



Lesson 3

REMIX STORAGE FACTORY

Hikmah Nisya - 1103184094
Radzis Araaf Jaya Jamaludin - 1103184234
Raudhatul Rafiqah Assyahiddini - 1103180225

Preparing Tools

Beberapa hal yang perlu digunakan dalam lesson ini.

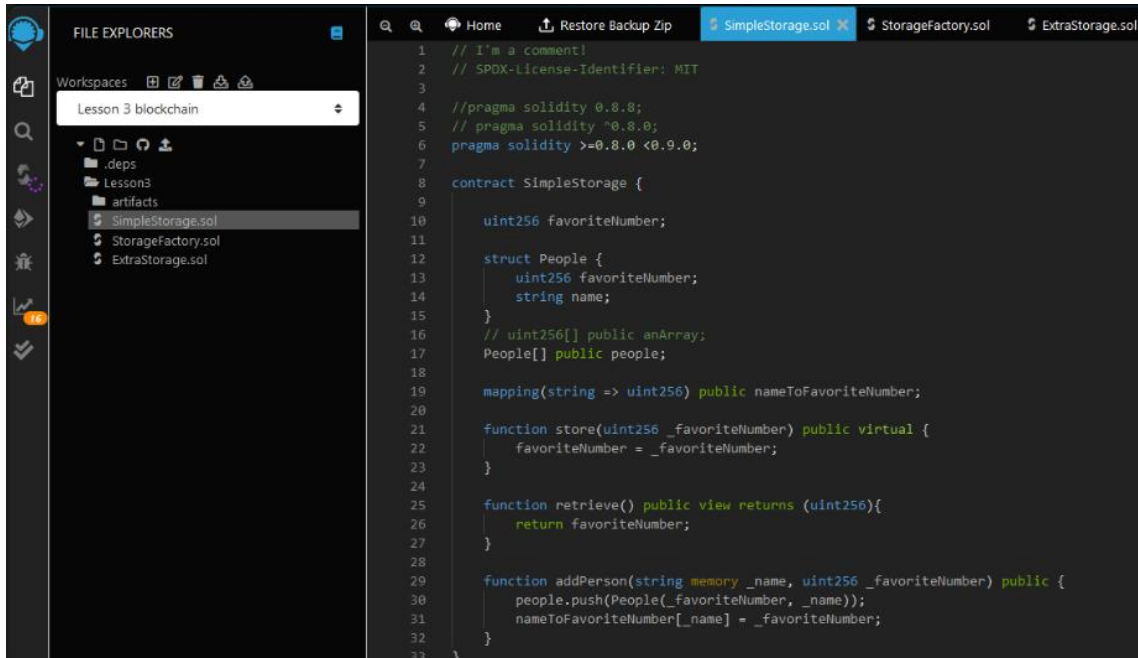
1. github : digunakan untuk kepentingan dalam menyimpan code dan menyimpan dokumentasi code yang telah kita buat.
2. Remix IDE : Remix IDE ini digunakan sebagai tempat code editor.
3. Solidity : Bahasa yang digunakan saat kita mempelajari Blockchain.
4. Waktu : kita membutuhkan banyak waktu untuk mempelajari blockchain, maka dari itu waktu adalah salah satu hal yang sangat dibutuhkan.

Membuat File pada REMIX IDE

Hal pertama yang dilakukan adalah membuka website Remix IDE : <https://remix.ethereum.org>. selanjutnya kita akan membuat workspace baru agar bersih dari template default yang diberikan oleh remix. Selanjutnya membuat 3 file yang ekstensi “.sol” seperti:

1. SimpleStorage.sol
2. StorageFactory.sol
3. ExtraStorage.sol

File ini nantinya akan kita gunakan.



Simple Storage

Selanjutnya kita akan mempersiapkan code yang telah kita buat pada SimpleStorage. Pada code disamping, digunakan compiler solidity lebih dari 0.7.0 hingga kurang dari versi 0.9.0. kemudian selanjutnya membuat nama kontrak dengan SimpleStorage dengan menyimpan variabel uint256, mapping variable, dan fungsi smart contract yang telah dibuat.

