Block Chain Final Project Report

> **区块链大作业：基于区块链的供应链金融平台**
>
> 区块链天然具有防篡改，可追溯等特性，这些特性决定其更容易受金融领域的青睐

小组成员： 18342102 伍浩源 18342137 赵文序

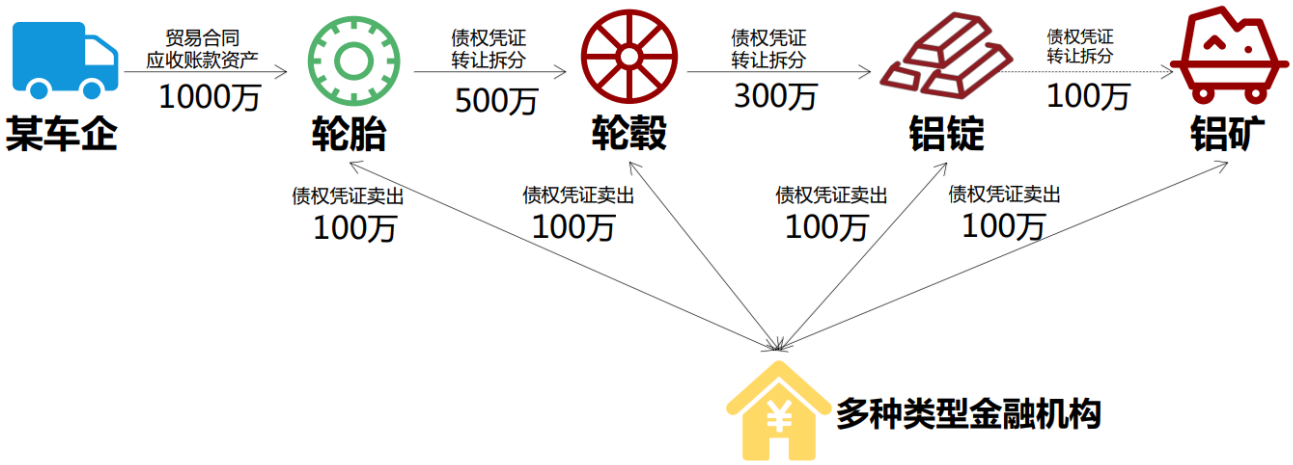
一、项目背景

1.1.总体介绍

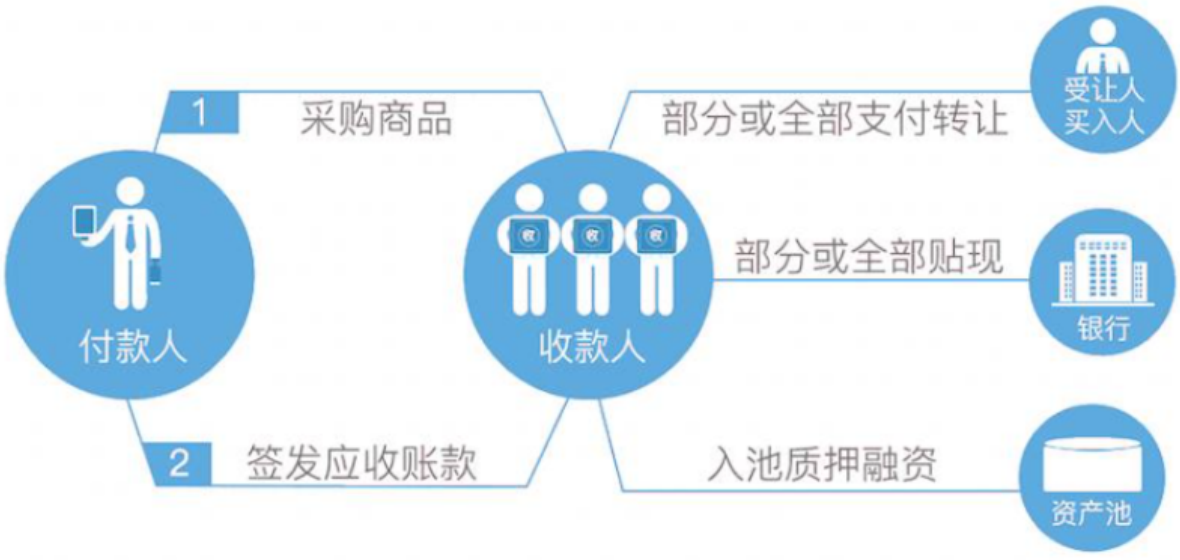
基于区块链、智能合约等，实现基于区块链的供应链金融平台。

1.2 必要功能

基于已有的开源区块链系统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。

1.3 加分项

- 功能：除了必要功能之外，实现更多的功能，限定：供应链金融领域相关
- 底层：改进区块链平台底层或自行开发区块链（共识算法等）
- 合约：实现链上数据隐私保护（同态加密、属性加密等）
- 前端：友好高效的用户界面
- 其他有挑战性的创新

二、方案设计

2.1 引言

区块链具有**不可修改**的特性，智能合约具有**运行内容不可篡改**且**记录**的特性，可以很方便的查找到可信任的账单信息，从而让小公司可以使用欠债方为大公司的债券而在银行进行融资或者使用债券转让的方式去购买其他公司的产品。而在智能合约中只需要设计一个债券系统，让小公司能使用债券向另一个公司购买产品，同时这

另一个公司持有的债券的欠款方不是这个小公司，而是这个小公司持有的债券的大公司欠款方。这样就能实现信用的传递，同时债权人和欠款人的信息都可以很方便的查看和管理。

2.2 设计内容

- 智能合约设计
- 链端合约部署
- 后端以及API接口设计
- 前端界面设计

2.3 具体实现

设计伊始，我们针对四大功能进行了面向对象抽象：

1. 确定的对象：应收款账单
2. 对象所具有的**属性**：债主、欠债人、企业名称、开始日期、还款日期、应收款金额
3. 对象所具有的**行为**（从功能抽象）：签名应收款账单、应收款转让、评估企业风承、支付余款

具体深入设计：

系统由很多个公司company组成,其中有1个银行公司,其他都为普通公司,两者能使用的函数不同.

