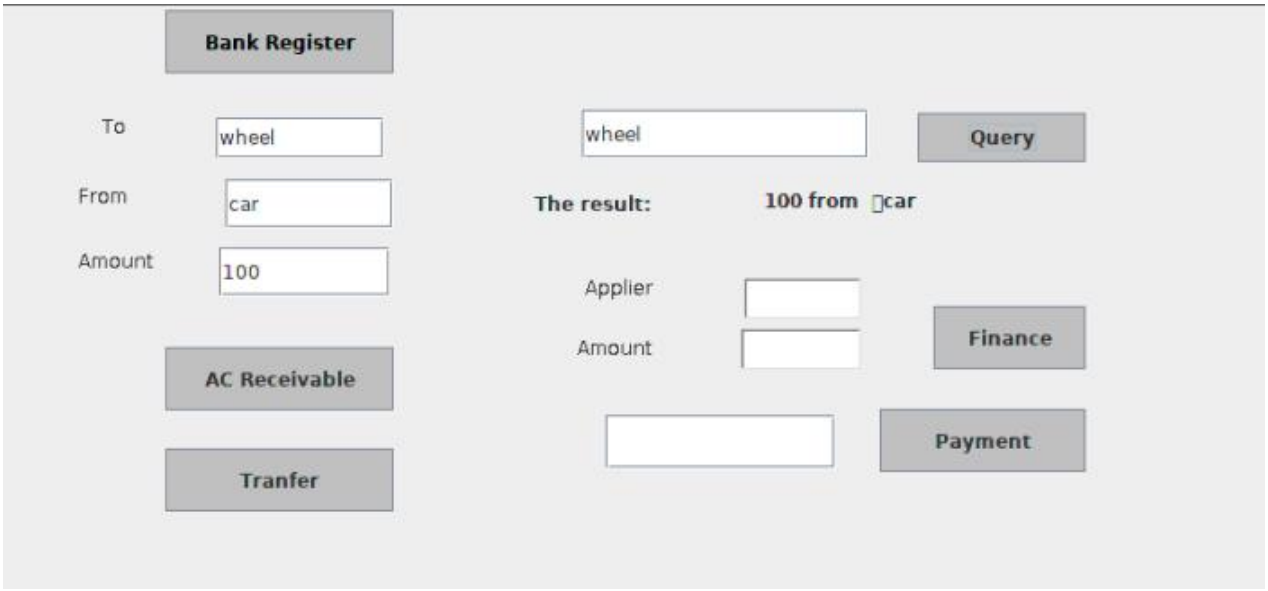
首先是银行注册并部署合约，点击 Bank Register 即可



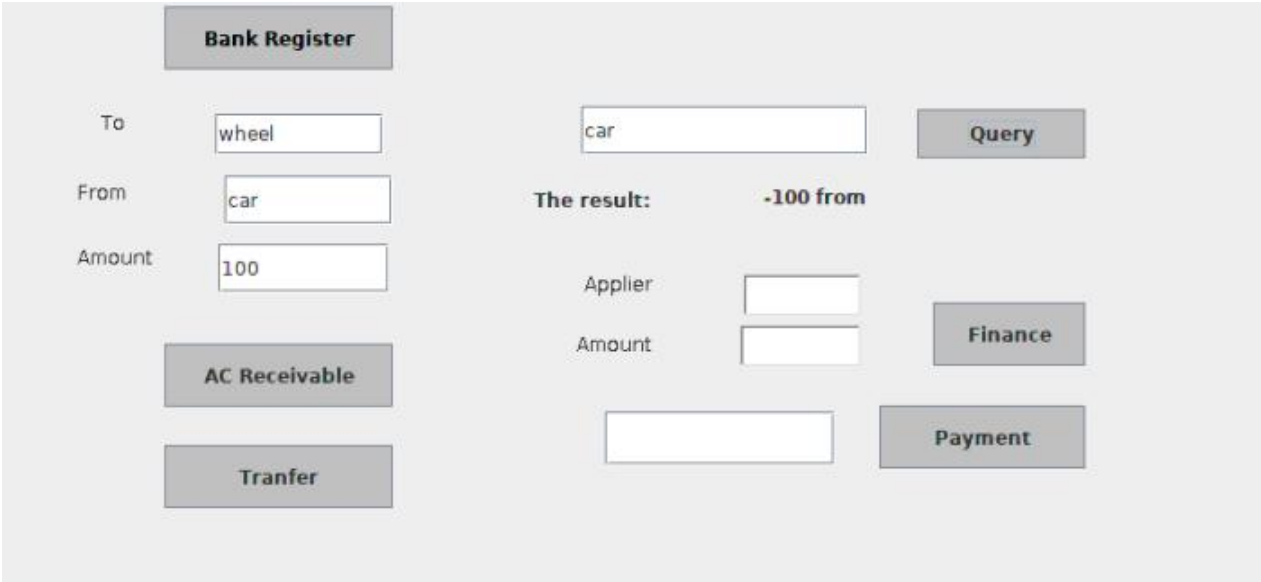
接着要实现功能 1，即实现采购商品—签发应收账款交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。用户 car 向用户 wheel 购买轮胎并签订了欠单，输入用户擦绒，wheel 以及欠单金额，并点击 AC Receivable 即可。



