区块链大作业实验报告

年级	专业	学号	姓名
大四	智能科学与技术	16337121	李瑞成

Github Address

https://github.com/RogerLrc/Blockchain

项目背景

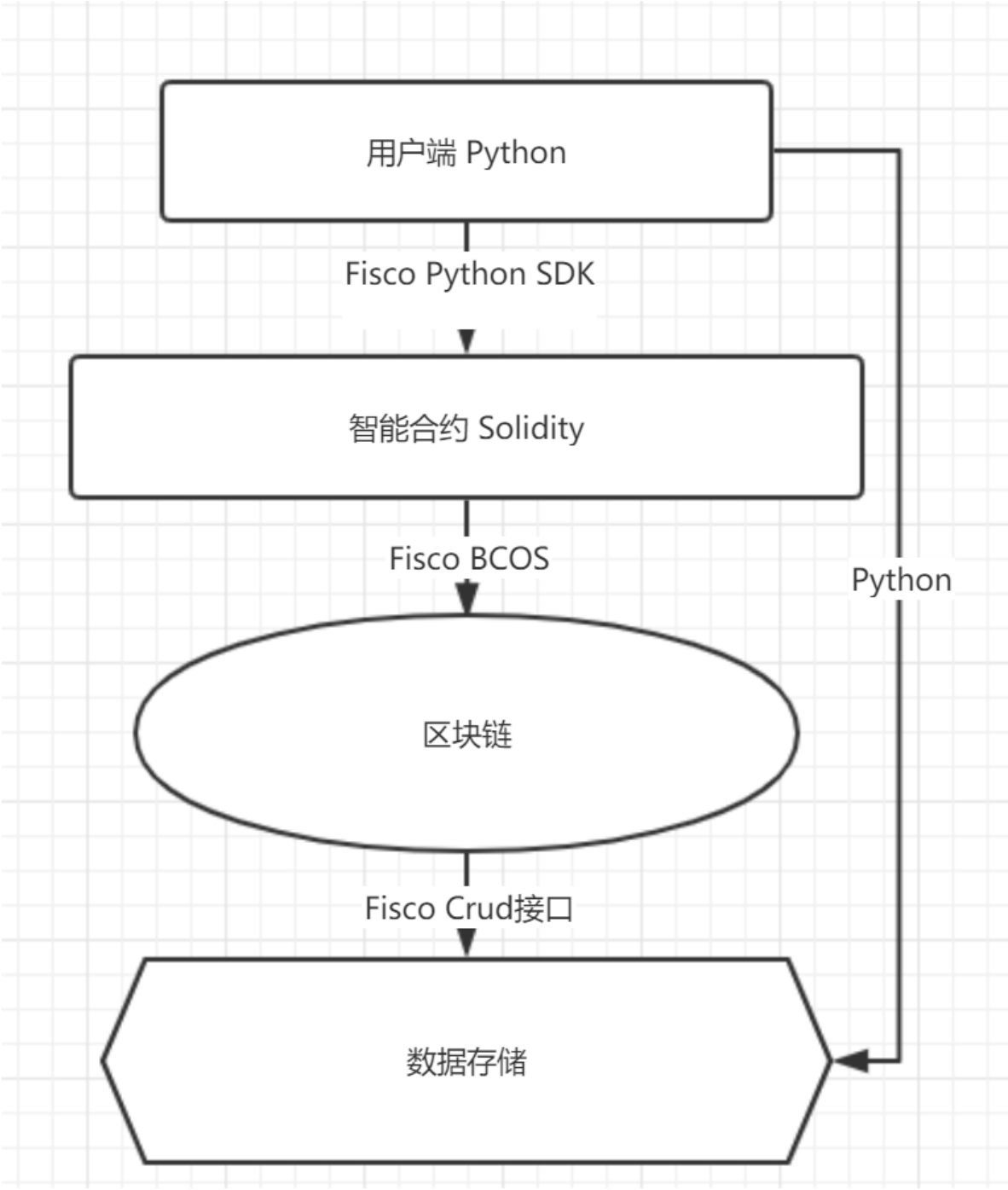
项目任务是基于区块链平台fisco搭建一个供应链系统，供应链上的企业可以通过使用账款单据进行交易，账款单据可以从供应链的中上游向下游企业转移。

