Activity 5: Creating an IOC.

Solidity basics

Members of address:

- **balance:** account balance.
- **code:** code for smart contract account.
- **send:** send given amount of Wei to Address, **returns false on failure**, forwards 2300 gas.
- **transfer:** send given amount of Wei to Address, **reverts on failure** forwards 2300 gas.
- **call(bytes memory):** low-level CALL with the given payload, returns success condition and return data, forwards all available gas.
- **delegatecall(bytes memory) and staticcall**

Special functions

There are two special functions associated with eth transfers *receive*() and *fallback*(). Both are marked public payable. Each contract has its own balance and may receive eth in the following ways:

- **constructor payable:** If the constructor of a contract is payable, when the contract is deployed the *msg.value* of the transaction calling the constructor will be deposited in the contract balance.
- **function payable:** If a function is payable, when the function is called with *msg.value*, *msg.value* will be added to the contract balance.
- **receive function**: each contract may have one receive function. This function is executed after a call to the contract with empty calldata. If the contract has no payable function defined it must have a receive function in order to receive eth.
- **fallback function**: each contract may have one fallback function. This function is executed after a call to the contract if none of the other function match the call or if calldata is empty and there is not receive function. If the contract has no payable function defined or no receive function it must have a fallback function in order to receive eth. It is recommended to define a receive function even if a fallback function is defined to distinguish between transfers and other types of calls.

Inheritance

Solidity supports multiple inheritance, polymorphism and overriding. Keywords **this** and **super** have the standard semantics: current contract, the parent of the current contract in the inheritance hierarchy. Keywords virtual and override are also used with the standard meaning: functions not yet implemented and functions the override the implementation from the base class.

“When a contract inherits from other contracts, only a single contract is created on the blockchain, and the code from all the base contracts is compiled into the created contract.” [1]

IOC.

An **Initial Coin Offer** is a fundraising model that involves exchanging digital currencies for new issued coins or tokens, most commonly for ERC20 tokens or other types of tokens. Investors are informed about the project's purposes in a white paper. If they decide to participate in the fundraising, they are rewarded with tokens or coins that they may further use as planned by the project initiators.

EXERCISE

1. Work on files IOC.sol, IOCNotPayable, Ownable, Pausable, MyERC20, ContractCalls:

- Deploy the contract IOC. Add a *msg.value* when calling the constructor. Check contract's balance.
- Try to pause the contract from an account different form the owner of the account.
- From another account buy some tokens by calling the function *buy*. Check contract's balance.
- Deploy contract ContractCalls ContractCall with constructor argument the address of IOC contract.
- Test ContractCalls functions:
    - Call *callTransfer*, *callSend, callFallback, callReceive* and check logs.
- Deploy the contract IOCNotPayable. Deploy contract ContractCall with constructor argument the address of IOCNotPayable contract.
- Test ContractCalls functions:
    - Call *callTransfer*, *callSend, callFallback, callReceive*.

2. Create an ICO with the following specifications:

- Investors will receive a number of tokens in exchange for ethers (wei) at a specified price. The type of tokens is the type of a newly created ERC20 token)
- Each unit of ERC20 token has a price. An investor can **deposit** the amount of ether (wei) equivalent to the desired number of tokens.
- The fundraising has the following parameters: ***startDate***, ***endDate***, ***owner***, ***unitPrice*** and ***minTokens***. The owner is the person that manages the event. minTokens represent the minimum number of tokens required by investors for the new ERC20 token to be emitted.
- If, between startDate and endDate, investors deposit an amount of wei enough to cover minToken, the the owner deploys the ERC20 token and distributes tokens to the investors.
- If, between startDate and endDate, the funds deposited by investors are insufficient to reach minTokens, the owner refunds the sums to the respective investors.

3. DoS scenario AuctionBad.sol AutionAttacker.sol

Bibliography

[1] https://docs.soliditylang.org/_/downloads/en/latest/pdf/

[2] https://consensys.github.io/smart-contract-best-practices