

Lottery Smart Contract Project

Program for the Lottery_SmartContract :-

```
// SPDX-License-Identifier: GPL-3.0
```

```
pragma solidity 0.8.16;           //older version for the “block.difficulty” feature...
```

```
contract lottery{
```

```
    //entities - manager, players and winner
```

```
    address public manager;
```

```
    address payable[] public players;
```

```
    address payable public winner;
```

```
    constructor(){
```

```
        manager=msg.sender;
```

```
    }
```

```
    //participate for players
```

```
    function participate() public payable{
```

```
        require(msg.value==1 ether,"Please pay one ether only! ");    //fee to enter the lottery = 1 ether
```

```
        players.push(payable(msg.sender));    //to enter if the condition is true
```

```
    }
```

```
    //check how much ether is present only for manager
```

```
    function getBalance() public view returns (uint){
```

```
        require(manager==msg.sender,"Sorry, you are not the Manager");
```

```
        return address(this).balance;
```

```
    }
```

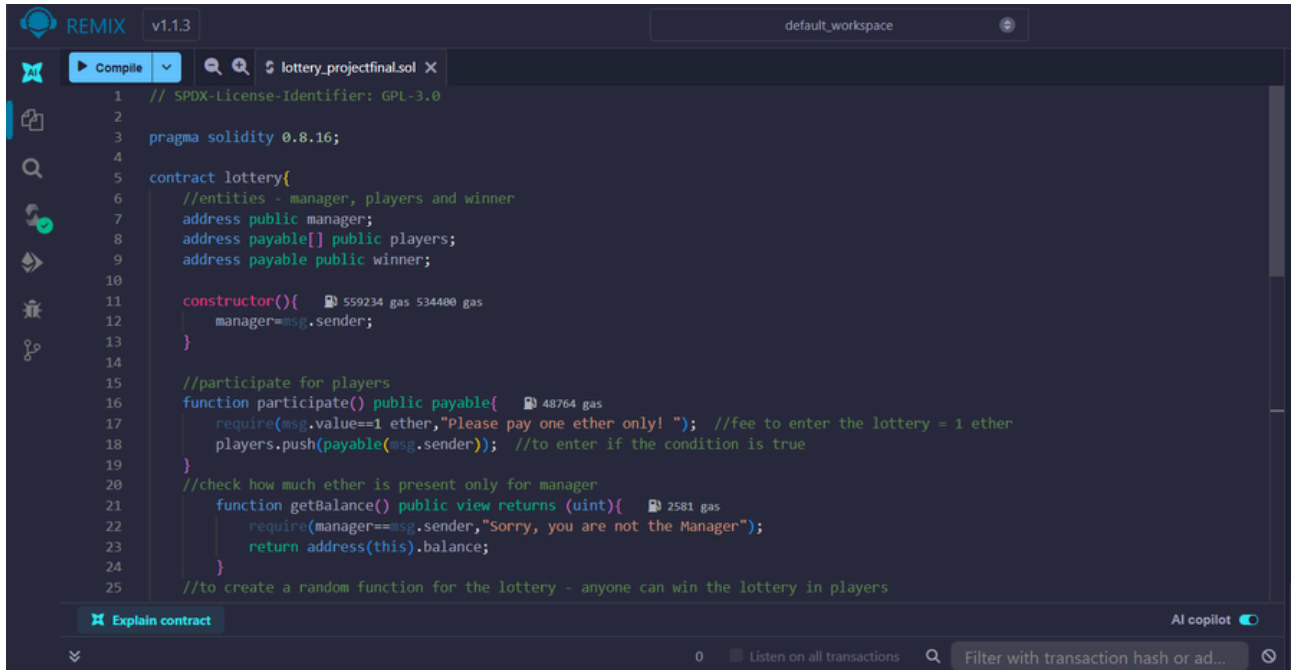
```
    //to create a random function for the lottery - anyone can win the lottery in players
```

```
function random() internal view returns(uint){  
    //generates a random number  
    return uint(keccak256(abi.encodePacked(block.difficulty,block.timestamp,players.length)));  
}
```

```
function pickWinner() public{  
    require(manager==msg.sender,"You are not the manager");  
    require(players.length>=3,"Players are less than 3");  
  
    uint r=random();  
    uint index = r%players.length;  
    winner=players[index];  
    winner.transfer(getBalance());  
    players= new address payable[](0);           //to initialize the players array back to 0  
}  
}
```

