SOLIDITY PROGRAMMING

Karachi Institute of Technology and Entrepreneurship (KITE)

Session 3: ERC20 Basics

Date: 12th November 2021

Instructor: Syed Faisal ur Rahman

Readings

https://www.tutorialspoint.com/solidity/index.htm

https://www.dappuniversity.com/articles/solidity-tutorial

https://101blockchains.com/solidity-tutorial/

https://www.geeksforgeeks.org/introduction-to-solidity/?ref=leftbar-rightbar

https://betterprogramming.pub/learn-solidity-functions-ddd8ea24c00d

https://www.bitdegree.org/learn/solidity-types

https://www.tutorialspoint.com/solidity/solidity_mappings.htm

https://medium.com/coinmonks/what-the-hack-is-memory-and-storage-in-solidity-6b9e62577305

https://www.ops.gov.ie/app/uploads/2021/01/Blockchain-Develop-Deploy-and-Test-Your-First-Smart-Contract.pdf

https://ethereumbuilders.gitbooks.io/guide/content/en/solidity_tutorials.html

DEVELOPMENT ENVIRONMENT

https://remix.ethereum.org/

READING: ERC20 BASIC CONTRACT EXAMPLE

https://www.toptal.com/ethereum/create-erc20-token-tutorial

https://ethereumdev.io/using-safe-math-library-to-prevent-from-overflows/

READING: EVENT

https://www.geeksforgeeks.org/what-are-events-in-solidity/

https://www.tutorialspoint.com/solidity/solidity_events.htm

TASK 1: ER20 BASIC CODE

Step 1: Login to your MetaMask account using the google extension.
Step 2: Open remix.
Step 3: Type or Copy/Paste the code below.
Step 4: Compile the code.
Step 5: In the “Deploy & run” set and amount (e.g 100 tokens)in the field given next to the deploy button.
Step 6: In the “Deploy & run”, choose Inject Web3 options and then press “Deploy” button to see how the contract works.
Step 7: Below the “Deployed Contracts”, select your contract.
Step 8: Transfer tokens from one account to the other and check balance.

Code

```
//ref: https://www.toptal.com/ethereum/create-erc20-token-tutorial
//ref:
https://gist.github.com/giladHaimov/8e81dbde10c9aeff69a1d683ed6870be#file-
basicerc20-sol

pragma solidity ^0.5.0;

contract ERC20Test {

    string public constant name = "ERC20Basic";
    string public constant symbol = "ATC20";
    uint8 public constant decimals = 18;


    event Approval(address indexed tokenOwner, address indexed spender, uint
tokens);
    event Transfer(address indexed from, address indexed to, uint tokens);


    mapping(address => uint256) balances;

    mapping(address => mapping (address => uint256)) allowed;

    uint256 totalSupply_;

    using SafeMath for uint256;


   constructor(uint256 total) public {
      totalSupply_ = total;
      balances[msg.sender] = totalSupply_;
    }

    function totalSupply() public view returns (uint256) {
      return totalSupply_;
    }

    function balanceOf(address tokenOwner) public view returns (uint) {
        return balances[tokenOwner];
    }

    function transfer(address receiver, uint numTokens) public returns
(bool) {
        require(numTokens <= balances[msg.sender]);
        balances[msg.sender] = balances[msg.sender].sub(numTokens);
        balances[receiver] = balances[receiver].add(numTokens);
        emit Transfer(msg.sender, receiver, numTokens);
        return true;
    }

    function approve(address delegate, uint numTokens) public returns (bool)
{
        allowed[msg.sender][delegate] = numTokens;
        emit Approval(msg.sender, delegate, numTokens);
        return true;
    }

    function allowance(address owner, address delegate) public view returns
(uint) {
        return allowed[owner][delegate];
    }

    function transferFrom(address owner, address buyer, uint numTokens)
public returns (bool) {
        require(numTokens <= balances[owner]);
        //This is to avoid gas estimation error which comes when we run it
from a test environment where owner==msg.sender
        if (owner!=msg.sender)
        {
                    require(numTokens <= allowed[owner][msg.sender]);
                    balances[owner] = balances[owner].sub(numTokens);
                    allowed[owner][msg.sender] =
allowed[owner][msg.sender].sub(numTokens);
             balances[buyer] = balances[buyer].add(numTokens);

        }

    else {
        balances[owner] = balances[owner].sub(numTokens);
        balances[buyer] = balances[buyer].add(numTokens);
    }
        emit Transfer(owner, buyer, numTokens);
        return true;
    }
}

library SafeMath {
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
      assert(b <= a);
      return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256) {
      uint256 c = a + b;
      assert(c >= a);
      return c;
    }
}
```

TASK 2: EXPLORE THE RINKEBY TESTNET FOR CONTRACTS AND TRANSACTIONS

Got to: https://rinkeby.etherscan.io/address/0x4ed7853321de989d025b222416ddfdaaec856b6d

This is a test contract and check the activity which I deployed using the code given above.

