第9章 智能合约抽奖

目录

CONTENTS

CONTENTS

章节

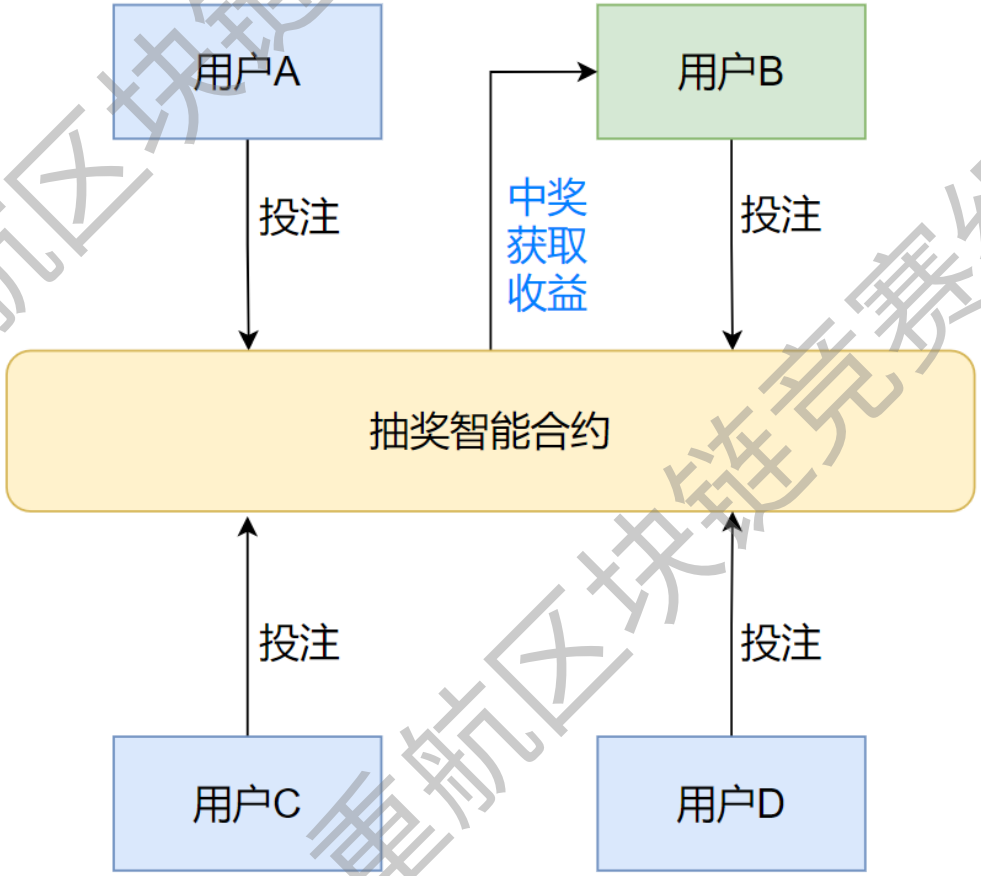
简单抽奖-合约设计

1.1 合约设计

腾讯云

合约设计

- 本章节完成一个抽奖智能合约，用户可以进行投注，等待开奖时抽取中奖用户，中奖用户获取收益



合约设计

- 抽奖合约中包括以下全局变量，事件和函数

CONTENTS

章节

简单抽奖-合约编写

2.1 合约编写

腾讯云

合约编写

- 初始化抽奖合约，使用solidity 0.6.10版本，抽奖合约名为Lottery

```
// 声明solidity版本为0.6.10
pragma solidity 0.6.10;

// 定义Lottery合约
contract Lottery {

}
```

合约编写

- 定义全局变量，包括合约创建者，是否停止投注，当前期数，单次抽奖价格，抽奖结构体

```
// 合约创建者
address owner;
// 是否停止投注
bool public closed;
// 当前抽奖期数
uint public currentSequnece = 1000;
// 单次抽奖价格
uint public price = 10;

// 抽奖结构体
struct Bet {
    address betUser; // 投注者
    bytes3  betStr; // 投注内容
    uint sequence; // 抽奖期数
}
```

合约编写

- 定义全局变量，包括当前所有投注，用户的投注，用户的余额，历史中奖者，当期中奖者

```
// 当前所有的投注
Bet[] currentBets;

// 用户的投注
mapping(address => Bet[]) userBets;
// 用户的余额
mapping(address => uint) balance;

// 历史所有中奖者
address[] public allWinners;

// 当前期中奖者
address public currentWinner;
```

