Creating Smart Contracts with Solidity

Creating an ERC20 Token

Solidity Inheritance, libraries and more

Inheritance in Solidity

- Just like many other OOP languages Solidity supports multiple inheritance, polymorphism,virtual functions
  - Multiple inheritance means a smart contract can have more than one parent smart contract in it’s inheritance chain
  - Polymorphism means that when there is a function call, the function being called is searched for and matched by both name and parameter types
  - Virtual functions allow for overriding(re-defining) the implementation of a function in an inherited smart contract

Types of contracts in Inheritance

- Regular contract
- Abstract contract - can have functions that are not defined
- Interface- similar to Abstract contracts but they all their functions must not be defined. They can inherit only from other interfaces. No state variables, constructor or modifier allowed.
- Library – this is like a regular contract but it is deployed at a public address, instead of being deployed to the smart contract address itself. In a sense they are similar to a shared library in a OS (.dll or .so)

Abstract Contract

- An abstract contract is declared using the **abstract** keyword
- Cannot be instantiated directly, they can only be derived from
- Abstract contracts decouple the definition of a contract from its implementation providing better extensibility

Interfaces

- Interface is created using the **interface** keyword
- Very similar to an abstract class , but all their functions must be undefined
- All their functions are implicitly **virtual** and overriding them doesn’t require the **override** keyword

Inheritance Example

```
abstract contract Machine {
        function start() virtual public;
}

interface PayablePerItem {
        function payForItem() external;
}

contract VendingMachine is Machine, PayablePerItem {
        function start() {  ..... }
        function payForItem()  { ..... }
}

contract CoffeMachine is Machine {
        function start() {  ..... }
}
```

Function Modifiers

- Function modifiers provide a way to alter the behaviour of a function in a declarative way. This is a little bit like aspect oriented programming or the decorators in languages like Python and Javascript
- Modifiers are also inheritable and can be overridden like functions when marked virtual

Function Modifiers Example

```
contract Owned {
        constructor() { owner = msg.sender; }
        address owner;
        modifier onlyOwner {
                require(msg.sender == owner, "Only owner can call this function.");
               _; // this placeholder designated where the original function will be called
        }
}

contract OwnedMachine is Machine {
        function start() public onlyOwner {  ..... }
}
```

Libraries

- Key difference from regular SC inheritance is that they can access only the state of the calling contract if it’s passed explicitly
- Library Functions can be called directly only if the function is **view** or **pure** , else a delegatecall is used by the compiler. The main implication for this are the security risks, because delegatecall gives access to storage of the caller to the callee.

Units

- For convenience the following units are predefined in Solidity
  - wei, gwei, ether - 1 wei == 1 , 1 gwei == 1e9, 1 ether == 1e18
  - seconds, minutes, hours, days, weeks – 60 seconds == 1 minute, etc.

Special variables and functions

- block - different properties of the current block
- msg – information about the current method invocation
- tx - properties of the current transaction
- blockhash(uint blockNumber) – get the hash of a recent block in the blockchain
- gasleft() - remaining gas in the trasnaction
- See more at https://docs.soliditylang.org/en/latest/cheatsheet.html

ERC 20

What is ERC

- Short for Ethereum Request for Comments
- A way to propose and discuss changes and standards for the application level of the network
- They live as part of the broader EIP (Ethereum Enhancement Proposals) at https://eips.ethereum.org/
- Formalised process of submission and review

Tokens

- A token on the Ethereum network can represent virtually anything that can be represented by physical tokens or coins
  - Fiat currency
  - Shares in a company
  - Reputation points
  - Etc.
- Fungible and non fungible
  - Fungible tokens are such that every token is the same as any other, all tokens are exactly the same and it’s their amount that is significant
  - Non fungible tokens (NFT) are tokens where each token is unique and has unique properties.

ERC20

- ERC20 is a standard for fungible tokens
- Provides a standard for both the developers of smart contract and wallets to interact with the tokens
    - Transfer tokens between accounts
    - Checking the current balance
    - Check the total supply of tokens in the network
    - Approval of access to token in a certain account to a third party account

ERC20 interface

- function name() public view returns (string)
- function symbol() public view returns (string)
- function decimals() public view returns (uint8)
- function totalSupply() public view returns (uint256)
- function balanceOf(address _owner) public view returns (uint256 balance)
- function transfer(address _to, uint256 _value) public returns (bool success)
- function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
- function approve(address _spender, uint256 _value) public returns (bool success)
- function allowance(address _owner, address _spender) public
- event Transfer(address indexed _from, address indexed _to, uint256 _value)
- event Approval(address indexed _owner, address indexed _spender, uint256 _value)

ERC20 Implementations

- Any smart contract implementing the ERC20 interface is considered an ERC20 token
- Since the functionality is mostly the same in 99% of the cases, open source libraries can be used for easier creation and better security
- https://www.openzeppelin.com/ provides OSS implementations for many different Tokens including ERC20

Using hardhat for development

Hardhat

- A set of Javascript libraries and tools to help with Smart Contracts development
- Allows for easily running and debugging Smart Contacts
- Support for debugging
- Support for testing
- Many plugins that extends it’s capabilities

Code time