

# 한 번에 끝내는 블록체인 개발 A to Z

---

## Chapter 3

### Lottery 컨트랙트 v1 개발

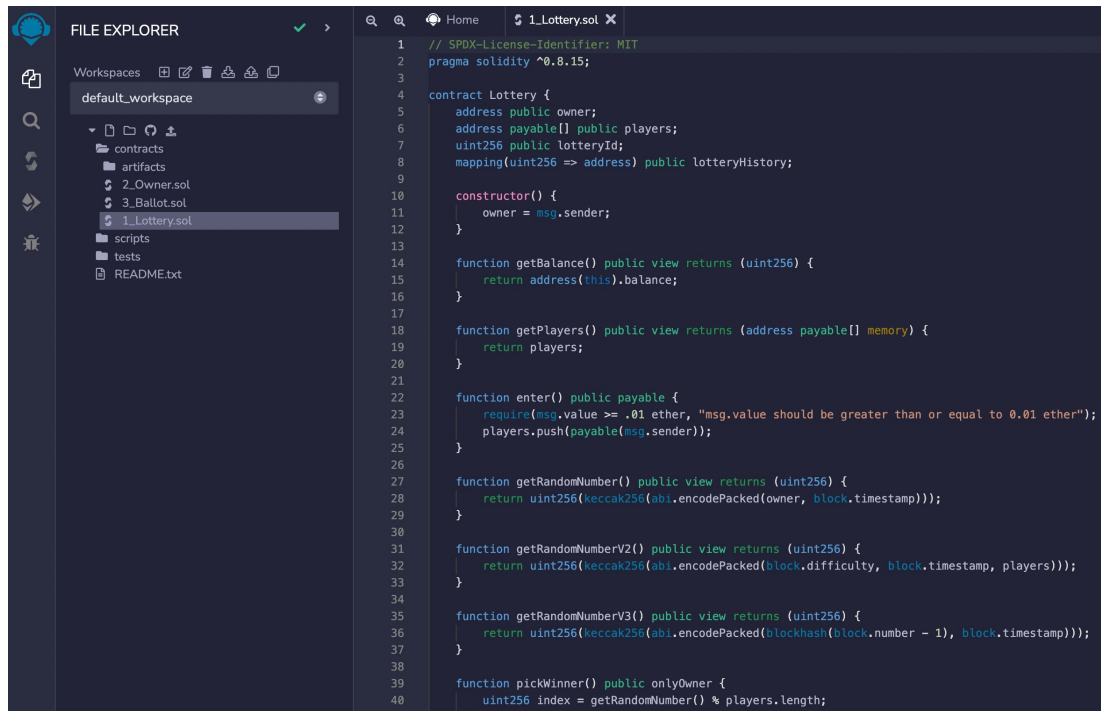
Chapter 3

Lottery 컨트랙트 v1 개발

# Lottery 컨트랙트 테스트하기 - Remix IDE

# Remix IDE에 파일 준비하기

- default\_workspace에 contracts/  
디렉토리 밑에 1번 파일을 Lottery 파일로  
변경 (파일 새로 만들어도 무방함)
- Lottery.sol 컨트랙트 내용 복사해오기

