

# Solidity Fundamentals 1

# Lottery Demo

## Flow:

- ☐ The manager deploys the smart contract on blockchain network
- ☐ Each player must pay at least 0.01 ether to join
- ☐ Everyone can see the list of player
- ☐ The manager choose winner randomly from the list

1

Create a new contract

# Use lottery.sol

Paste your code here

2

3

Compile your code

Start to compile

☒ Auto compile

Lottery

Details

Publish on Swarm

Static Analysis raised 4 warning(s) that requires your attention. Click here to show the warning(s).

browser/lottery.sol:18:21: Warning: This function only accepts  
return uint(sha256(block.difficulty, block.number, p

```
1 pragma solidity 0.4.24;
2
3 contract Lottery {
4     address public manager;
5     address[] public players;
6
7     constructor() public {
8         manager = msg.sender;
9     }
10
11     function enterLottery() public payable {
12         require(msg.value > .01 ether, "You do not have enough Ethereum to join the lottery");
13
14         players.push(msg.sender);
15     }
16
17     function random() private view returns (uint){
18         return uint(sha256(block.difficulty, block.number, players));
19     }
20
21     function pickWinner() public restricted {
22         uint index = random() % players.length;
23         players[index].transfer(address(this).balance);
24         players = new address[](0);
25     }
26
27     function getPlayers() public view returns (address[]){
28         return players;
29     }
30 }
```

[2] only remix transactions, script

Search transactions

## 4 Switch to Run tab

The screenshot displays the Remix IDE interface. On the left, the 'browser/lottery.sol' file is open, showing Solidity code for a lottery contract. The code includes a constructor, an 'enterLottery' function, a 'random' function, a 'pickWinner' function, and a 'getPlayers' function. The 'Run' tab is selected in the top right. The 'Environment' dropdown is set to 'JavaScript VM'. The 'Account' dropdown shows '0xca3...a733c (100 ether)'. The 'Gas limit' is set to '3000000' and the 'Value' is '0 wei'. The 'Lottery' contract is selected in the dropdown below. The 'Deploy' button is highlighted. The 'Transactions recorded' section shows 0 transactions. The 'Deployed Contracts' section shows a message: 'Currently you have no contract instances to interact with.'

```
1 pragma solidity 0.4.24;
2
3 contract Lottery {
4     address public manager;
5     address[] public players;
6
7     constructor() public {
8         manager = msg.sender;
9     }
10
11     function enterLottery() public payable {
12         require(msg.value > .01 ether, "You do not have enough Ethereum to join the lottery");
13
14         players.push(msg.sender);
15     }
16
17     function random() private view returns (uint){
18         return uint(sha256(block.difficulty, block.number, players));
19     }
20
21     function pickWinner() public restricted {
22         uint index = random() % players.length;
23         players[index].transfer(address(this).balance);
24         players = new address[](0);
25     }
26
27     function getPlayers() public view returns (address[]){
28         return players;
29     }
30 }
```

Environment: JavaScript VM VM (-) i 5 choose

Account: 0xca3...a733c (100 ether) 6 choose

Gas limit: 3000000

Value: 0 wei

Lottery

Deploy 7 Deploy smart contract to blockchain

Load contract from Address At Address

Transactions recorded: 0

Deployed Contracts

Currently you have no contract instances to interact with.

[2] only remix transactions, script Search transactions

+

📁

⏪

+

browser/lottery.sol

»

» browse

» config

```
3 contract Lottery {
4   address public manager;
5   address[] public players;
6
7   constructor() public {
8     manager = msg.sender;
9   }
10
11  function enterLottery() public payable {
12    require(msg.value > .01 ether, "You do not have enough Ethereum to join the lotte
13
14    players.push(msg.sender);
15  }
16
17  function random() private view returns (uint){
18    return uint(sha256(block.difficulty, block.number, players));
19  }
20
21  function pickWinner() public retriected {
22    uint index = random() % players.length;
23    players[index].transfer(address(this).balance);
24    players = new address[](0);
25  }
26
27  function getPlayers() public view returns (address[]){
28    return players;
29  }
30
31  modifier retriected(){
32
```

🔍 [2] only remix transactions, script

🔍 Search transactions

to	Lottery.pickWinner() 0x5e72914535f202659083db3a02c984188fa26e9f
gas	3000000 gas
transaction cost	24218 gas

