Introduction to blockchain and Ethereum smart contracts

Making your own digital token via a token issuing smart contract

Written by Peter, Ho Man Fai

Learning outcomes

After you complete this lab, you will be able to:

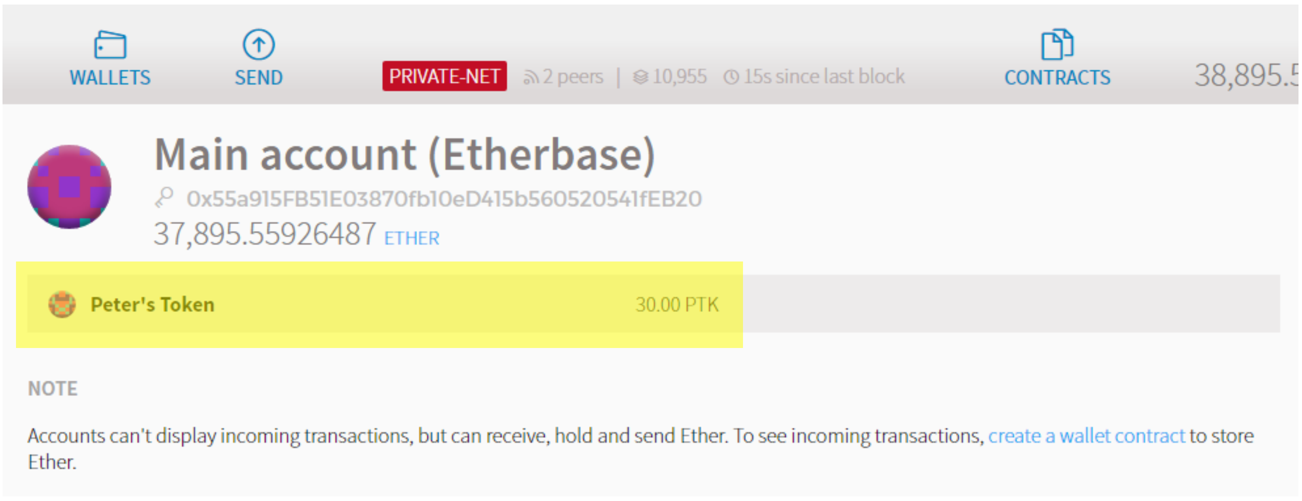
1. Implement a smart contract that can distribute custom tokens in your Ethereum network.
2. Manage and transfer custom tokens in the Ethereum Wallet.
3. Learn new Solidity programming techniques and practice your programming skills.

Please open the DigitalTokenAuthority.sol file and read through the source code. There are also some programming comments written inside the file for you to learn how the program works.

To test and run the smart contract, copy the source code and deploy it to your private blockchain network via your Ethereum Wallet.

Smart contract as a custom token

Besides using the cryptocurrency Ether in your Ethereum private network, you can also build your own custom currency, or token, using smart contracts. **Tokens in the Ethereum ecosystem can represent any fungible tradable good: coins, loyalty points, gold certificates, in game items, etc.**



Main account (Etherbase)

0x55a915FB51E03870fb10eD415b560520541fEB20

37,895.55926487 ETHER

Peter's Token 30.00 PTK

NOTE

Accounts can't display incoming transactions, but can receive, hold and send Ether. To see incoming transactions, create a wallet contract to store Ether.

All tokens implement some basic features in a standard way. To let your custom token being recognized by and compatible with the Ethereum Wallet, you have to follow some standards in your smart contract. **To create a standardized custom token, your smart contract must have four variables and one function:**

```
string name;
string symbol;
uint8 decimals;
mapping(address => uint256) balanceOf;

function transfer(address _to, uint256 _amount)
```

The usages of these variables are explained as follows:

- name: The name of your custom token.
- symbol: The abbreviation of your custom token. For example: USD, HKD, EUR…… You can also input "%" as the symbol, meaning your custom token is measured in percentage.
- decimals: The amount of decimal places your custom token supports. For example, if you want people to transfer your token in 12.15687, then the decimals value will be 5.
- mapping(address => uint256): A storage that stores the amount of custom token owned by different users.

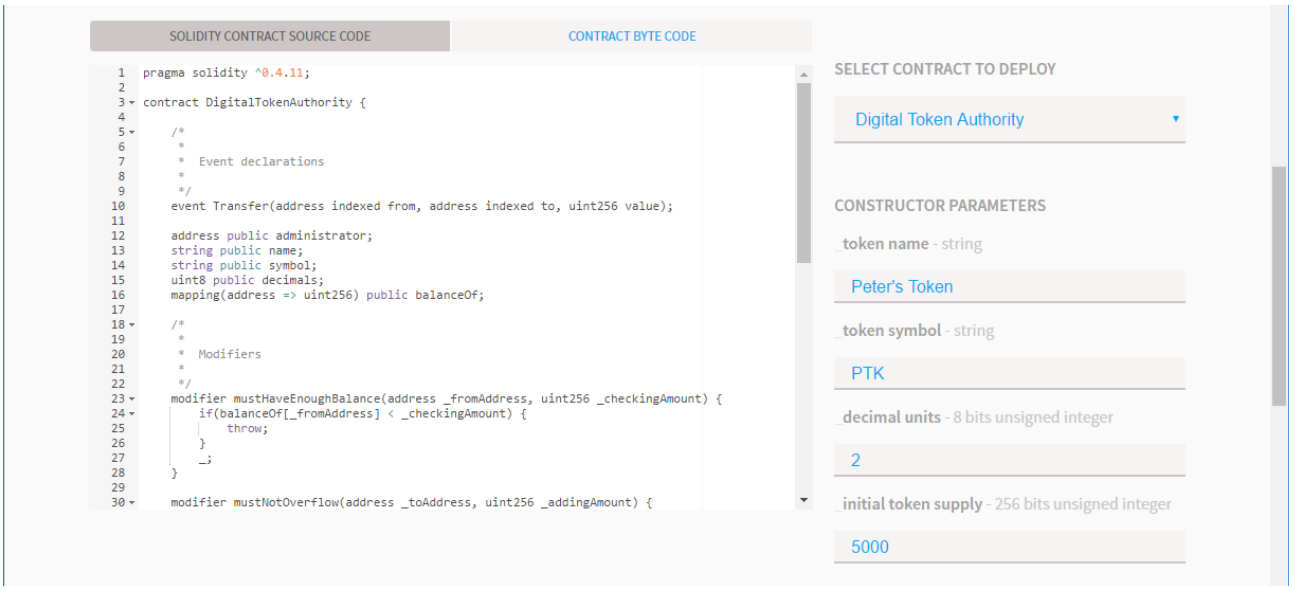
The function "transfer" must have one address and one uint256 input parameters. The function implements how token is being transfer from one account to another.

