

## Blockchain HW2

- 项目介绍

- 项目名称：汽车租赁Dapp

- 项目内容：

该项目创建了一个智能合约，在该合约中发行NFT集合，每个NFT代表一辆车。

该项目构建了一个汽车租赁系统，调用部署在本地链上的智能合约的方法来处理相关的逻辑。实现包括获取一辆汽车（NFT），借用一辆汽车，获取当前可租用汽车列表，查询汽车主人以及借用人等功能。

为了测试方便项目有一个汽车的airdrop接口，用户调用这个接口就可以直接获得一辆车。

- 项目功能：

该项目能够连接metamask中的账户，连接部署在本地链上的合约，调用相关逻辑完成以下功能：

- 查看自己拥有的汽车列表。查看当前还没有被借用的汽车列表
    - 查询一辆汽车的主人，以及该汽车当前的借用者
    - 选择并借用某辆未被租用的汽车一段时间

- 实现分析：

NFT的实现

- 要发行一个NFT，需要引用ERC721合约，由ERC721合约派生出MyERC4907进行发行

- 合约变量介绍

```
struct UserInfo
{
    address user;    // address of user role
    uint64 expires; // unix timestamp, user expires
}
mapping (uint256 => UserInfo) internal _users;
uint256 amount;
```

UserInfo结构存储用户信息；\_users存储每个token和用户的对应关系；amount存储当前发行的NFT总量

- 合约关键方法介绍

```
function airdrop(address User) public{
    _mint(User,amount);
    amount++;
    UserInfo memory user;
    user.user=User;
    user.expires=1697778000;
    _users[amount-1]=user;
}
function getAmount() public view returns(uint256){
    return amount;
}
```

airdrop方法：将该用户地址与token进行绑定，增加amount，设置user信息并将user与token绑定；

getAmount方法：获取当前发行的NFT总量

- 接下来就是管理汽车借用系统合约（BorrowYouCar）的设计
  - 合约变量介绍

```
struct Car {
    address owner;
    address borrower;
    uint256 borrowUntil;
}
mapping(uint256 => Car) public cars;
MyERC4907 public my=new
MyERC4907("CarTokens","CarTokensSymbol");
```

Car结构存储用户信息，内容包括该车辆的主人，借用人和借用期限；cars存储每个token和用户的对应关系；my用来引用MyERC4907中的方法

- 合约关键方法介绍

```
function GetCarList() view external returns(uint256[] memory){
    uint16 count=0;
    for(uint16 i=0;i<my.getAmount();i++)
    {
        if(cars[i].owner==address(msg.sender))
            count++;
    }
    uint256[] memory cars=new uint256[](count);
    count=0;
    for(uint16 i=0;i<my.getAmount();i++)
    {
        if(cars[i].owner==address(msg.sender)){
            cars[count]=i;
            count++;
        }
    }
    return cars;
}
```

GetCarList方法，获取当前用户的拥有的汽车，先遍历一遍cars获取该用户拥有的汽车数量，再设置一个动态数组，将汽车对应的token存在该数组中返回

```
function GetALLCarList() view external returns(uint256[]
memory){
    uint256[] memory cars=new uint256[](my.getAmount());
    for(uint16 i=0;i<my.getAmount();i++)
    {
        cars[i]=i;
    }
    return cars;
}
```

GetALLCarList方法：获取当前发行的所有汽车，直接将cars中存储的token全部返回

```
function GetALLCarListNB(uint64 now) view external
returns(uint256[] memory){
    uint16 count=0;
    for(uint16 i=0;i<my.getAmount();i++)
    {
        if(cars[i].borrowUntil<now)
            count++;
    }
    uint256[] memory carsS=new uint256[](count);
    count=0;
    for(uint16 i=0;i<my.getAmount();i++)
    {
        if(cars[i].borrowUntil<now){
            carsS[count]=i;
            count++;
        }
    }
    return carsS;
}
```