Compile

Run

Settings

Analysis

Debugger

Support

Account

0xca3...a733c (99.999999999998246669)

Gas limit

3000000

Value

0.1

ether

8

Money to send

Lottery

Deploy

Load contract from Address

At Address

Transactions recorded: 4

Deployed Contracts

Lottery at 0x5e7...26e9f (memory)

enterLottery

pickWinner

getPlayers

9

Join the lottery

0: address[]:

0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,0x4B0897b0513fdC7C541B6c

manager

browser/lottery.sol

Compile Run Settings Analysis Debugger Support

Gas limit 3000000  
Value 0 ether

```
7 constructor() public {  
8     manager = msg.sender;  
9 }  
10  
11 function enterLottery() public payable {  
12     require(msg.value > .01 ether, "You do not have enough Ethereum to join the lottery");  
13  
14     players.push(msg.sender);  
15 }  
16  
17 function random() private view returns (uint){  
18     return uint(sha256(block.difficulty, block.number, players));  
19 }  
20  
21 function pickWinner() public restricted {  
22     uint index = random() % players.length;  
23     return players[index];  
24 }
```

Lottery

Deploy

Load contract from Address At Address

Transactions recorded: 5

Deployed Contracts

Lottery at 0x5e7...26e9f (memory)

enterLottery

pickWinner

getPlayers

0: address[]:

0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,0x4B0897b0513fdC7C541B6c

manager

players uint256

transact to Lottery.pickWinner pending ...

Successful transaction

[vm] from:0xca3...a733c  
to:Lottery.pickWinner() 0x5e7...26e9f value:0 wei  
data:0x5d4...95aea logs:0 hash:0xc67...93385 Debug

transact to Lottery.enterLottery pending ...

Failed transaction

[vm] from:0xca3...a733c  
to:Lottery.enterLottery() 0x5e7...26e9f  
value:10000000000000000 wei data:0xc1a...f5785 logs:0  
hash:0xd31...3bb72 Debug

transact to Lottery.enterLottery error: VM error: revert

# Mapping

## Example:

- Store a list of address
- Each address has a unique ID
- Given the ID, anyone can see the address

Key Value

ID 1	Address 1
ID 2	Address 2
ID 3	Address 3

```
mapping(uint => address) public addressList
```

## Syntax:

mapping(keyType => valueType) visibility varName

- **keyType**: Data type of key (uint)
- **valueType**: Data type of value (address)
- **visibility**: public, private
- **varName**: name of the reference(pointer) to the mapping object

*See test2.sol*

# Mapping

## Example:

- Store a list of address **array**
- Each address array has a unique ID
- Given the ID, anyone can see the address **array**

Key Value

ID 1	Address Array 1
ID 2	Address Array 2
ID 3	Address Array 3

```
mapping(uint => address[]) public addressArrayList
```

\* The [] declares an array

## Access values stored in mapping

**varName[key]**

\*We can not get all the keys and values in a mapping object. The value can only be accessed by key.

***See test3.sol***



# Mapping

## Problem:

- Store a list of unsigned integer array.
- Each unsigned integer array is linked to an address
- Given the address, anyone can access the integer array

Key	Value
Address 1	Integer array 1
Address 2	Integer array 2
Address 3	Integer array 3