Important note:
You should ALWAYS follow the names given here. If you do not follow, the Ethereum Wallet will not be able to detect the custom token created in your contract.

Deploy a custom token

Deploying a custom token in Ethereum Wallet is like how you deploy a normal contract. However, there are few more steps involved.

1. When you input your contract source code in the "Deploy contract" page, you have to fill in the constructor's default value. The following shows an example.

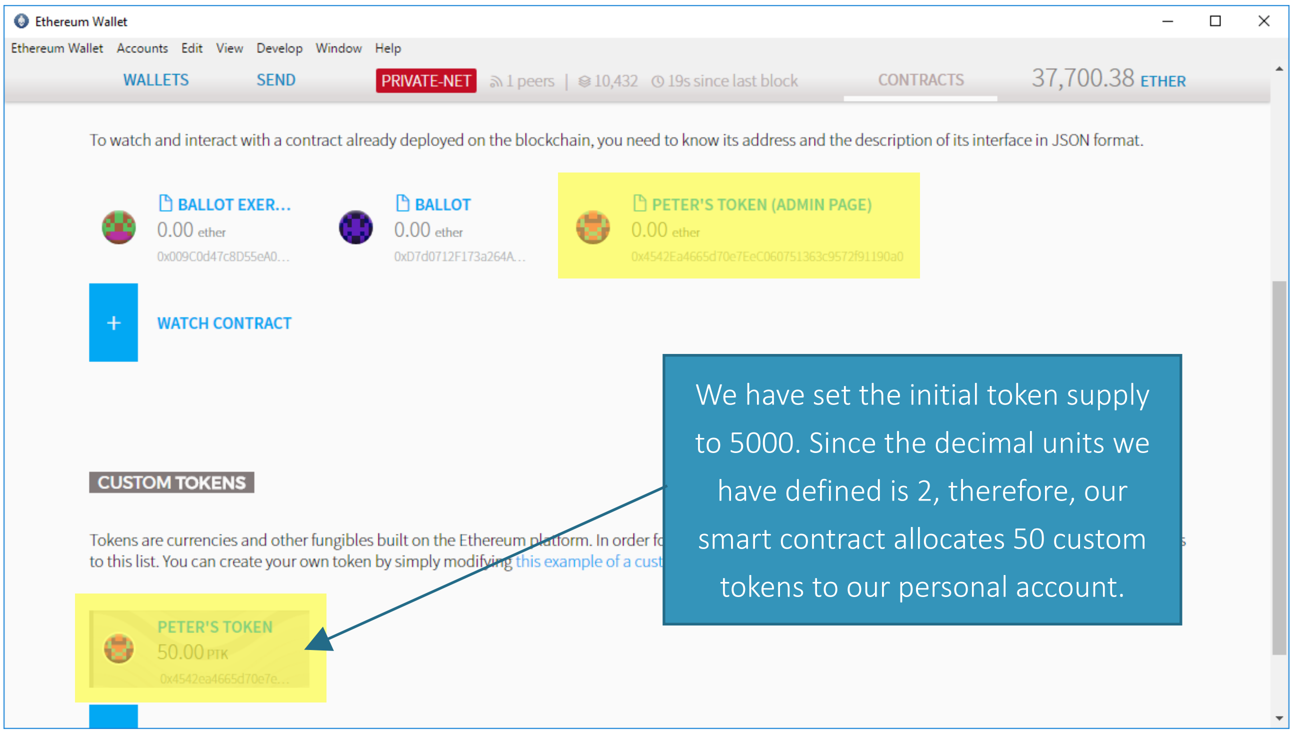


```
pragma solidity ^0.4.11;

contract DigitalTokenAuthority {

    /*
     *
     *  Event declarations
     *
     */
    event Transfer(address indexed from, address indexed to, uint256 value);

    address public administrator;
    string public name;
    string public symbol;
    uint8 public decimals;
    mapping(address => uint256) public balanceOf;

    /*
     *
     *  Modifiers
     *
     */
    modifier mustHaveEnoughBalance(address _fromAddress, uint256 _checkingAmount) {
        if(balanceOf[_fromAddress] < _checkingAmount) {
            throw;
        }
        _;
    }

    modifier mustNotOverflow(address _toAddress, uint256 _addingAmount) {
```

SELECT CONTRACT TO DEPLOY
Digital Token Authority

CONSTRUCTOR PARAMETERS
_token name - string
Peter's Token
_token symbol - string
PTK
_decimal units - 8 bits unsigned integer
2
_initial token supply - 256 bits unsigned integer
5000

2. After you deploy the contract, not only you can see your contract is on the "custom contract" section, but also **you can see another item on the "custom tokens" section too**.


Ethereum Wallet
Ethereum Wallet Accounts Edit View Develop Window Help
WALLETS SEND PRIVATE-NET 1 peers | 10,432 19s since last block CONTRACTS 37,700.38 ETHER

To watch and interact with a contract already deployed on the blockchain, you need to know its address and the description of its interface in JSON format.

BALLOT EXER... 0.00 ether 0x009C0d47c8D55eA0...
BALLOT 0.00 ether 0xD7d0712F173a264A...
PETER'S TOKEN (ADMIN PAGE) 0.00 ether 0x4542Ea4665d70e7EeC060751363c9572f91190a0