Implementation of the program :-

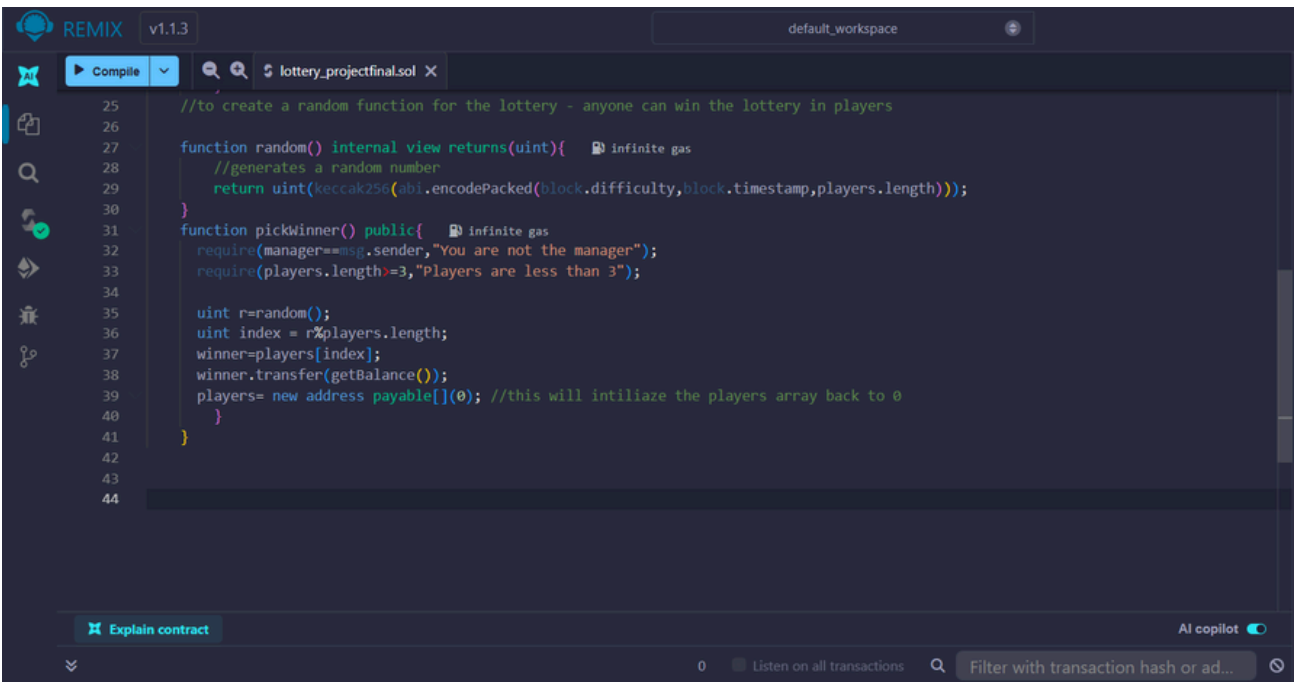
In Solidity :-



The screenshot shows the REMIX IDE interface with a file named 'lottery_projectfinal.sol' open. The code defines a 'lottery' contract with the following components:

- License:** // SPDX-License-Identifier: GPL-3.0
- Pragma:** pragma solidity 0.8.16;
- Entities:** address public manager; address payable[] public players; address payable public winner;
- Constructor:** Takes no arguments, sets manager to msg.sender. Gas: 559234 gas 534400 gas.
- participate() function:** Takes no arguments, requires msg.value == 1 ether, pushes msg.sender to the players array. Gas: 48764 gas.
- getBalance() function:** Public view function returning uint, requires msg.sender == manager. Gas: 2581 gas.
- Comments:** //to create a random function for the lottery - anyone can win the lottery in players