- 银行
    - 发行货币：即给某个公司发钱
    - 公司查询：查看指定公司的信息,公司金钱等,公司名字,公司地址,公司类型等
- 公司
    - 公司注册：唯一的公司名称作为公司标识，包含初始注册资金
    - 公司转账：本公司转账给其他公司
    - 查询信息：查看公司信息
    - 查询账单：查看公司的账单信息
    - 创建账单：创建欠款人的账单
    - 账单转移：转移债权人的账单
    - 偿还账单：偿还欠款人的账单
    - 银行融资：欠款人用自己的账单向银行申请融资

2.3.1 智能合约

第一步.设计智能合约

存储设计

FISCO BCOS提供合约CRUD接口开发模式，可以通过合约创建表，并对创建的表进行增删改查操作。

接口设计

按照业务的设计目标，需要实现资产注册，转账，查询功能，对应功能的接口如下：

- 合约接口

```
// 加载合约地址
public static Asset load(String contractAddress, Client client, CryptoKeyPair
credential) {
    return new Asset(contractAddress, client, credential);
}

// 部署合约
public static Asset deploy(Client client, CryptoKeyPair credential) throws
ContractException {
    return deploy(Asset.class, client, credential,
getBinary(client.getCryptoSuite()), "");
}
```

- 资产注册，转账，查询接口

```
// 用户信息上链接口
public static class RegisterEventEventResponse {
    public TransactionReceipt.Logs log;

    public BigInteger ret;

    public String account;

    public BigInteger asset_value;
}
```

```
// 资产转移接口
public static class TransferEventEventResponse {
    public TransactionReceipt.Logs log;

    public BigInteger ret;

    public String from_account;

    public String to_account;

    public BigInteger amount;
}
```

```
// 交易添加接口
public static class AddTransactionEventEventResponse {
    public TransactionReceipt.Logs log;

    public BigInteger ret;
```

```
    public String id;

    public String acc1;

    public String acc2;

    public BigInteger money;
}
```

```
// 更新交易接口
public static class UpdateTransactionEventEventResponse {
    public TransactionReceipt.Logs log;

    public BigInteger ret;

    public String id;

    public BigInteger money;
}
```

```
// 欠条拆分接口
public static class SplitTransactionEventEventResponse {
    public TransactionReceipt.Logs log;

    public BigInteger ret;

    public String old_id;

    public String new_id;

    public String acc;

    public BigInteger money;
}
```

第二步.开发源码

根据我们第一步的存储和接口设计，创建一个Asset的智能合约，实现注册、转账、查询功能，并引入一个叫Table的系统合约，这个合约提供了CRUD接口。合约内容具体如下：

```
contract TableFactory {
    function openTable(string memory) public view returns (Table) {} //open table
    function createTable(string memory, string memory, string memory) public
returns (int256) {} //create table
}
```

```
//select condition
contract Condition {
    function EQ(string memory, int256) public view{}
    function EQ(string memory, string memory) public view{}

    function NE(string memory, int256) public view{}
    function NE(string memory, string memory) public view{}

    function GT(string memory, int256) public view{}
    function GE(string memory, int256) public view{}

    function LT(string memory, int256) public view{}
    function LE(string memory, int256) public view{}

    function limit(int256) public view{}
    function limit(int256, int256) public view{}
}

//one record
contract Entry {
    function getInt(string memory) public view returns (int256) {}
    function getUInt(string memory) public view returns (uint256) {}
    function getAddress(string memory) public view returns (address) {}
    function getBytes64(string memory) public view returns (bytes1[64] memory) {}
    function getBytes32(string memory) public view returns (bytes32) {}
    function getString(string memory) public view returns (string memory) {}

    function set(string memory, int256) public {}
    function set(string memory, uint256) public {}
    function set(string memory, string memory) public {}
    function set(string memory, address) public {}
}

//record sets
contract Entries {
    function get(int256) public view returns (Entry) {}
    function size() public view returns (int256) {}
}

//Table main contract
contract Table {
    function select(string memory, Condition) public view returns (Entries) {}
    function insert(string memory, Entry) public returns (int256) {}
    function update(string memory, Entry, Condition) public returns (int256) {}
    function remove(string memory, Condition) public returns (int256) {}

    function newEntry() public view returns (Entry) {}
    function newCondition() public view returns (Condition) {}
}

contract KVTableFactory {
    function openTable(string memory) public view returns (KVTable) {}
    function createTable(string memory, string memory, string memory) public
```

```
returns (int256) {}
}

//KVTable per permiary key has only one Entry
contract KVTable {
    function get(string memory) public view returns (bool, Entry) {}
    function set(string memory, Entry) public returns (int256) {}
    function newEntry() public view returns (Entry) {}
}
```

第三步.编译智能合约

.sol的智能合约需要编译成ABI和BIN文件才能部署至区块链网络上。有了这两个文件即可凭借Java SDK进行合约部署和调用。但这种调用方式相对繁琐，需要用户根据合约ABI来传参和解析结果。为此，控制台提供的编译工具不仅可以编译出ABI和BIN文件，还可以自动生成一个与编译的智能合约同名的合约Java类。这个Java类是根据ABI生成的，帮助用户解析好了参数，提供同名的方法。当应用需要部署和调用合约时，可以调用该合约类的对应方法，传入指定参数即可。使用这个合约Java类来开发应用，可以极大简化用户的代码。