WATCH CONTRACT

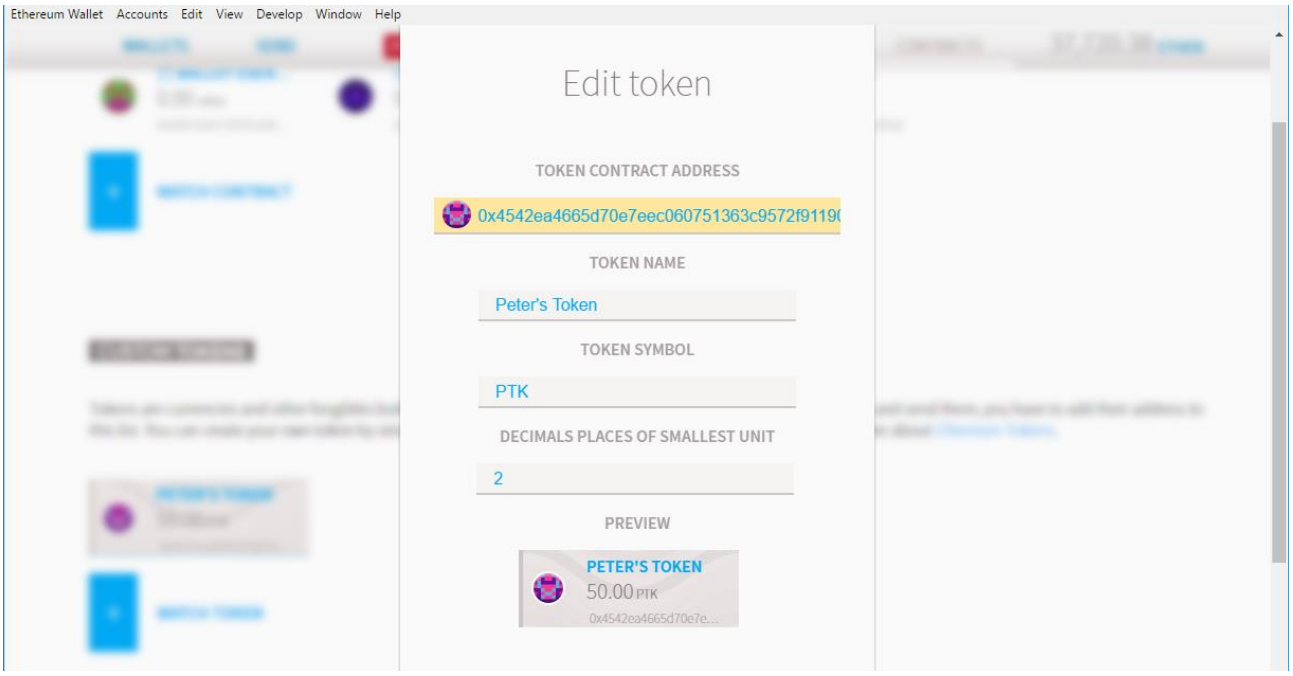
CUSTOM TOKENS

Tokens are currencies and other fungibles built on the Ethereum platform. In order fo... to this list. You can create your own token by simply modifying this example of a cust...

PETER'S TOKEN 50.00 PTK 0x4542ea4665d70e7e...

We have set the initial token supply to 5000. Since the decimal units we have defined is 2, therefore, our smart contract allocates 50 custom tokens to our personal account.

3. Click your custom token $\rightarrow$ You shall see a popup that shows the settings of the token.


Ethereum Wallet Accounts Edit View Develop Window Help

Edit token

TOKEN CONTRACT ADDRESS
0x4542ea4665d70e7eec060751363c9572f9119

TOKEN NAME
Peter's Token

TOKEN SYMBOL
PTK

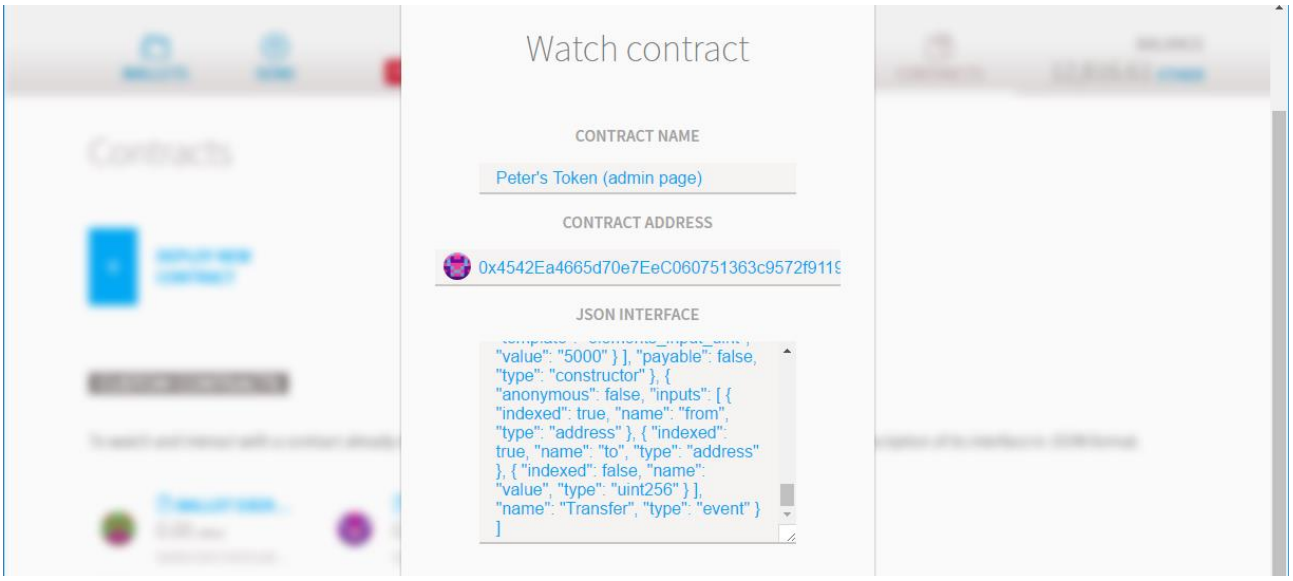
DECIMALS PLACES OF SMALLEST UNIT
2

PREVIEW
PETER'S TOKEN 50.00 PTK 0x4542ea4665d70e7e...

Custom Token section in Ethereum Wallet

If you follow all the standards in your smart contract, the Ethereum Wallet will be able to detect it as a custom token and automatically add your token to the "custom token" section.

4. To make your second computer use your custom token, you may add the smart contract to the "custom contract" section in your second computer as usual.



