

REPUBLIQUE DE CÔTE D'IVOIRE



UNION – DISCIPLINE – TRAVAIL

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique (MESRS)



RAPPORT DE TRAVAUX PRATIQUES

REMIX – ETHEREUM IDE

Réalisé par :

AHO Yesso Michée

Elève-Ingénieur en Informatique 3^{ème} année

Enseignant-Chercheur pédagogique :

Mr DJICKO Bonnai

Année académique : 2023-2024

I. SOMMAIRE

II. INTRODUCTION.....	3
III. BASIC SYNTAX.....	4
IV. PRIMITIVE DATA TYPES.....	5
V. VARIABLES.....	6
VI. FUNCTIONS.....	7
1. Reading and Writing to a State Variable.....	7
2. View and Pure.....	7

II. INTRODUCTION

The screenshot shows the Remix IDE interface. On the left, the 'Compiler' panel is active, displaying the version '0.8.22+commit.4fc1097e' and options for 'Auto compile' and 'Hide warnings'. Below these are buttons for 'Compile introduction.sol', 'Compile and Run script', 'Publish on Ipfs', 'Publish on Swarm', and 'Compilation Details'. The 'CONTRACT' dropdown shows 'Counter (introduction.sol)'. The main editor displays the Solidity code for the 'Counter' contract:

```
2 pragma solidity ^0.8.3;
3
4 contract Counter {
5     uint public count;
6
7     // Function to get the current count
8     function get() public view returns (uint) { 2459 gas
9         return count;
10    }
11
12    // Function to increment count by 1
13    function inc() public { infinite gas
14        count += 1;
15    }
16
17    // Function to decrement count by 1
18    function dec() public { infinite gas
19        count -= 1;
20    }
21 }
```

The screenshot shows the Remix IDE interface with the 'Deploy' panel on the left. The 'Deploy' button is highlighted, and the 'At Address' section is active. The 'Transactions recorded' section shows three transactions: a constructor call, an 'inc' call, and a 'get' call. The 'Deployed Contracts' section shows the 'COUNTER AT 0XD91...39138 (MEMORY)' contract with buttons for 'dec', 'inc', 'count', and 'get'. The 'Low level interactions' section shows the 'CALLDATA' for the 'get' call. The main editor displays the Solidity code for the 'Counter' contract, and the 'Debug' console shows the execution details of the transactions:

```
[vm] from: 0x583...eddC4 to: Counter.(constructor) value: 0 wei data: 0x68...68033 logs: 0 hash: 0x84e...4f25c
transact to Counter.inc pending ...

[vm] from: 0x583...eddC4 to: Counter.inc() 0xd91...39138 value: 0 wei data: 0x371...303c0 logs: 0 hash: 0x0fa...bc7c4
transact to Counter.inc pending ...

[vm] from: 0x583...eddC4 to: Counter.inc() 0xd91...39138 value: 0 wei data: 0x371...303c0 logs: 0 hash: 0x7f3...03b6d
call to Counter.get

CALL [call] from: 0x58380da6a701c568545dcfc803fc8b75f56beddC4 to: Counter.get() data: 0x6d4...ce63c
```

The screenshot shows the Remix IDE interface with the 'Deploy' panel on the left. The 'Deploy' button is highlighted, and the 'At Address' section is active. The 'Transactions recorded' section shows three transactions: a constructor call, an 'inc' call, and a 'get' call. The 'Deployed Contracts' section shows the 'COUNTER AT 0XD91...39138 (MEMORY)' contract with buttons for 'dec', 'inc', 'count', and 'get'. The 'Low level interactions' section shows the 'CALLDATA' for the 'get' call. The main editor displays the Solidity code for the 'Counter' contract, and the 'Debug' console shows the execution details of the transactions:

```
[call] from: 0x58380da6a701c568545dcfc803fc8b75f56beddC4 to: Counter.get() data: 0x6d4...ce63c
transact to Counter.dec pending ...

[vm] from: 0x583...eddC4 to: Counter.dec() 0xd91...39138 value: 0 wei data: 0xb3b...cfa82 logs: 0 hash: 0x540...7485b
call to Counter.get

CALL [call] from: 0x58380da6a701c568545dcfc803fc8b75f56beddC4 to: Counter.get() data: 0x6d4...ce63c
call to Counter.get

CALL [call] from: 0x58380da6a701c568545dcfc803fc8b75f56beddC4 to: Counter.get() data: 0x6d4...ce63c
```