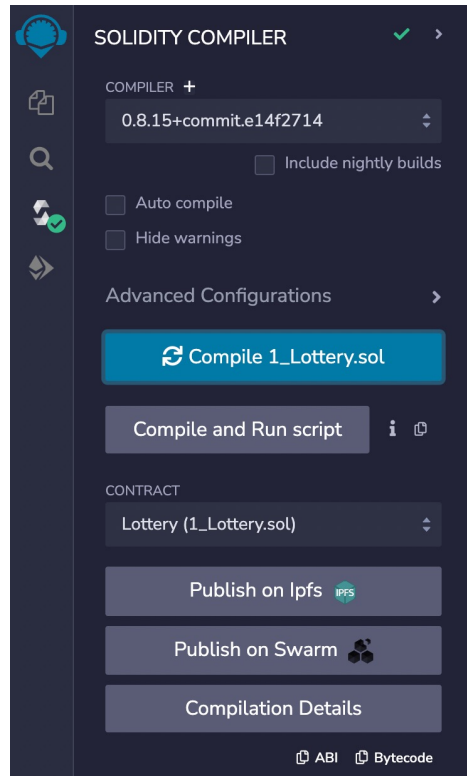


The screenshot displays the Remix IDE interface. On the left, the 'FILE EXPLORER' panel shows the 'default\_workspace' with a directory structure including 'contracts', 'artifacts', 'scripts', and 'tests'. The '1.Lottery.sol' file is selected under the 'contracts' directory. The main editor area on the right shows the content of '1.Lottery.sol', which is a Solidity contract for a lottery. The code includes a pragma statement for Solidity version ^0.8.15, a contract definition for 'Lottery', a constructor, and several public functions: 'getBalance', 'getPlayers', 'enter', 'getRandomNumber', 'getRandomNumberV2', 'getRandomNumberV3', and 'pickWinner'.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.15;
3
4 contract Lottery {
5     address public owner;
6     address payable[] public players;
7     uint256 public lotteryId;
8     mapping(uint256 => address) public lotteryHistory;
9
10    constructor() {
11        owner = msg.sender;
12    }
13
14    function getBalance() public view returns (uint256) {
15        return address(this).balance;
16    }
17
18    function getPlayers() public view returns (address payable[] memory) {
19        return players;
20    }
21
22    function enter() public payable {
23        require(msg.value >= .01 ether, "msg.value should be greater than or equal to 0.01 ether");
24        players.push(payable(msg.sender));
25    }
26
27    function getRandomNumber() public view returns (uint256) {