Watch contract

CONTRACT NAME

Peter's Token (admin page)

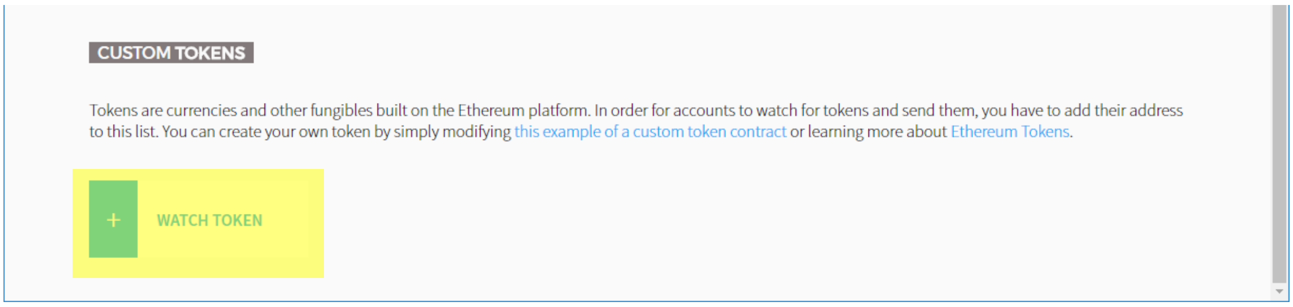
CONTRACT ADDRESS

0x4542Ea4665d70e7EeC060751363c9572f911

JSON INTERFACE

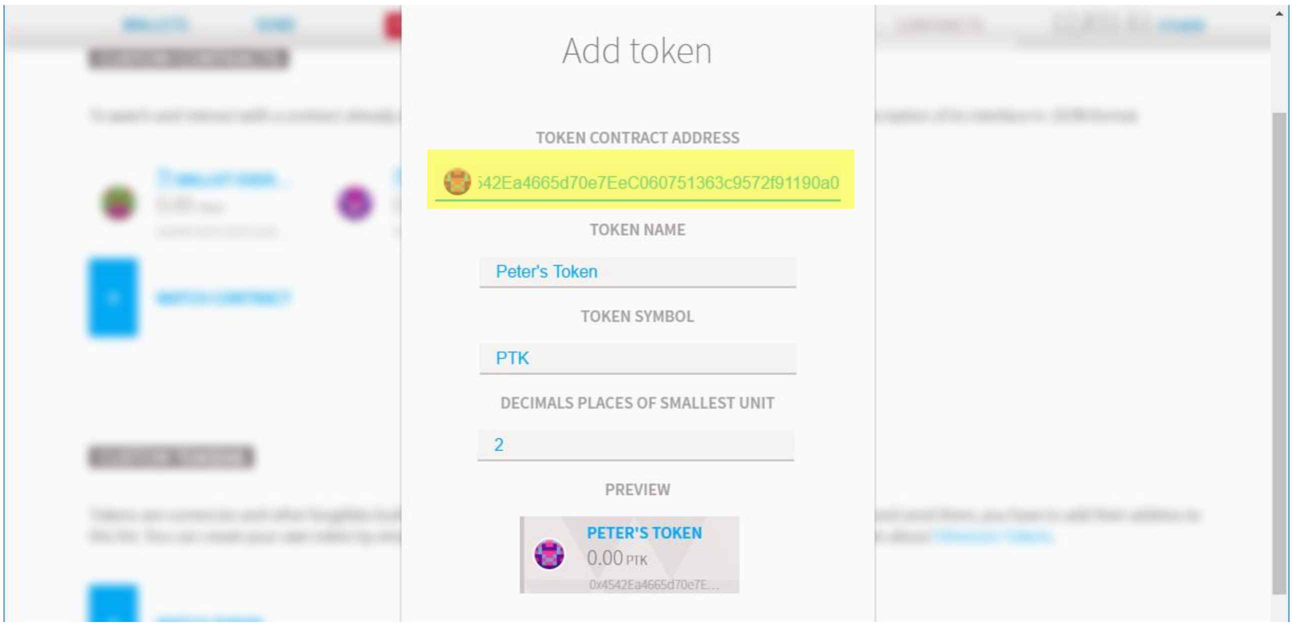
```
"value": "5000" } ], "payable": false,
"type": "constructor" }, {
"anonymous": false, "inputs": [ {
"indexed": true, "name": "from",
"type": "address" }, { "indexed":
true, "name": "to", "type": "address"
}, { "indexed": false, "name":
"value", "type": "uint256" } ],
"name": "Transfer", "type": "event" }
]
```

5. Furthermore, you also need to add the smart contract to the "custom token" section by specifying the smart contract's address. Once you fill in the address, the rest of the inputs will be filled automatically.



CUSTOM TOKENS

Tokens are currencies and other fungibles built on the Ethereum platform. In order for accounts to watch for tokens and send them, you have to add their address to this list. You can create your own token by simply modifying this example of a custom token contract or learning more about Ethereum Tokens.