2.3.2 链端合约

- 合约基本构建

```
constructor() public {
  // 构造函数中创建t_asset表
  createTable();
}

function createTable() private {
  TableFactory tf = TableFactory(0x1001);
  // 资产管理表, key : account, field : asset_value
  // |   资产账户(主键)      |     信用额度       |
  // |-------------------- |-------------------|
  // |        account      |    asset_value    |
  // |---------------------|-------------------|
  //
  // 创建表
  tf.createTable("t_asset", "account", "asset_value");
  // 交易记录表, key: id, field: acc1, acc2, money, status
  // | 交易单号(key) | 债主 | 借债人 | 债务金额 |   状态   |
  // |-------------|------|-------|---------|---------|
  // |     id      | acc1 | acc2  |  money  | status  |
  // |-------------|------|-------|---------|---------|
  tf.createTable("t_transaction", "id","acc1, acc2, money, status");
}

// 返回t_asset
function openAssetTable() private returns(Table) {
  TableFactory tf = TableFactory(0x1001);
  Table table = tf.openTable("t_asset");
  return table;
```

```
}

// 返回t_transaction
function openTransactionTable() private returns(Table) {
  TableFactory tf = TableFactory(0x1001);
  Table table = tf.openTable("t_transaction");
  return table;
}

function select(string account) public constant returns(int256, int256) {
  // 打开表
  Table table = openAssetTable();
  // 查询
  Entries entries = table.select(account, table.newCondition());
  int256 asset_value = 0;
  if (0 == uint256(entries.size())) {
      return (-1, asset_value);
  } else {
      Entry entry = entries.get(0);
      return (0, int256(entry.getInt("asset_value")));
  }
}

function select_transaction(string id) public constant returns(int256[],
bytes32[]) {
  // 打开表
  Table table = openTransactionTable();
  Entries entries = table.select(id, table.newCondition());

  int256[] memory int_list = new int256[](3);
  bytes32[] memory str_list = new bytes32[](2);
  if (0 == uint256(entries.size())) {
      int_list[0] = -1;
      return (int_list, str_list);
  } else {
      Entry entry = entries.get(0);
      int_list[1] = entry.getInt("money");
      int_list[2] = entry.getInt("status");
      str_list[0] = entry.getBytes32("acc1");
      str_list[1] = entry.getBytes32("acc2");
      return (int_list, str_list);
  }
}

function register(string account, int256 asset_value) public returns(int256){
  int256 ret_code = 0;
  int256 ret = 0;
  int256 temp_asset_value = 0;

  (ret, temp_asset_value) = select(account);
  if(ret != 0) {
      Table table = openAssetTable();

      Entry entry = table.newEntry();
```

```
        entry.set("account", account);
        entry.set("asset_value", int256(asset_value));

        int count = table.insert(account, entry);
        if (count == 1)
            ret_code = 0;
        else
            ret_code = -2;
    } else
        ret_code = -1;

    emit RegisterEvent(ret_code, account, asset_value);

    return ret_code;
}
```

- 添加交易

```
function addTransaction(string id, string acc1, string acc2, int256 money) public
returns(int256){
    int256 ret_code = 0;
    int256 ret = 0;
    bytes32[] memory str_list = new bytes32[](2);
    int256[] memory int_list = new int256[](3);

    (int_list, str_list) = select_transaction(id);
    if(int_list[0] != int256(0)) {
        Table table = openTransactionTable();

        Entry entry0 = table.newEntry();
        entry0.set("id", id);
        entry0.set("acc1", acc1);
        entry0.set("acc2", acc2);
        entry0.set("money", int256(money));
        entry0.set("status", int256(money));

        int count = table.insert(id, entry0);
        if (count == 1) {
            ret = transfer(acc2,acc1,money);
            if(ret != 0) ret_code = -3;
            else ret_code = 0;
        } else ret_code = -2;
    } else ret_code = -1;

    emit AddTransactionEvent(ret_code, id, acc1, acc2, money);

    return ret_code;
}
```

- 更新交易信息

```
function updateTransaction(string id, int256 money) public returns(int256,
string[]){
    int256 ret_code = 0;
    bytes32[] memory str_list = new bytes32[](2);
    int256[] memory int_list = new int256[](3);
    string[] memory acc_list = new string[](2);
    (int_list, str_list) = select_transaction(id);
    acc_list[0] = byte32ToString(str_list[0]);
    acc_list[1] = byte32ToString(str_list[1]);

    if(int_list[0] == 0) {
        if(int_list[2] < money){
            ret_code = -2;
            emit UpdateTransactionEvent(ret_code, id, money);
            return (ret_code, acc_list);
        }

        Table table = openTransactionTable();

        Entry entry0 = table.newEntry();
        entry0.set("id", id);
        entry0.set("acc1", byte32ToString(str_list[0]));
        entry0.set("acc2", byte32ToString(str_list[1]));
        entry0.set("money", int_list[1]);
        entry0.set("status", (int_list[2] - money));

        int count = table.update(id, entry0, table.newCondition());
        if(count != 1) {
            ret_code = -3;
            emit UpdateTransactionEvent(ret_code, id, money);
            return (ret_code,acc_list);
        }

        int256 temp =
transfer(byte32ToString(str_list[0]),byte32ToString(str_list[1]),money);
        if(temp != 0){
            ret_code = -4 * 10 + temp;
            emit UpdateTransactionEvent(ret_code, id, money);
            return (ret_code,acc_list);
        }

        ret_code = 0;

    } else ret_code = -1;

    emit UpdateTransactionEvent(ret_code, id, money);

    return (ret_code,acc_list);
}
```

- 转移资产

```
function transfer(string from_account, string to_account, int256 amount) public
returns(int256) {
    int ret_code = 0;
    int256 ret = 0;
    int256 from_asset_value = 0;
    int256 to_asset_value = 0;

    (ret, from_asset_value) = select(from_account);
    if(ret != 0) {
        ret_code = -1;
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;

    }

    (ret, to_asset_value) = select(to_account);
    if(ret != 0) {
        ret_code = -2;
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    if(from_asset_value < amount) {
        ret_code = -3;
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    if (to_asset_value + amount < to_asset_value) {
        ret_code = -4;
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    Table table = openAssetTable();

    Entry entry0 = table.newEntry();
    entry0.set("account", from_account);
    entry0.set("asset_value", int256(from_asset_value - amount));

    int count = table.update(from_account, entry0, table.newCondition());
    if(count != 1) {
        ret_code = -5;
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    Entry entry1 = table.newEntry();
    entry1.set("account", to_account);
    entry1.set("asset_value", int256(to_asset_value + amount));

    table.update(to_account, entry1, table.newCondition());
```

```
    emit TransferEvent(ret_code, from_account, to_account, amount);

    return ret_code;
}
```

