**LAB 4: Lottery Contract with Injected Web3**

**Overview:**

This purpose of this lab is to help design and create second smart contract called Lottery. It will also guide you on deploying and testing of Lottery Contract using Injected Web3 environment in Remix. This lab will also guide you on sharing your deployed Lottery contract with others.

**Prerequisites:**

You should have completed Lab 1 – Metamask Setup, created an account in Metamask wallet and have some ether in Rinkeby Test Network. You should also have completed Lab 3 – Solidity and Remix IDE, learned how to compile smart contract in Remix.

**Lottery Contract Design:**

This next contract we're going to work on is going to be a lottery or like a raffle. We're going to create a contract that we're going to call a lottery. It will have a prize pool and a list of people who have entered for the prize pool. So, let's imagine that we have two players player one in player two over here. These two players can send in some amount of money to our contract as soon as someone sends money into the contract. They will then be recorded as a player in the game.

**A screenshot of a cell phone

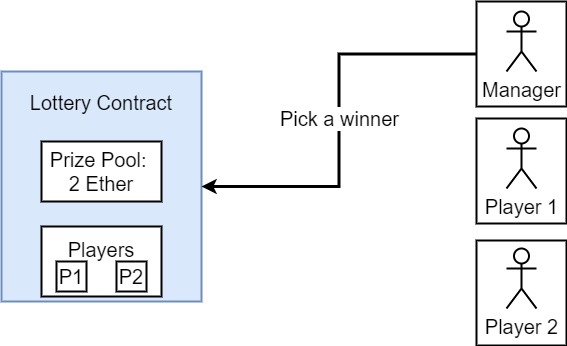
Description automatically generated**

So this player's box right here has P1 and P2 which represent player 1 and Player 2. The 1 ether that they send into the contract will be held in a prize pool.

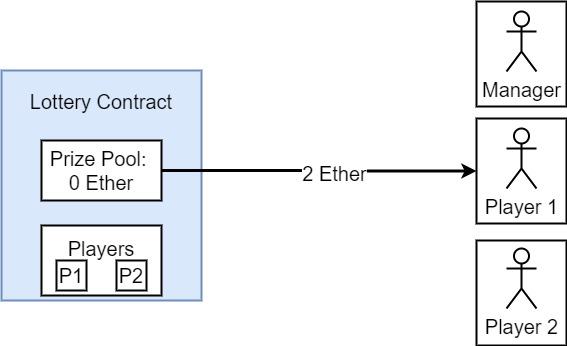
A close up of a clock

Description automatically generated

At some point in time after some number of people have entered the contract or entered the contest, a third person or a third party of sorts who we will refer to as a manager will tell the contract to pick a winner.



When this manager tells the contract to pick a winner and one thing I want to make clear about this is that it's not the manager who's picking the winner. The manager is telling the contract to pick a winner at that point. The contracts will look at its list of participants it will pick one of these two players right here and it will send all of the money out of the prize pool to that particular winner. At that point the lottery contract then resets and then becomes ready to accept a new list of players and repeat itself all over again. So essentially it's a self-repeating contract that can be used for playing a tiny little lottery or a raffle whatever you want to call it.



The following will be basic outline of the basic variables and functions we might need on our lottery contract.

A screenshot of a cell phone

Description automatically generated

**Lottery Contract Solidity Code:**

pragma solidity ^0.4.17;

contract Lottery

{

address public manager;

address[] public players;

function Lottery() public

{

manager = msg.sender;

}

function enter() public payable

{

require(msg.value > .01 ether);

players.push(msg.sender);

}

function random() private view returns (uint)

{

return uint(keccak256(block.difficulty, now, players));

}

function pickWinner() public restricted

{

uint index = random() % players.length;

players[index].transfer(this.balance);

players = new address[](0);

}

modifier restricted()

{

require(msg.sender == manager);

\_;

}

function getPlayers() public view returns (address[])

{

return players;

}

}

**Deploying and Testing a Smart Contract**

Once you’ve compiled Lottery contract in Remix, you can use the “run” tab to deploy it. The “environment” drop-down gives three options for where to deploy the contract:

* **JavaScript VM** - This lets you run your contract directly in the browser using a JavaScript implementation of the Ethereum virtual machine (EVM). This is great for simple testing but doesn’t allow anyone else to interact with your contract. The Ethereum network is made up of “nodes” which store and update the blockchain via a consensus protocol. All interactions with the blockchain involve communicating with one of these nodes.
* **Injected Web3** - Web3 is the interface for interacting with an Ethereum node. If you’re using the MetaMask browser extension, it injects an implementation of Web3 into every web page. This option will let you use that injected implementation to deploy to a test network or the main Ethereum network.
* **Web3 Provider** - This option connects directly to an Ethereum node via HTTP. If you’re running your own node (or something like ganache), you can use this option to connect to it.