Add token

TOKEN CONTRACT ADDRESS

542Ea4665d70e7EeC060751363c9572f91190a0

TOKEN NAME

Peter's Token

TOKEN SYMBOL

PTK

DECIMALS PLACES OF SMALLEST UNIT

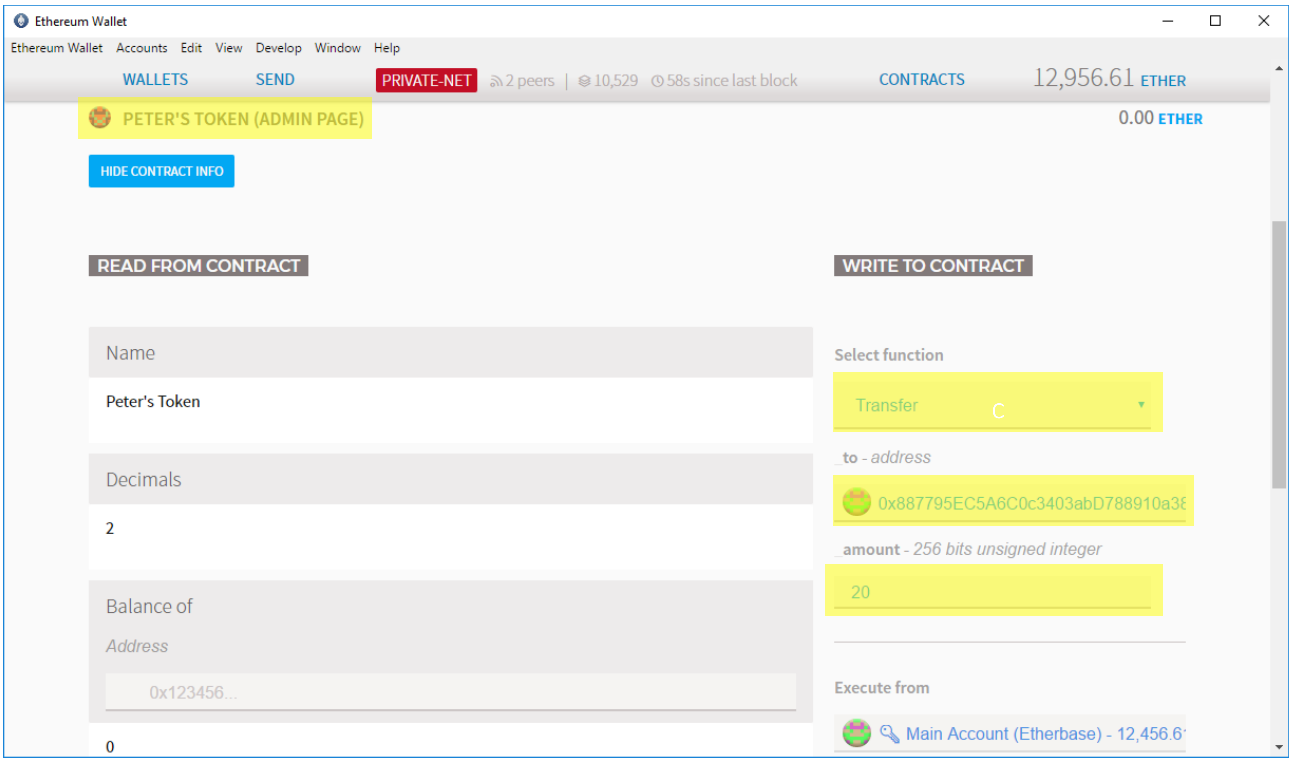
2

PREVIEW

PETER'S TOKEN
0.00 PTK
0x4542Ea4665d70e7E...

Transfer custom tokens in Ethereum Wallet

To transfer custom tokens, you can enter the contract's page → select the transfer function → input the target personal account address and the transfer amount → hit execute.

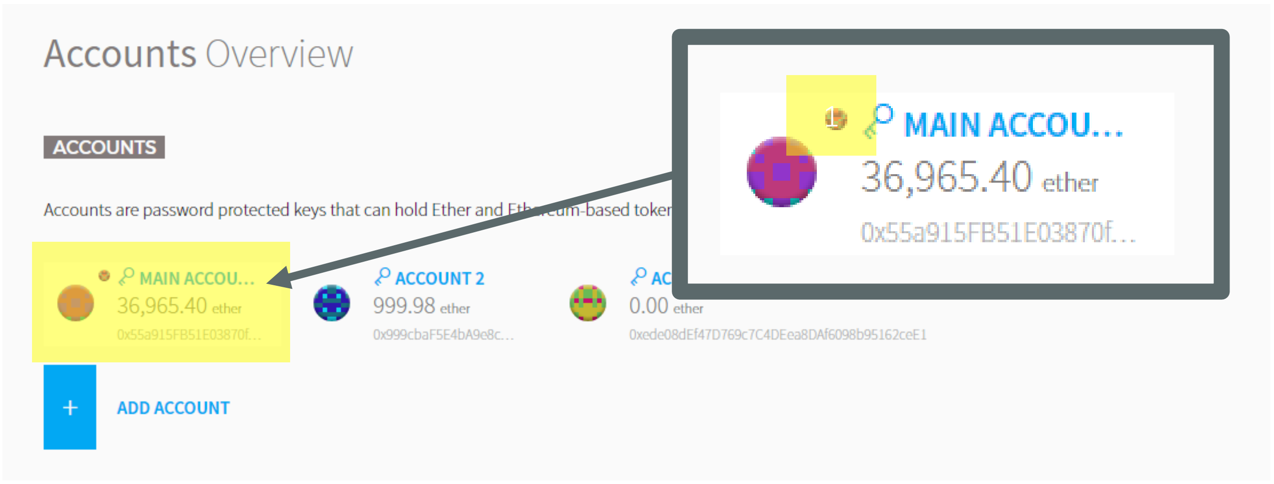


Name
Peter's Token
Decimals
2
Balance of
Address
0x123456...
0

Select function
Transfer
_to - address
0x887795EC5A6C0c3403abD788910a38
_amount - 256 bits unsigned integer
20
Execute from
Main Account (Etherbase) - 12,456.6

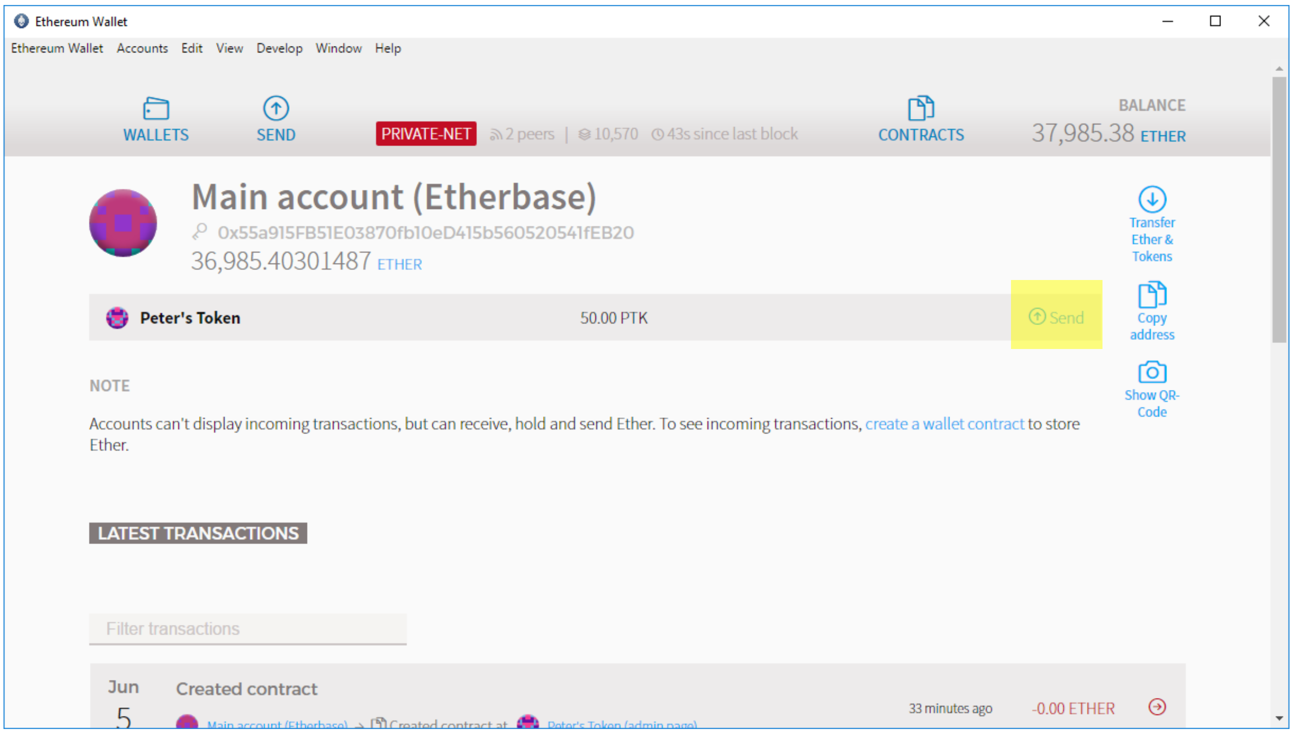
Besides this method, we can use the built-in feature provided by the Ethereum Wallet. Please note that the following method only works if your smart contract follows all the standards aforementioned.

