区块链大作业：基于区块链的供应链金融平台项目设计说明

课程名称	**区块链原理与技术**	**任课老师**	**黄华威**
年级	2018级	专业	软件工程
小组	Group 6	大作业	基于区块链的供应链金融平台
组员	黄进，胡梓渊，喻勇强	学号	18342029，18342026，18342123
开始日期	2020.11.30	完成日期	2020.12.14

一、设计要求 （以下功能具体的测试将在 功能测试文档 里给出）

- 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。
- 根据提供的供应链场景，基于FISCO-BCOS设计相关的智能合约并详细解释智能合约是如何解决提出的问题
- 即介绍如何设计智能合约，以方便实现四个功能，相当于对设计的 Asset.sol 合约文件进行源码解读

本次链端智能合约的编写主要参考了 **FISCO BCOS** 官方文档，参考的教程文档如下：

- 构建第一个区块链应用
- FISCO BCOS零基础入门，五步轻松构建应用
- 智能合约编写之Solidity的基础特性

二、智能合约存储设计

- 基于已有的开源区块链系统FISCO-BCOS（https://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。**
- **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。

- **注：** `Asset.sol` 合约的实现需要引入 FISCO BCOS 提供的一个系统合约接口文件 `Table.sol`，该系统合约文件中的接口由 FISCO BCOS 底层实现。当业务合约需要操作 CRUD 接口时，均需要引入该接口合约文件。
- 比如以下 `api`，用于创建、打开表；进行查询、插入、更新和删除

```
contract TableFactory {
    function openTable(string) public constant returns (Table); //open table
    function createTable(string,string,string) public returns(int); //create table
}
```

```
//Table main contract
contract Table {
    //select api
    function select(string, Condition) public constant returns(Entries);
    //insert api
    function insert(string, Entry) public returns(int);
    //update api
    function update(string, Entry, Condition) public returns(int);
    //remove api
    function remove(string, Condition) public returns(int);

    function newEntry() public constant returns(Entry);
    function newCondition() public constant returns(Condition);
}
```

1. 设计表

- 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。
- 针对本应用需要设计一个存储资产管理的表`t_asset`，该表字段如下：
    - account: 主键，资产账户 (string类型)
    - asset_value: 资产金额 (int256类型)

  其中account是主键，即操作`t_asset`表时需要传入的字段，区块链根据该主键字段查询表中匹配的记录。

- 除此之外还需要设计一个存储交易记录的表`t_receipt`，该表字段如下：
    - id：主键，标识交易的唯一 ID (string类型)
    - borrower：借债人 (string类型)
    - debtee：债权人 (string类型)
    - money：交易债务金额 (int256类型)
    - status：未结清金额 (int256类型)

2. 相关函数

① 构造函数

- 部署并初始化合约
- 用于建表和注册第三方可信机构账号（银行），设计让银行的资产额度非常大

```
string Bank_name = "bank";
int256 Bank_asset_value = 99999;

constructor() public {
    // 构造函数中创建t_asset表
    createTable();
    // 注册银行账号
    register(Bank_name, Bank_asset_value);
}
```

② 建表

- 根据设计的表结构建表

```
function createTable() public returns(int) {
    int count = 0;
    TableFactory tf = TableFactory(0x1001);
    // 资产管理表, key : account, field : asset_value
    // |   资产账户(主键)    |    资产额度    |
    // |--------------------|----------------|
    // |       account      |   asset_value  |
    // |--------------------|----------------|
    //
    // 创建资产管理表
    count += tf.createTable("t_asset", "account", "asset_value");

    // 交易记录表, key: id, field: borrower, debtee, money, status
    // | 交易单号(key) | 借债人 | 债权人 | 债务金额 |  状态  |
    // |-------------|------|-------|---------|---------|
    // |      id     | borrower | debtee  |  money  | status  |
    // |-------------|------|-------|---------|---------|
    // 创建交易记录表
    count += tf.createTable("t_receipt", "id","borrower, debtee, money, status");
    emit CreateTableEvent(count);
    return count;
}
```