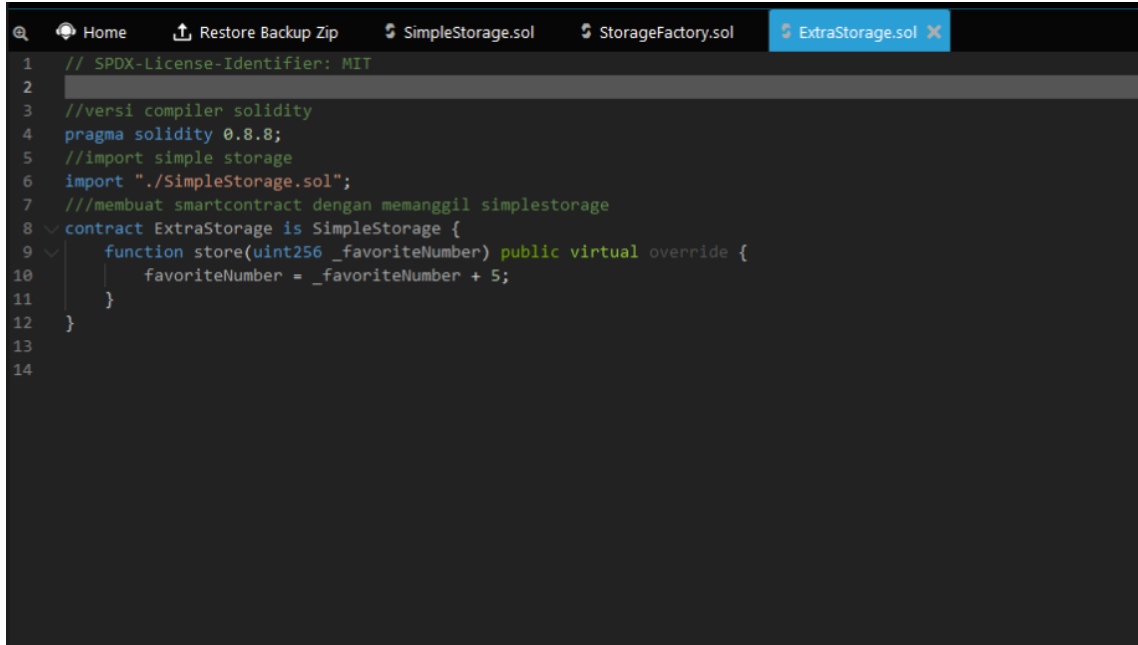
```
1 // SPDX-License-Identifier: MIT
2
3 // Versi compiler solidity yang akan digunakan
4 pragma solidity >=0.8.0 <0.9.0;
5
6 // Nama smartcontract yang akan di buat
7 contract SimpleStorage {
8     // Jenis variable yang digunakan menggunakan uint256
9     uint256 favoriteNumber;
10
11     // Membuat structure People dengan variabel dalamnya adalah uint 256 dan string
12     struct People {
13         uint256 favoriteNumber;
14         string name;
15     }
16     // uint256[] public anArray;
17     // array people akan dibuka menjadi public
18     People[] public people;
19
20     //proses mapping nilai string menjadi uint256 yang bersifat public dan disimpan pada nameToFavoriteNumber
21     mapping(string => uint256) public nameToFavoriteNumber;
22
23     //Fungsi store yang gunanya menyimpan favorite number dan bersifat public virtual
24     function store(uint256 _favoriteNumber) public virtual {
25         favoriteNumber = _favoriteNumber;
26     }
27
28     //Fungsi retrieve yang gunanya adalah mengambil nilai value dari favorite number
29     function retrieve() public view returns (uint256){
30         return favoriteNumber;
31     }
32
33     //Fungsi addPerson yang gunanya adalah menyimpan data orang baru beserta nomor favoritnya
34     function addPerson(string memory _name, uint256 _favoriteNumber) public {
35         people.push(People(_favoriteNumber, _name));
36         nameToFavoriteNumber[_name] = _favoriteNumber;
37     }
38 }
```

Storage Factory

Selanjutnya kita akan mempersiapkan code untuk StorageFactory. Pada code di samping, digunakan compiler solidity lebih dari 0.8.0. kemudian membuat nama kontrak dengan StorageFactory dengan menyimpannya ke SimpleStorage dan bersifat public. Selain itu dibuat juga fungsi pada kontrak seperti fungsi createsimplecontract yang bersifat public dengan fungsi membuat SC, lalu fungsi sfstore untuk menyimpan simplestorageindex dan simplestoragenumber secara public pada array dan fungsi sfget untuk mengambil nilai simplestorage index.

```
1 // SPDX-License-Identifier: MIT
2
3 // Versi compiler solidity yang akan digunakan
4 pragma solidity ^0.8.0;
5
6 //import smartcontract yang sebelumnya kita buat
7 import "./SimpleStorage.sol";
8
9 //nama smartcontract
10 contract StorageFactory {
11
12     //membuat array simplestorage dan bersifat public
13     SimpleStorage[] public simpleStorageArray;
14     //membuat fungsi create simplestoragecontract dengan fungsi membuat kontrak yang bersifat public
15     function createSimpleStorageContract() public {
16         SimpleStorage simpleStorage = new SimpleStorage();
17         simpleStorageArray.push(simpleStorage);
18     }
19     //membuat fungsi sfstore dengan fungsi menyimpan simplestorageindex dan simplestoragenumber secara public dan disimpan pada array
20     function sfStore(uint256 _simpleStorageIndex, uint256 _simpleStorageNumber) public {
21         // Address
22         // ABI
23         // SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).store(_simpleStorageNumber);
24         simpleStorageArray[_simpleStorageIndex].store(_simpleStorageNumber);
25     }
26     //membuat fungsi sfget dengan fungsi mengambil nilai dari simplestorageindex dan mengembalikan array nilainya
27     function sfGet(uint256 _simpleStorageIndex) public view returns (uint256) {
28         // return SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).retrieve();
29         return simpleStorageArray[_simpleStorageIndex].retrieve();
30     }
31 }
32
```