- 欠条拆分

```
function splitTransaction(string old_id, string new_id, string acc, int256 money)
public returns(int256) {
    int256 ret_code = 0;
    int256 ret = 0;
    int temp = 0;
    bytes32[] memory str_list = new bytes32[](2);
    int256[] memory int_list = new int256[](3);
    string[] memory acc_list = new string[](2);
    (int_list, str_list) = select_transaction(old_id);

    if(int_list[0] == 0) {
        (ret, temp) = select(acc);
        if(ret != 0) {
            ret_code = -5;
            emit SplitTransactionEvent(ret_code, old_id, new_id, acc, money);
            return ret_code;
        }

        if(int_list[2] < money){
            ret_code = -2;
            emit SplitTransactionEvent(ret_code, old_id, new_id, acc, money);
            return ret_code;
        }

        (ret,acc_list) = updateTransaction(old_id, money);
        if (ret != 0) {
            ret_code = -4;
            emit SplitTransactionEvent(ret_code, old_id, new_id, acc, money);
            return ret_code;
        }
        ret = addTransaction(new_id, acc, byte32ToString(str_list[1]), money);
        if (ret != 0) {
            ret_code = -3;
            emit SplitTransactionEvent(ret_code, old_id, new_id, acc, money);
            return ret_code;
        }

    } else ret_code = -1;


    emit SplitTransactionEvent(ret_code, old_id, new_id, acc, money);
    return ret_code;
}
```

2.3.3 后端设计

后端设计主要在AssetClient中实现

初始化代码的主要功能为构造Client与CryptoKeyPair对象，这两个对象在创建对应的合约类对象(调用合约类的deploy或者load函数)时需要使用。

```
static Logger logger = LoggerFactory.getLogger(AssetClient.class);

private BcosSDK bcosSDK;
private Client client;
private CryptoKeyPair cryptoKeyPair;

public void initialize() throws Exception {
  @SuppressWarnings("resource")
  ApplicationContext context =
      new ClassPathXmlApplicationContext("classpath:applicationContext.xml");
  bcosSDK = context.getBean(BcosSDK.class);
  client = bcosSDK.getClient(1);
  cryptoKeyPair = client.getCryptoSuite().createKeyPair();
  client.getCryptoSuite().setCryptoKeyPair(cryptoKeyPair);
  logger.debug("create client for group1, account address is " +
cryptoKeyPair.getAddress());
}
```

- 连接到区块链节点

```
public void deployAssetAndRecordAddr() {

  try {
    Asset asset = Asset.deploy(client, cryptoKeyPair);
    System.out.println(
        " deploy Asset success, contract address is " +
asset.getContractAddress());

    recordAssetAddr(asset.getContractAddress());
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();
    System.out.println(" deploy Asset contract failed, error message is  " +
e.getMessage());
  }
}

public void recordAssetAddr(String address) throws FileNotFoundException,
IOException {
  Properties prop = new Properties();
  prop.setProperty("address", address);
  final Resource contractResource = new ClassPathResource("contract.properties");
  FileOutputStream fileOutputStream = new
```

```
FileOutputStream(contractResource.getFile());
  prop.store(fileOutputStream, "contract address");
}

public String loadAssetAddr() throws Exception {
  // load Asset contact address from contract.properties
  Properties prop = new Properties();
  final Resource contractResource = new ClassPathResource("contract.properties");
  prop.load(contractResource.getInputStream());

  String contractAddress = prop.getProperty("address");
  if (contractAddress == null || contractAddress.trim().equals("")) {
    throw new Exception(" load Asset contract address failed, please deploy it
first. ");
  }
  logger.info(" load Asset address from contract.properties, address is {}",
contractAddress);
  return contractAddress;
}
```

- 获取交易信息

```
public boolean queryAssetAmount(String assetAccount) {
  try {
    String contractAddress = loadAssetAddr();

    Asset asset = Asset.load(contractAddress, client, cryptoKeyPair);
    Tuple2<BigInteger, BigInteger> result = asset.select(assetAccount);
    if (result.getValue1().compareTo(new BigInteger("0")) == 0) {
      System.out.printf(" Your credit limit is: %s \n", result.getValue2());
      return true;
    } else {
      System.out.printf(" %s asset account is not exist \n", assetAccount);
      return false;
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();
    logger.error(" queryAssetAmount exception, error message is {}",
e.getMessage());

    System.out.printf(" query asset account failed, error message is %s\n",
e.getMessage());
  }
  return false;
}

public void queryAssetTransaction(String t_id) {
  try {
    String contractAddress = loadAssetAddr();
```

```
    Asset asset = Asset.load(contractAddress, client, cryptoKeyPair);
    Tuple2<List<BigInteger>, List<byte[]>> result =
asset.select_transaction(t_id);
    if (result.getValue1().get(0).compareTo(new BigInteger("0")) == 0) {
      String temp1 = new String(result.getValue2().get(0));
      String temp2 = new String(result.getValue2().get(1));
      System.out.printf("Transaction\n ID: " + t_id + "; Acc1: " + temp1 + ";
Acc2: " + temp2 + "; Money: " + result.getValue1().get(1) + "; Current: " +
result.getValue1().get(2) + "\n");
    } else {
      System.out.printf("Transaction %s is not exist \n", t_id);
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();
    logger.error(" queryAssetAmount exception, error message is {}",
e.getMessage());

    System.out.printf(" query asset account failed, error message is %s\n",
e.getMessage());
  }
}
```