③ 打开表

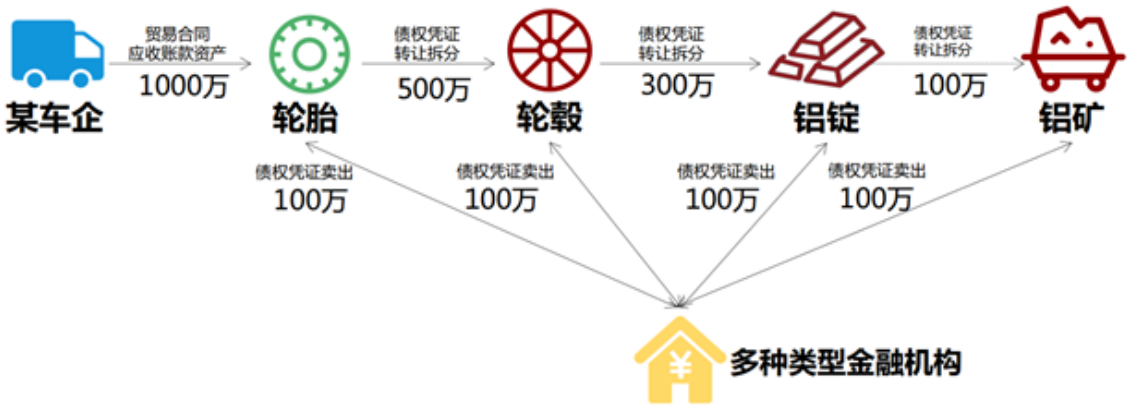
```
// 返回资产管理表
function openAssetTable() private returns(Table) {
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("t_asset");
    return table;
}

// 返回交易记录表
function openReceiptTable() private returns(Table) {
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("t_receipt");
    return table;
}
```

三、智能合约功能接口设计

1. 要实现的基本功能：

- 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。



2. 四个基础功能

① 账户注册 (register)

- 输入账户名和资产金额进行注册

```
    /*
    描述 : 资产注册
    参数 :
            account :       资产账户
            asset_value  : 资产金额
    返回值:
            0  资产注册成功
            -1 资产账户已存在
            -2 其他错误
    */
```

```
function register(string account, int256 asset_value) public returns(int256){
    int256 ret_code = 0;
    int256 ret = 0;
    int256 temp_asset_value = 0;
    // 查询账户是否存在
    (ret, temp_asset_value) = selectAccount(account);
    if(ret != 0) {
        Table table = openAssetTable();

        Entry entry_register = table.newEntry();
        entry_register.set("account", account);
        entry_register.set("asset_value", int256(asset_value));
        // 插入
        int count = table.insert(account, entry_register);
        if (count == 1) {
            // 成功
            ret_code = 0;
        } else {
            // 失败? 无权限或者其他错误
            ret_code = -2;
            table.remove(account, table.newCondition());
        }
    } else {
        // 账户已存在
        ret_code = -1;
    }

    emit RegisterEvent(ret_code, account, asset_value);

    return ret_code;
}
```

② 查询账户信息 (selectAccount)

- 可以根据账户名查询一个账户是否存在，它的资产金额是多少

```
/*
    描述 : 根据资产账户查询资产金额
    参数 :
            account : 资产账户
    返回值:
            参数一: 成功返回 0, 账户不存在返回 -1
            参数二: 第一个参数为 0 时有效，代表资产金额
*/
```

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

③ 查询交易信息 (selectReceipt)

- 可以根据交易编号 ID 查询一笔交易是否存在，它的交易金额和未结清金额是多少，债权人和借债人是谁

```
/*
    描述 : 根据交易 ID 查询交易信息
    参数 :
            id : 交易编号
    返回值:
            参数一: 若成功返回 0
            参数二: 若成功数组首个元素为初始交易金额, 第二个为未结清金额
            参数三: 若成功则第一个元素为债权人, 第二个为借债人
    */
```

```
function selectReceipt(string id) public constant returns(int256, int256[], string[]) {
    // 打开表
    Table table = openReceiptTable();
    // 查询
    Entries entries_receipt = table.select(id, table.newCondition());

    // ret_code
    int256 ret_code = 0;
    // money, status
    int256[] memory info_list = new int256[](2);
    // borrower, debtee
    string[] memory user_list = new string[](2);
    if (0 == int256(entries_receipt.size())) {
        ret_code = -1;
        return (ret_code, info_list, user_list);
    } else {
        Entry entry = entries_receipt.get(0);
        info_list[0] = entry.getInt("money");
        info_list[1] = entry.getInt("status");
        user_list[0] = entry.getString("borrower");
        user_list[1] = entry.getString("debtee");
        return (ret_code, info_list, user_list);
    }
}
```