接着对 wheel 的信息进行查询，发现 wheel 从 car 上获得了 100 的账单：

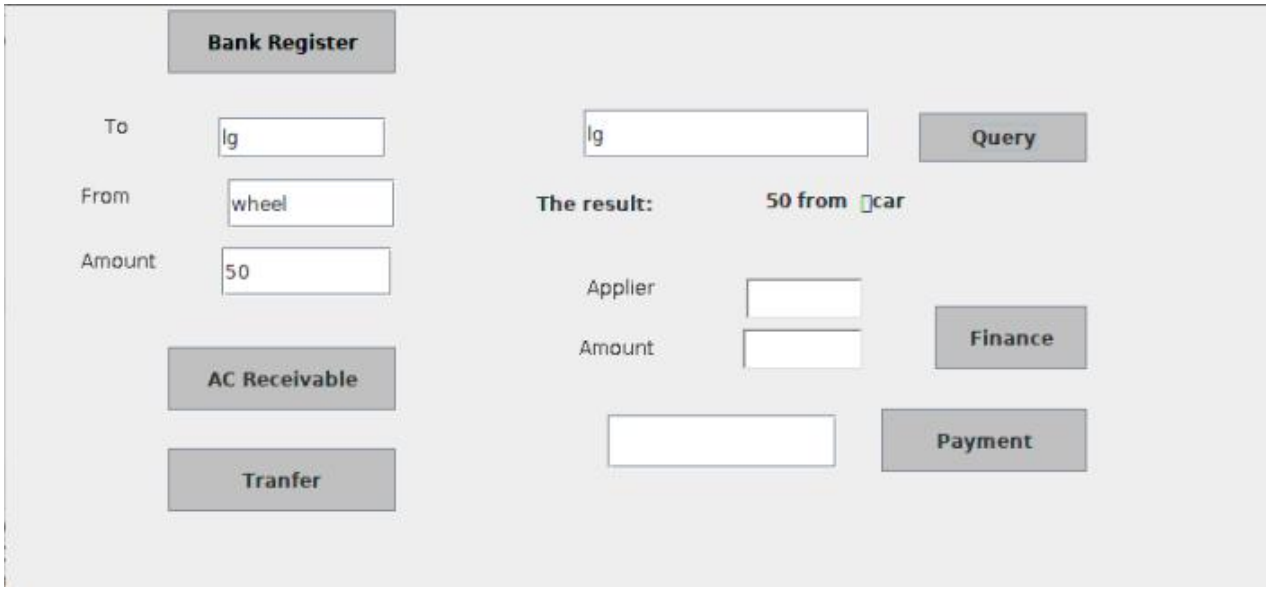


接着对 car 进行查询，发现 car 有欠单 100，也就是-100：

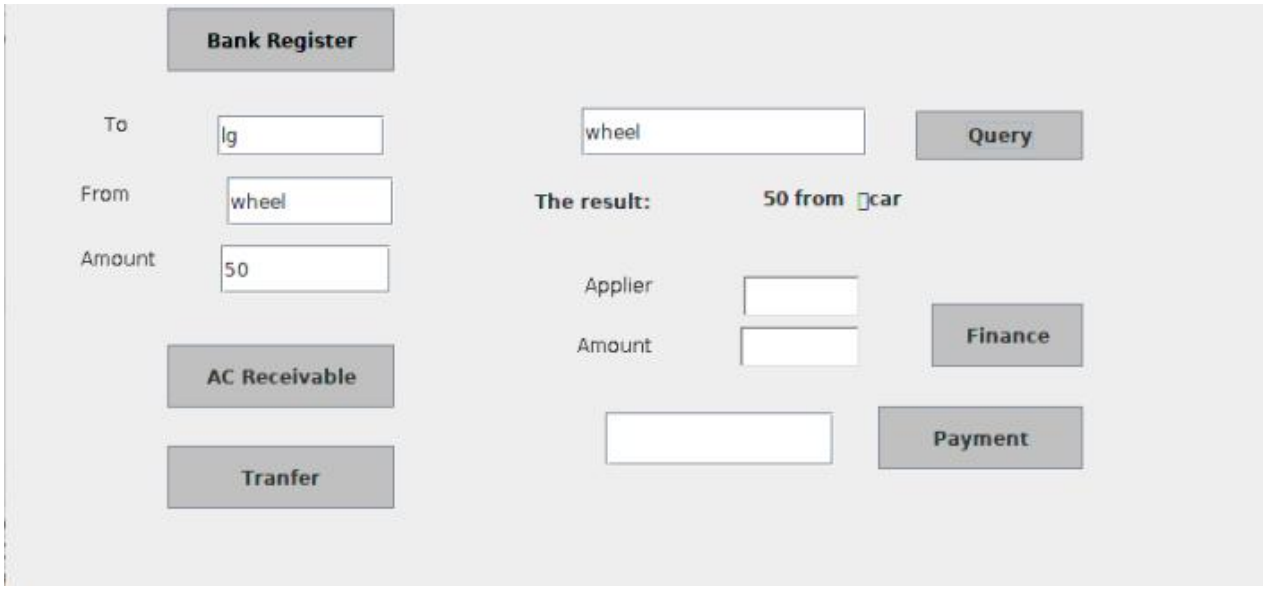


