

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 intiliaze the players array back to 0  
}  
}
```

Implementation of the program :-

In Solidity :-

The screenshot shows the REMIX IDE interface with the file 'lottery_projectfinal.sol' open. The code defines a lottery contract with a constructor, a 'participate' function for players to enter with 1 ether, a 'getBalance' view function for the manager, and a random number generation function. The code is annotated with comments explaining its purpose.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity 0.8.16;

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
}
```

The screenshot shows the REMIX IDE interface with the file 'lottery_projectfinal.sol' open. The code now includes a 'random' internal view function that generates a random number using the keccak256 hash of the block's difficulty and timestamp. It also includes a 'pickWinner' function that picks a random player as the winner and transfers the collected ether to the winner. The code is annotated with comments explaining its purpose.

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

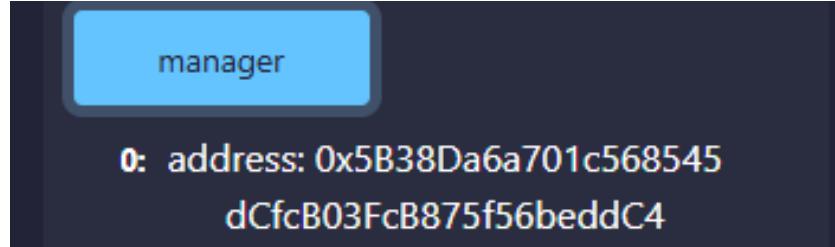
function random() internal view returns(uint){ infinite gas
    //generates a random number
    return uint(keccak256(abi.encodePacked(block.difficulty,block.timestamp,players.length)));
}

function pickWinner() public{
    require(manager==msg.sender,"You are not the manager");
    require(players.length>=3,"Players are less than 3");

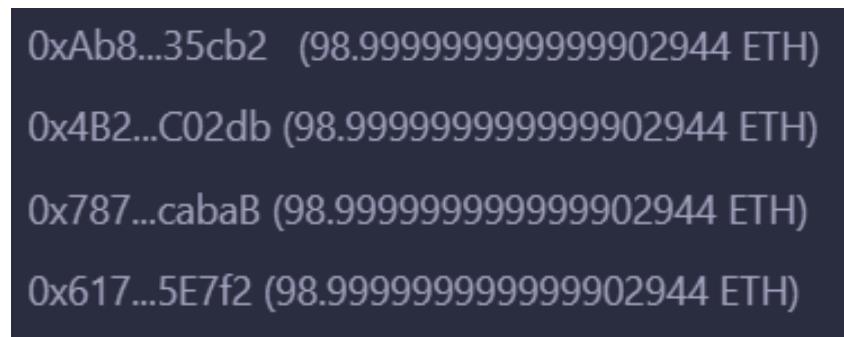
    uint r=random();
    uint index = r%players.length;
    winner=players[index];
    winner.transfer(getBalance());
    players= new address payable[](0); //this will intilizae the players array back to 0
}
```

Deployment of the program :-

Manager in the Lottery :-



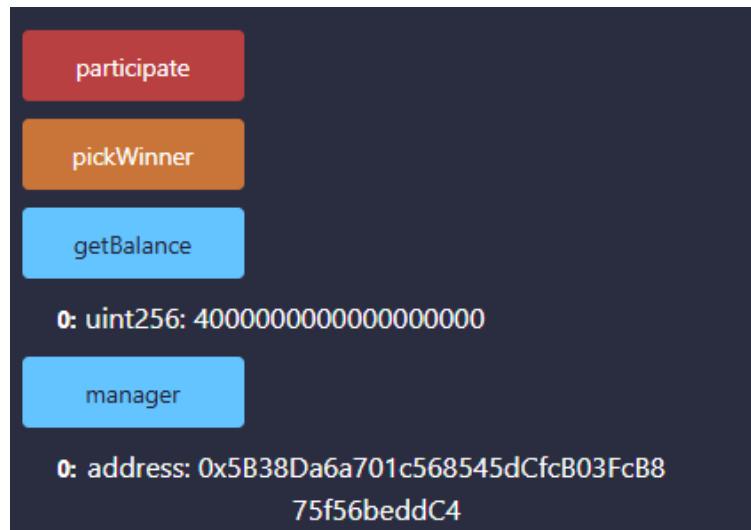
Players in the lottery (Total-4):-



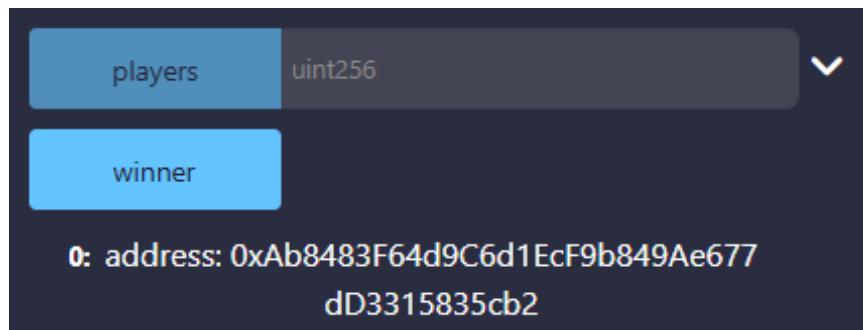
Balance after adding fees to participate in the lottery :-



Checking the Balance by the Address of Manager :-



Getting the Winner of the Lottery :-



Transactions History :-

[vm] from: 0x5B3...eddC4 to: lottery.(constructor) value: 0 wei data: 0x608...00033 logs: 0 hash: 0x53b...2703a transact to lottery.participate pending ...	Debug
[vm] from: 0xAb8...35cb2 to: lottery.participate() 0xd8b...33fa8 value: 10000000000000000000000000000000 wei data: 0xd11...711a2 logs: 0 hash: 0x995...0bfc8 transact to lottery.participate pending ...	Debug
[vm] from: 0x4B2...C02db to: lottery.participate() 0xd8b...33fa8 value: 10000000000000000000000000000000 wei data: 0xd11...711a2 logs: 0 hash: 0x068...fb006 transact to lottery.participate pending ...	Debug
[vm] from: 0x787...cabab to: lottery.participate() 0xd8b...33fa8 value: 10000000000000000000000000000000 wei data: 0xd11...711a2 logs: 0 hash: 0xac...dbd32 transact to lottery.participate pending ...	Debug
[vm] from: 0x617...5E7f2 to: lottery.participate() 0xd8b...33fa8 value: 10000000000000000000000000000000 wei data: 0xd11...711a2 logs: 0 hash: 0x824...a5a5a call to lottery.getBalance	Debug
call [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: lottery.getBalance() data: 0x120...65fe0 transact to lottery.pickWinner pending ...	Debug
[vm] from: 0x5B3...eddC4 to: lottery.pickWinner() 0xd8b...33fa8 value: 0 wei data: 0x5d4...95aea logs: 0 hash: 0x355...a35a6 call to lottery.winner	Debug

Winner Address has got the amount :-

0x5B3...eddC4 (99.99999999999856766 ETH)
0xAb8...35cb2 (102.999999999999868744 ETH)
0x4B2...C02db (98.99999999999902944 ETH)
0x787...cabab (98.99999999999902944 ETH)
0x617...5E7f2 (98.99999999999902944 ETH)
0x17F...8c372 (99.99999999999956922 ETH)