④ 资产转移 (transfer)

- 将金额从一个账户转移到另一个

```
/*
    描述 : 资产转移
    参数 :
            from_account : 转移资产账户
            to_account :   接收资产账户
            asset_value :  转移金额
    返回值:
            0  资产转移成功
            -1 转移资产账户不存在
            -2 接收资产账户不存在
            -3 金额不足
            -4 金额溢出
            -5 其他错误
    */
```

```
function transfer(string from_account, string to_account, int256 asset_value) public returns(int256) {
    // 查询转移资产账户信息
    int ret_code = 0;
    int256 ret = 0;
    int256 from_asset_value = 0;
    int256 to_asset_value = 0;

    // 转移账户是否存在?
    (ret, from_asset_value) = selectAccount(from_account);
    if(ret != 0) {
        ret_code = -1;
        // 转移账户不存在
        emit TransferEvent(ret_code, from_account, to_account, asset_value);
        return ret_code;

    }

    // 接受账户是否存在?
    (ret, to_asset_value) = selectAccount(to_account);
    if(ret != 0) {
        ret_code = -2;
        // 接收资产的账户不存在
        emit TransferEvent(ret_code, from_account, to_account, asset_value);
        return ret_code;
    }

    if(from_asset_value < asset_value) {
        ret_code = -3;
        // 转移资产的账户金额不足
        emit TransferEvent(ret_code, from_account, to_account, asset_value);
        return ret_code;
    }
```

```
    if (to_asset_value + asset_value < to_asset_value) {
        ret_code = -4;
        // 接收账户金额溢出
        emit TransferEvent(ret_code, from_account, to_account, asset_value);
        return ret_code;
    }

    Table table = openAssetTable();

    Entry entry_from = table.newEntry();
    entry_from.set("account", from_account);
    entry_from.set("asset_value", int256(from_asset_value - asset_value));
    // 更新转账账户
    int count = table.update(from_account, entry_from, table.newCondition());
    if(count != 1) {
        ret_code = -5;
        // 失败? 无权限或者其他错误?
        emit TransferEvent(ret_code, from_account, to_account, asset_value);
        return ret_code;
    }

    Entry entry_to = table.newEntry();
    entry_to.set("account", to_account);
    entry_to.set("asset_value", int256(to_asset_value + asset_value));
    // 更新接收账户
    table.update(to_account, entry_to, table.newCondition());

    emit TransferEvent(ret_code, from_account, to_account, asset_value);

    return ret_code;
}
```