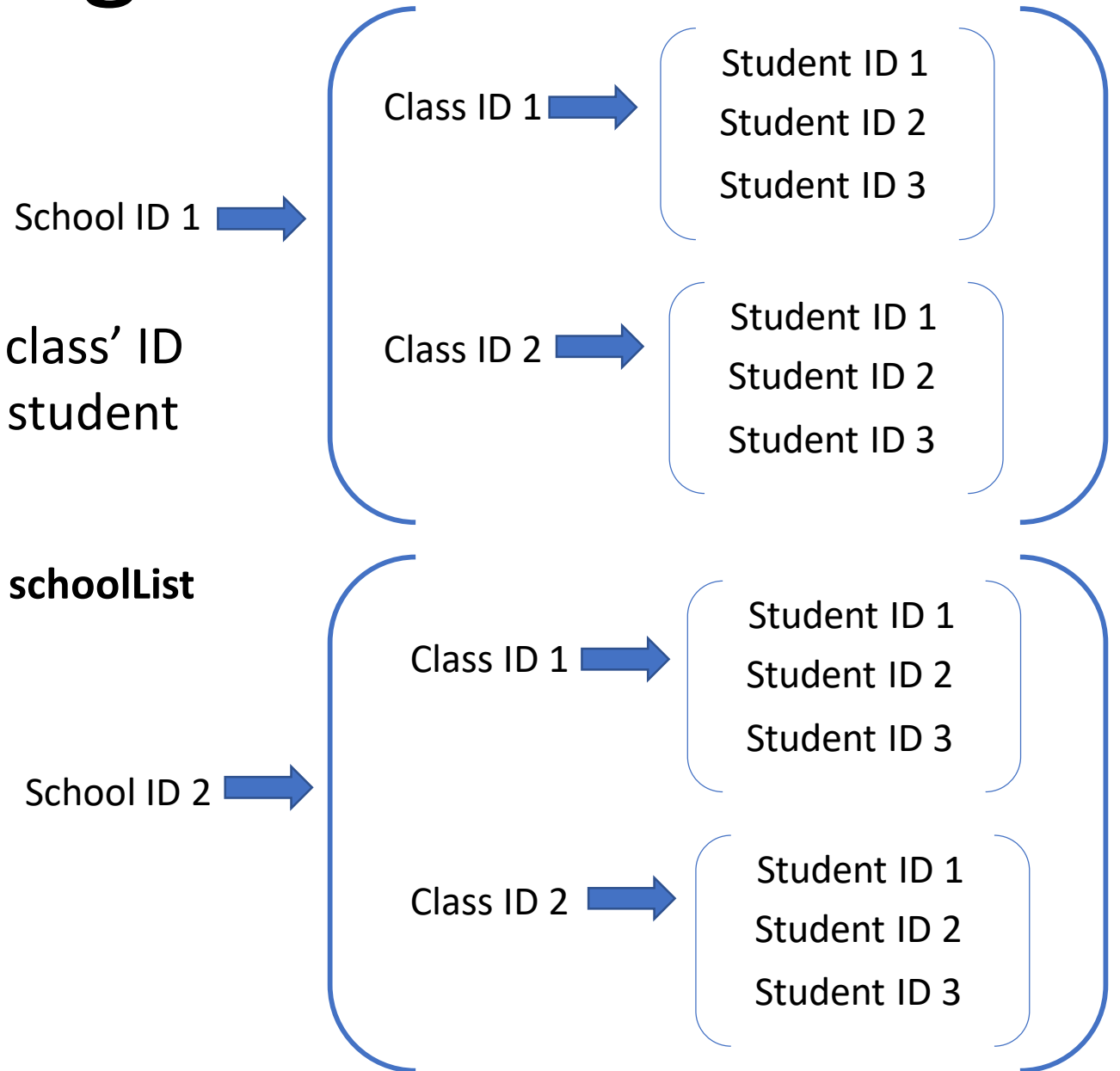
# Mapping of Mapping

## Example:

- ❑ Store a list of school ID
- ❑ Each school ID is used to access a list of class' ID
- ❑ Each class'ID can give access to a list of student ID

`mapping(uint => (mapping(uint => uint[])) public schoolList`

*See test4.sol*



# Mapping of Mapping

## **Problem**

- ☐ Store a list of school address
- ☐ Each school address is used to access a list of class' address
- ☐ Each class 's address can give access to an array of student ID

# Struct

*See test5.sol*

## Example:

- ❑ Each student has an ID and address of his father and address of his close friend)

```
struct Student {  
    uint ID  
    address father  
    address closeFriend  
}
```

## Syntax:

```
struct structName {  
    propertyType1 propertyName1  
    propertyType2 propertyName2  
    ...  
}
```

*See test6.sol*

# Mapping of Struct

## Example:

- ☐ Store a list of student addresses
- ☐ Each student address is mapped to a student object that contains his ID and address of his father

# Mapping of Struct

## Problem

- ☐ Store a list of school address
- ☐ Each school address is used to access a list of class' address
- ☐ Each class 's address can give access to an array of student objects
- ☐ Each student object contains his ID and address of his father