共享单车系统智能合约介绍

共享单车系统主要包括系统合约和用户合约，下面将详细的分别介绍这两个合约：

- 用户合约
- 系统合约

用户合约

用户合约是区块链上单车的一种存在形态，赋予及体现了单车所拥有的一些属性，如单车的所有者，租赁情况，买卖情况，地处坐标以及相对于相对于初始距离的一些状态指示位，下面就是本次demo演示的用户合约代码，后续在此基础上用户可以添加其他属性来对单车增添更多的状态以及更为丰富的承载体，做到了灵活多变。

1. 用户合约代码

```
contract user_bicycle_info {

    #区块链上单车的属性，用户可以根据需求灵活添加更多属性
    address owner_addr;      #拥有者
    uint public  rent_flag;  #租赁属性
    uint public  rent_value;
    uint public  sell_flag;  #买卖属性
    uint public  sell_value;
    uint public origin_distance;  #位置属性
    uint public update_distance;
    uint public out_of_bound;
    uint public reward_value;  #奖励属性

   function user_bicycle_info () {
   owner_addr = msg.sender;
   rent_value = 2;
   rent_flag = 0;
   sell_flag = 0;
   sell_value = 300;
   origin_distance = 0;
   update_distance = 0;
   out_of_bound = 0;
   reward_value = 1;

  }
}
```

系统管理合约

系统合约是对区块链该系统上所有单车的一个智能管理，包括用户注册单车，租赁单车、还车，两方交易单车以及奖励系统。用户需要将自己已发布的用户合约地址注册到该系统上并同时缴纳一定的押金，系统也会将该地址以列表的形式展示给其他用户供其租赁或者直接买卖交易，采用提前约定规则的方式来对用户以及用户合约进行系统管理。下面是系统合约的代码及部分注解。

1. 系统合约代码

```
pragma solidity ^0.4.4;

contract system_manage {
   address public organizer;  #系统管理者（即系统合约发布者）
   uint public user_amount;   #系统用户数
```

```
    uint public deposit_quota; #押金
    uint public balance;
    address[] public user_list;    #用户列表
    address[] public user_contract_list; #用户合约列表
    address[] public used_contract_list;
    address[] public far_contract_list;
    mapping (address => uint) public balances;

    struct Bicycle_info{
         address owner_addr;
         uint rent;
         uint rent_value;
         uint sell;
         uint sell_value;
         uint origin_distance;
         uint update_distance;
         uint out_of_bound;
         uint reward_value;
     }

    mapping (address => uint) public registrantsPaid;
    mapping (address => Bicycle_info) public depository;
    mapping (address => address) public rent_info_map;
    mapping (address => address) public sell_info_map;

    event Deposit(address _from, uint _user_amount);  #用于日志记录
    event Refund(address _to, uint _user_amount);

    function system_manage() {
    organizer = msg.sender;
    user_amount = 0;
    deposit_quota = 500;
   }

   #改变系统押金数
   function changeQuota(uint newquota) public {
     if (msg.sender != organizer) {
         return;
     }
     deposit_quota = newquota;
   }

    #用户充值函数
   function charge() public returns (bool success){
      balances[msg.sender] = 100000;
      return true;
 }

    #注册函数
   function register(address contract_address,uint user_deposit_amount) public returns (bool success)  {
     if (user_deposit_amount != deposit_quota) {
         return false;
     }
     registrantsPaid[msg.sender] = user_deposit_amount;
     depository[contract_address].owner_addr = msg.sender;
     depository[contract_address].rent = 0;
     depository[contract_address].rent_value = 100;
     depository[contract_address].origin_distance = 0;
     depository[contract_address].out_of_bound = 0;
     depository[contract_address].update_distance = 0;
     depository[contract_address].reward_value = 50;
     user_amount++;
     balances[msg.sender]-=deposit_quota;
     own_list.push(contract_address);
     user_list.push(msg.sender);
     user_contract_list.push(contract_address);
     Deposit(msg.sender, user_deposit_amount);
     return true;
   }

  #购买函数
   function buy(address contract_addr) public returns (bool success) {
     if(depository[contract_addr].rent != 0) {
          return false;
```

```
     }
    if(!depository[contract_addr].owner_addr.send(depository[contract_addr].sell_value)){
        throw;
    }
    depository[contract_addr].sell=1;
    depository[contract_addr].owner_addr=msg.sender;
    sell_info_map[depository[contract_addr].owner_addr]=msg.sender;
    balances[msg.sender]-=300;
    return true;
  }

  #还车函数
  function refund(address contract_addr) public returns (bool success) {
    if(rent_info_map[msg.sender] != depository[contract_addr].owner_addr) {
        return false;
    }
    if(depository[contract_addr].rent !=1 ){
        return false;
    }
    depository[contract_addr].rent=0;
    depository[contract_addr].update_distance = 1000;
    if(depository[contract_addr].update_distance-depository[contract_addr].origin_distance>800){
         far_contract_list.push(contract_addr);
         depository[contract_addr].out_of_bound = 1;
     }
    if(!msg.sender.send(deposit_quota)){
        throw;
    }
    registrantsPaid[msg.sender] = 0;
    balances[msg.sender]-=100;
    user_amount--;
    Refund(msg.sender, user_amount);
    return true;
  }

  #租赁函数
  function rent(address select_contract_address,uint rent_value ) public returns (bool success) {
     if(depository[select_contract_address].rent != 0) {
         return false;
     }
     if(rent_value != depository[select_contract_address].rent_value){
         return false;
     }
     registrantsPaid[msg.sender] = deposit_quota;
     if(!depository[select_contract_address].owner_addr.send(depository[select_contract_address].rent_value)){
         throw;
     }
     rent_info_map[msg.sender]=depository[select_contract_address].owner_addr;
     user_list.push(msg.sender);
     used_contract_list.push(select_contract_address);
     user_amount++;
     depository[select_contract_address].rent=1;
     balances[msg.sender]-=deposit_quota;
     return true;
  }

  #退押金函数
  function withdraw(address contract_addr) public returns (bool success) {
    if(depository[contract_addr].rent ==1 ){
        return false;
    }
    if(msg.sender != depository[contract_addr].owner_addr){
        return false;
    }
    if(!msg.sender.send(deposit_quota)){
        throw;
    }
    balances[msg.sender] +=deposit_quota;
    registrantsPaid[msg.sender] = 0;
    user_amount--;
    return true;
  }


```

```
154.   #奖励函数
155.   function reward(address contract_addr) public returns (bool success) {
156.     if(depository[contract_addr].out_of_bound !=1 ){
157.         return false;
158.     }
159.     if(!msg.sender.send(depository[contract_addr].reward_value)){
160.         throw;
161.     }
162.     depository[contract_addr].update_distance = 0;
163.     depository[contract_addr].out_of_bound = 0;
164.
165.     return true;
166.   }
167.
168.   #系统合约销毁功能
169.   function destroy() {
170.     if (msg.sender == organizer) {
171.       suicide(organizer);
172.     }
173.   }
174. }
```