3. 四个交易功能（以下行为会记录在链上）

① 签发应收账款交易上链（添加交易 createReceipt）

- 通过调用资产转移函数 `transfer` 实现创建新交易功能
- 只有上游企业（资产额度足够大）才能发起交易，上游企业与下游企业进行交易，两者之间签订了一笔交易

```
/*
    描述 : 签发应收账款交易上链（添加交易）
    参数 :
            id :        交易编号
            borrower : 借债人
            debtee :    债权人
            transaction_money:      初始交易金额
    返回值:
            0  交易添加成功
            -1 交易已存在
            -2 资产转移错误
            -3 借债人资产额度不够
            -4 其他错误
```

```
14      */
```

```
function createReceipt(string id, string borrower, string debtee, int256 transaction_money) public returns(int256){
    int256 ret_code = 0;
    int256 ret = 0;
    int256 asset_value = 0;

    int isExist = 0;
    int256[] memory info_list = new int256[](2);
    string[] memory user_list = new string[](2);

    // 查询交易是否存在
    (isExist, info_list, user_list) = selectReceipt(id);
    if(isExist != int256(0)) {
        // 打开表
        Table table = openReceiptTable();
        // 查询
        Entry entry_receipt = table.newEntry();
        entry_receipt.set("id", id);
        entry_receipt.set("borrower", borrower);
        entry_receipt.set("debtee", debtee);
        entry_receipt.set("money", int256(transaction_money));
        entry_receipt.set("status", int256(transaction_money));

        // 插入
        int count = table.insert(id, entry_receipt);
        if (count == 1) {
            // 查询借债人的资产额度
            (ret, asset_value) = selectAccount(borrower);
            if (asset_value > 1000) {
                // 将借债人的资产额度转移一部分给债权人
                ret = transfer(borrower,debtee,transaction_money);
                // 资产额度转让失败
                if(ret != 0) {
                    ret_code = -2;
                } else {
                    ret_code = 0;
                }
            } else {
                // 借债人资产额度不够;
                ret_code = -3;
            }
        } else {
            // 失败? 无权限或者其他错误
            ret_code = -4;
        }

    } else {
        // 交易已存在
        ret_code = -1;
    }

    // 如果交易创建失败，将记录清除
    if(ret_code != 0) {
        table.remove(id, table.newCondition());
    }

    emit CreateReceiptEvent(ret_code, id, borrower, debtee, transaction_money);

    return ret_code;
}
```

② 应收账款的转让上链（拆分交易 divideRecipt）

- 假设一个用户是一笔交易的债权人，现在他想转让到另一笔向别的用户借债的新交易中，可以将不超过原交易未结清金额的额度转移给另外一个用户
- 通过转让原交易借债人的应收账款给新交易的债权人，实现交易转移，新债权人会从该用户那获得同等额度的资产
- debtee 把 borrower 借的债转让给 newDebtee，debtee 资产额度转移给 newDebtee，分为两步
    1. borrower 还债给 debtee `(repayReceipt)`
    2. borrower 向 newDebtee 借债 `(createReceipt)`

```
/*
    描述 ：  应收账款的转让上链（拆分交易）
    参数 ：
            original_id:  需要转让的交易 ID
            new_id:       新创建的交易 ID
            newDebtee:    新创建交易的债权人
            divide_money:         交易拆分的金额
    返回值 ：
             0 交易转让成功
            -1 转让的交易不存在
            -2 账户不存在
            -3 转让的金额大于原交易未结算的金额
            -4 资产返还错误
            -5 新交易创建不成功
    */
```

```
function divideReceipt(string original_id, string new_id, string newDebtee, int256 divide_money) public returns(int256) {
    int256 ret_code = 0;
    int256 ret = 0;


    int256 isExist = 0;
    int256[] memory info_list = new int256[](2);
    string[] memory user_list = new string[](2);
    // 查询该欠条是否存在
    (isExist, info_list, user_list) = selectReceipt(original_id);

    if(isExist == 0) {
        int temp = 0;
        // 查询账户是否存在
        (ret, temp) = selectAccount(newDebtee);
        if(ret != 0) {
            ret_code = -2;
            emit DivideReceiptEvent(ret_code, original_id, new_id, newDebtee, divide_money);
            return ret_code;
        }

        // 转让的金额大于原交易未结算的金额
        if(divide_money > info_list[1]){
            ret_code = -3;
            emit DivideReceiptEvent(ret_code, original_id, new_id, newDebtee, divide_money);
            return ret_code;
        }

        // debtee 把 borrower 借的债转让给 newDebtee，debtee 资产额度转移给 newDebtee，分为两步
        // 1. borrower 还债给 debtee
```

```
        ret = repayReceipt(original_id, divide_money);
        if (ret != 0) {
            ret_code = -4;
            emit DivideReceiptEvent(ret_code, original_id, new_id, newDebtee, divide_money);
            return ret_code;
        }

        // 2. borrower 向 newDebtee 借债
        ret = createReceipt(new_id, user_list[0], newDebtee, divide_money);
        if (ret != 0) {
            // 新交易创建不成功
            ret_code = -5;
            emit DivideReceiptEvent(ret_code, original_id, new_id, newDebtee, divide_money);
            return ret_code;
        }

    } else {    // 转让的交易不存在
        ret_code = -1;
    }

    emit DivideReceiptEvent(ret_code, original_id, new_id, newDebtee, divide_money);
    return ret_code;
}
```