**Sharing Your Deployed Contract with Others**

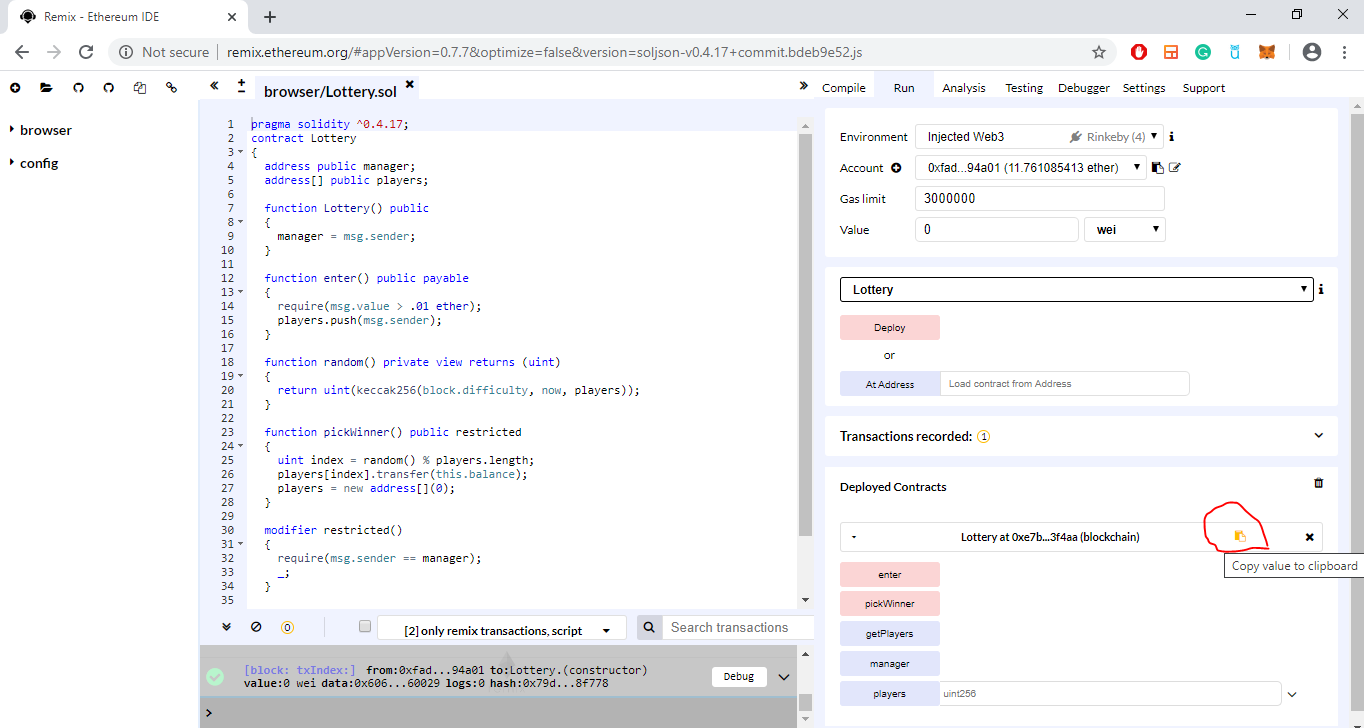
For someone to interact with your contract, they need to know two things:

* The address of the deployed contract.
* The contract’s Application Binary Interface (ABI).