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity 0.8.16;
4
5 contract lottery{
6     //entities - manager, players and winner
7     address public manager;
8     address payable[] public players;
9     address payable public winner;
10
11     constructor(){
12         manager=msg.sender;
13     }
14
15     //participate for players
16     function participate() public payable{
17         require(msg.value==1 ether,"Please pay one ether only! "); //fee to enter the lottery = 1 ether
18         players.push(payable(msg.sender)); //to enter if the condition is true
19     }
20
21     //check how much ether is present only for manager
22     function getBalance() public view returns (uint){
23         require(manager==msg.sender,"Sorry, you are not the Manager");
24         return address(this).balance;
25     }
26
27     //to create a random function for the lottery - anyone can win the lottery in players
```



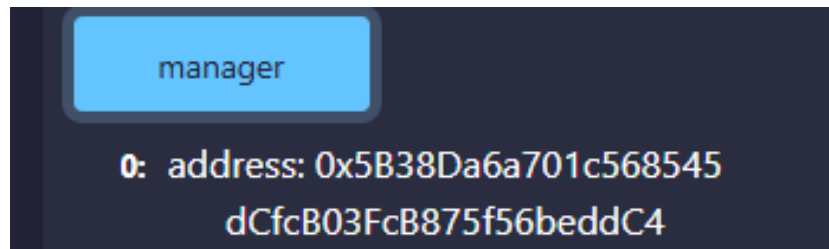
The screenshot shows the continuation of the 'lottery' contract in the REMIX IDE. The code includes the following components:

- random() function:** Internal view function returning uint, generates a random number using keccak256. Gas: Infinite gas.
- pickWinner() function:** Public function, requires msg.sender == manager and players.length >= 3. It uses random() to pick a winner and transfer the balance. Gas: Infinite gas.
- Comments:** //this will initialize the players array back to 0

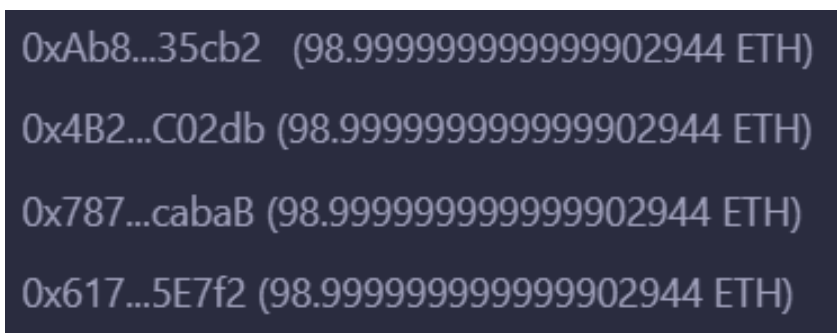
```
25 //to create a random function for the lottery - anyone can win the lottery in players
26
27 function random() internal view returns(uint){
28     //generates a random number
29     return uint(keccak256(abi.encodePacked(block.difficulty,block.timestamp,players.length)));
30 }
31
32 function pickWinner() public{
33     require(manager==msg.sender,"You are not the manager");
34     require(players.length>=3,"Players are less than 3");
35
36     uint r=random();
37     uint index = r%players.length;
38     winner=players[index];
39     winner.transfer(getBalance());
40     players= new address payable[](0); //this will initialize the players array back to 0
41 }
42
43
44
```

Deployment of the program :-

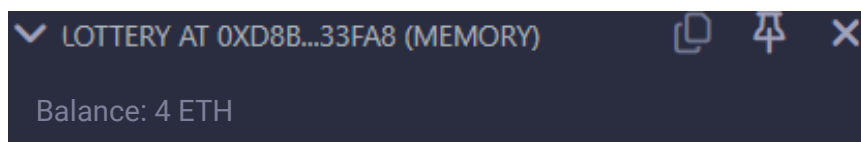
Manager in the Lottery :-



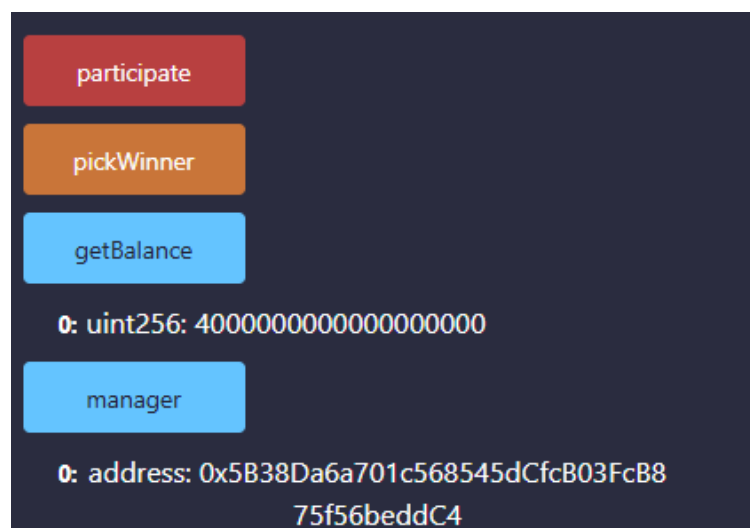
Players in the lottery (Total-4):-



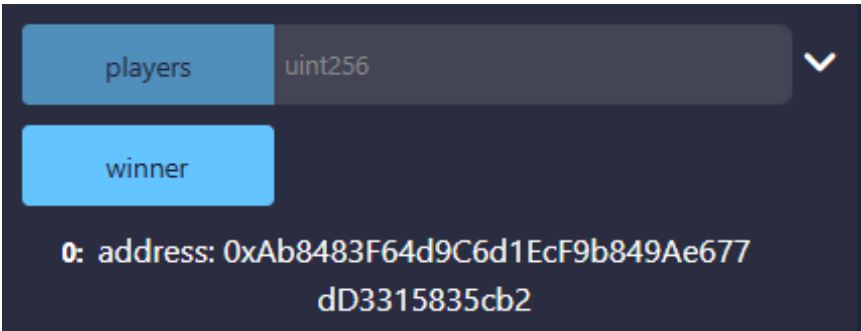
Balance after adding fees to participate in the lottery :-



Checking the Balance by the Address of Manager :-



Getting the Winner of the Lottery :-



Transactions History :-



Winner Address has got the amount :-