28        return uint256(keccak256(abi.encodePacked(owner, block.timestamp)));
29    }
30
31    function getRandomNumberV2() public view returns (uint256) {
32        return uint256(keccak256(abi.encodePacked(block.difficulty, block.timestamp, players)));
33    }
34
35    function getRandomNumberV3() public view returns (uint256) {
36        return uint256(keccak256(abi.encodePacked(blockhash(block.number - 1), block.timestamp)));
37    }
38
39    function pickWinner() public onlyOwner {
40        uint256 index = getRandomNumber() % players.length;
41    }
42 }
```

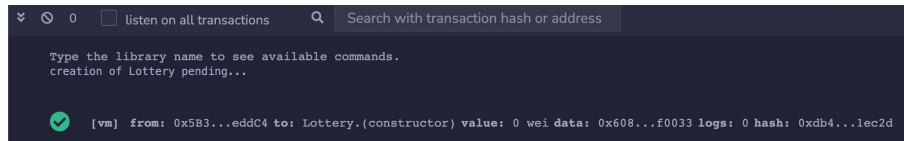
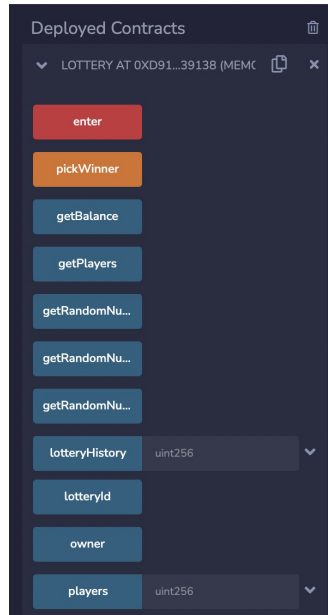
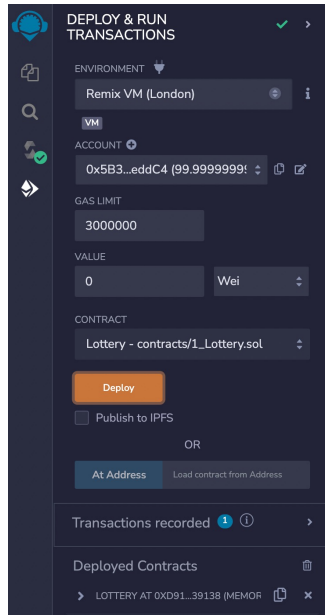
# 컨트랙트 컴파일하기

- Solidity Compiler 탭에서 Compiler 드롭다운 클릭 후, 0.8.15 버전 선택
- 파란색 Compile 1\_Lottery.sol 버튼 클릭



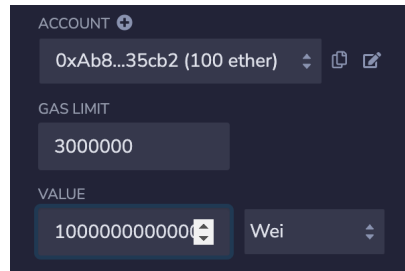
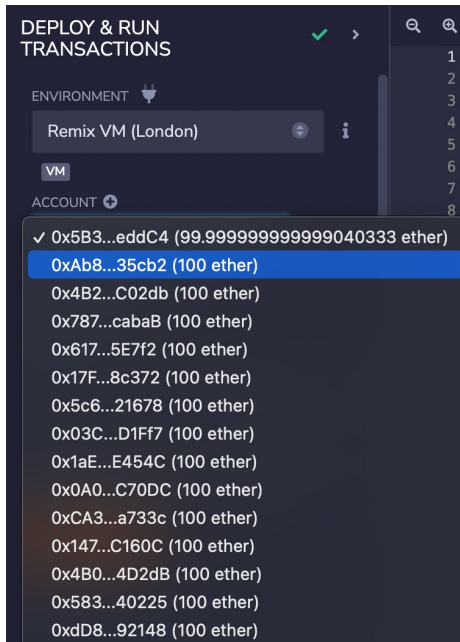
# 컨트랙트 배포하기

- Deploy & Run Transactions 탭에서 Contract 부분에 Lottery 컨트랙트가 로드되어있는지 체크 후 Deploy 버튼 클릭
- 배포 성공하면 터미널에 초록색 체크 모양 로그가 뜬
- 배포되고 나면 Deploy 버튼 아래에 Deployed Contracts 부분에 'Lottery At 0x...' 가 생김
- 앞에 '>' 모양 클릭하면, Lottery 컨트랙트의 함수들이 뜬



# 컨트랙트 테스트하기 - enter()

- Account 부분에서 2번째 계정으로 바꿔주기 (첫번째 계정은 owner로만 사용하기 위함)
- Value 부분에서  $10^{16}$  WEI (= 0.01 ETH) 입력 (cf. 1 ETH =  $10^{18}$  WEI)
  - Remix에선 소수점 지원 x
  - <https://eth-converter.com/> (eth <-> wei converter)
- enter() 클릭
  - 터미널에 트랜잭션 성공 로그 출력

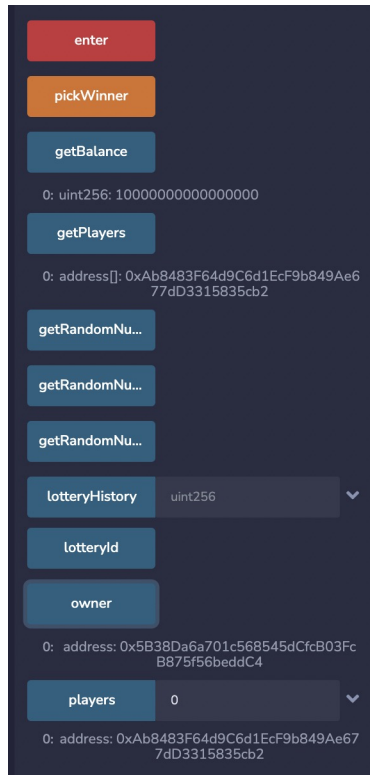