- 个人信息上链

```
public void registerAssetAccount(String assetAccount, BigInteger amount) {
  try {
    String contractAddress = loadAssetAddr();

    Asset asset = Asset.load(contractAddress, client, cryptoKeyPair);
    TransactionReceipt receipt = asset.register(assetAccount, amount);
    List<Asset.RegisterEventEventResponse> response =
asset.getRegisterEventEvents(receipt);
    if (!response.isEmpty()) {
      if (response.get(0).ret.compareTo(new BigInteger("0")) == 0) {
        System.out.printf(
            " register asset account success => asset: %s, value: %s \n",
assetAccount, amount);
      } else {
        System.out.printf(
            " register asset account failed, ret code is %s \n",
response.get(0).ret.toString());
      }
    } else {
      System.out.println(" event log not found, maybe transaction not exec. ");
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();

    logger.error(" registerAssetAccount exception, error message is {}",
```

```
e.getMessage());
    System.out.printf(" register asset account failed, error message is %s\n",
e.getMessage());
  }
}
```

- 添加交易信息

```
public void addAssetTransaction(String t_id, String acc1, String acc2, BigInteger
money) {
  try {
    String contractAddress = loadAssetAddr();

    Asset asset = Asset.load(contractAddress, client, cryptoKeyPair);
    TransactionReceipt receipt = asset.addTransaction(t_id, acc1, acc2, money);
    List<AddTransactionEventEventResponse> response =
asset.getAddTransactionEventEvents(receipt);
    if (!response.isEmpty()) {
      if (response.get(0).ret.compareTo(new BigInteger("0")) == 0) {
        System.out.printf(" Add transaction success! id:" + t_id+"\n");
      } else {
        System.out.printf(" Add transaction failed, ret code is %s \n",
            response.get(0).ret.toString());
      }
    } else {
      System.out.println(" event log not found, maybe transaction not exec. ");
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();

    logger.error(" registerAssetAccount exception, error message is {}",
e.getMessage());
    System.out.printf(" register asset account failed, error message is %s\n",
e.getMessage());
  }
}
```

- 更新交易信息

```
public void updateAssetTransaction(String t_id, BigInteger money) {
  try {
    String contractAddress = loadAssetAddr();
    Asset asset = Asset.load(contractAddress, client, cryptoKeyPair);
    TransactionReceipt receipt = asset.updateTransaction(t_id, money);
    List<UpdateTransactionEventEventResponse> response =
asset.getUpdateTransactionEventEvents(receipt);
    // Tuple2<BigInteger, List<String>> result =
```

```
asset.getUpdateTransactionOutput(asset.updateTransaction(t_id, money).send());

    if (!response.isEmpty()) {
      if (response.get(0).ret.compareTo(new BigInteger("0")) == 0) {
        System.out.printf(" Update transaction success.\n" );
      } else {
        System.out.printf(" Update transaction failed, ret code is %s \n",
            response.get(0).ret.toString());
      }
    } else {
      System.out.println(" event log not found, maybe transaction not exec. ");
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();

    logger.error(" registerAssetAccount exception, error message is {}",
e.getMessage());
    System.out.printf(" register asset account failed, error message is %s\n",
e.getMessage());
  }
}
```

- 资产转移

```
public void transferAsset(String fromAssetAccount, String toAssetAccount,
BigInteger amount) {
  try {
    String contractAddress = loadAssetAddr();
    Asset asset = Asset.load(contractAddress, client, cryptoKeyPair);
    TransactionReceipt receipt = asset.transfer(fromAssetAccount, toAssetAccount,
amount);
    List<Asset.TransferEventEventResponse> response =
asset.getTransferEventEvents(receipt);
    if (!response.isEmpty()) {
      if (response.get(0).ret.compareTo(new BigInteger("0")) == 0) {
        System.out.printf(
            " transfer success => from_asset: %s, to_asset: %s, amount: %s \n",
            fromAssetAccount, toAssetAccount, amount);
      } else {
        System.out.printf(
            " transfer asset account failed, ret code is %s \n",
response.get(0).ret.toString());
      }
    } else {
      System.out.println(" event log not found, maybe transaction not exec. ");
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();

    logger.error(" registerAssetAccount exception, error message is {}",
```

```
e.getMessage());
    System.out.printf(" register asset account failed, error message is %s\n",
e.getMessage());
  }
}
```

- 欠条拆分

```
public void splitAssetTransaction(String old_id, String new_id, String acc,
BigInteger money) {
  try {
    String contractAddress = loadAssetAddr();
    Asset asset = Asset.load(contractAddress, client, cryptoKeyPair);
    TransactionReceipt receipt = asset.splitTransaction(old_id, new_id, acc,
money);
    List<SplitTransactionEventEventResponse> response =
asset.getSplitTransactionEventEvents(receipt);
    if (!response.isEmpty()) {
      if (response.get(0).ret.compareTo(new BigInteger("0")) == 0) {
        System.out.printf(" Split transaction success! old_id: "+ old_id +"
new_id: "+new_id+"\n");
      } else {
        System.out.printf(" Split transaction failed, ret code is %s \n",
            response.get(0).ret.toString());
      }
    } else {
      System.out.println(" event log not found, maybe transaction not exec. ");
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    // e.printStackTrace();

    logger.error(" registerAssetAccount exception, error message is {}",
e.getMessage());
    System.out.printf(" register asset account failed, error message is %s\n",
e.getMessage());
  }
}
```

- 使用说明

```
public static void Usage() {
  System.out.println(" Usage:");
  System.out.println(
      "\t java -cp conf/:lib/*:apps/* org.fisco.bcos.asset.client.AssetClient
deploy");
  System.out.println(
      "\t java -cp conf/:lib/*:apps/* org.fisco.bcos.asset.client.AssetClient
```

```
query account");
  System.out.println(
      "\t java -cp conf/:lib/*:apps/* org.fisco.bcos.asset.client.AssetClient
register account value");
  System.out.println(
      "\t java -cp conf/:lib/*:apps/* org.fisco.bcos.asset.client.AssetClient
transfer from_account to_account amount");
  System.exit(0);
}
```