接着进行功能 2，实现应收账款的转让上链，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还

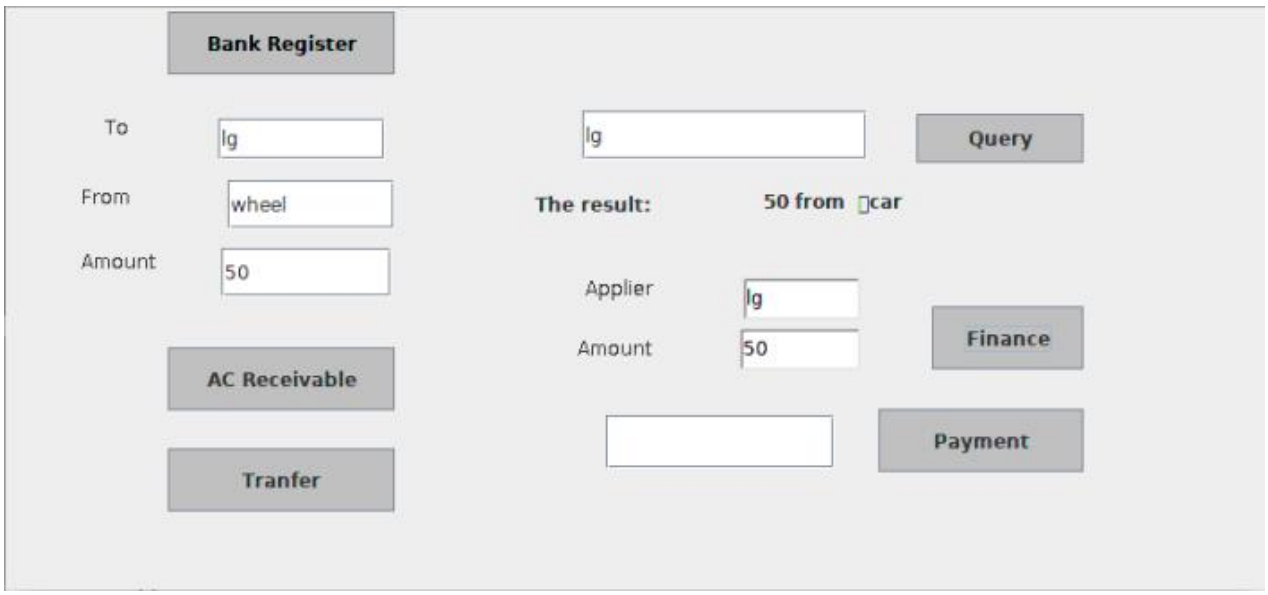
钱款。输入用户轮毂厂 lg 以及 wheel，并输入转让金额 