- 合约代码编写

```
contract Digitalpoint {
    enum Actor{ Aviation, Bank, Market, Petroleum, Client }

    struct Aviation{
        bytes32 ID;
        bytes32 Name;
        uint pointbalance;
    }

    struct Bank{
        bytes32 ID;
        bytes32 Name;
        uint pointbalance;
    }

    struct Market{
        bytes32 ID;
        bytes32 Name;
        uint pointbalance;
    }
```

```
    struct Petroleum{
        bytes32 ID;
        bytes32 Name;
        uint pointbalance;
    }

    struct Client{
        bytes32 ID;
        bytes32 Name;
        uint[] pointbalances;
        uint unionpaybalance;
    }

    struct Commodity{
        bytes32 ID;
        bytes32 Name;
        uint value;
    }

    mapping(bytes32 => Aviation) aviationMap;
    mapping(bytes32 => Bank) bankMap;
    mapping(bytes32 => Market) marketMap;
    mapping(bytes32 => Petroleum) petroleumMap;
    mapping(bytes32 => Client) clientMap;
    mapping(bytes32 => Commodity) commodityMap;

function newAviation(bytes32 ID, bytes32 Name, uint pointbalance) returns (bool, bytes32){
    Aviation aviation = aviationMap[ID];
    if(aviation.ID != 0x0){
        return (false,"this ID has been occupied!");
    }
    aviation.ID = ID;
    aviation.Name = Name;
    aviation.pointbalance = pointbalance;
    return (true,"success");
}

function newBank(bytes32 ID, bytes32 Name, uint pointbalance) returns (bool, bytes32){
    Bank bank = bankMap[ID];
    if(bank.ID != 0x0){
        return (false,"this ID has been occupied!");
    }
    bank.ID = ID;
    bank.Name = Name;
    bank.pointbalance = pointbalance;
    return (true,"success");
}
```

```
function newMarket(bytes32 ID, bytes32 Name, uint pointbalance) returns (bool, bytes32){
    Market market = marketMap[ID];
    if(market.ID != 0x0){
        return (false,"this ID has been occupied!");
    }
    market.ID = ID;
    market.Name = Name;
    market.pointbalance = pointbalance;
    return (true,"success");
}

function newPetroleum(bytes32 ID, bytes32 Name, uint pointbalance) returns (bool, bytes32){
    Petroleum petroleum = petroleumMap[ID];
    if(petroleum.ID != 0x0){
        return (false,"this ID has been occupied!");
    }
    petroleum.ID = ID;
    petroleum.Name = Name;
    petroleum.pointbalance = pointbalance;
    return (true,"success");
}

 function newCommodity(bytes32 ID, bytes32 Name, uint value) returns (bool, bytes32){
     Commodity commodity = commodityMap[ID];
     if(commodity.ID != 0x0){
         return (false,"this ID has been occupied!");
     }
     commodity.ID = ID;
     commodity.Name = Name;
     commodity.value = value;
     return (true,"success");
 }

 function newClient(bytes32 ID, bytes32 Name, uint[] pointbalances, uint unionpaybalance) returns (bool, bytes32){
     Client client = clientMap[ID];
     if(client.ID != 0x0){
         return (false,"this ID has been occupied!");
     }
     client.ID = ID;
     client.Name = Name;
     client.pointbalances = pointbalances;
     client.unionpaybalance = unionpaybalance;
     return (true,"success");
 }

 function Queryclientbalance(bytes32 ID) returns(bool,bytes32,bytes32,uint[],uint){
     Client client = clientMap[ID];
     return (true,"Success",client.Name,client.pointbalances,client.unionpaybalance);
 }
```

```
function exchangeMoneyToPoints(bytes32 ID,uint amount,uint n) returns(bool,bytes32){
    Client client = clientMap[ID];
    client.unionpaybalance -= amount;
    client.pointbalances[n-1] += amount;
    return (true,"success");
}

function exchangepoints(bytes32 ID1, uint amount1, uint n, bytes32 ID2, uint amount2, uint m) returns(bool, bytes32){
    Client client1 = clientMap[ID1];
    Client client2 = clientMap[ID2];
    client1.pointbalances[n-1] -= amount1;
    client2.pointbalances[n-1] += amount1;
    client1.pointbalances[m-1] += amount2;
    client2.pointbalances[m-1] -= amount2;
    return (true,"success");
}

function pointstransaction(bytes32 ID1, uint n, bytes32 ID2) returns(bool, bytes32){
    Client client1 = clientMap[ID1];
    Commodity commodity = commodityMap[ID2];
    client1.pointbalances[n-1] -= commodity.value;
    return (true,"Purchase succeeded");
}
```

