# 说明文档

191250080 李子怡

## 代码结构

### 创建项目对象

包含项目ID、项目名字、项目创建者、地址与贡献者映射、项目贡献者数、项目所有贡献者列表、每获取一贡献度需要贡献的金额、每次的购买的收益、项目总收益、项目总贡献度

```
struct project{
    uint256 id;
    string name;
    address creator;
    mapping(address => contributor) contributors;
    uint256 contributorNum;
    address [] allContributors;        //项目所有的贡献者
    uint256 weiPerContri;           // 每获取一贡献度需要贡献的金额
    uint256 perBalance;    // 每次的购买的收益
    uint256 totalBalance;  //项目总收益
    uint256 totalContri;     // 项目总贡献度
}
```

### 创建贡献者对象

包含贡献者地址、是否在项目内、在项目中的总贡献度、以及在项目中的贡献比率

```
struct contributor {
    address addr;
    bool isIn;
    uint256 contribution;      // 总贡献度
    uint256 contributionRate;       //贡献比率
}
```

### 项目初始化方法

用户可以通过`createProject`方法对项目进行初始化，并将自己作为贡献者加入项目中

```
// 项目初始化
function createProject (string memory name, uint256 weiPerContri) public returns(uint256) {
    uint256 id = projectID;
    // 初始化项目
    projects[id].id = id;
    projects[id].name = name;
    projects[id].creator = msg.sender;
    projects[id].weiPerContri = weiPerContri;
    projectsKeys.push(id);
    // 初始化创建者
    projects[id].contributors[msg.sender].addr = msg.sender;
    projects[id].contributors[msg.sender].isIn = true;
    projectID ++;
    return id;
}
```

**加入项目方法**

用户通过`joinPro`方法将自己加入指定的项目（输入的项目ID）中、并通过输入的value，以及每获取一贡献度需要贡献的金额计算保存用户的贡献度

```
// 加入项目，购买贡献度
function joinPro(uint256 id) public payable returns(uint){
    project storage pro = projects[id];
    uint256 contriToBuy = msg.value / pro.weiPerContri;
    if (!projects[id].contributors[msg.sender].isIn)
    {
        projects[id].contributors[msg.sender].isIn = true;
        projects[id].contributors[msg.sender].addr = msg.sender;
        projects[id].contributorNum += 1;
        projects[id].allContributors.push(msg.sender);
    }

    projects[id].contributors[msg.sender].contribution += contriToBuy;
    projects[id].totalContri += contriToBuy;
    //从加入者的账户转入合约当中
    projects[id].totalBalance += msg.value;
    payable(address(this)).transfer(msg.value);
    //payable(address(this)).transfer(msg.value);

    return contriToBuy;
}
```

**购买项目并触发以太币分配方法**

用户通过`buyProject`方法触发对项目的购买，同时调用`sendBouns`方法对参与项目的贡献者进行以太币的分配

```
// 购买项目并触发分配
function buyProject(uint256 _projectID) public payable {
    projects[_projectID].totalBalance += msg.value;
    projects[_projectID].perBalance = msg.value;
    //从购买人账户转到合约账户中
    payable(address(this)).transfer(msg.value);
    sendBouns(_projectID);
}
```

```
// 分配收入
function sendBouns(uint256 _projectID) public payable {
    uint256 proContri = projects[_projectID].totalContri;
    for(uint256 i=0;i<projects[_projectID].contributorNum;i++){
        address add=projects[_projectID].allContributors[i];
        uint256 personalContri = projects[_projectID].contributors[add].contribution;
        address payable _payableAddr = payable( projects[_projectID].contributors[add].addr);
        uint256 bounsAmount =  projects[_projectID].perBalance * personalContri / proContri * 80 / 100;
        //从合约中转到每个贡献者用户中
        (_payableAddr).transfer(bounsAmount);
    }
}
```