Extra Storage

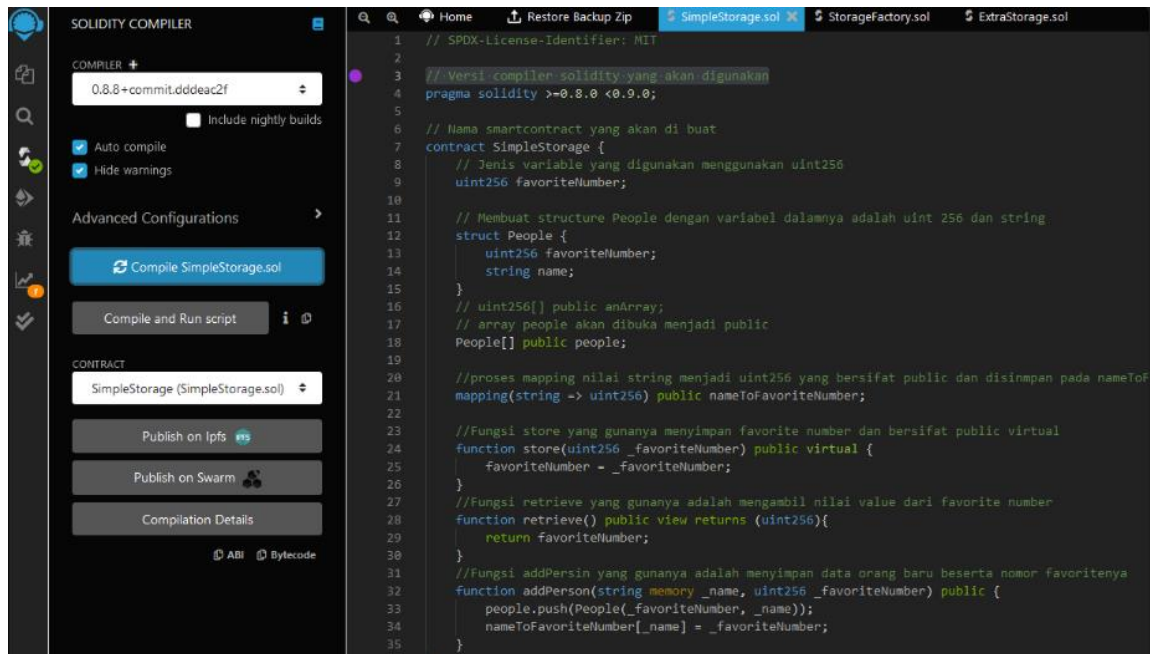


```
1 // SPDX-License-Identifier: MIT
2
3 //versi compiler solidity
4 pragma solidity 0.8.8;
5 //import simple storage
6 import "../SimpleStorage.sol";
7 ///membuat smartcontract dengan memanggil simplestorage
8 contract ExtraStorage is SimpleStorage {
9     function store(uint256 _favoriteNumber) public virtual override {
10         favoriteNumber = _favoriteNumber + 5;
11     }
12 }
13
14
```

Kemudian kita akan mempersiapkan code kita pada ExtraStorage. Pada code disamping digunakan compiler solidity lebih dari 0.8.0 lalu membuat nama kontrak dengan ExtraStorage dan memanggil SimpleStorage terlebih dahulu. Kemudian pada smart contractnya sendiri akan mengocerride nilai favoritenumbr dengan menambahkan lima pada nilai favoritenumbr sebelumnya.

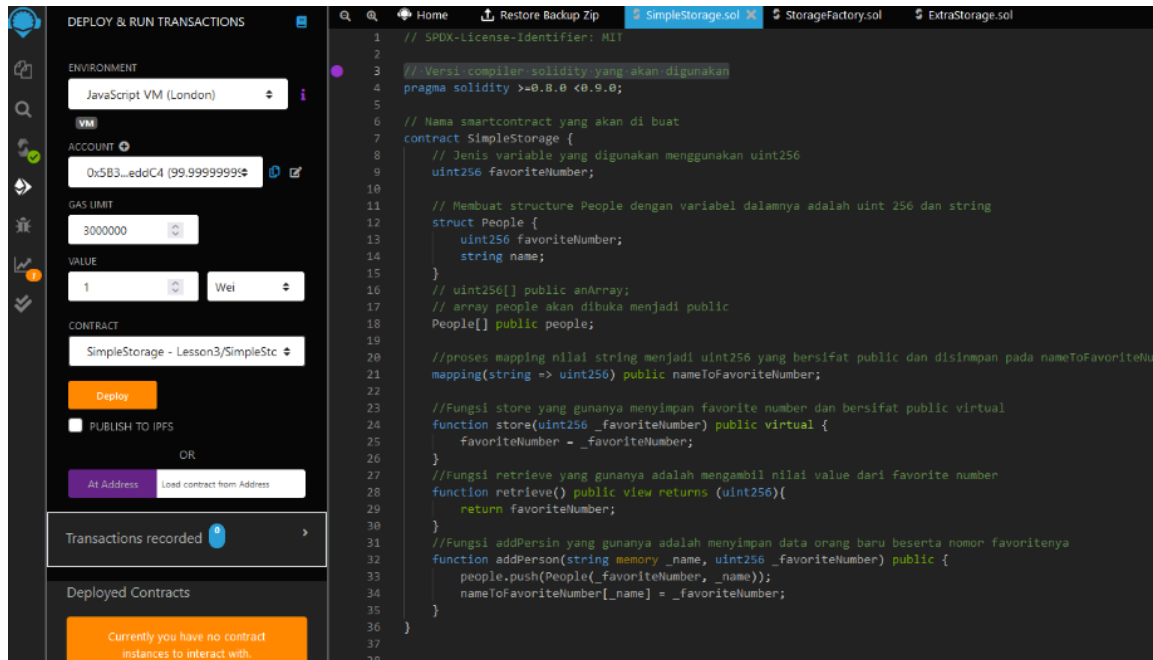
Compile Simple Storage

Tahap ini, kita melakukan compile terlebih dahulu code kita yang telah dibuat sebelumnya.