6. **Go back to Wallet's page → Find the personal account that stores your custom tokens.** To find that personal account, you may look for a small ball icon located at the top-right corner of the personal account item.



Accounts are password protected keys that can hold Ether and Ethereum-based tokens

7. Click that personal account → Now, you should have noticed that one addition row has appeared to show how many custom tokens you own → Mouse over that row → Click "Send".



Main account (Etherbase)

0x55a915FB51E03870fb10eD415b560520541fEB20

36,985.40301487 ETHER

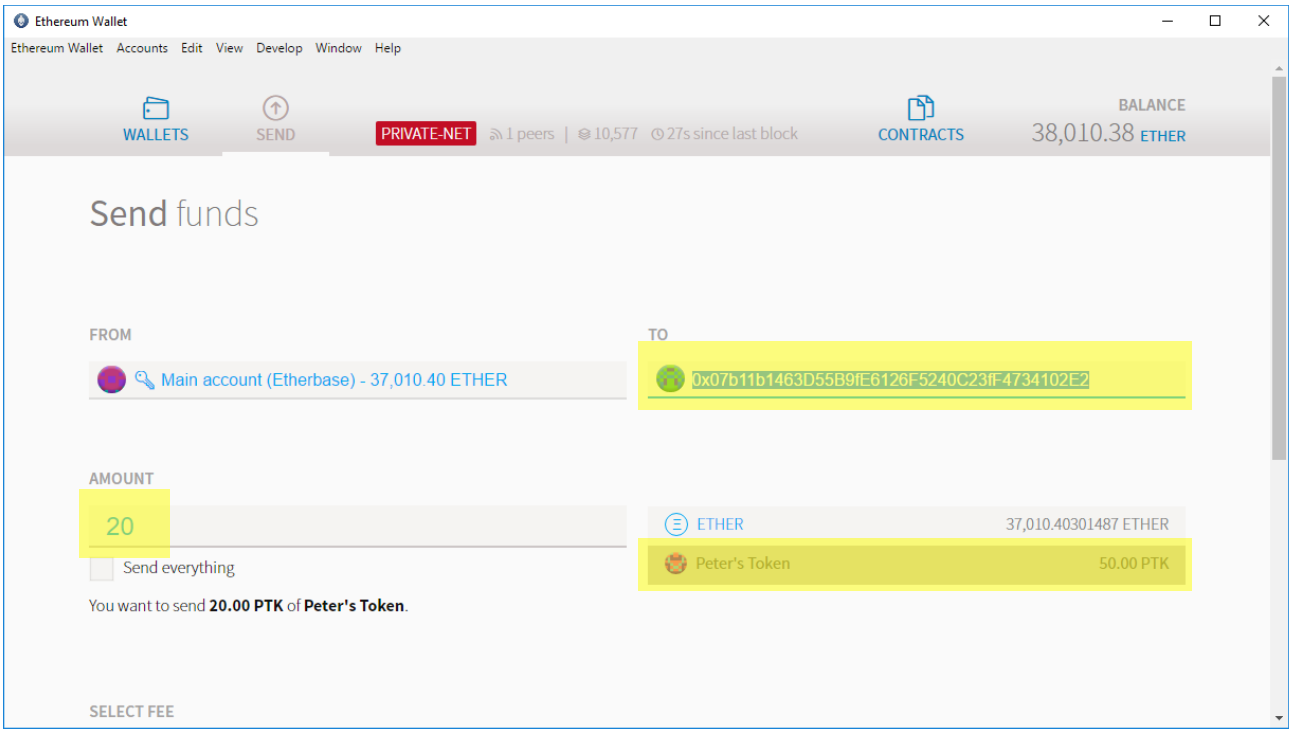
Peter's Token 50.00 PTK

NOTE

Accounts can't display incoming transactions, but can receive, hold and send Ether. To see incoming transactions, create a wallet contract to store Ether.

Jun 5 Created contract 33 minutes ago -0.00 ETHER

8. Sending your custom token to other address in this page is similar to how you transfer Ether. Input the recipient account's address → Input the amount → Select your custom token → Hit send.


Ethereum Wallet
Ethereum Wallet Accounts Edit View Develop Window Help
WALLETS
SEND
PRIVATE-NET
1 peers | 10,577 27s since last block
CONTRACTS
BALANCE 38,010.38 ETHER