**支持转账的payable方法**

```
fallback() external payable {}

receive() external payable {}
```

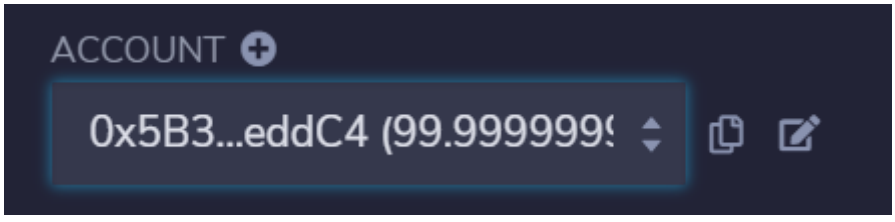
**获取当前用户账户金额的getBanlance方法**

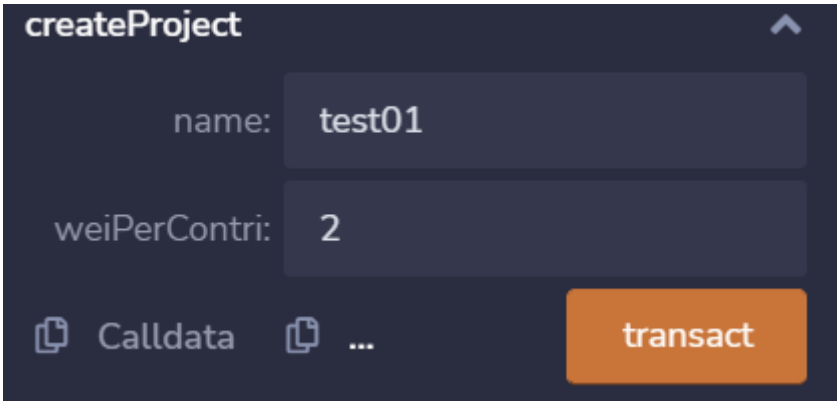
```
function getBanlance() view public returns(uint256){
    return (address(this).balance);
}
```