GetALLCarListNB方法：获取当前发行的还未被借出的汽车，先遍历一遍cars获取数量，再存储值并返回，与GetCarList逻辑相似

```
function GetCarListNB(uint64 now) view external
returns(uint256[] memory){
    uint16 count=0;
    for(uint16 i=0;i<my.getAmount();i++)
    {

        if(cars[i].owner==address(msg.sender)&&cars[i].borrowUntil<now)
            count++;
    }
    uint256[] memory carsS=new uint256[](count);
    count=0;
    for(uint16 i=0;i<my.getAmount();i++)
    {

        if(cars[i].owner==address(msg.sender)&&cars[i].borrowUntil<now){
            carsS[count]=i;
            count++;
        }
    }
    return carsS;
}
```

GetCarListNB方法：获取当前用户拥有的还未被借出的汽车，先遍历一遍cars获取数量，再存储值并返回，与GetCarList逻辑相似

```
function GetRentCarList(uint64 now) view external
returns(uint256[] memory){
    uint16 count=0;
```

```

        for(uint16 i=0;i<my.getAmount();i++)
        {

            if(cars[i].borrower==address(msg.sender)&&cars[i].borrowUntil>now)

                count++;
        }
        uint256[] memory carS=new uint256[](count);
        count=0;
        for(uint16 i=0;i<my.getAmount();i++)
        {

            if(cars[i].borrower==address(msg.sender)&&cars[i].borrowUntil>now){

                cars[count]=i;
                count++;
            }
        }
        return carS;
    }

```

GetRentCarList方法：获取当前用户借用的汽车，先遍历一遍cars获取数量，再存储值并返回，与GetCarList逻辑相似

```

function GetOwner(uint256 CAR) external returns(address){
    if (CAR>=my.getAmount()){
        return address(0);
    }
    return cars[CAR].owner;
}

```

GetOwner方法：获取某辆汽车的主人的私钥，若该车不存在，则返回0.

```

function GetBorrower(uint256 CAR,uint64 now) external
returns(address){
    if (CAR>=my.getAmount()){
        return address(0);
    }

    if(cars[CAR].borrower==address(0)||cars[CAR].borrowUntil<now){
        return address(0);
    }
    return cars[CAR].borrower;
}

```

GetBorrower方法：获取某辆汽车的借用者公钥

```
function BorrowCar(uint256 car, uint64 now,uint64 duration)
external returns(bool,string memory){
    if(my.getAmount()<=car)
        return (false,"there is no such car");
    else if(cars[car].borrowUntil>=now)
        return (false,"this car has been borrowed");
    else{
        cars[car].borrowUntil=now+duration;
        cars[car].borrower=msg.sender;
        return (true,"borrow successfully");
    }
}
```

BorrowCar方法：借用一辆车至某期限

```
function GetaCar() public {
    my.airdrop(msg.sender);
    Car memory car;
    car.owner=msg.sender;
    car.borrower=address(0);
    car.borrowUntil=0;
    cars[my.getAmount()-1]=car;
}
```

GetaCar方法调用MyERC4907中的airdrop方法为当前用户发行一辆车

- 项目运行

- 环境配置

- 在电脑中下载ganache，配置本地链
- 创建一个metamask账户，连接本地链，导入账户
- 在contracts文件夹下用npm安装hardhat依赖项

```
npm install --save-dev hardhat
```

- 前端配置

控制台进入frontend安装依赖项

```
npm install
```

- 运行项目

- 编译合约

进入contracts文件夹编译合约

```
npx hardhat compile
```

- 将合约部署和前端配置

```
npx hardhat run --network ganache scripts/deploy.ts
```

控制台会输出相关合约的地址：

将该地址输入frontend/contract-addresses.json文件中

并将contracts/artifacts/contracts/BorrowYourCar.sol/BorrowYourCar.json和  
contracts/artifacts/contracts/MyERC4907/BorrowYourCar.json中的内容

分别粘贴在frontend/BorrowYourCar.sol/BorrowYourCar.json和  
frontend/MyERC4907.sol/MyERC4907.json中

- 运行前端

```
npm run dev
```

- 运行过程

初始界面

## 汽车租用系统

租车功能

所租车辆:  租车时间(单位为秒):

查询功能

查询车辆:

点击连接一个账户




http://localhost:5173

## 与 MetaMask 连接

选择要在此网站上使用的账户



全部选择 

新账户



Account 1 (0xac1...6597)  
0 ETH



Account 2 (0x4d7...da9a)  
0 ETH



Account 3 (0xada...373b)  
99.91716824 ETH

只连接您信任的网站。 [了解更多](#)

取消

下一步

选择提前导入metamask的本地链上的用户

连接成功后便可以显示相关的车辆信息（当前用户还未拥有任何汽车）：

# 汽车租赁系统

租车功能

所租车辆:

租车时间(单位为秒):

借用一辆车

查询功能

查询车辆:




查询一辆车

刷新当前汽车列表

用户: [ "0xada44d30c77e8e1a5a4e7f3eff447209d7e1373b" ] 你好!