③ 利用应收账款向银行融资上链（融资 financeReceipt）

- 把应收账款转让给银行，从而从银行获得融资
- 所有企业都可以使用自身的资产额度向银行请求融资，获得一笔不多于自身资产额度的金额。
- 在我们设计的智能合约中将这种行为视为该企业与银行之间创建了一笔交易 `(createReceipt)`，将企业的资产额度转移给银行

```
/*
    描述 ： 利用应收账款向银行融资上链（融资）
    参数 ：
            id ：           融资编号
            financer ：     融资人
            finance_money:  金额
    返回值：
             0 融资成功
            -1 融资人不存在
            -2 融资额大于融资人目前的资产额度
            -3 其他错误
    */
```

```
function financeReceipt(string id, string financer , int256 finance_money) public returns(int256) {
    int256 ret_code = 0;
    int256 ret = 0;

    int256 isExist = 0;
    int256 asset_value = 0;

    // 查询融资人是否存在
    (isExist, asset_value) = selectAccount(financer);

    if(isExist == 0) { // 融资人存在
        // 融资人目前的资产额度大于等于融资额
        if (asset_value >= finance_money) {
            // 向银行申请融资
            ret = createReceipt(id, financer, "bank", finance_money);
            if (ret != 0) {
                ret_code = -3;
                emit FinanceReceiptEvent(ret_code, id, financer, finance_money);
                return ret_code;
            }
        } else {
            ret_code = -2;
            emit FinanceReceiptEvent(ret_code, id, financer, finance_money);
            return ret_code;
        }

    } else { // 融资人不存在
        ret_code = -1;
    }
    emit FinanceReceiptEvent(ret_code, id, financer, finance_money);
    return ret_code;
}
```

④ 应收账款支付结算上链（结算交易 repayReceipt）

- 通过资产转移来移除借款人手中的应收账款，借债人和债权人资产额度变化结算的金额
- 在一笔交易中，作为借债人的企业可以通过还钱给债主来让交易中的未结清金额减少，并从债权人那返还等额的资产额度。当未结清金额为 0 时，该交易支付结算完毕

```
/*
    描述 : 应收账款支付结算上链（结算交易）
    参数 :
            id : 交易编号
            repay_money: 金额
    返回值:
             0 交易结算成功
            -1 交易不存在
            -2 还债金额大于借款
            -3 其他错误
            -4 资产返还错误
    */
```

```
function repayReceipt(string id, int256 repay_money) public returns(int256){
    int256 ret_code = 0;

    int256 isExist = 0;
    int256[] memory info_list = new int256[](2);
    string[] memory user_list = new string[](2);
    // 查询该交易是否存在
    (isExist, info_list, user_list) = selectReceipt(id);

    if(isExist == 0) { // 交易存在

        // 还款金额大于欠款
        if(repay_money > info_list[1]){
            ret_code = -2;
            emit RepayReceiptEvent(ret_code, id, repay_money);
            return (ret_code);
        }

        // 更新交易状态
        Table table = openReceiptTable();
        Entry entry0 = table.newEntry();
        entry0.set("id", id);
        entry0.set("borrower", user_list[0]);
        entry0.set("debtee", user_list[1]);
        entry0.set("money", info_list[0]);
        // 未结算金额减少
        entry0.set("status", (info_list[1] - repay_money));
```

```
        // 更新交易信息
        int count = table.update(id, entry0, table.newCondition());
        if(count != 1) {
            ret_code = -3;
            // 失败? 无权限或者其他错误?
            emit RepayReceiptEvent(ret_code, id, repay_money);
            return (ret_code);
        }

        // 资产返还
        int256 temp = transfer(user_list[1], user_list[0], repay_money);
        if(temp != 0){
            ret_code = -4;
            emit RepayReceiptEvent(ret_code, id, repay_money);
            return (ret_code);
        }

        ret_code = 0;

    } else { // 交易不存在
        ret_code = -1;
    }
    emit RepayReceiptEvent(ret_code, id, repay_money);

    return ret_code;
}
```