## 项目运行

**0x5B用户创建项目**

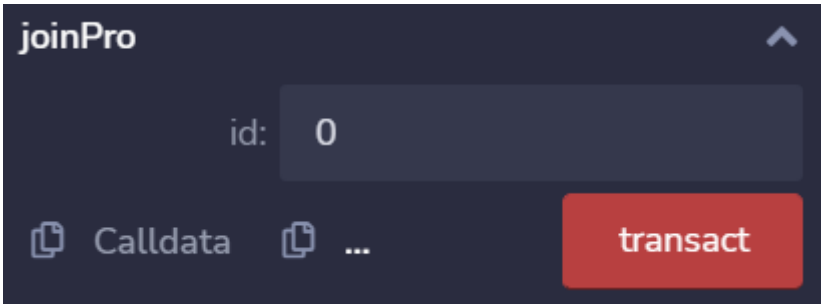
设定项目名称`test01`，项目每获取一贡献度需要贡献的金额为2wei

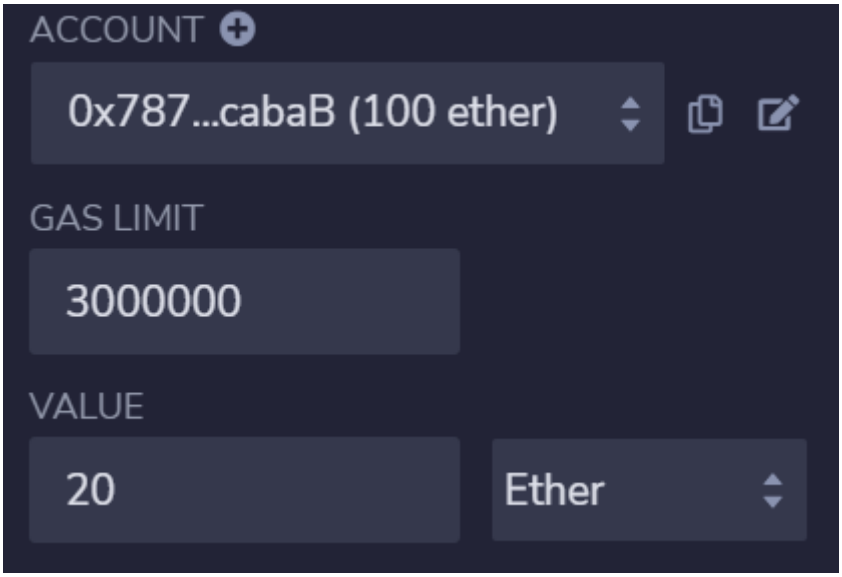





**0x78用户加入项目**

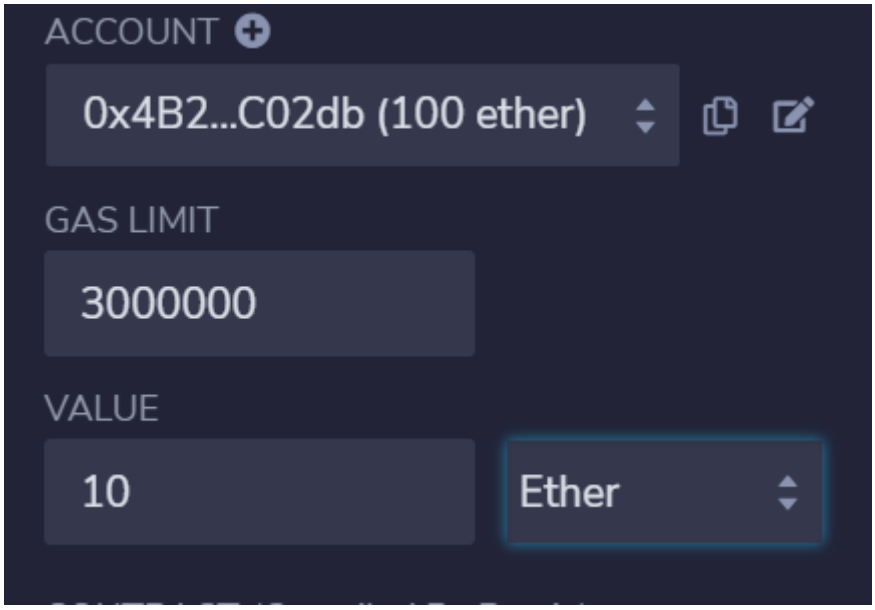
加入项目ID为0的项目，并用20以太购买项目贡献度