Send funds

FROM

Main account (Etherbase) - 37,010.40 ETHER

TO

0x07b11b1463D55B9fE6126F5240C23fF4734102E2

AMOUNT

20

Send everything

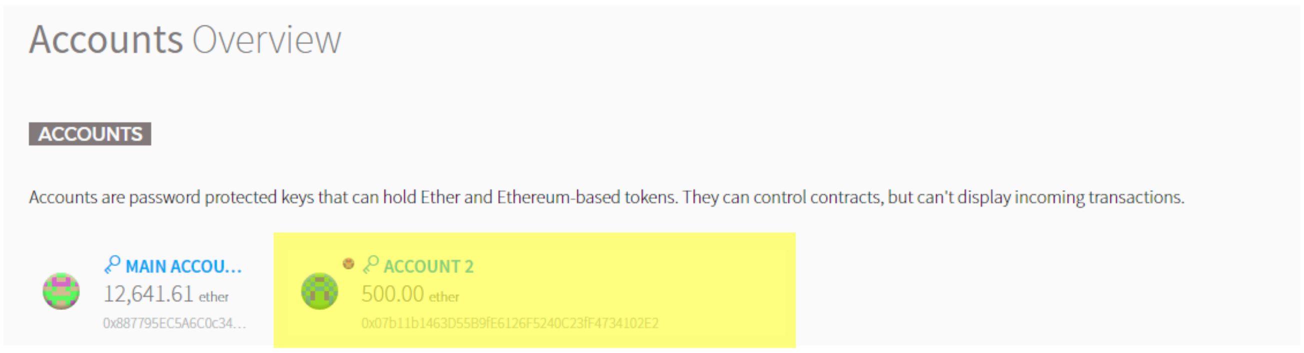
You want to send **20.00 PTK** of **Peter's Token**.

ETHER 37,010.40301487 ETHER

Peter's Token 50.00 PTK

SELECT FEE

9. After a block is being mined, your custom token should have been transferred to your second account. Go to your second account's wallet page and verify.



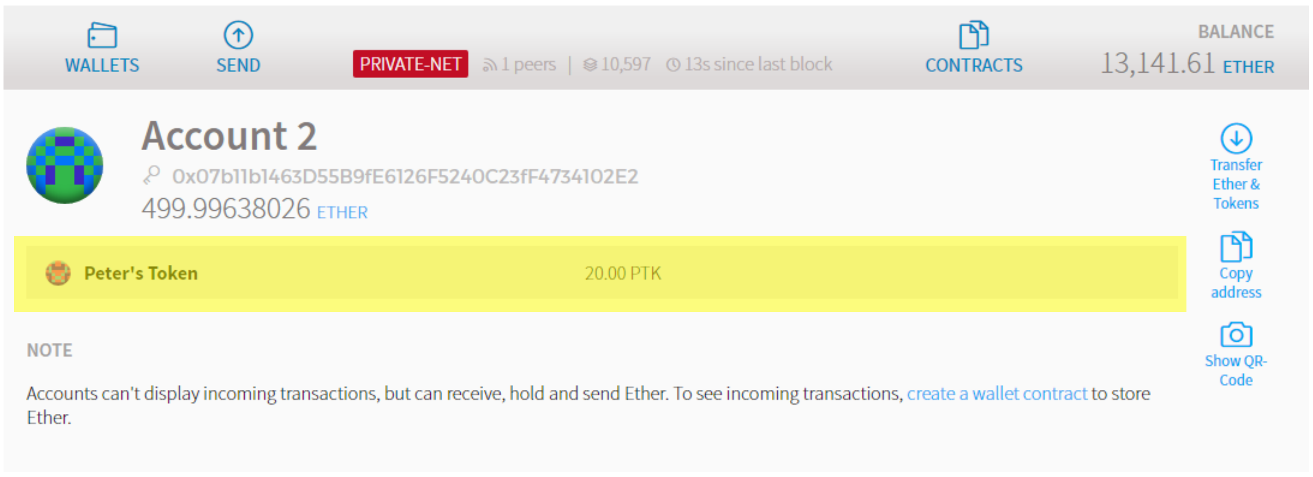
Accounts Overview

ACCOUNTS

Accounts are password protected keys that can hold Ether and Ethereum-based tokens. They can control contracts, but can't display incoming transactions.

MAIN ACCOU...
12,641.61 ether
0x887795EC5A6C0c34...

ACCOUNT 2
500.00 ether
0x07b11b1463D55B9fE6126F5240C23fF4734102E2



Account 2
0x07b11b1463D55B9fE6126F5240C23fF4734102E2
499.99638026 ETHER

Peter's Token 20.00 PTK

NOTE

Accounts can't display incoming transactions, but can receive, hold and send Ether. To see incoming transactions, create a wallet contract to store Ether.

Solidity programming techniques

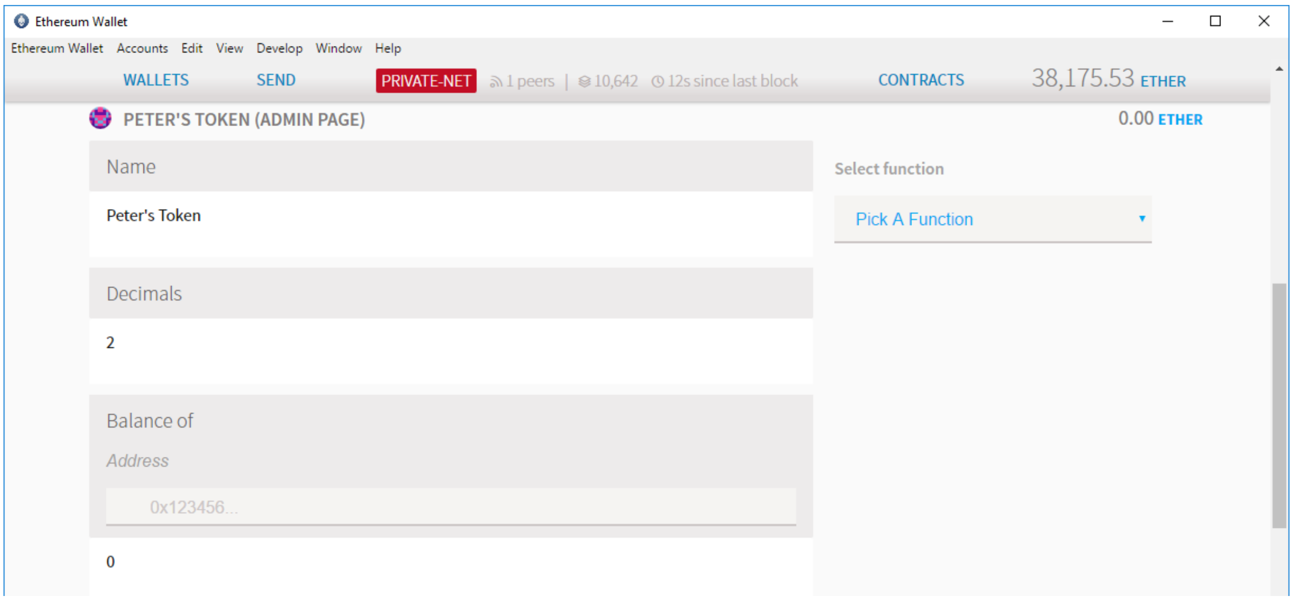
Public keyword

If you want to show some smart contract values in the Ethereum Wallet, you can implement several constant functions. It is very tedious for programmers to write getter functions for each of the variable.