```
transact to Lottery.enter pending ...
```

```
[vm] from: 0xAb8...35cb2 to: Lottery.enter() 0xd91...39138 value: 10000000000000000 wei data: 0xe97...dcb62 logs: 0 hash: 0x783...7a009
```

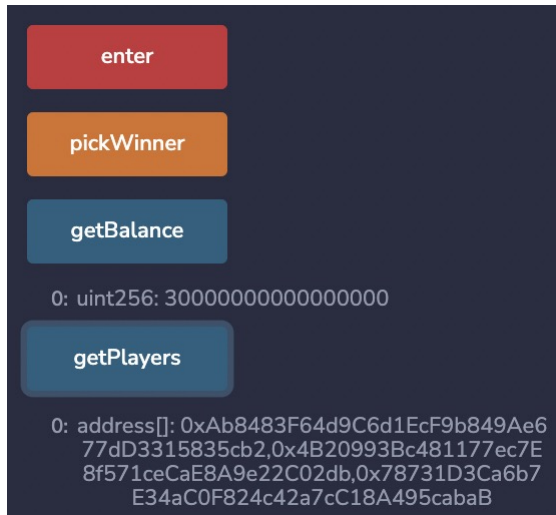
# 컨트랙트 테스트하기 - enter()

- enter() 잘 되었는지 view 함수로 결과  
확인해보기
  - getBalance():  $10^{16}$  WEI (= 0.01 ETH)
  - getPlayers(): 2번째 account
  - owner(): 1번째 account
  - players[0]: 2번째 account



# 컨트랙트 테스트하기 - enter()

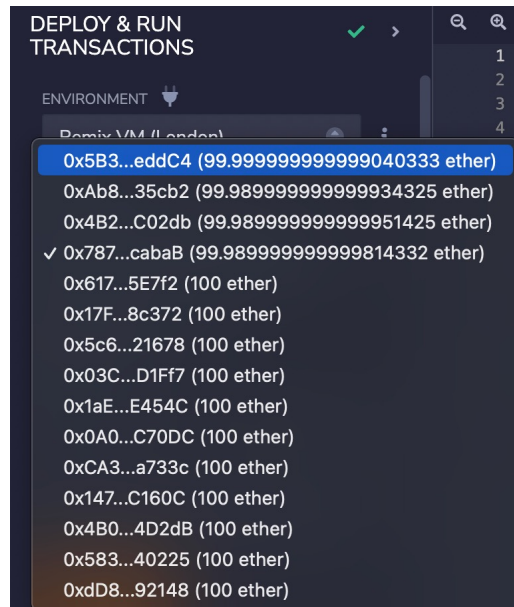
- 3, 4번째 account로도 똑같이 enter() 호출
- view 함수로 결과 확인
  - getBalance():  $30^{16}$  WEI (= 0.03 ETH)
  - getPlayers(): 2,3,4번째 accounts





## - pickWinner()

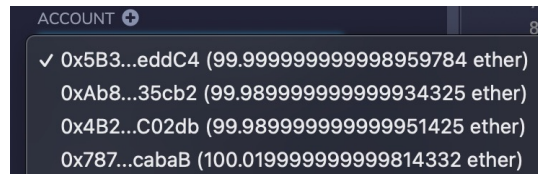
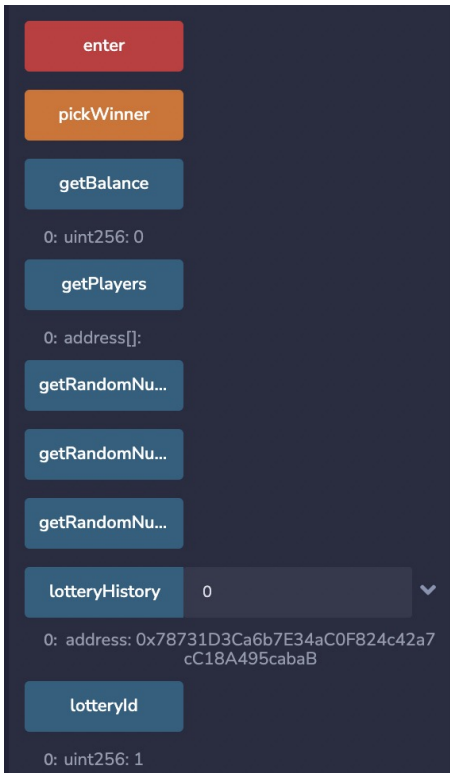
- pickWinner()는 owner만 호출할 수 있으므로 account를 owner account로 변경
- pickWinner() 클릭



# 컨트랙트 테스트하기

## - pickWinner()

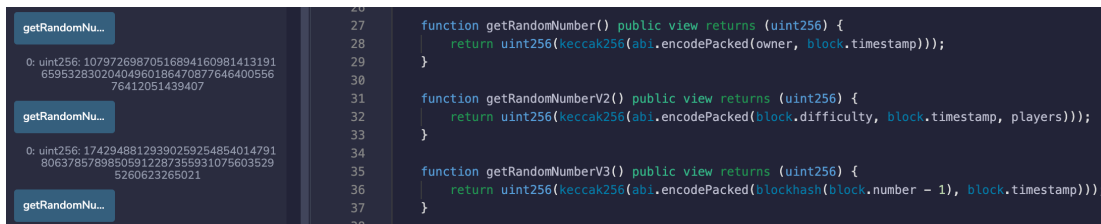
- pickWinner()는 owner만 호출할 수 있으므로 account를 owner account로 변경
- pickWinner() 클릭
- view 함수로 결과 확인
  - getBalance(): 0
  - getPlayers(): []
  - lotteryHistory[0]: 4번째 account
  - lotteryId: 1



- Account 부분에 0x787... 보면 ETH balance가 증가한 것 확인 가능 (+ 0.03 ETH)

# 컨트랙트 테스트하기 - getRandomNumber()

- getRandomNumber(): block.timestamp  
값에 의해 여러번 클릭해보면 1초씩  
지날때마다 랜덤값이 변경되는 것 확인 가능
- getRandomNumberV2():  
getRandomNumber()와 같이 매 초마다  
업데이트 되는 것 확인 가능
- getRandomNumberV3(): Remix VM  
상에선 blockhash()가 안되는 이슈가 있어서  
값이 안뜸



The screenshot displays a development environment with two panels. The left panel shows the results of three consecutive calls to the `getRandomNumber()` function, each returning a unique 256-bit hexadecimal value. The right panel shows the Solidity source code for the contract, including three functions: `getRandomNumber()`, `getRandomNumberV2()`, and `getRandomNumberV3()`. The `getRandomNumber()` function uses `block.timestamp` for randomness, while `getRandomNumberV3()` uses `blockhash(block.number - 1, block.timestamp)`.