Go to: https://rinkeby.etherscan.io/address/0x76c62eb54bec8132cff0bcbbb3aa1022f33fb9ac

Check the activity. This is my public address.

Go to: https://rinkeby.etherscan.io/tx/0xf390f13382a465cd48b9fb5d582443b12689551950a955638c2587a943533249

This is a test transaction using the ERC20 token (ASC20).

Go to: https://rinkeby.etherscan.io/token/0x4ed7853321de989d025b222416ddfdaaec856b6d

This is the test token we created.

TASK 3: ERC20 USING INTERFACE

//ref: https://www.toptal.com/ethereum/create-erc20-token-tutorial

//ref: https://gist.github.com/giladHaimov/8e81dbde10c9aeff69a1d683ed6870be#file-basicerc20-sol

```
// https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/token/ERC20/IERC20.sol

pragma solidity ^0.5.0;

interface IERC20 {

    function totalSupply() external view returns (uint);


    function balanceOf(address account) external view returns (uint);


    function transfer(address recipient, uint amount) external returns (bool);


    function allowance(address owner, address spender) external view returns (uint);


    function approve(address spender, uint amount) external returns (bool);
    function transferFrom(
        address sender,
        address recipient,
        uint amount
    ) external returns (bool);


    event Transfer(address indexed from, address indexed to, uint value);
    event Approval(address indexed owner, address indexed spender, uint value);
}


contract MyERC20 is IERC20  {


    string public constant name = "ERC20Basic";
    string public constant symbol = "ATC20";
    uint8 public constant decimals = 18;



    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
    event Transfer(address indexed from, address indexed to, uint tokens);



    mapping(address => uint256) balances;

    mapping(address => mapping (address => uint256)) allowed;

    uint256 totalSupply_;
```

```
using SafeMath for uint256;


constructor(uint256 total) public {
         totalSupply_ = total;
         balances[msg.sender] = totalSupply_;
}

function totalSupply() public view returns (uint256) {
         return totalSupply_;
}

function balanceOf(address tokenOwner) public view returns (uint) {
   return balances[tokenOwner];
}

function transfer(address receiver, uint numTokens) public returns (bool) {
   require(numTokens <= balances[msg.sender]);
   balances[msg.sender] = balances[msg.sender].sub(numTokens);
   balances[receiver] = balances[receiver].add(numTokens);
   emit Transfer(msg.sender, receiver, numTokens);
   return true;
}

function approve(address delegate, uint numTokens) public returns (bool) {
   allowed[msg.sender][delegate] = numTokens;
   emit Approval(msg.sender, delegate, numTokens);
```

```
        return true;
    }

    function allowance(address owner, address delegate) public view returns (uint) {
        return allowed[owner][delegate];
    }

    function transferFrom(address owner, address buyer, uint numTokens) public returns (bool) {
        require(numTokens <= balances[owner]);
        //This is to avoid gas estimation error which comes when we run it from a test environment where owner==msg.sender
        if (owner!=msg.sender)
        {
                require(numTokens <= allowed[owner][msg.sender]);
                balances[owner] = balances[owner].sub(numTokens);
                allowed[owner][msg.sender] = allowed[owner][msg.sender].sub(numTokens);
                balances[buyer] = balances[buyer].add(numTokens);

        }

        else
        {
                balances[owner] = balances[owner].sub(numTokens);
                balances[buyer] = balances[buyer].add(numTokens);
        }
        emit Transfer(owner, buyer, numTokens);
        return true;
    }
}


library SafeMath {
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
      assert(b <= a);
      return a - b;
    }


    function add(uint256 a, uint256 b) internal pure returns (uint256) {
      uint256 c = a + b;
      assert(c >= a);
      return c;
    }
}
```

READING: ERC20 CONTINUE

https://solidity-by-example.org/app/erc20/

https://ethereum.org/en/developers/tutorials/transfers-and-approval-of-erc-20-tokens-from-a-solidity-smart-contract/

https://dev.to/stermi/how-to-create-an-erc20-token-and-a-solidity-vendor-contract-to-sell-buy-your-own-token-4j1m

https://docs.appery.io/docs/eth-app-example-part1

https://www.quicknode.com/guides/solidity/how-to-create-and-deploy-an-erc20-token