Thankfully, the Solidity provides the public keyword to ease this problem.

```
address public administrator;
string public name;
string public symbol;
uint8 public decimals;
mapping(address => uint256) public balanceOf;
```

By adding a "public" keyword after the type of a variable, the Solidity compiler will automatically generate getter functions for those variables. So, you may not need to write any constant getter functions again for you to show values in the Ethereum Wallet.


Ethereum Wallet
Ethereum Wallet Accounts Edit View Develop Window Help
WALLETS
SEND
PRIVATE-NET
1 peers | 10,642 12s since last block
CONTRACTS
38,175.53 ETHER
PETER'S TOKEN (ADMIN PAGE)
0.00 ETHER
Select function
Pick A Function

Name

Peter's Token

Decimals

2

Balance of

Address

0x123456...

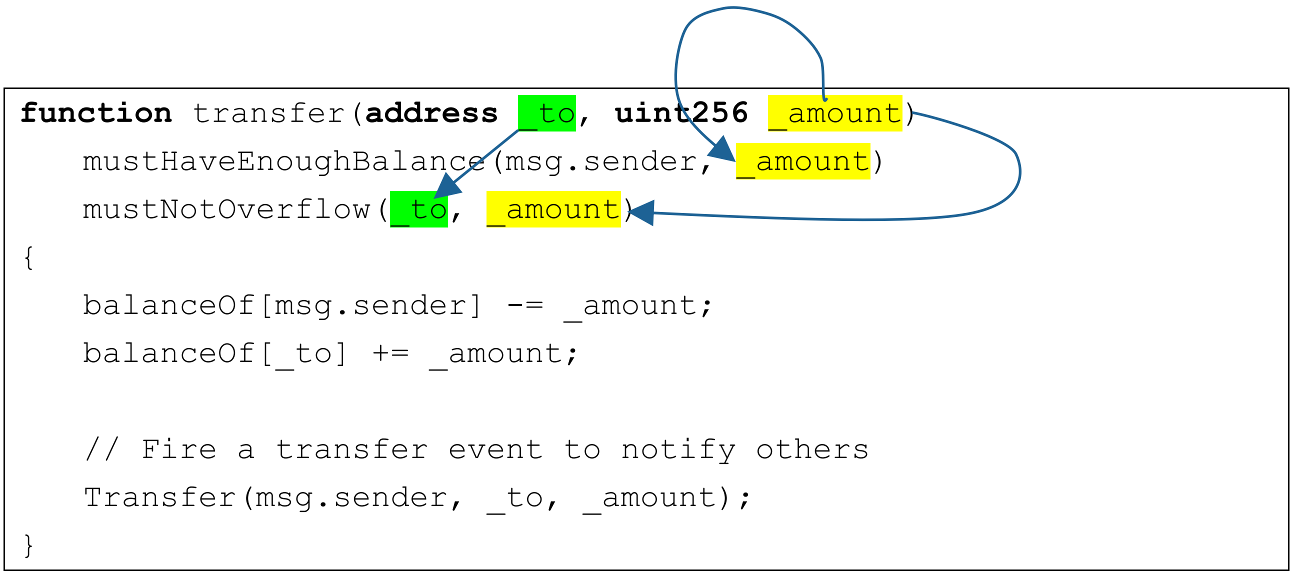
0

Important note:

Omitting the "public" keyword does not mean the variable is invisible (like the behavior of the "private" keyword in any Java programs). Please remember one of the characteristics of blockchain is public accessibility. Either you declare a variable with the "public" keyword or not, other users will have the right to read your variable's value.

Passing function inputs to a modifier

It is possible for you to pass any function inputs (i.e. parameters) to a modifier. The arrows below indicate how the names are referenced.



```
function transfer(address _to, uint256 _amount)
    mustHaveEnoughBalance(msg.sender, _amount)
    mustNotOverflow(_to, _amount)
{
    balanceOf[msg.sender] -= _amount;
    balanceOf[_to] += _amount;

    // Fire a transfer event to notify others
    Transfer(msg.sender, _to, _amount);
}
```

Integer overflow checking

Integer overflow is a serious issue when dealing with currencies or custom tokens. **In order to prevent it, we have to make sure the added amount of balance must be greater than the current balance.**

To have a good practice, you should check for overflow problems whenever you manipulate any balances in your custom token contract.

```
modifier mustNotOverflow(
    address _toAddress,
    uint256 _addingAmount)
{
    if(balanceOf[_toAddress] + _addingAmount <
        balanceOf[_toAddress])
    {
        throw;
    }
    _;
}
```

If you do not know what is integer overflow, please check out the Wikipedia for more information: https://en.wikipedia.org/wiki/Integer_overflow.

Exercises

Complete the following requirements.

1. Create a new smart contract called "MyDigitalCoupon" that implements all the standard features of a custom token. *(Tip: Try not to copy-and-paste the example source code directly, you will learn more if you program the custom token from stretch)*

2. Create a new non-constant function called "issueNewCoupon" with the followings.

Inputs	1. An address of the coupon recipient. 2. An unsigned integer amount of coupon for issuance.
Pre-conditions	Only executes when the caller is the administrator (i.e. contract creator), and the issuance does not cause any integer overflow.
Algorithms	1. Add the balance of the input address by the input amount.

3. Create a new non-constant function called "burnCoupon" with the followings.

Inputs	1. An address of which coupon is going to be destroyed. 2. An unsigned integer amount of coupon to be destroyed.
Pre-conditions	Only executes when the caller is the administrator (i.e. contract creator), and the input account has enough balance.
Algorithms	1. Decrease the balance of the input address by the input amount.

4. Deploy this new custom token. Name it as "PolyU Restaurant Coupon" and use "PCP" as the token symbol. The decimal places of the token are 0, and issue 1000 PCP to your personal account.

5. Transfer 250 PCP from your personal account to another one, and verify the successfulness of the transfer in your Ethereum Wallet.

References

1. Create your own crypto-currency
   https://www.ethereum.org/token