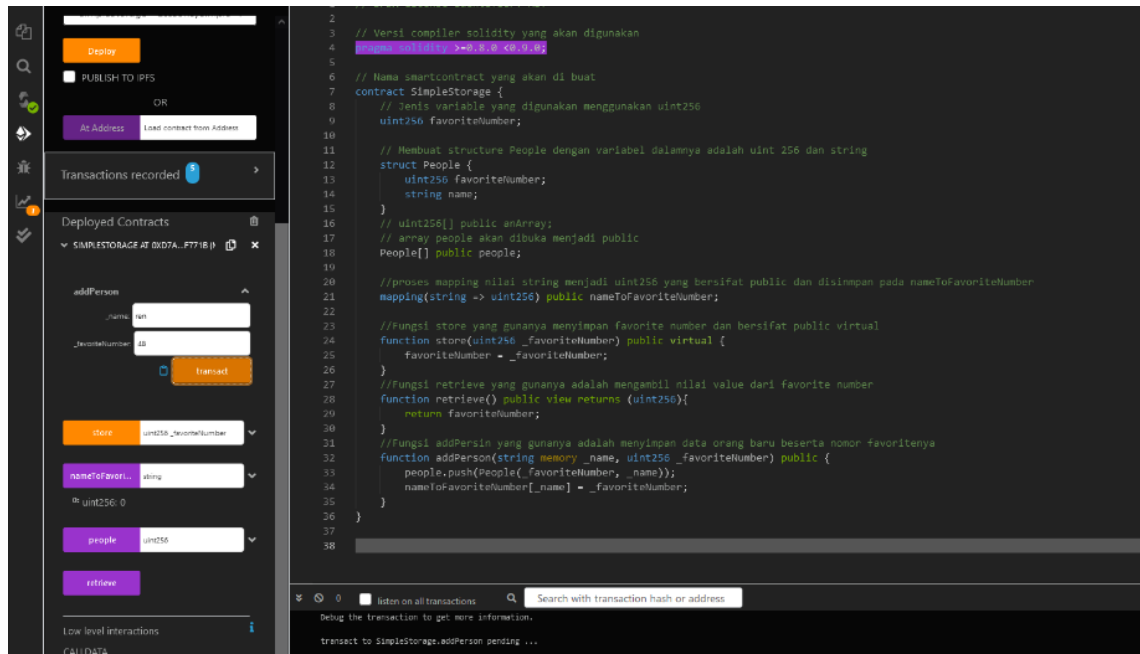
Deploy Simple Storage

Tahap ini kita akan melakukan deploy smart contract ke dalam jaringan blockchain tetapi hanya untuk test network saja. Kemudian pada environment kita dapat menggunakan apa yang telah tersedia di remix, tetapi pada test ini digunakan environment london.



Explore Simple Storage SC

Pertama, kita akan mencoba menyimpan value nama orang ke dalam blockchain melalui smart contract.



```
1
2
3 // Versi compiler solidity yang akan digunakan
4 pragma solidity ^0.8.0;
5
6 // Nama smartcontract yang akan di buat
7 contract SimpleStorage {
8     // jenis variable yang digunakan menggunakan uint256
9     uint256 favoriteNumber;
10
11     // Membuat structure People dengan variabel dalamnya adalah uint 256 dan string
12     struct People {
13         uint256 favoriteNumber;
14         string name;
15     }
16     // uint256[] public anArray;
17     // array people akan dibuka menjadi public
18     People[] public people;
19
20     //proses mapping nilai string menjadi uint256 yang bersifat public dan disimpan pada nameToFavoriteNumber
21     mapping(string -> uint256) public nameToFavoriteNumber;
22
23     //fungsi store yang gunanya menyimpan favorite number dan bersifat public virtual
24     function store(uint256 _favoriteNumber) public virtual {
25         favoriteNumber = _favoriteNumber;
26     }
27
28     //fungsi retrieve yang gunanya adalah mengambil nilai value dari favorite number
29     function retrieve() public view returns (uint256){
30         return favoriteNumber;
31     }
32
33     //fungsi addPerson yang gunanya adalah menyimpan data orang baru beserta nomor favoritnya
34     function addPerson(string memory _name, uint256 _favoriteNumber) public {
35         people.push(People(_favoriteNumber, _name));
36         nameToFavoriteNumber[_name] = _favoriteNumber;
37     }
38 }
```

Explore Simple Storage SC

Kedua, kita akan mencoba menyimpan value nomor favorite orang tersebut ke dalam blockchain melalui smart contract.

```
// Versi compiler solidity yang akan digunakan
// versi solidity >=0.8.0 <0.9.0
// Nama smartcontract yang akan di buat
contract SimpleStorage {
    // Jenis variable yang digunakan menggunakan uint256
    uint256 favoriteNumber;

    // Membuat structure People dengan variabel dalamnya adalah uint 256 dan string
    struct People {
        uint256 favoriteNumber;
        string name;
    }
    // uint256[] public anArray;
    // array people akan dibuka menjadi public
    People[] public people;

    //proses mapping nilai string menjadi uint256 yang bersifat public dan disimpan pada nameToFavoriteNumber
    mapping(string => uint256) public nameToFavoriteNumber;

    //Fungsi store yang gunanya menyimpan favorite number dan bersifat public virtual
    function store(uint256 _favoriteNumber) public virtual {
        favoriteNumber = _favoriteNumber;
    }

    //Fungsi retrieve yang gunanya adalah mengambil nilai value dari favorite number
    function retrieve() public view returns (uint256){
        return favoriteNumber;
    }

    //Fungsi addPerson yang gunanya adalah menyimpan data orang baru beserta nomor favoritnya
    function addPerson(string memory _name, uint256 _favoriteNumber) public {
        people.push(People(_favoriteNumber, _name));
        nameToFavoriteNumber[_name] = _favoriteNumber;
    }
}
```

[vn] From: 0x533...edc04 to: SimpleStorage.addPerson(string,uint256) 0x07A...F771B value: 0 wei data: 0x6f7...00000 logs: 0 hash: 0xf52...55e63
transact to SimpleStorage.store pending ...

Explore Simple Storage SC

Ketiga, kita mencoba memanggil hasil return, jika kita memberikan input nama orang yang sebelumnya ke dalam blockchain, apakah nilai yang dibalikan sama seperti nilai yang telah di input. Jika benar, maka smart contract yang telah dibuat sudah benar.