```
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
function getRandomNumber() public view returns (uint256) {  
    return uint256(keccak256(abi.encodePacked(owner, block.timestamp)));  
}  
  
function getRandomNumberV2() public view returns (uint256) {  
    return uint256(keccak256(abi.encodePacked(block.difficulty, block.timestamp, players)));  
}  
  
function getRandomNumberV3() public view returns (uint256) {  
    return uint256(keccak256(abi.encodePacked(blockhash(block.number - 1), block.timestamp)));  
}
```

getRandomNu...  
0: uint256: 10797269870516894160981413191  
65953283020404960186470877646400566  
76412051439407  
  
getRandomNu...  
0: uint256: 17429488129390259254854014791  
80637857898505912287355931075603529  
5260623265021  
  
getRandomNu...

# 컨트랙트 테스트하기 - getRandomNumberV3()

- 로컬 블록체인 노드 사용해서 테스트하기
- Ganache CLI 실행: `$ ganache`
- Environment 부분에서 Ganache Provider 선택
- 다시 Deploy 하기 (블록체인 네트워크가 변경되었으므로)
- getRandomNumber() 함수들 클릭해보면 이번엔 V3에서도 랜덤값 반환하는 것 확인 가능
- Ganache는 트랜잭션이 생성될 때에만 블록이 마이닝되므로, 랜덤 함수들 값 변화 없을 것