2.3.4 交互界面设计

- 登陆界面设计

```
public boolean login()
{
  int choice;
  String account, password, again;
  Console console = System.console();
  System.out.println("------Welecome to the FISCO-BCOS Trade Platform------\n");
  System.out.println("Plz enter:\n1:SIGN IN\t2:SIGN UP\t0:quit()");
  if (scanner.hasNextInt()){
      choice = scanner.nextInt();
      switch(choice){
          case 0:
              this.status = false;
              return false;

          case 1:
              account = (String)scanner.nextLine();
              System.out.print("------SIGN IN------\nID: ");
              account = (String)scanner.nextLine();
              System.out.print("Password:");
              password = new String(console.readPassword());
              if (map.get(account)!=null && map.get(account).compareTo(password)
== 0) {
                  current = account;
                  System.out.print("Sign in successfully! Wait for key...");
                  again = (String)scanner.nextLine();
                  return true;
              } else {
                  System.out.print("No account or wrong password! Wait for
key...");
                  again = (String)scanner.nextLine();
                  return false;
              }

          case 2:
              account = (String)scanner.nextLine();
              System.out.print("------REGISTER------\nID: ");
              account = (String)scanner.nextLine();
```

```
                System.out.print("Password:");
                password = new String(console.readPassword());
                System.out.print("Input Again:");
                again = new String(console.readPassword());
                if (password.compareTo(again)==0 && map.get(account)==null) {
                    map.put(account, password);
                    writeToFile();
                    readFile();
                    System.out.print("Register success! Wait for key...");
                    again = (String)scanner.nextLine();
                    return false;
                } else {
                    System.out.print("Register failed! Wait for key...");
                    again = (String)scanner.nextLine();
                    return false;
                }



            default:
                System.out.print("Invalid input! Wait for key...");
                again = (String)scanner.nextLine();
                return false;
        }
    }
    else {
        System.out.print("Invalid input! Wait for key...");
        return false;
    }
}
```

- 交互界面主要设计

```
Scanner scanner = new Scanner(System.in);
CLI platform = new CLI();


AssetClient client = new AssetClient();
client.initialize();
client.deployAssetAndRecordAddr();
int transaction_id = 1000000;

while (platform.getStatus()) {
  while (platform.getStatus() == true && platform.login() == false);

  if (!platform.getStatus()) break;

  String x;
  Map<String,String> map = platform.getMap();

  for (String key : map.keySet()) {
    if (client.queryAssetAmount(key) == false) {
```

```
        if (key.compareTo("bank") != 0) {
          if (key.compareTo("acc0") == 0)
            x = "10000";
          else
            x = "1000";
        } else
          x = "1000000";

        client.registerAssetAccount(key,new BigInteger(x));
      }
    }

    platform.clearFile();
    boolean judge = true;

    while(judge){
      platform.msg();
      int choice, int1, int2;
      String str1,str2,str3, str4;

      if(scanner.hasNextInt()){
        choice = scanner.nextInt();
        switch(choice){
          case 0:
            platform.setCurrentNull();
            judge = false;

            break;
          case 1:
            System.out.println("------Check my credit------\n");
            client.queryAssetAmount(platform.getCurrent());
            System.out.print("Wait for key...");
            str4 = (String)scanner.nextLine();

            break;
          case 2:
            System.out.println("------Trade with others------\n");
            str1 = (String)scanner.nextLine();
            System.out.print("Trader Account: ");
            str1 = (String)scanner.nextLine();
            System.out.print("Transaction Amount: ");
            int1 = scanner.nextInt();
            str3 = int1+"";
            str2 = transaction_id +"";
            transaction_id += 1;
            client.addAssetTransaction(str2,str1, platform.getCurrent() ,new
BigInteger(str3));
            System.out.print("Wait for key...");
            str4 = (String)scanner.nextLine();

            break;

          case 3:
            System.out.println("------Financing/Loan------\n");
```

```
            str1 = (String)scanner.nextLine();
            System.out.print("Financing/Loan Amount: ");
            int1 = scanner.nextInt();
            str1 = transaction_id +"";
            transaction_id += 1;
            str2 = int1+"";
            client.addAssetTransaction(str1, "bank", platform.getCurrent(),new
  BigInteger(str2));
            System.out.print("Wait for key...");
            str4 = (String)scanner.nextLine();

            break;

          case 4:
            System.out.println("------IOU Spilt------\n");
            str1 = (String)scanner.nextLine();
            System.out.print("Beneficiary: ");
            str1 = (String)scanner.nextLine();
            System.out.print("Original ID: ");
            str2 = (String)scanner.nextLine();
            System.out.print("Transfer amount: ");
            int1 = scanner.nextInt();
            str3 = transaction_id+"";
            transaction_id+=1;
            str4 = int1 + "";
            client.splitAssetTransaction(str2,str3,str1,new BigInteger(str4));
            System.out.print("Wait for key...");
            str4 = (String)scanner.nextLine();
            break;

          case 5:
            System.out.println("------Transfer/Loan Repayment------\n");
            str1 = (String)scanner.nextLine();
            System.out.print("Trade ID: ");
            str1 = (String)scanner.nextLine();
            System.out.print("Transfer Amount: ");
            int1 = scanner.nextInt();
            str2 = transaction_id +"";
            transaction_id += 1;
            str3 = int1+"";
            client.updateAssetTransaction( str1, new BigInteger(str3));
            System.out.print("Wait for key...");
            str4 = (String)scanner.nextLine();

            break;

          case 6:
            System.out.print("------Trade Check------\n");
            str1 = (String)scanner.nextLine();
            System.out.print("Original ID: ");
            str1 = (String)scanner.nextLine();
            client.queryAssetTransaction(str1);
            System.out.print("Wait for key...");
            str4 = (String)scanner.nextLine();
```

```
        default:
          System.out.print("Invalid input! Wait for key...");
          str1 = (String)scanner.nextLine();
          break;
      }
    }
  }
}
```

三、 加分项

登录注册界面

```
------Welecome to the FISCO-BCOS Trade Platform------

Plz enter:
1:SIGN IN       2:SIGN UP       0:quit()
1
------SIGN IN------
ID: wuhy68
Password:
Sign in successfully! Wait for key...
```

```
------Welecome to the FISCO-BCOS Trade Platform------

Plz enter:
1:SIGN IN       2:SIGN UP       0:quit()
2
------REGISTER------
ID: hahaha
Password:
Input Again:
```