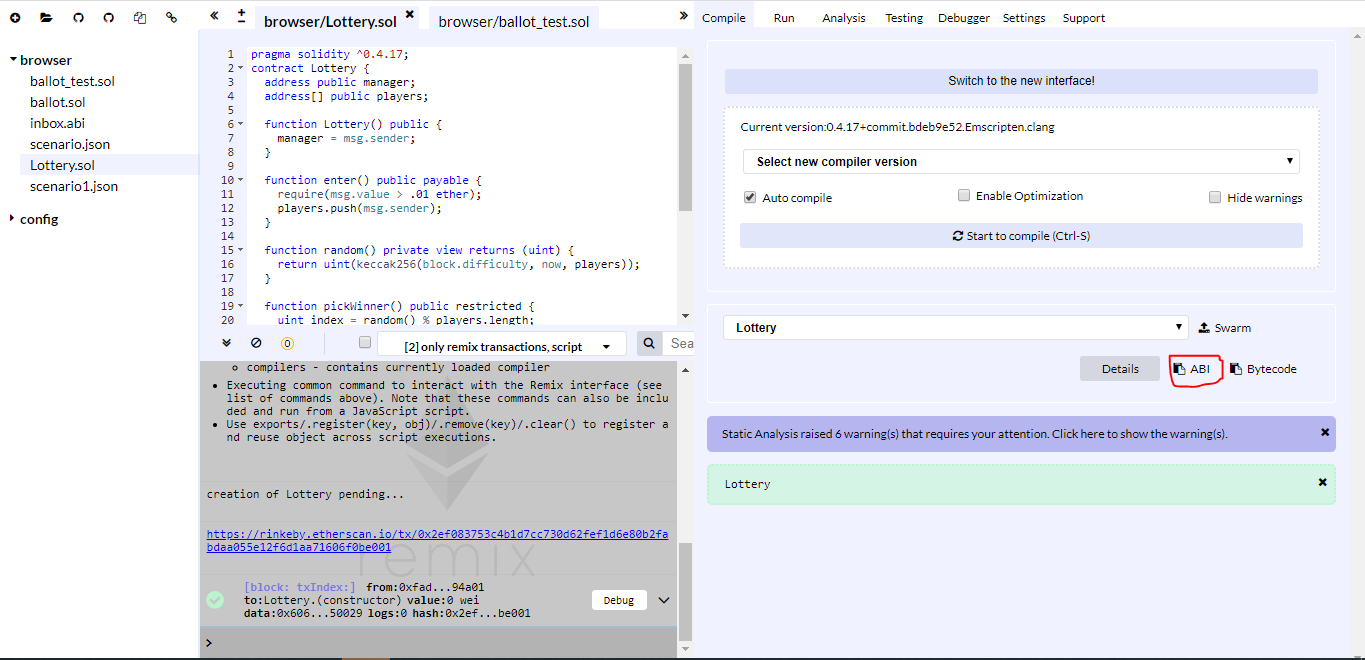
You’ll get an address when you deploy your contract to a public network–either a test network or the main Ethereum network. When using Remix, a good way to do that is to deploy using MetaMask by choosing the “Injected Web3” environment.

To deploy to the Rinkeby test network, you’ll need ether in an account there to pay for gas. Ether on a test network has no real monetary value because it’s extremely easy to mine. You can fund your Rinkeby account for free via the Rinkeby faucet.

The line that appears when your contract has been deployed has a clipboard icon on the right. (You may need to expand the right-hand panel to make that icon visible.) Clicking that icon will copy the contract’s address to your clipboard.



The ABI is what tells callers what functions are available, what arguments they take, and what they return. The ABI is generated for you by the Solidity compiler, and Remix exposes it from the “Compile” tab. Click “ABI” next to your contract Details section to copy the interface to your clipboard.



After you copy the ABI code, and paste it in Notepad, you will be able to see code something like below:



After you share the contract address and the ABI Code with the person, then they need to perform the following things so that they can interact with your contract:

1. Person have to go to Remix and create a new .abi file. And paste the ABI code you just share with them.

Click the “+” button on the top left screen.

A screenshot of a computer

Description automatically generated

A screenshot of a social media post

Description automatically generated

1. After that, they need to go to “Run” Tab and on At Address then need to paste the contract address you just shared. Click “At Address” button after you paste the contract address.

A screenshot of a computer

Description automatically generated

You will see warning that if you want to interact with the contract. Click “OK”

A screenshot of a social media post

Description automatically generated

After that, you will see the contract on the right-hand side of the screen. Now you can interact with the lottery contract.

A screenshot of a computer

Description automatically generated

1. Person can enter and play the lottery, but they have to mention how much value they want to enter on lottery. They can enter the value on the Value Tab and also choose if they want to use wei, gwei or ether as a value.

A screenshot of a cell phone

Description automatically generated