The screenshot displays a web interface on the left and Solidity code on the right. The web interface, titled 'Deployed Contracts', shows a contract named 'SIMPLESTORAGE AT 0xD7A...F7718'. It features three main sections: 'addPerson' with input fields for '_name' (value: 'an') and '_favoriteNumber' (value: '48'), a 'transact' button, and a 'store' section with a '_favoriteNumber' input (value: '48') and a 'transact' button. Below these is a 'nameToFavoriteNumber' section with a '_name' input (value: 'an') and a 'call' button. A dropdown menu for 'people' is set to 'uint256', and a 'retrieve' button is at the bottom. The Solidity code on the right is a SimpleStorage contract. It uses the Solidity compiler version 0.8.0. It defines a 'SimpleStorage' contract with a 'uint256 favoriteNumber' variable. It includes a 'struct People' with 'uint256 favoriteNumber' and 'string name' fields. A 'public' array 'people' is declared. The 'mapping' 'nameToFavoriteNumber' maps 'string' to 'uint256'. The 'store' function is a 'public virtual' function that takes a 'uint256 _favoriteNumber' and updates 'favoriteNumber'. The 'retrieve' function is a 'public view returns (uint256)' function that returns 'favoriteNumber'. The 'addPerson' function is a 'public' function that takes a 'string memory _name' and a 'uint256 _favoriteNumber', pushes a 'People' struct to the 'people' array, and updates 'nameToFavoriteNumber[_name]'.

```
2 // Versi compiler solidity yang akan digunakan
3 // Solidity 0.8.0
4
5
6 // Nama smartcontract yang akan di buat
7 contract SimpleStorage {
8     // Jenis variable yang digunakan menggunakan uint256
9     uint256 favoriteNumber;
10
11     // Membuat structure People dengan variabel dalamnya adalah uint 256 dan string
12     struct People {
13         uint256 favoriteNumber;
14         string name;
15     }
16     // uint256[] public anArray;
17     // array people akan dibuka menjadi public
18     People[] public people;
19
20     //proses mapping nilai string menjadi uint256 yang bersifat public dan disimpan pada nameToFavoriteNumber
21     mapping(string => uint256) public nameToFavoriteNumber;
22
23     //fungsi store yang gunanya menyimpan favorite number dan bersifat public virtual
24     function store(uint256 _favoriteNumber) public virtual {
25         favoriteNumber = _favoriteNumber;
26     }
27
28     //fungsi retrieve yang gunanya adalah mengambil nilai value dari favorite number
29     function retrieve() public view returns (uint256){
30         return favoriteNumber;
31     }
32
33     //fungsi addPerson yang gunanya adalah menyimpan data orang baru beserta nomor favoritnya
34     function addPerson(string memory _name, uint256 _favoriteNumber) public {
35         people.push(People(_favoriteNumber, _name));
36         nameToFavoriteNumber[_name] = _favoriteNumber;
37     }
38 }
```

Transaction hash: 0x58380d6a701c5685454c7c883f8875f50ed8c4 to: SimpleStorage.nameToFavoriteNumber(string) data: 0x8ba...00000

Explore Simple Storage SC

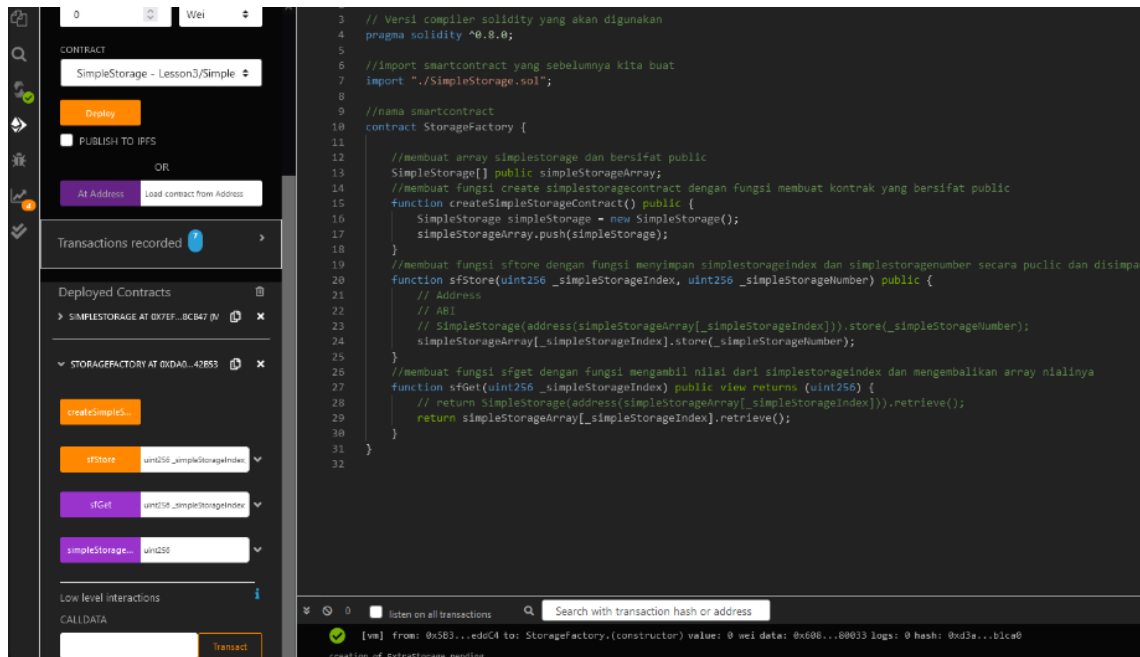
Keempat, kita mencoba memanggil hasil return, jika memberikan input nomor favorite orang, dan outputnya adalah input yang sebelumnya maka code sudah benar.