III. BASIC SYNTAX

The screenshot shows the LEARNETH IDE interface. On the left, a sidebar contains a home icon, a menu icon, and a progress indicator for 2/19 sections. The main text area on the left provides explanations for Solidity keywords: `pragma` (specifying compiler version), `contract` (defining a contract), `state variable` (defining a variable), `statically typed` (language characteristic), `visibility` (access control), and `data types` (variable types). An **Assignment** section is also present. The right pane shows the `basicSyntax.sol` file with the following code:

```
1 // SPDX-License-Identifier: MIT
2 // compiler version must be greater than or equal to 0.8.3 and less than 0.9.0
3 pragma solidity ^0.8.3;
4
```

Below the code editor is a transaction log with a search bar. It displays two transactions:

- CALL** [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c
transact to Counter.dec pending ...
- VM** [vm] from: 0x583...eddC4 to: Counter.dec() 0xd91...39138 value: 0 wei data: 0xb3b...cfa82 logs: 0 hash: 0x540...7485b

This screenshot shows the same LEARNETH IDE interface, but the `basicSyntax.sol` file now contains a contract definition:

```
2 // compiler version must be greater than or equal to 0.8.3 and less than 0.9.0
3 pragma solidity ^0.8.3;
4
5 contract MyContract {
6     string public name = "Alice";
7 }
```

The transaction log at the bottom shows a different set of transactions:

- CALL** [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c
transact to Counter.dec pending ...
- VM** [vm] from: 0x583...eddC4 to: Counter.dec() 0xd91...39138 value: 0 wei data: 0xb3b...cfa82 logs: 0 hash: 0x540...
call to Counter.get

IV. PRIMITIVE DATA TYPES

```
1 Like uint, different ranges are available from int8 to int256
2 */
3 int8 public i8 = -1;
4 int public i256 = 456;
5 int public i = -123; // int is same as int256
6
7 address public addr = 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c;
8 address public newAddr = 0x742d35Cc6634C0532925a3b844Bc454e4438f44e;
9
```

```
int8 public i8 = -1;
int public neg = -4;
int public i256 = 456;
int public i = -123; // int is same as int256
```

```
uint8 public newU = 0; // the smallest
```

V. VARIABLES

```
contract Variables {  
    // State variables are stored on the blockchain.  
    string public text = "Hello";  
    uint public num = 123;  
  
    uint public blockNumber;
```

```
function doSomething() public { 22338 gas  
    // Local variables are not saved to the blockchain.  
    uint i = 456;  
  
    // Here are some global variables  
    uint timestamp = block.timestamp; // Current block timestamp  
    address sender = msg.sender; // address of the caller  
  
    blockNumber = block.number; // the assignment!!!  
}
```

VI. FUNCTIONS

1. Reading and Writing to a State Variable

```
contract SimpleStorage {  
    // State variable to store a number  
    uint public num;  
  
    bool public b = true;
```

```
function get_b() public view returns (bool) {  
    return b;  
}
```

2545 gas

The screenshot shows the Remix IDE interface. On the left, the 'set' tab is active, displaying the state of the SimpleStorage contract. The state variables are listed as follows:

- b**: bool true (with a 'b - call' button)
- get**: uint256 5
- get_b**: bool true
- num**: uint256 0

On the right, the 'transaction history' panel is visible, showing a list of transactions. The first transaction is a call to SimpleStorage.b() with data 0x4df...7e3d0. The second transaction is a call to SimpleStorage.get() with data 0x6d4...ce3c.

2. View and Pure

```
function addToX2(uint y) public {  
    x = x + y;  
}
```

infinite gas

addToX2	5	⌵
add	uint256 i, uint256 j	⌵
addToX	uint256 y	⌵
x		
0: uint256: 6		