50，接着对轮毂厂 lg 进行查询，发现现在 lg 的账单金额变为 50 来自车厂，说明账单转让成功：



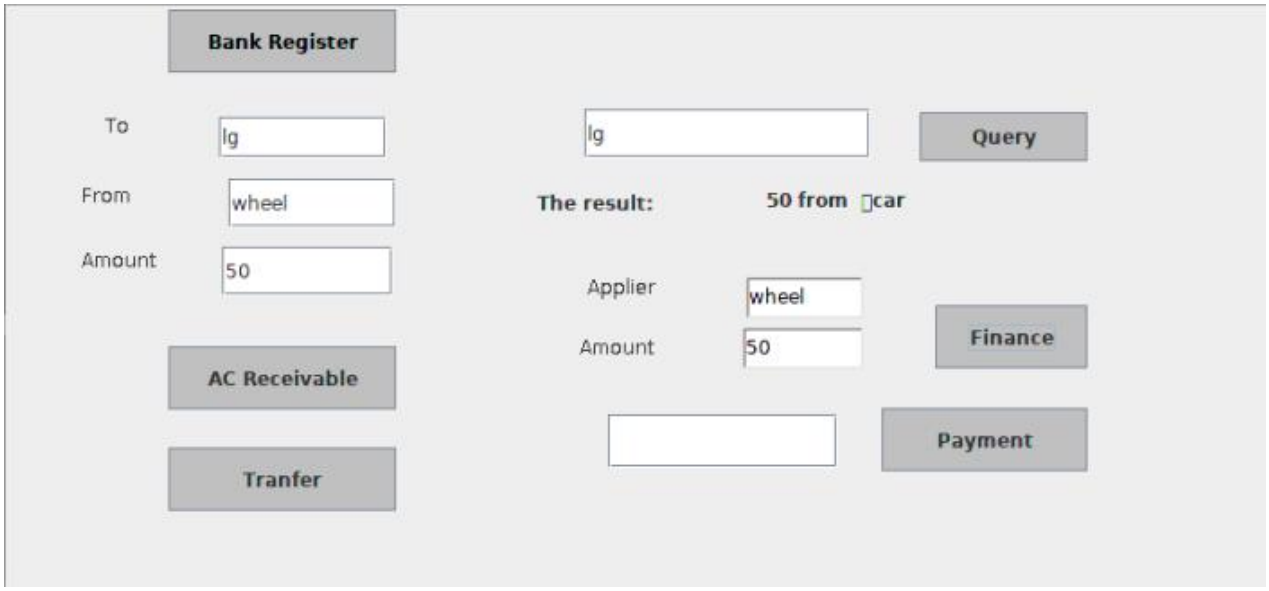
继续对 wheel 进行查询，发现 wheel 的账单从来自 car 的 100 变为 50，说明转让出去了 50，说明账单转让成功：