The image shows a web interface on the left and Solidity code on the right. The web interface has a dark theme and includes sections for 'addPerson', 'store', 'nameToFavoriteNumber', and 'people'. Each section has input fields and buttons. The 'addPerson' section has inputs for 'name' and 'favoriteNumber' with a 'transact' button. The 'store' section has an input for 'favoriteNumber' and a 'transact' button. The 'nameToFavoriteNumber' section has an input for 'name' and a 'call' button. The 'people' section has an input for 'people' and a 'call' button. There are also 'retrieve' buttons for each section. The Solidity code on the right is for a 'SimpleStorage' contract. It includes a pragma statement for Solidity version, a contract definition, a struct 'People', an array 'people', a mapping 'nameToFavoriteNumber', and functions 'store', 'retrieve', and 'addPerson'. The 'store' function takes a 'uint256 favoriteNumber' and sets it to 'nameToFavoriteNumber'. The 'retrieve' function returns the 'favoriteNumber' from 'nameToFavoriteNumber'. The 'addPerson' function takes a 'string memory name' and a 'uint256 favoriteNumber', pushes a 'People' struct to the 'people' array, and sets 'nameToFavoriteNumber[name]' to 'favoriteNumber'.

```
1 // Versi compiler solidity yang akan digunakan
2 pragma solidity >=0.8.0 <0.9.0;
3
4 // Nama smartcontract yang akan di buat
5
6 contract SimpleStorage {
7     // Jenis variable yang digunakan menggunakan uint256
8     uint256 favoriteNumber;
9
10
11     // Membuat structure People dengan variabel dalamnya adalah uint 256 dan string
12     struct People {
13         uint256 favoriteNumber;
14         string name;
15     }
16     // uint256[] public anArray;
17     // array people akan dibuka menjadi public
18     People[] public people;
19
20     //proses mapping nilai string menjadi uint256 yang bersifat public dan disimpan pada nameToFavoriteNumber
21     mapping(string => uint256) public nameToFavoriteNumber;
22
23     //Fungsi store yang gunanya menyimpan favorite number dan bersifat public virtual
24     function store(uint256 _favoriteNumber) public virtual {
25         favoriteNumber = _favoriteNumber;
26     }
27
28     //Fungsi retrieve yang gunanya adalah mengambil nilai value dari favorite number
29     function retrieve() public view returns (uint256){
30         return favoriteNumber;
31     }
32
33     //Fungsi addPerson yang gunanya adalah menyimpan data orang baru beserta nomor favoritnya
34     function addPerson(string memory _name, uint256 _favoriteNumber) public {
35         people.push(People(_favoriteNumber, _name));
36         nameToFavoriteNumber[_name] = _favoriteNumber;
37     }
38 }
```

Deploy Storage Factory

Jika berhasil kita akan melihat smart contract yang kita telah buat pada tab deploy contract, pada deploy contract dapat melihat fitur-fitur yang kita buat pada smart contract.



Explore Storage Factory

Pertama, kita akan membuat code yang mana kita dapat membuat sc lalu menyimpan nilai ke blockchain, pada Langkah pertam ini kita mencoba untuk membuat sc dengan index yang mulai dari nol.

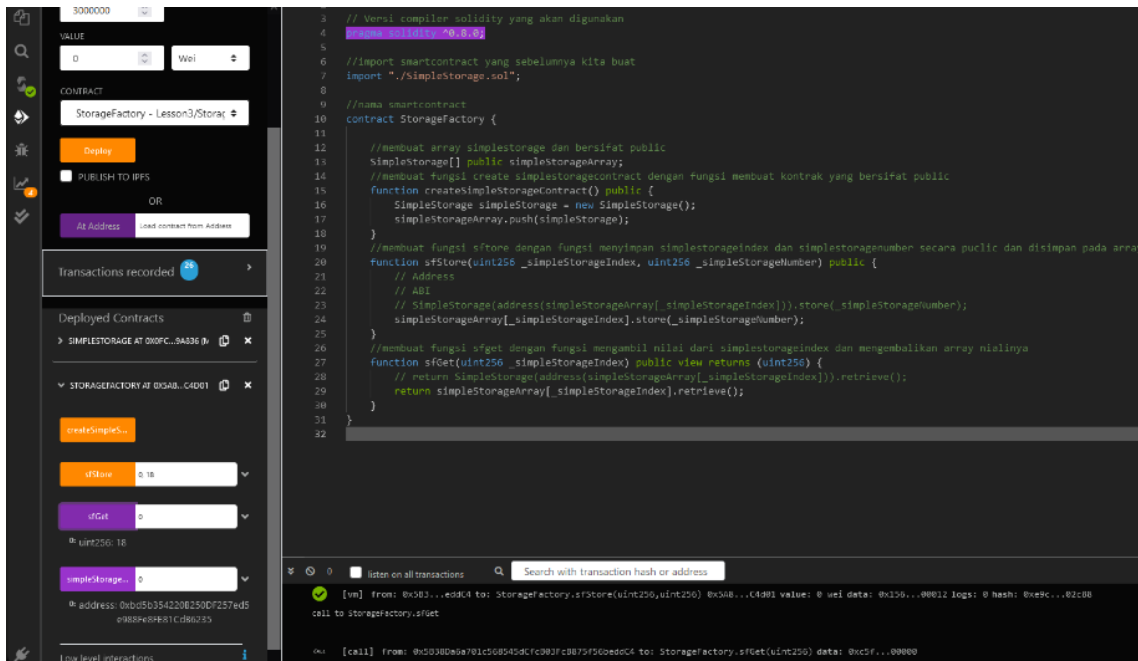
The screenshot displays the Remix IDE interface. On the left, the 'CONTRACT' tab shows the 'StorageFactory' contract with a 'Deploy' button. Below it, the 'Deployed Contracts' section lists the deployed contract 'STORAGEFACTORY AT 0x5AL...C40B1'. The 'Deployed Contracts' section also shows the 'createSimpleStorage' function with a dropdown menu for 'uint256 _simpleStorageIndex' and a 'store' button. The 'StorageFactory' contract is also listed with a dropdown menu for 'uint256: 0' and a 'simpleStorage' button. On the right, the 'Solidity' tab shows the source code for the 'StorageFactory' contract. The code defines a 'SimpleStorage' struct and a 'SimpleStorageArray' array. It includes functions for creating a simple storage contract, storing a value, and retrieving a value. The code is as follows:

```
1 // SPDX-License-Identifier: MIT
2
3 // Versi compiler solidity yang akan digunakan
4 // solidity ^0.8.0
5
6 //import smartcontract yang sebelumnya kita buat
7 import './SimpleStorage.sol';
8
9 //nama smartcontract
10 contract StorageFactory {
11
12     //membuat array simplestorage dan bersifat public
13     SimpleStorage[] public simpleStorageArray;
14     //membuat fungsi create simplestoragecontract dengan fungsi membuat kontrak yang bersifat public
15     function createSimpleStorageContract() public {
16         SimpleStorage simpleStorage = new SimpleStorage();
17         simpleStorageArray.push(simpleStorage);
18     }
19     //membuat fungsi sStore dengan fungsi menyimpan simpleStorageIndex dan simpleStorageNumber secara public dan disimpan pada array
20     function sStore(uint256 _simpleStorageIndex, uint256 _simpleStorageNumber) public {
21         // Address
22         // ABI
23         // simpleStorage(address(simpleStorageArray[_simpleStorageIndex])).store(_simpleStorageNumber);
24         simpleStorageArray[_simpleStorageIndex].store(_simpleStorageNumber);
25     }
26     //membuat fungsi sGet dengan fungsi mengambil nilai dari simpleStorageIndex dan mengembalikan array nilainya
27     function sGet(uint256 _simpleStorageIndex) public view returns (uint256) {
28         // return SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).retrieve();
29         return simpleStorageArray[_simpleStorageIndex].retrieve();
30     }
31 }
32
```