使用MySQL引擎

修改存储配置为MySQL

通过群组配置文件group.[group_id].ini的storage配置项可配置MySQL，详细可参考这里. 本教程中，所有节点均以root用户名连接同一个本机MySQL数据库，真实业务场景中，可按需修改数据库相关配置(包括MySQL的IP和端口，连接MySQL的用户名和密码等)。

```
# 修改存储类型为mysql
sed -i 's/type=rocksdb/type=mysql/g'
~/fisco/nodes/127.0.0.1/node*/conf/group.1.ini

# 配置数据库用户名和密码(本教程中，所有节点均以root的用户名连接同一个数据库，root用户密码为123456)
sed -i 's/db_username=/db_username=root/g'
~/fisco/nodes/127.0.0.1/node*/conf/group.1.ini
sed -i 's/db_passwd=/db_passwd=123456/g'
```

```
~/fisco/nodes/127.0.0.1/node*/conf/group.1.ini

# -----配置每个区块链节点在MySQL中创建的库名-----
# 配置node0的数据库名称为db_node0
sed -i 's/db_name=/db_name=db_node0/g'
~/fisco/nodes/127.0.0.1/node0/conf/group.1.ini

# 配置node1的数据库名称为db_node1
sed -i 's/db_name=/db_name=db_node1/g'
~/fisco/nodes/127.0.0.1/node1/conf/group.1.ini

# 配置node2的数据库名称为db_node2
sed -i 's/db_name=/db_name=db_node2/g'
~/fisco/nodes/127.0.0.1/node2/conf/group.1.ini

# 配置node3的数据库名称为db_node3
sed -i 's/db_name=/db_name=db_node3/g'
~/fisco/nodes/127.0.0.1/node3/conf/group.1.ini
```

储存加密

联盟链的数据，只对联盟内部成员可见。落盘加密，保证了运行联盟链的数据，在硬盘上的安全性。一旦硬盘被带出联盟链自己的内网环境，数据将无法被解密。落盘加密是对节点存储在硬盘上的内容进行加密，加密的内容包括：合约的数据、节点的私钥。

启动Key Manager

```
./key-manager 8150 123xyz
```

配置dataKey

配置dataKey的节点，必须是新生成，未启动过的节点。

执行脚本，定义dataKey，获取cipherDataKey

```
cd key-manager/scripts
bash gen_data_secure_key.sh 127.0.0.1 8150 123456

CiherDataKey generated: ed157f4588b86d61a2e1745efe71e6ea
Append these into config.ini to enable disk encryption:
[storage_security]
enable=true
key_manager_ip=127.0.0.1
key_manager_port=8150
cipher_data_key=ed157f4588b86d61a2e1745efe71e6ea
```

得到cipherDataKey，脚本自动打印出落盘加密需要的ini配置(如下)。此时得到节点的cipherDataKey：cipher_data_key=ed157f4588b86d61a2e1745efe71e6ea 将得到的落盘加密的ini配置，写入节点配置文件（config.ini ）中。

```
vim nodes/127.0.0.1/node0/config.ini
Copy to clipboard
```

修改[storage_security]中的字段如下。

```
[storage_security]
enable=true
key_manager_ip=127.0.0.1
key_manager_port=8150
cipher_data_key=ed157f4588b86d61a2e1745efe71e6ea
```

加密节点私钥 执行脚本，加密节点私钥

```
cd key-manager/scripts
# 参数：ip port 节点私钥文件 cipherDataKey
bash encrypt_node_key.sh 127.0.0.1 8150 ../../nodes/127.0.0.1/node0/conf/node.key
ed157f4588b86d61a2e1745efe71e6ea
```

执行后，节点私钥自动被加密，加密前的文件备份到了文件node.key.bak.xxxxxx中，请将备份私钥妥善保管，并删除节点上生成的备份私钥

```
[INFO] File backup to "nodes/127.0.0.1/node0/conf/node.key.bak.1546502474"
[INFO] "nodes/127.0.0.1/node0/conf/node.key" encrypted!
```

四、功能测试

编译

- 运行区块链节点

```
bash fisco/nodes/127.0.0.1/start_nodes.sh
```

- 运行assey app

```
cd fisco/asset-app/dist
bash asset_run.sh deploy
```

4.1 账户余额

```
1
------Check my credit------

 Your credit limit is: 1500
Wait for key...Dear wuhy68, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```

4.2 进行交易

```
2
------Trade with others------

T          wuhy68
T          unt: 500
 Add transaction success! id:1000000
Wait for key...Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```


4.3 欠条拆分

```
4
------IOU Spilt------

Beneficiary: zhaowx
Original ID: 1000001
Transfer amount: 500
 Split transaction success! old_id: 1000001 new_id: 1000002
Wait for key...Dear wuhy68, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```

4.4 银行融资

```
3
------Financing/Loan------

Financing/Loan Amount: 1000
 Add transaction success! id:1000001
Wait for key...Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```


4.5 还款功能

```
5
------Transfer/Loan Repayment------

Trade ID: 1000001
Transfer Amount: 500
 Update transaction success.
Wait for key...Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```

4.6 查询账单