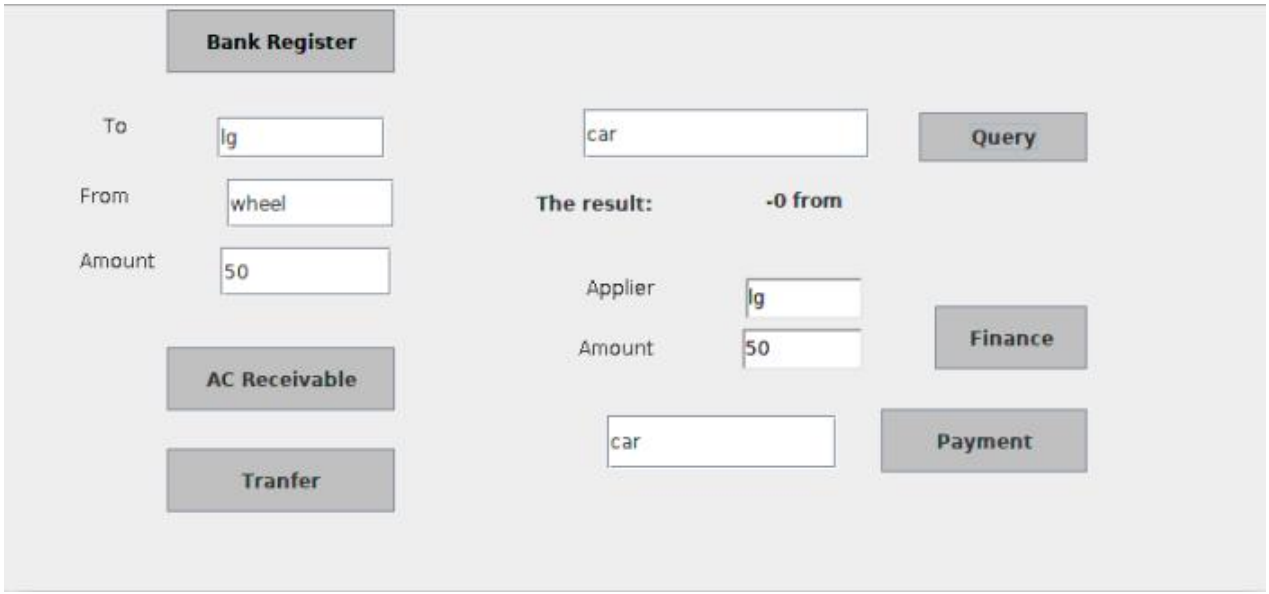
接着进行功能 3，利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。输入申请人 wheel 以及融资金额，发现融资成功，说明银行发现 wheel 持有 car 的账单并认为信誉度足够，可以融资：