At the bottom, the 'Logs' tab shows the transaction details for the 'createSimpleStorage' function call, including the transaction hash and the data returned by the function.

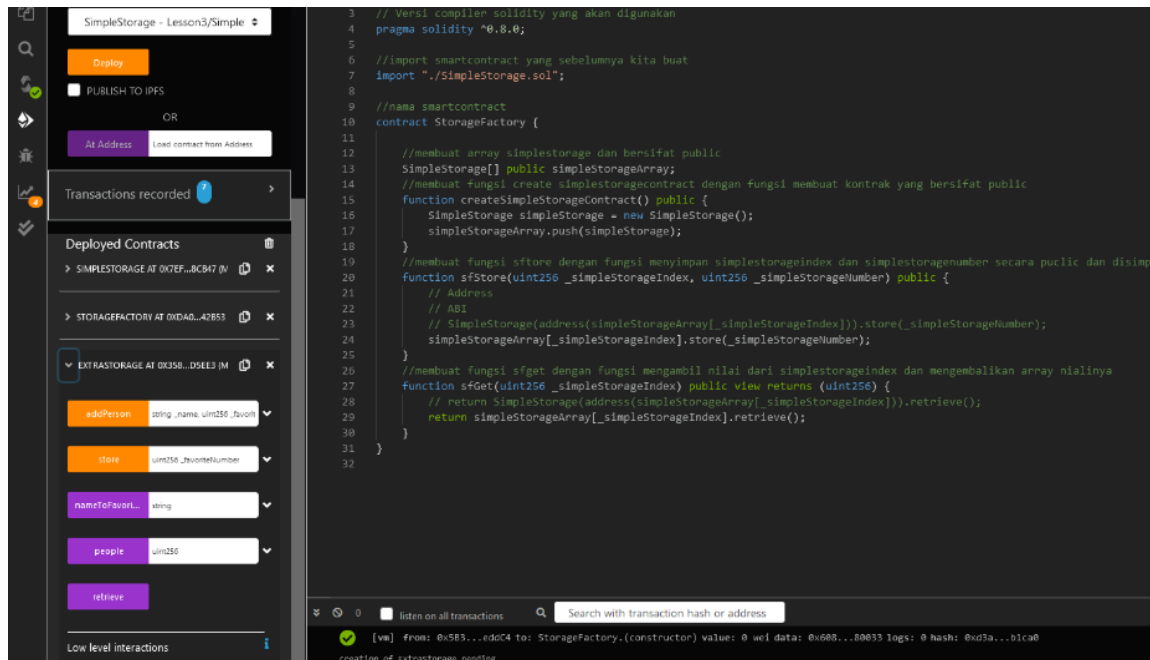
Explore Storage Factory

Kedua, kita dapat menyimpan nomor favorite dan disimpan pada index sc tertentu pada blockchain, disini digunakan index nol dan menyimpan nomor favorite delapa belas. Jika kita memanggil nilai delapan belas tadi yang diinput sebelumnya.



Mendeploy Extra Storage

Ketiga, jika berhasil maka kita dapat melihat smart contract yang telah dibuat pada tab deployed contracts, pada deployed contracts, kita dapat melihat fitur-fitur yang dibuat pada smart contract.



Explore Extra Storage

Pada smart contract extra storage kita memiliki fungsi override nilai yang kita miliki, pada SC ini ditambahkan nilai favorite number ditambah lima dari nilai awal, dapat dilihat Ketika menyimpan value 69 pada nama ren Ketika di get masih bernilai 69.