• 接口解释

五种角色：4 类企业（航空公司，银行，购物超市，石油公司）；1 种消费者；

一种商品结构，用于积分兑换；

各种初始化函数；

Queryclientbalance：返回消费者虚拟币

exchangeMoneyToPoints：消费以获取虚拟币

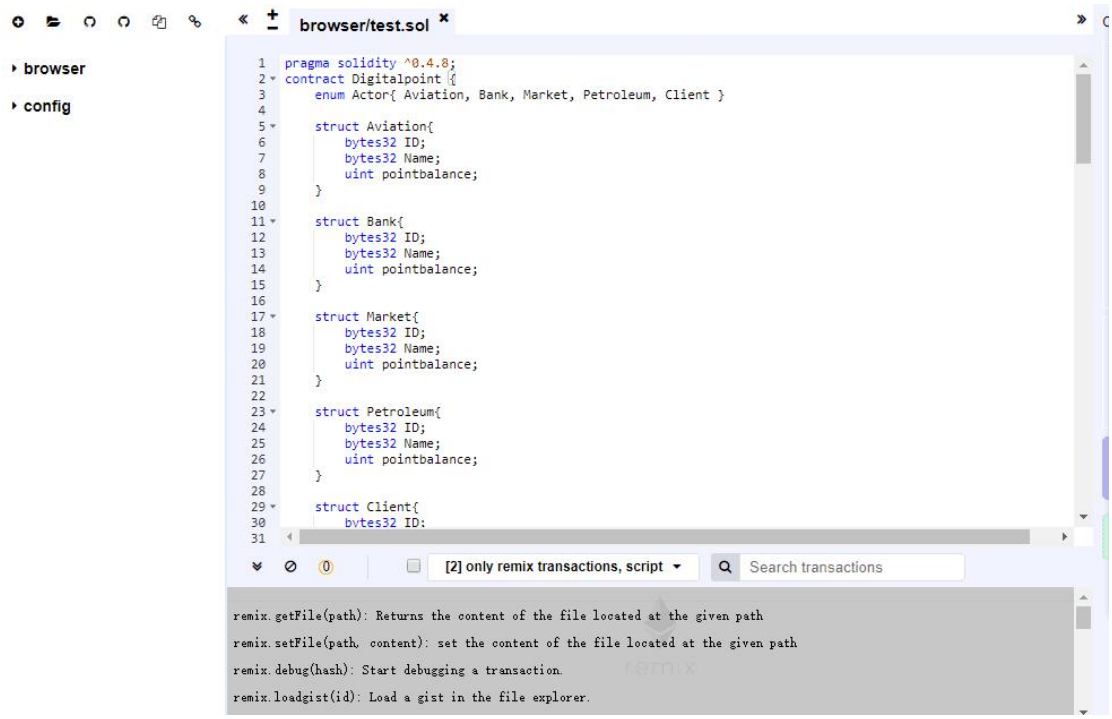
Exchangepoints：消费者之间交换虚拟币

Pointstransaction：虚拟币消费以兑换奖品

• 合约代码编译

使用 Remix 在线工具编译

1. 在浏览器中打开 Remix 工具
2. 点击左上角+，新生成一个文件.sol；
3. 将之前在本地写的.sol 中的代码拷贝到此文件中



点击 compile -> start to compile，然后点击 Details 按钮，拷贝 json 数据

• 合约代码部署

1. 部署合约前解锁账户

```
> user1 = eth.accounts[0]
"0xda037c3458e875bc7a03140b1b7bcb6a7ec8f280"
> personal.unlockAccount(user1)
Unlock account 0xda037c3458e875bc7a03140b1b7bcb6a7ec8f280
Passphrase:
true
```

2.复制 Remix 编译生成的 web3deploy 代码粘贴到 geth 控制台

```
> var digitalpoint = digitalpointContract.new(
   {
     from: web3.eth.accounts[0],
     data: '0x6060604052346100005 75b610d178061001960003960 00f300606060405236156100a2576000357c0100000000
0000000000000000000000000000000000000000000000000900463ffffffff16806309de1912146100a75780634fb3437714610105
80635d09b6f6146101635780637 35c452f146101c157806378a278501461023a57806380d66932146102d857806390d48a9d1461
365780639a6676c6146103ea578063ca1370f514610443578063e58dbfe7146104a2575b610000565b34610000576100dc600480
3560001916906020019091908035600019169060200190919080359060200190919050506105 65b6040518083151515158152
2001826000191660001916815260200192505050604051809103 90f35b346100005761013a6004808035600019169060200190 91
8035906020019091908035600019169060200190919050506105 b8565b604051808315151515815260200182600019166000191 6
526020019250505060405180910390f35b3461000057610198600480803560001916906020019091908035600019169060200190
```

```
018190555060017f737563636573730000000000000000000000000000000000000000000000000000000925092505b5093509391505056
00a165627a7a7230582053c89848360cf0f5dcba89d9d87c577af252fe34b7a587653f2e4d3db411d2a10029',
    gas: '4700000'
  }, function (e, contract){
   console.log(e, contract);
   if (typeof contract.address !== 'undefined') {
        console.log('Contract mined! address: ' + contract.address + ' transactionHash: ' + contract.tran
sactionHash);
    }
})
INFO [12-16|03:22:23.396] Setting new local account                address=0x8B3963dd2c95d9dc8aE938AA3343C
14d094D7ef8
INFO [12-16|03:22:24.631] Submitted contract creation              fullhash=0x0a47dfc0322ae0d9b333c2ca653f
a83fe4f6a27b7169bcebedb84484c395a46e contract=0xDEd9af26148550391f1B69e9EAD219956d3f9C07
```

返回 address，即合约的地址，部署成功。

遇到的问题：

1.部署的时候控制台返回 authentication needed: password or unlock undefined

解决办法：unlock 账户

2.部署的时候控制台返回 exceeds block gas limit undefined

解决办法：修改创世区块内的 gasLimited 为 0xffffffff