具体需要实现的四个功能为：

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。

方案设计

整体设计方案如图：



用户端设计

针对给任务设计两种python对象，company和bank。

Company表示供应链中上下游的企业

```
class Company:

    def __init__(self, name):

    # 获取当前正在进行的交易信息
    def getTransaction(self):

    # 创建贷款账单交易
```

```
    def makeDebt(self, to_name, value, duration):

    # 创建账单转移交易
    def makeTransfer(self, from_name, to_name, value, duration):

    # 创建金融融资交易
    def makeFinance(self, value, duration):

    # 确认交易
    def confirmTrade(self, from_name, value, duration):
```

Bank表示银行等提供融资服务的金融机构

```
class Bank:

    def __init__(self, name):

    # 显示正在进行的融资交易
    def showFinance(self):

    # 创建金融融资
    def makeFinance(self, from_name, value, duration):
```

针对该具体的任务创建了四个账户分别为car，wheel，hub，bank分别代表供应链上中下游的企业和银行。

链端设计

该项目相较于一般的支付任务的关键特征是不会及时支付，付款方会创造一个账款单据，并且该单据可以在下游企业之间进行转让。为此设计了contract合约Book用来存储账款单据并进行相关操作

```
contract Book{
    // 账单创建者地址，银行融资时通过检查账单创建者是否为可信任账户决定是否发放融资
    address public debtMaker;

    // debt用来存储该账单中debtMaker到期应付给中下游企业的账单金额。
    // 如果没有账单转移，debt只存储应还给中游企业的金额。
    // 如果中游企业进行了账单转移，则debt按转移顺序依次添加相应账单。
    int[] debt;
    int value;

    // nameMap将企业用户地址映射为debt中的index
    mapping(address => uint) public nameMap;

    event DebtChanged();
```

```
    event DebtPaid();

    constructor() public{}
    function makeDebt(address debtOwner, int number) public
returns(bool){}
    function transferDebt(address from, address to, int number)
public returns(bool){}
    function payDebt(address debtOwner, int number) public
returns(bool){}

}
```

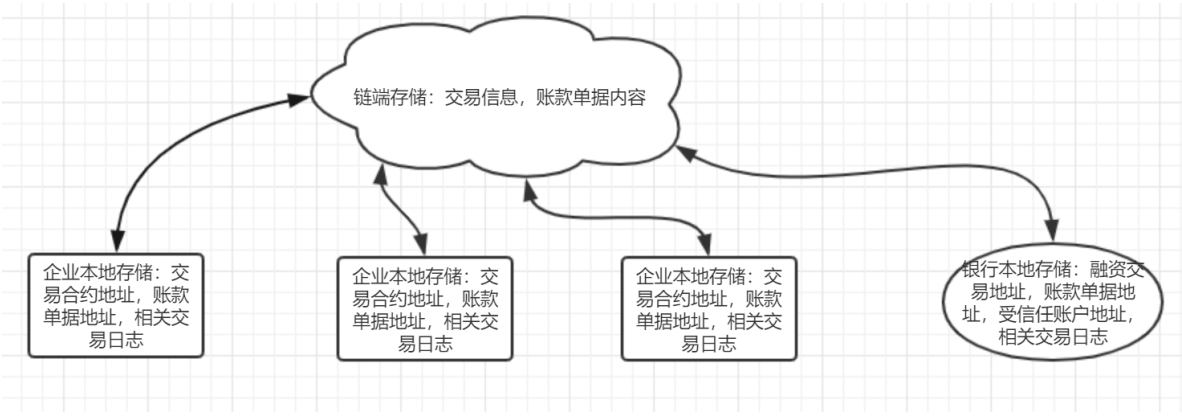
基于该账款单据合约设计相应的交易合约Trade

```
pragma solidity ^0.4.4

contract Trade{
    //账款单据合约的地址
    address book;

    address buyer;
    address seller;
    int value;

    //账款单据有效期
    int expiration;

    //交易类型，分别为账单交易、账单转移交易和银行融资，依次对应功能一二三
    enum TradeType {Debt, Transfer, Finance}
    TradeType type;

    event TradeConfirmed();
    event TradeFinished();

    modifier onlySeller(){
        require(
            msg.sender == seller,
            "Only seller can call this."
        );
        _;
    }

    // 检查账单有效期，有效期过了之后卖方可以从核心企业获取相应欠款，对应归能
四。
    modifier timeExpired(){
        require(
            now >= expiration,
            "Debt time hasn't expired."
        );
```

```
        _;
    }

    // 创建交易，并存储账款单据
   constructor(address _book, int _value, int duration, string _type)
public{}

    // 确认交易金额，对应融资交易银行还要确认账款单据的有效性
    function confirmTrade() public returns(bool){}

    // 账单到期付款，调用Book里的payDebt函数完成付款
    function finishTrade() public onlySeller timeExpired
returns(bool){}

    function toType(string) private returns(TradeType){}

}
```