合约编写

- 定义事件，在合约调用时记录日志

```
// 事件日志，记录中奖者及所有中奖人数
event Winner(address winner, uint winnerCount);
// 事件日志，记录投注用户、投注内容、投注期数
event UserBet(address betUser, bytes3 betStr, uint betSeq);
```

- 定义修饰符，限制合约函数执行

```
// 修饰符，限制合约创建者操作
modifier onlyOwner() {
    require(msg.sender == owner);
    _;
}
```

合约编写

- 定义构造函数，在合约执行时初始化owner变量

```
// 构造函数，初始化owner合约创建者变量
constructor() public {
    owner = msg.sender;
}
```

合约编写

- 定义deposit函数和withdraw函数，处理用户存入和取出余额逻辑

```
// 存入余额
function deposit(uint value) public {
    balance[msg.sender] += value;
}
```

```
// 取出余额
function withdraw() public returns (uint) {
    uint value = balance[msg.sender];
    balance[msg.sender] = 0;
    return value;
}
```

合约编写

- 定义bet函数，用户进行投注，每次投注时会扣除10余额

```
function bet(bytes3 betStr) public {
    // 判断是否停止投注
    require(closed == false, "已停止投注");
    // 判断余额是否充足
    require(balance[msg.sender] >= price, "余额不足");
    // 减少余额
    balance[msg.sender] -= price;
    Bet memory item  = Bet({
        betUser: msg.sender,
        betStr: betStr,
        sequence: currentSequnece
    });
    // 将投注信息添加到currentBets和userBets中
    currentBets.push(item);
    userBets[msg.sender].push(item);
    emit UserBet(msg.sender, betStr, currentSequnece);
}
```

合约编写

- 定义getUserBets函数，功能为获取用户历史的所有投注

```
// 获取用户历史所有投注
function getUserBets() public view returns (bytes3[] memory, uint[] memory){
    // 获取用户历史所有投注bets数组
    Bet[] memory bets = userBets[msg.sender];
    // 获取数组长度
    uint length = bets.length;
    // 初始化strs和seqs数组
    bytes3[] memory strs = new bytes3[](length);
    uint[] memory seqs = new uint[](length);
    // 填充strs和seqs数组
    for(uint i = 0; i <length; i++){
        Bet memory item = bets[i];
        strs[i]=(item.betStr);
        seqs[i] = (item.sequence);
    }
    return (strs, seqs);
}
```

合约编写

- 定义getUserBetsCount函数，功能为获取用户历史投注次数

```
// 获取用户历史投注次数
function getUserBetsCount() public view returns (uint){
    return userBets[msg.sender].length;
}
```

- 定义random函数，功能为在当期奖池中随机抽取一个中奖地址

```
// 在当期奖池中抽取一个地址
function random() private view returns (address){
    // 随机生成一个索引值
    uint randIdx = (block.number^block.timestamp) % currentBets.length;
    // 根据索引取出中奖地址
    Bet memory item = currentBets[randIdx] ;
    return item.betUser;
}
```

合约编写

- 定义closeAndFindWinner函数，功能为开奖，在函数内部执行random函数，将中奖收益转移给获奖者

```
// 开奖，获奖者获取2倍投注的收益
function closeAndFindWinner() public onlyOwner {
    // 限制处于投注未截止状态
    require(closed == false);
    // 限制投注人数必须大于等于2人
    require(currentBets.length >= 2);
    closed = true;
    // 获取中奖地址
    currentWinner = random();
    // 中奖地址获取2倍收益
    balance[currentWinner] += price * 2;
    // 将中奖者添加到历史中奖者数组中
    allWinners.push(currentWinner);
    emit Winner(currentWinner, allWinners.length);
}
```

合约编写

- 定义reOpen函数，功能为初始化奖池，重新接受用户投注

```
// 初始化奖池，重新接受投注
function reOpen() public onlyOwner {
    // 限制处于投注截止状态
    require(closed == true);
    closed = false;
    // 删除本期投注中的所有数据
    for (uint i = 0; i < currentBets.length; i++){
        delete userBets[currentBets[i].betUser];
    }
    // 初始化当前所有投注数组
    delete currentBets;
    // 初始化当前中奖者变量
    delete currentWinner;
}
```