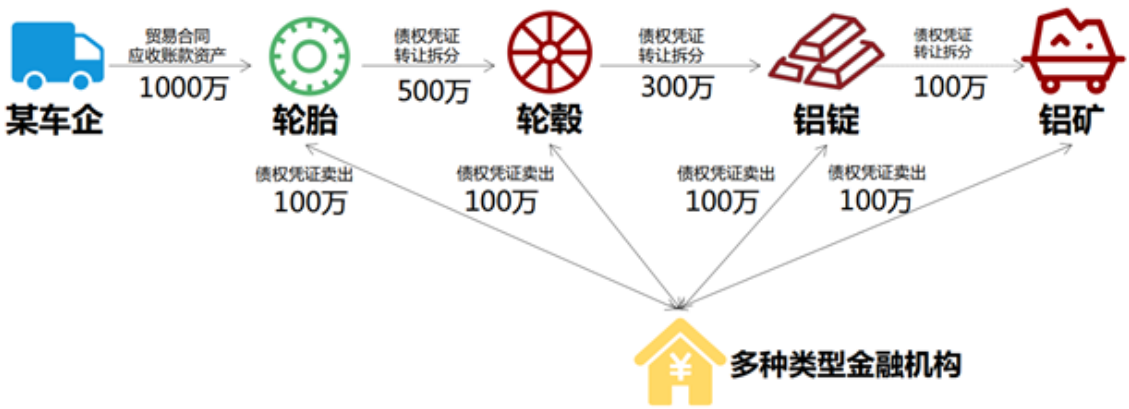
4. 事件

- **事件** 功能类似日志，记录了一个事件的发生，会被记录到区块链中，客户端可以通过web3订阅这些事件。
- 设计的 Asset.sol 智能合约包括以下事件：

```
// event
// 建表
event CreateTableEvent(int count);
// 资产注册
event RegisterEvent(int256 ret, string account, int256 asset_value);
// 资产额度转移
event TransferEvent(int256 ret, string from_account, string to_account, int256 asset_value);
// 签发应收账款交易上链（添加交易）
event CreateReceiptEvent(int256 ret, string id, string borrower, string debtee, int256 money);
// 应收账款的转让上链（拆分交易）
event DivideReceiptEvent(int256 ret, string original_id, string new_id, string newDebtee, int256 money);
// 利用应收账款向银行融资上链（融资）
event FinanceReceiptEvent(int256 ret, string id, string financer, int256 money);
// 应收账款支付结算上链（返还交易）
event RepayReceiptEvent(int256 ret, string id, int256 money);
```

四、方案设计事例（以下设计方案事例的具体实现将在 功能测试文档 里给出）

- 在这次的项目中，需要用到的在链上的对象一共有四个， 分别是上游企业车企、中游轮胎公司、下游轮毂公司和银行。
- 合约实现以下了四个交易功能， 即添加交易上链、 交易转让上链、 融资上链和支付结算上链
- 以车企，轮胎厂，轮毂厂，和银行间的交易作为例子：它们的资产额度假设分别为 A0, B0, C0, D0，注意银行的资产额度非常大



1. 某上游企业车企 与 中游企业轮胎厂进行一笔交易，交易金额为 1000 万，此时通过 `createReceipt` 函数创建一笔交易，交易过后，车企的资产额度变为 A1 = A0 - 1000，轮胎厂的变为 B1 = B0 + 1000
2. 轮胎厂通过拆分转让和车企的交易， 与 轮毂厂产生一笔新的交易，交易金额为 500 万，通过 `divideReceipt` 来实现。这时轮胎厂的资产额度变为 B2 = B1 - 500，轮毂厂变为 C1 = C0 + 500
3. 轮胎厂再向银行融资 500 万 `financeReceipt`，此时轮胎厂资产额度变为 B3 = B2 - 500，银行变为 D1 = D0 + 500
4. 车企向轮胎厂还款 `repayReceipt`，还款金额为 500 万，车企的资产额度变为 A2 = A1 + 500，轮胎厂 B4 = B3 - 500