```
6
------Trade Check------
Original ID: 1000001
Transaction
 ID: 1000001; Acc1: bank; Acc2: acc0; Money: 1000; Current: 1000
Wait for key...
```

五、实验流程

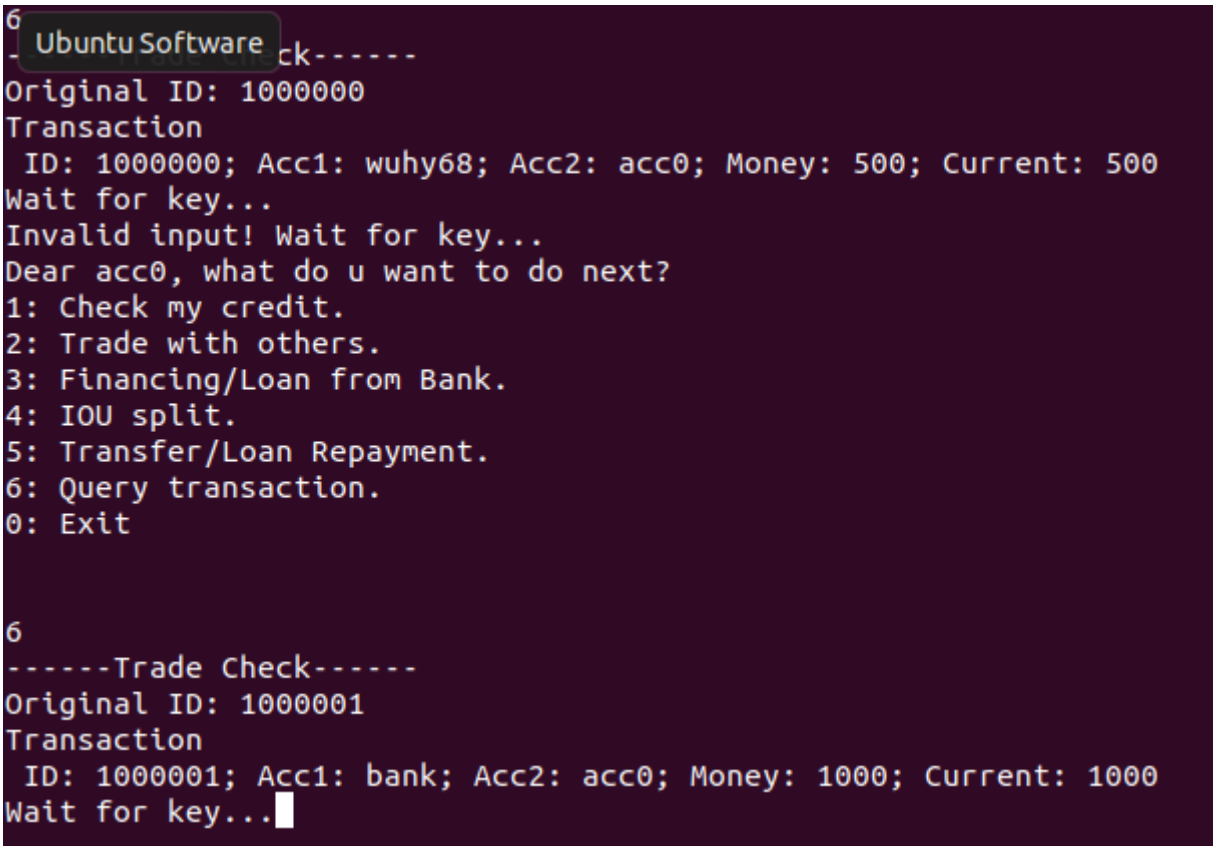
acc0: 与wuhy68进行交易

```
2
------Trade with others------

T          wuhy68
T          unt: 500
 Add transaction success! id:1000000
Wait for key...Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```

acc0: 向银行融资

```
3
------Financing/Loan------

Financing/Loan Amount: 1000
 Add transaction success! id:1000001
W          Dear acc0, what do u want to do next?
1          dit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```

acc0： 查询以上两笔交易


6
Ubuntu Software
ck------
Original ID: 1000000
Transaction
ID: 1000000; Acc1: wuhy68; Acc2: acc0; Money: 500; Current: 500
Wait for key...
Invalid input! Wait for key...
Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
6
------Trade Check------
Original ID: 1000001
Transaction
ID: 1000001; Acc1: bank; Acc2: acc0; Money: 1000; Current: 1000
Wait for key...

zhaowx：拆分欠条

```
4
------IOU Spilt------

Beneficiary: zhaowx
Original ID: 1000001
Transfer amount: 500
 Split transaction success! old_id: 1000001 new_id: 1000002
Wait for key...Dear wuhy68, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```

```
6
------Trade Check------
Original ID: 1000001
Transaction
 ID: 1000001; Acc1: bank; Acc2: acc0; Money: 1000; Current: 1000
Wait for key...
Invalid input! Wait for key...
Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit


6
------Trade Check------
Original ID: 1000001
Transaction
 ID: 1000001; Acc1: bank; Acc2: acc0; Money: 1000; Current: 500
```

zhaowx & acc0 查询差分后的账单

```
6
------Trade Check------
Original ID: 1000001
Transaction
 ID: 1000001; Acc1: bank; Acc2: acc0; Money: 1000; Current: 500
Wait for key...
Invalid input! Wait for key...
Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit


6
------Trade Check------
Original ID: 1000002
Transaction
 ID: 1000002; Acc1: zhaowx; Acc2: acc0; Money: 500; Current: 0
Wait for key...
```

zhaowx & acc0 还款

```
5
------Transfer/Loan Repayment------

Trade ID: 1000001
Transfer Amount: 500
 Update transaction success.
Wait for key...Dear acc0, what do u want to do next?
```

zhaowx & acc0 查询还款后的账单

```
6
------Trade Check------
Original ID: 1000002
Transaction
 ID: 1000002; Acc1: zhaowx; Acc2: acc0; Money: 500; Current: 0
Wait for key...
Invalid input! Wait for key...
Dear acc0, what do u want to do next?
1: Check my credit.
2: Trade with others.
3: Financing/Loan from Bank.
4: IOU split.
5: Transfer/Loan Repayment.
6: Query transaction.
0: Exit
```

Help

六、心得体会

通过完成本次区块链大作业，我们了解了 FISCO BOCOS 的区块链平台。作为一个非区块链底层开发者，我认为该区块链平台还是有较大的提升空间的。由于API接口适应性问题，本次作业对于一些没有Java基础的同学就不太友好。通过在 FISCO 的官网的学习，我们全面地了解到了其系统设计。体会到了区块链这一技术在未来的广阔前景，本次因为个人精力和水平实在有限，最终制品的完成度上还是有很多不尽如人意的地方，但是也从中收获了不少知识，锻炼了代码能力，更加深入地了解了区块链这一技术。