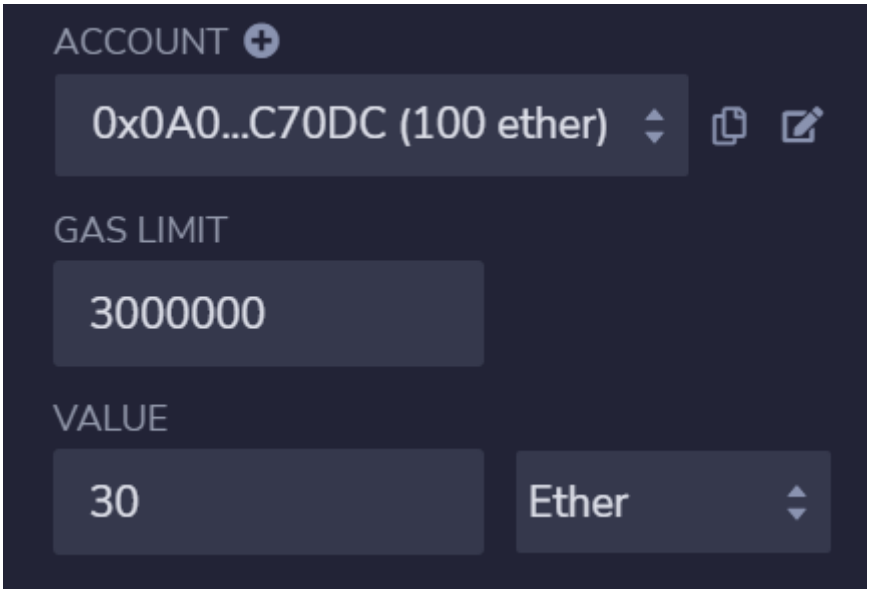

**0x4B用户加入项目**

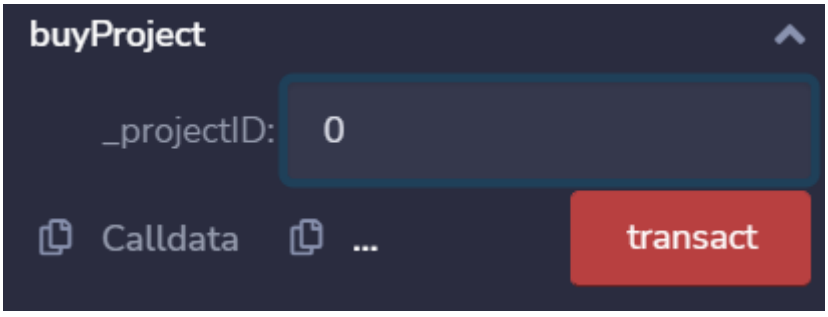
加入项目ID为0的项目，并用10以太购买项目贡献度




**0x0A用户购买项目**

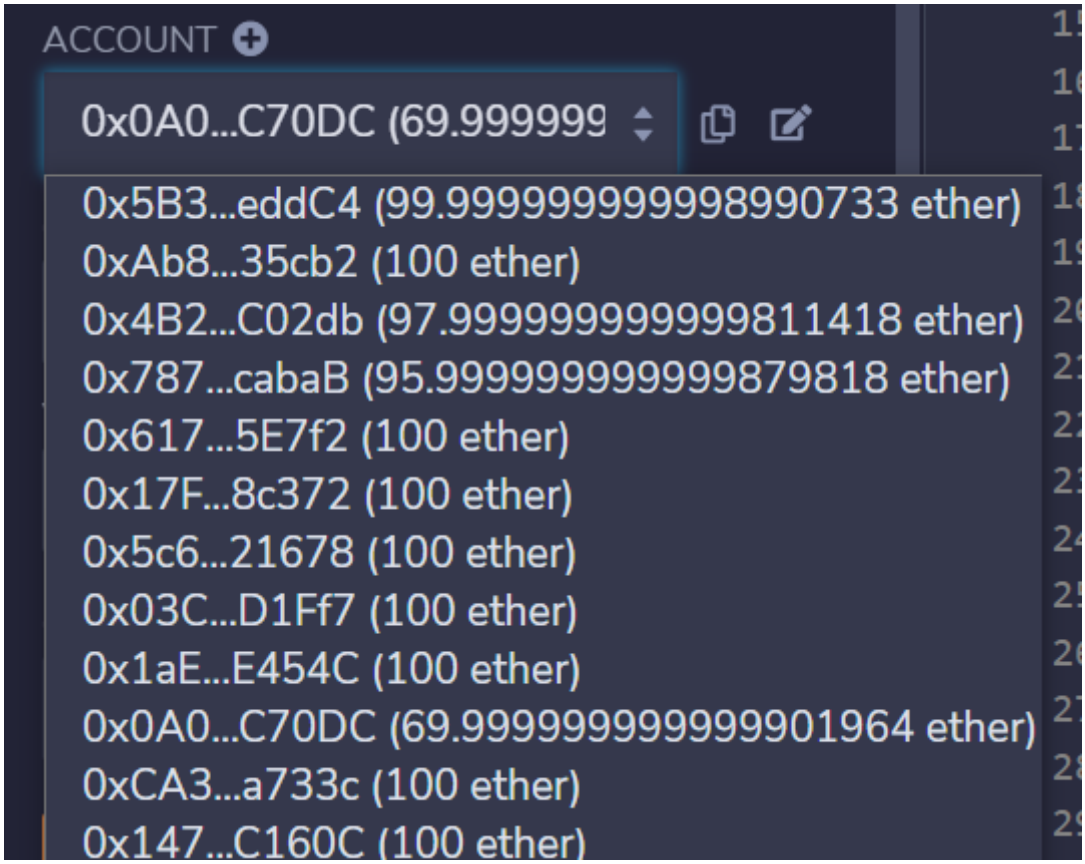
花费30以太，购买项目ID为0的项目




**查看贡献度**

触发金额分配方法，将以太分给项目的贡献者




**`getBanlance`方法**

获取当前用户账户中的金额