CONTENTS

章节

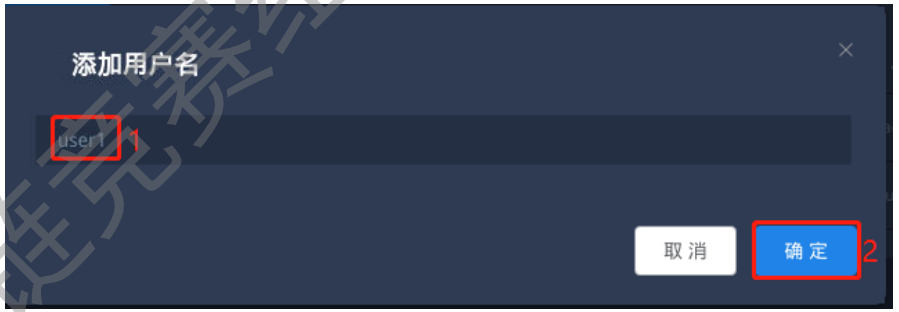
简单抽奖-合约测试

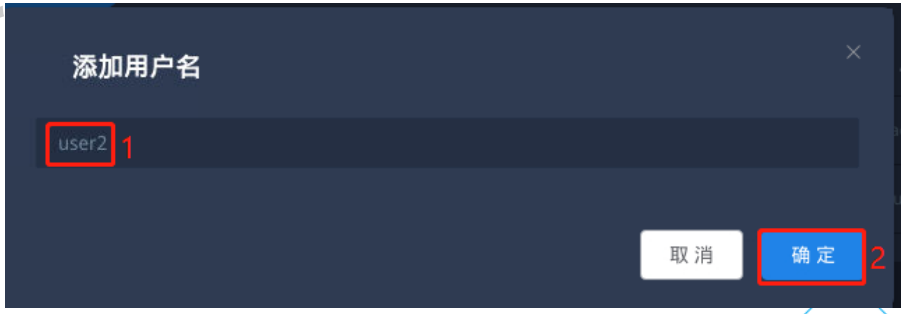
3.1 合约测试

腾讯云

合约测试

- 合约部署前，创建三个测试账户，分别为admin，user1，user2





合约测试

- user1、user2、user3三个用户对应的地址

合约测试

- 合约编译完成后，使用管理员admin账户部署，部署完成后执行两次deposit函数，向user1和user2地址分别存入100余额

合约测试

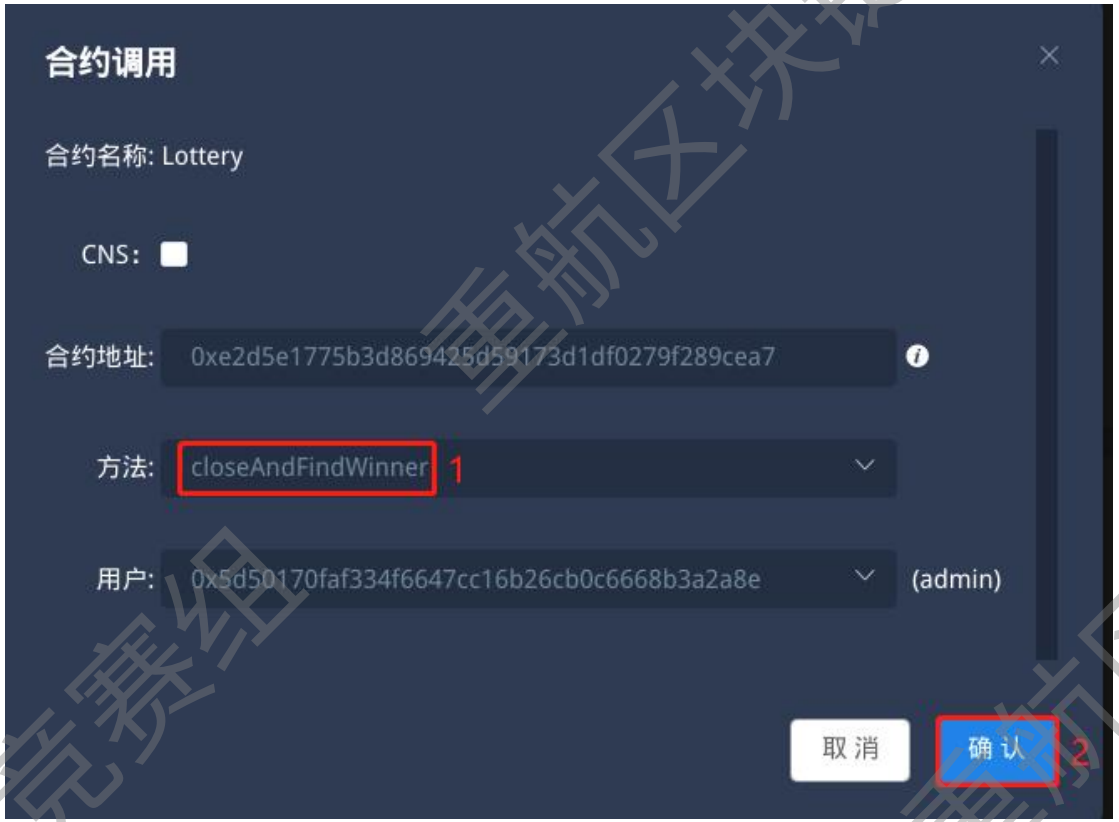
- 开始投注，user1地址执行bet函数投注，投注内容为0x112233，投注完成后，UserBet事件记录了日志