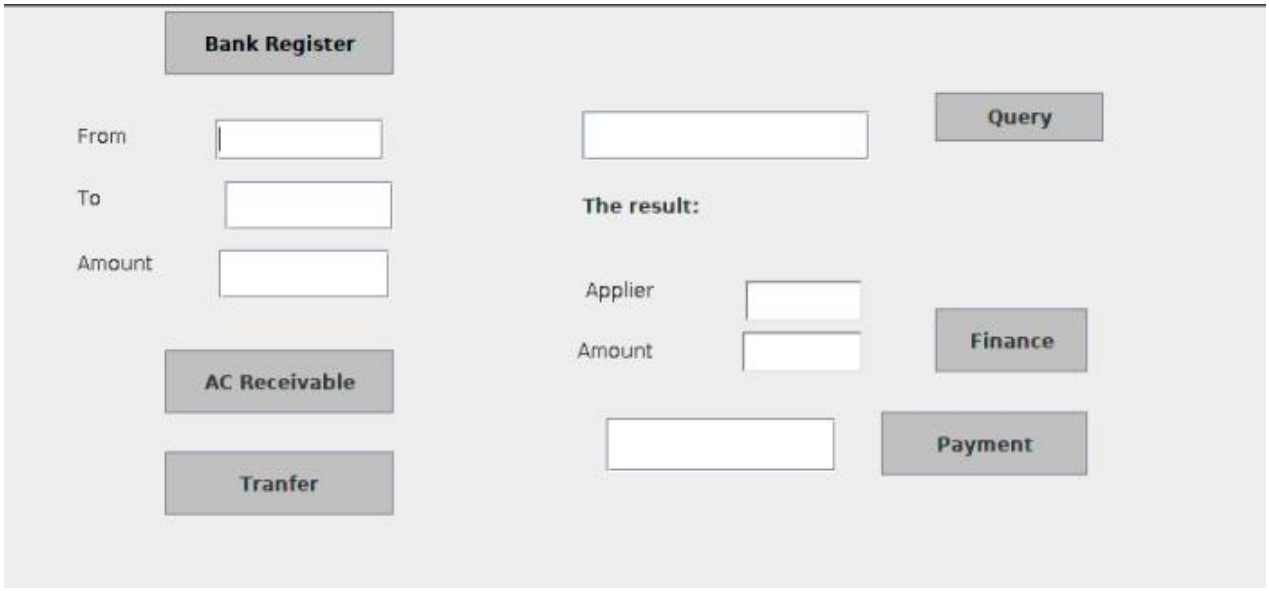
接着轮毂厂 lg 进行融资，输入申请人 lg 以及融资金额，发现融资成功，说明银行发现 lg 持有 car 的账单并认为信誉度足够，可以融资：



最后实现功能 4，即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。输入有欠款的公司 car，执行 payment 成功，对 car 的账单进行查询，发现现在变成 0，说明已经没有欠款，全部支付完成。



四、 界面展示



五、 心得体会

本次是第一次开发区块链的相关应用，这确实是一次很困难的作业。首先我跟着 fisco 的教程学习了如何开发一个简单的区块链应用，学习并跟着他进行操作后得到了一个简单的项目文件，接着我就在这个基础上实现了自己的区块链应用，完成了供应链金融的相关功能。实现了相应的前端，后端以及链端。

这次作业的第一个难点就是不懂 java，之前完全没有接触过 java，包括项目的运行，代码的编写等等等等都要从头开始学习，还有的问题就是对于供应链金融的区块链应用的整个框架非常不熟悉，虽然上次已经编写好了合约，但是在这次大作业之中还是没有一个区块链应用的整体概念，导致我在一开始对于如何完成非常迷茫，不知道应该如何完成，在阅读相关文档以及咨询了同学之后才渐渐对应用的整体框架有了一些简单的认识，前端的实现也是很困难的部分，我并没有实现过很多前端应用，对于前端的应用也非常不熟悉，虽然很多同学都是利用前端类似 VUE 框架来实现的，但是我确实对于这方面不是很了解，所以我只好通过 JAVA 来编写了一个 UI 并实现了一个界面，总体来说做的不是很美观。

在这次作业中我也收获到了很多，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，也学习到了如何查询官方文档，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。同时我也对 fisco 以及整个区块链应用的框架以及执行等等有了一个比较清晰的认识。在我的理解中区块链就相当于一个特殊的数据库，前端向后端发出请求，然后后端程序通过操作区块链来得到前端需要的返回值或者执行前端要求的操作。但是区块链也与数据库不同，这也使得区块链有越来越多的应用，应用场景也更加广泛。

总的来说，通过本次大作业更加清楚地了解到了区块链的概念及其应用场景，但如果要对区块链有更深入的了解就仍然需要深入学习。

Github 地址：https://github.com/TempterCyn/BlockChainFinalProject