Lecture 21 - Solidity - Contract for Hw 7, 8, 9

News

1. Yield Curve has Inverted https://seekingalpha.com/article/4251080-inverted-yield-curve-important
2. Demand for Insurance and Blockchain https://www.ccn.com/bitcoin-expertise-exploding-among-insurance-professionals-in-2019-study "A few vendors, like Black Insurance, an Estonian firm, are offering more radical blockchain solutions, with a goal to upend the entire insurance system. In this model, a broker, not an insurance carrier, creates a new insurance product, then lists that product through Black Insurance to gauge interest in capital markets. Risk is transferred directly to the capital markets, disintermediating (eliminating) traditional insurance carriers entirely."
3. Whiskey tracked. https://cointelegraph.com/news/over-130-year-old-liquor-company-william-grant-sons-to-track-whiskey-on-blockchain

Solidity

Version of the compiler

```
pragma solidity >=0.4.21 <0.6.0;
```

Versions of Tools

```
$ truffle --version
Truffle v5.0.9 - a development framework for Ethereum
    ...
$ node --version
v10.15.3
$ npm --version
6.4.1
$ solc --version
solc, the solidity compiler commandline interface
Version: 0.5.7+commit.6da8b019.Darwin.appleclang
```

Config File

```
{
    ...
    "Eth_URL_ws"                : "ws://192.168.0.199:8546",
    "Eth_URL_rpc"                 : "http://192.168.0.199:8545",
    "Eth_Account_Address"        : "0x023e291a99d21c944a871adcc44561a58f99bdbc",
    "Eth_Account_KeyFile"        : "./keystore/UTC--2018-07-17T00-00-00.000000000Z--023
    "Eth_Contract_SignedData"    : "0x0134291399821c944a871adcc44561a58f99bdbc"
    ...
}
```

Example for Homework

```
pragma solidity >=0.4.21 <0.6.0;

import "openzeppelin-solidity/contracts/ownership/Ownable.sol";

contract SignedDataVersion01 is Ownable {

    address payable owner_address;
    uint256 private minPayment;

    mapping(uint256 => mapping(uint256 => bytes32)) dData;
    mapping(uint256 => mapping(uint256 => address)) dOowner;
    event DataChange(uint256 App, uint256 Name, bytes32 Value, address By);

    event ReceivedFunds(address sender, uint256 value,
        uint256 application, uint256 payFor);
    event Withdrawn(address to, uint256 amount);

    constructor() public {
        owner_address = msg.sender;
        minPayment = 1000;
    }

    modifier needMinPayment {
        require(msg.value >= minPayment,
            "Insufficient payment.  Must send more than minPayment.");
        _;
    }

    function init() public {
        minPayment = 1000;
    }

    function setMinPayment( uint256 _minPayment ) public onlyOwner {
        minPayment = _minPayment;
    }

    function getMinPayment() public onlyOwner view returns ( uint256 ) {
        return ( minPayment );
    }

    /**
     * @dev TODO if the data is empty, or if the msg.sender is the original
            createor of the data:
     *      then : save the msg.sender into dOwner, save the data into dData
     *             create a DataChange event.
     *      else : revert an error.
     */
    function setData ( uint256 _app, uint256 _name, bytes32 _data )
    public needMinPayment payable {
        // TODO-start - code for students to implement -- about 7 or 8 lines of cod
        // Create a temorary variable with dOwner[_app][_name]
        // If the msg.sender is the owner and or if tmp is address(0) "not assigned
        // Save the msg.sender to dOnwer
        // Save the data.
        // Emit DataChange event
        // Else
        // Revert an error
        // TODO-end
    }

    /**
     * @dev TODO return the data by looking up _app and _name in dData.
     */
    function getData ( uint256 _app, uint256 _name )
    public view returns ( bytes32 ) {
        // TODO-start - code for students to implement -- really just one l.o.c. -
        // Lookup the data and return it.
        // TODO-end
    }

    /**
     * @dev payable fallback
     */
    function () external payable {
        emit ReceivedFunds(msg.sender, msg.value, 0, 1);
    }

    /**
     * @dev genReceiveFunds - generate a receive funds event.
     */
    function genReceivedFunds ( uint256 application, uint256 payFor )
    public payable {
        emit ReceivedFunds(msg.sender, msg.value, application, payFor);
    }

    /**
     * @dev Withdraw contract value amount.
     */
    function withdraw( uint256 amount ) public onlyOwner returns(bool) {
        address(owner_address).transfer(amount);
        // owner_address.send(amount);
        emit Withdrawn(owner_address, amount);
        return true;
    }

    /**
     * @dev How much do I got?
     */
    function getBalanceContract() public view onlyOwner returns(uint256){
        return address(this).balance;
    }

    /**
     * @dev For futute to end the contract, take the value.
     */
    function kill() public onlyOwner {
        emit Withdrawn(owner_address, address(this).balance);
        selfdestruct(owner_address);
    }
}
```