合约测试

- 开始投注，user2地址执行bet函数投注，投注内容为0x123123，投注完成后，UserBet事件记录了日志

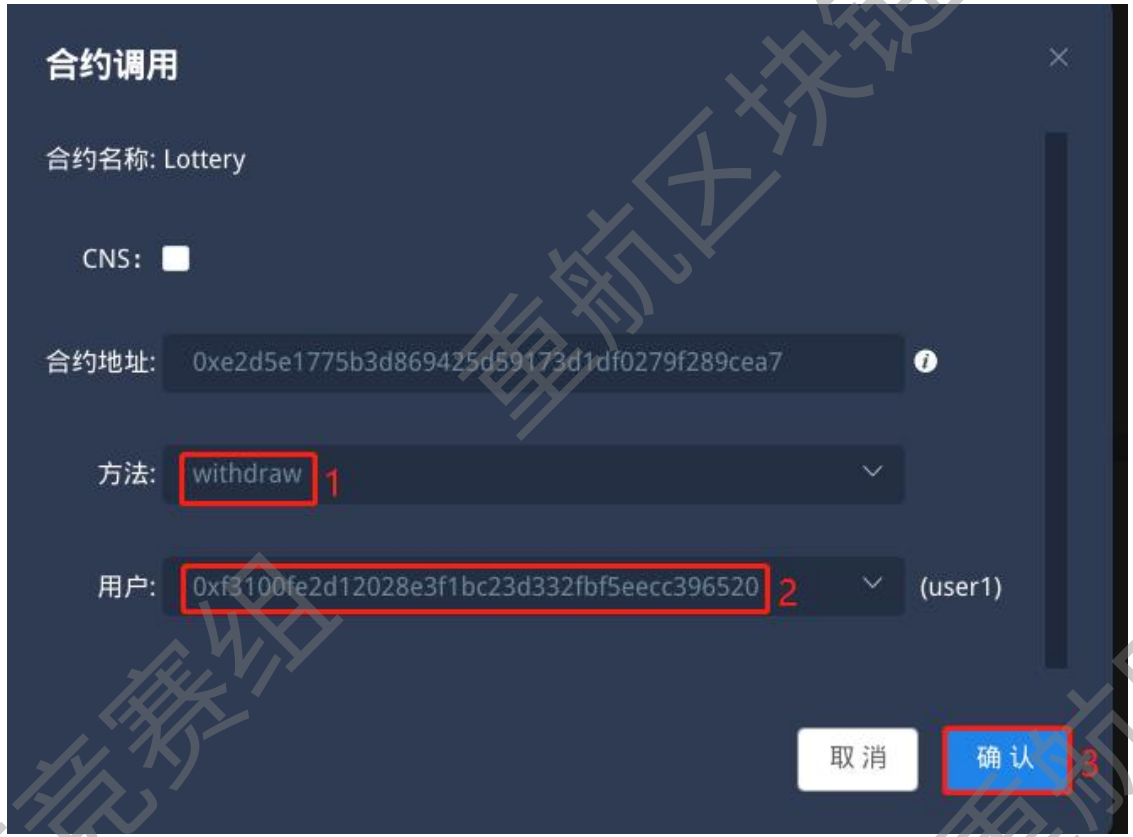
合约测试

- 投注完成后，使用管理员admin账户调用closeAndFindWinner函数，执行开奖流程，交易回执中显示中奖地址为user1



合约测试

- user1取出余额，执行withdraw函数，在交易回执中可以观察到取出了110余额（100本金+10收益）



总结

- 本节课我们学习了
  - 简单抽奖合约的设计
  - 简单抽奖合约的编写
  - 简单抽奖合约的测试

谢谢