数据存储设计

具体的存储内容为：



存储分为链上存储和本地存储，链上存储使用了solidity自带的基本数据结构和fisco提供的表结构，而本地存储则使用了python以及相关的库做文件管理。

功能测试

在本地用fisco-bcos创建了一个四个节点的区块链，然后将四个用户分配到四个节点上进行功能测试。因为时间不是很充足，所以没有设计用户界面，采用了命令行进行交互。

上游企业car

创建账款单据并向中游wheel发起交易

```
(python-sdk) sre@ubuntu:~/fisco$ python main.py
Please Enter Your Account Name:  car
Please Enter Your Password:  car
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  1
No transaction in progress
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  2
Format: trade_type from to value duration
Please enter your transaction:  debt car wheel 1000 1
INFO >> compile with solc compiler
INFO >> compile with solc compiler
Trade address:  0xb7a506877cd874b0dc0ed1a43762193b60a7b6d4
Transaction deployed
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  1
OwedDebt wheel 1000 1 0xb7a506877cd874b0dc0ed1a43762193b60a7b6d4
```

中游企业wheel

确认上游交易并利用上游的账款单据向下游企业hub发起转移交易

```
Please Enter Your Account Name:  wheel
Please Enter Your Password:  wheel
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  2
Format: trade_type from to value duration
Please enter your transaction:  confirm car wheel 1000 1
Transaction deployed
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  1
OwnedDebt car 1000 1 0xb7a506877cd874b0dc0ed1a43762193b60a7b6d4

--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  2
Format: trade_type from to value duration
Please enter your transaction:  transfer car hub 500 1
INFO >> compile with solc compiler
Trade address:  0x35ea6ab6035449acc289fdc313835e7c154ddb41
Transaction deployed
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  1
OwnedDebt car 1000 1 0xb7a506877cd874b0dc0ed1a43762193b60a7b6d4
TransferDebt car hub 500 1 0x35ea6ab6035449acc289fdc313835e7c154ddb41
```

下游企业hub

接受中游企业的交易并利用账款单据向银行发起融资

```
(python-sdk) sre@ubuntu:~/fisco$ python3 main.py
Please Enter Your Account Name:  hub
Please Enter Your Password:  hub
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  1
No transaction in progress
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  2
Format: trade_type from to value duration
Please enter your transaction:  confirm wheel hub 1000 1
Transaction deployed
--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  1
OwnedDebt wheel 1000 1 0x35ea6ab6035449acc289fdc313835e7c154ddb41

--------------------
1 - Show transcation in progress
2 - Make a new transaction
3 - quit
--------------------
Please enter your option:  2
Format: trade_type from to value duration
Please enter your transaction:  transfer hub bank 500 1
INFO >> compile with solc compiler
Trade address:  0xbbe16a7054c0f1d3b71f4efdb51b9e40974ad651
Transaction deployed
--------------------
```

