# Реализация лотереи при помощи смарт-контракта на Solidity

Анисова П.А., ТБЧ18-1м

# Правила лотереии

- Ограничение в 5 пользователей
- пользователь должен заплатить 0,1 эфира, чтобы присоединиться к лотерее
- один и тот же пользователь может присоединиться только один раз
- владелец договора может присоединиться к лотерее
- когда присоединяются 5 пользователей, выбирается победитель
- победитель получает все деньги
- новая лотерея начинается, когда выбран очередной победитель

# Функция присоединения к игре

```
function join() public payable {
    require(msg.value == 0.1 ether, "Must send 0.1 ether"); // проверка размера ставки
    require(joinedAlready(msg.sender) == false, "User already joined"); //проверка уникальности юзера
    participants[participantsCount] = msg.sender; //добавляем нового участника
    participantsCount++; //увеличиваем число участников
    if (participantsCount == 5) {
        selectWinner(); //если достигли 5 участников, то выбираем победителя
    }
```

# Проверка на уникальность юзера

```
function joinedAlready(address _participant) private view returns(bool) {
    for(uint i = 0; i < participantsCount; i++) { //проходимся по всем юзерам
        if (participants[i] == _participant) {
            return true; //нашли, выходим
        }
    }
    return false;//не нашли, выходим
}
```

# Выбор победителя

```
function selectWinner() private returns(address) {
    address winner = participants[randomNumber()]; //случайный выбор пользователя
    winner.transfer(address(this).balance); //переводим все деньги ему
    delete participants; //очищаем список участников
    participantsCount = 0; //обнуляем счетчик участников
    return winner; //возвращаем адрес победителя
  }
```

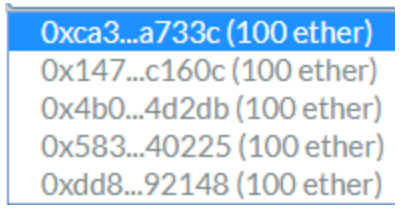
# Полный код

```
contract Lottery {
    address[5] participants;
    uint8 participantsCount = 0;
    uint randNonce = 0;

    function join() public payable {
        require(msg.value == 0.1 ether, "Must send 0.1 ether");
        require(joinedAlready(msg.sender) == false, "User already joined");
        participants[participantsCount] = msg.sender;
        participantsCount++;
        if (participantsCount == 5) {
            selectWinner();
        }
    }

    function joinedAlready(address _participant) private view returns(bool) {
        for(uint i = 0; i < participantsCount; i++) {
            if (participants[i] == _participant) {
                return true;
            }
        }
        return false;
    }

    function selectWinner() private returns(address) {
        address winner = participants[randomNumber()];
        winner.transfer(address(this).balance);
        delete participants;
        participantsCount = 0;
        return winner;
    }

    function randomNumber() private returns(uint) {
        uint rand = uint(keccak256(abi.encodePacked(now, msg.sender, randNonce))) % 5;
        randNonce++;
        return rand;
    }
}
```

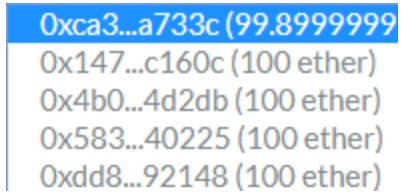
# Развертывание смарт-контракта

# Процесс игры

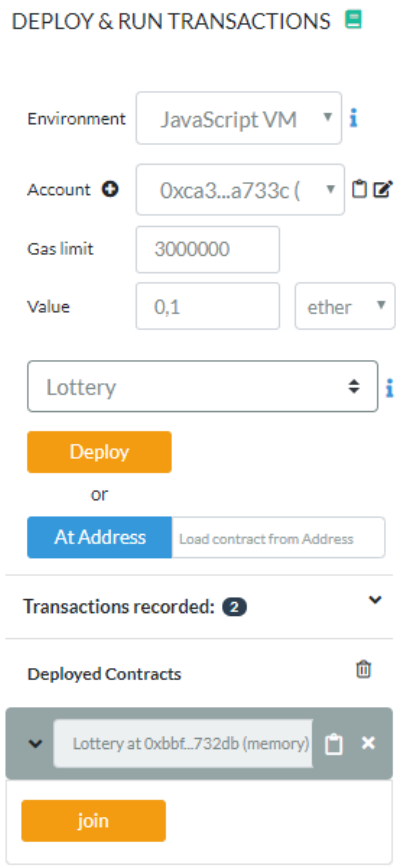
Аккаунты до вступления в лотерею




Первый участник (автор контракта) вступил в игру




Транзакция со вступлением в игру



# Конец игры

Все игроки присоединились к игре

```
transact to Lottery.join pending ...

[vm] from:0xca3...a733c to:Lottery.join() 0x692...77b3a value:100000000000000000 wei data:0xb68...8a363 logs:0 hash:0x086...fe038

transact to Lottery.join pending ...

[vm] from:0x147...c160c to:Lottery.join() 0x692...77b3a value:100000000000000000 wei data:0xb68...8a363 logs:0 hash:0xe0d...c0fea

transact to Lottery.join pending ...

[vm] from:0x4b0...4d2db to:Lottery.join() 0x692...77b3a value:100000000000000000 wei data:0xb68...8a363 logs:0 hash:0xca0...b5756

transact to Lottery.join pending ...

[vm] from:0x583...40225 to:Lottery.join() 0x692...77b3a value:100000000000000000 wei data:0xb68...8a363 logs:0 hash:0x62a...841eb

transact to Lottery.join pending ...

[vm] from:0xdd8...92148 to:Lottery.join() 0x692...77b3a value:100000000000000000 wei data:0xb68...8a363 logs:0 hash:0xfef...12371

>
```

Второй игрок выиграл 0,5 эфира