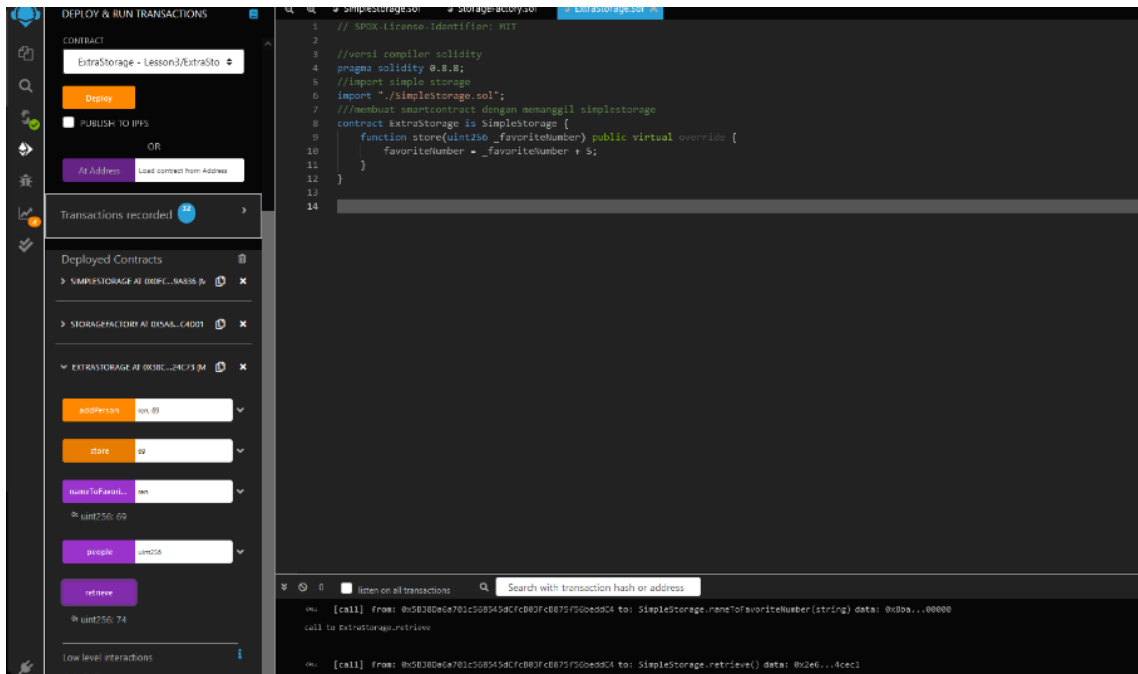
The screenshot shows a web interface for a smart contract named "ExtraStorage - Lesson3/ExtraSto". The interface includes a "Deploy" button, a "PUBLISH TO IPFS" checkbox, and a section for "Transactions recorded" showing 32 transactions. Below this, there is a list of "Deployed Contracts" including "SIMPLESTORAGE AT 0x0FC...9A836", "STORAGEFACTORY AT 0x5A8...C4D01", and "EXTRASTORAGE AT 0x38C...24C73". The "EXTRASTORAGE" contract is expanded, showing a form with fields for "addPerson" (value: ren, 69), "store" (value: 69), "nameToFavoriteNumber" (value: ren), and "people" (value: uint256: 69). There is also a "retrieve" button. The right side of the interface displays the Solidity code for the contract, which includes a pragma statement for Solidity 0.8.8, an import for "SimpleStorage.sol", and a function "store" that increments the "favoriteNumber" by 5. The bottom of the interface shows a transaction log with a green checkmark and the text: "[vn] from: 0x5B3...eddC4 to: SimpleStorage.store(uint256) 0x38c...24C73 value: 0 wei data: 0x605...00045 logs: 0 hash: 0x6b6...eaf4 call to ExtraStorage.nameToFavoriteNumber".

```
1 //version solidity
2
3 //versi compiler solidity
4 pragma solidity 0.8.8;
5 //import simple storage
6 import "../SimpleStorage.sol";
7 //membuat smartcontract dengan memanggil simplestorage
8 contract ExtraStorage is SimpleStorage {
9     function store(uint256 _favoriteNumber) public virtual override {
10         favoriteNumber = _favoriteNumber + 5;
11     }
12 }
13
14
```

[vn] from: 0x5B3...eddC4 to: SimpleStorage.store(uint256) 0x38c...24C73 value: 0 wei data: 0x605...00045 logs: 0 hash: 0x6b6...eaf4 call to ExtraStorage.nameToFavoriteNumber

Explore Extra Storage

Jika retrieve nilai dari SC extra storage kita akan mendapatkan nilai baru yaitu 75 yang berasal 69 + 5



The screenshot displays a web application for managing blockchain transactions. On the left sidebar, under 'Deployed Contracts', the 'EXTRASTORAGE' contract is selected. It shows fields for 'address' (0x08...), 'store' (69), 'favoriteNumber' (69), and 'retrieve' (69). The main area features a code editor with Solidity code for the 'EXTRASTORAGE' contract. The code includes a 'store' function that increments the 'favoriteNumber' by 5. Below the code editor, a transaction log shows a call to 'SimpleStorage.retrieve()' with a return value of 75.

```
1 // SPDX-License-Identifier: MIT
2
3 // versi compiler solidity
4 pragma solidity 0.8.0;
5 //import simple storage
6 import "./simplestorage.sol";
7
8 //membuat smartcontract dengan memanggil simplestorage
9 contract ExtraStorage is SimpleStorage {
10     function store(uint256 _favoriteNumber) public virtual override {
11         _favoriteNumber = _favoriteNumber + 5;
12     }
13 }
14
```

Transaction log:

```
[call] from: 0x08380De6e701c508545dcfcb03fc6875f50e6dC4 to: SimpleStorage.retrieve(string) data: 0x0ba...00000
call to ExtraStorage.retrieve

[call] from: 0x08380De6e701c508545dcfcb03fc6875f50e6dC4 to: SimpleStorage.retrieve() data: 0x0e6...4ee1
```