银行bank

确认下游企业帐款单据的有效性并发放融资

```
Please Enter Your Account Name:  bank
Please Enter Your Password:  bank
--------------------
1 - Show finance in progress
2 - Make a new finance
3 - quit
--------------------
Please enter your option:  2
Format: from value duration
Please enter your finance: hub 500 1
--------------------
1 - Show finance in progress
2 - Make a new finance
3 - quit
--------------------
Please enter your option:  1
Finance hub 500 1 0xbbe16a7054c0f1d3b71f4efdb51b9e40974ad651

--------------------
1 - Show finance in progress
2 - Make a new finance
3 - quit
--------------------
```

部分链上交易日志


```
>Confirm Trade-----------------
{'blockHash': '0xace355b9ea9641d27faaffb9ba5ac0aa7a223392266de1696cdec302267e31fe',
'blockNumber': '0x2c', 'contractAddress':
'0x0000000000000000000000000000000000000000', 'from':
'0x95198b93705e394a916579e048c8a32ddfb900f7', 'gasUsed': '0x569f', 'input':
'0xa124ab0400000000000000000000000000000000000000000000000000000000000003e8', 'logs':
[], 'logsBloom':
'0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
'output': '0x0000000000000000000000000000000000000000000000000000000000000000',
'status': '0x0', 'to': '0x35ea6ab6035449acc289fdc313835e7c154ddb41',
'transactionHash':
'0x048975562cf0e2fc3230abdda9483e7a442d182f342b88d844fd21224ac4b3ce',
'transactionIndex': '0x0'}

>Deploy Trade-------------------
{'blockHash': '0x9def099d560ee5fb1df32ae9d963276b6b2d312986b9d4d7d21ff2f221c6d59f',
'blockNumber': '0x2d', 'contractAddress':
'0xbbe16a7054c0f1d3b71f4efdb51b9e40974ad651', 'from':
'0x95198b93705e394a916579e048c8a32ddfb900f7', 'gasUsed': '0xe2120', 'input':
'0x608060405234801561001057600080fd5b50610cf88061002060003960000f3006080604052600436106
'logs': [], 'logsBloom':
'0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
'output': '0x', 'status': '0x0', 'to': '0x0000000000000000000000000000000000000000',
'transactionHash':
'0x17cbce6ddbb31466d672a19b4cc8bc7d61d7753e6fe83254a1cca6c19a35d46a',
'transactionIndex': '0x0'}
```


心得体会

Python SDK使用

因为python sdk没有详细的使用教程，所以需要自己看相关的源码和example。一开始不是很习惯这种学习SDK的方式，花了不少时间。并且在使用过程中也遇到了不少问题，比如python sdk自带的合约部署函数deploy不能传递constructor的参数，所以自己尝试重写了deploy函数，但是发现会代码一直提示传入的合约地址不合法，并且会造成内存泄漏。另外sdk接口似乎没有提供明确的python类型到solidity类型的转换，导致一开始测试合约时python类中自定义的一个类型始终无法传入合约。通过更加仔细地阅读源码，以上问题得到了解决。

总的来说这是一次很好的学习的经历，SDK的代码整体写的非常清楚，重要部分也给出了注释，非常感谢SDK的撰写者，我从中学到了很多。

用户的权限控制和隐私保护

我一开始的设想是在用户端和链端都明确地控制不同用户的权限，但是实施起来发现用户端的权限控制比较容易，而链端的权限控制因为对区块链知识掌握不足并没有实现。我的链端实现方式是上游企业的账款单据可以被所有该供应链上的所有企业获取，这不利于上游企业的隐私保护。比如一个账款单据的转移顺序为：A-B-C-D-E，那么对于E来说，他可以获取之前的所有转移交易，而比较好的做法应该是控制该账单对于E来说为：A----D-E，即E只应该知道将账单转移给他的企业D以及账单发起企业A。希望后续对区块链的学习能解决这个问题。