获取一辆车

切换当前账户

当前可租用车辆	当前用户拥有的车辆	当前用户借用的车辆	当前用户拥有且未出借的车辆
0 			
1 			
2 			

点击获取一辆车，即可连接metamask发送交易



? tes



Account 3



0xFCb...87A3

http://localhost:5173

0xFCb...87A3 : 合约交互 ⓘ



详细信息

数据

十六进制



建议的网站 > ⓘ

燃料 (估算) ⓘ

0.00077583

**0.00077583 ETH**

很可能在 15 秒 以内

最大费用: 0.00077583 ETH

共计

0.00077583

**0.00077583 ETH**

金额 + 燃料费

最大金额: 0.00077583 ETH

拒绝

确认

点击确认，发现所有车辆以及当前用户拥有的车辆有增加

# 汽车租赁系统

租车功能

所租车辆:

租车时间(单位为秒):

借用一辆车

查询功能

查询车辆:

查询一辆车

刷新当前汽车列表

用户: [ "0xada44d30c77e8e1a5a4e7f3eff447209d7e1373b" ] 你好!

获取一辆车

切换当前账户

当前可租用车辆	当前用户拥有的车辆	当前用户借用的车辆	当前用户拥有且未出借的车辆
<div>0</div>	<div>3</div>		<div>3</div>
<div>1</div>			
<div>2</div>			
<div>3</div>			

连接另一个用户（删除前一个用户，再点击切换当前账户）发现可租用车辆不变，但是当前用户的车辆状态发生变化

# 汽车租赁系统

租车功能

所租车辆:

租车时间(单位为秒):

借用一辆车

查询功能

查询车辆:

查询一辆车

刷新当前汽车列表

用户: [ "0x3beab58e1145d7c722ed9de3ca84492794d3f72b" ] 你好!

获取一辆车 切换当前账户

当前可租用车辆	当前用户拥有的车辆	当前用户借用的车辆	当前用户拥有且未出借的车辆
0			
1			
2			
3			
4			

在租车功能下租用3车辆，60秒钟，点击借用一辆车；交易完成后，可以看到：

# 汽车租赁系统

## 租车功能

所租车辆:

租车时间(单位为秒):

借用一辆车

## 查询功能

查询车辆:

查询一辆车

刷新当前汽车列表

用户: [ "0x3beab58e1145d7c722ed9de3ca84492794d3f72b" ] 你好!

获取一辆车

切换当前账户

当前可租车辆

0



1



2



4



当前用户拥有的车辆

当前用户借用的车辆

3



当前用户拥有且未出借的车辆

查询3车的主人和借用人:

## 查询功能

查询车辆:

查询一辆车

车辆主人: 0xADa44D30C77E8e1a5A4e7f3Eff447209D7e1373b

借车人: 0x3BeAb58e1145D7c722ED9DE3Ca84492794d3F72b

60秒后, 发现借用失效

# 汽车租赁系统

## 租车功能

所租车辆:

租车时间(单位为秒):

借用一辆车

## 查询功能

查询车辆:

车辆主人: 0xADa44D30C77E8e1a5A4e7f3Eff447209D7e1373b

借车人: 0x00

刷新当前汽车列表

用户: [ "0x3beab58e1145d7c722ed9de3ca84492794d3f72b" ] 你好!

获取一辆车

切换当前账户

当前可租车辆	当前用户拥有的车辆	当前用户借用的车辆	当前用户拥有且未出借的车辆
0 			
1 			
2 			
3 			
4 			

再次借用3车，并回到主人账户，可以看到车辆3已经借用出去了

localhost:5173 显示

该车不可借用

确定

# 汽车租赁系统

当我想再次借用车辆3时，显示该车不可借用