



Blockchain

Lessons 3 Remix Storage Factory

Membuat file pada REMIX

hal yang pertama dilakukan adalah membuka website remix(<https://remix.ethereum.org>) lalu kita membuat workspace baru agar bersih dari template yang default diberikan oleh remix.

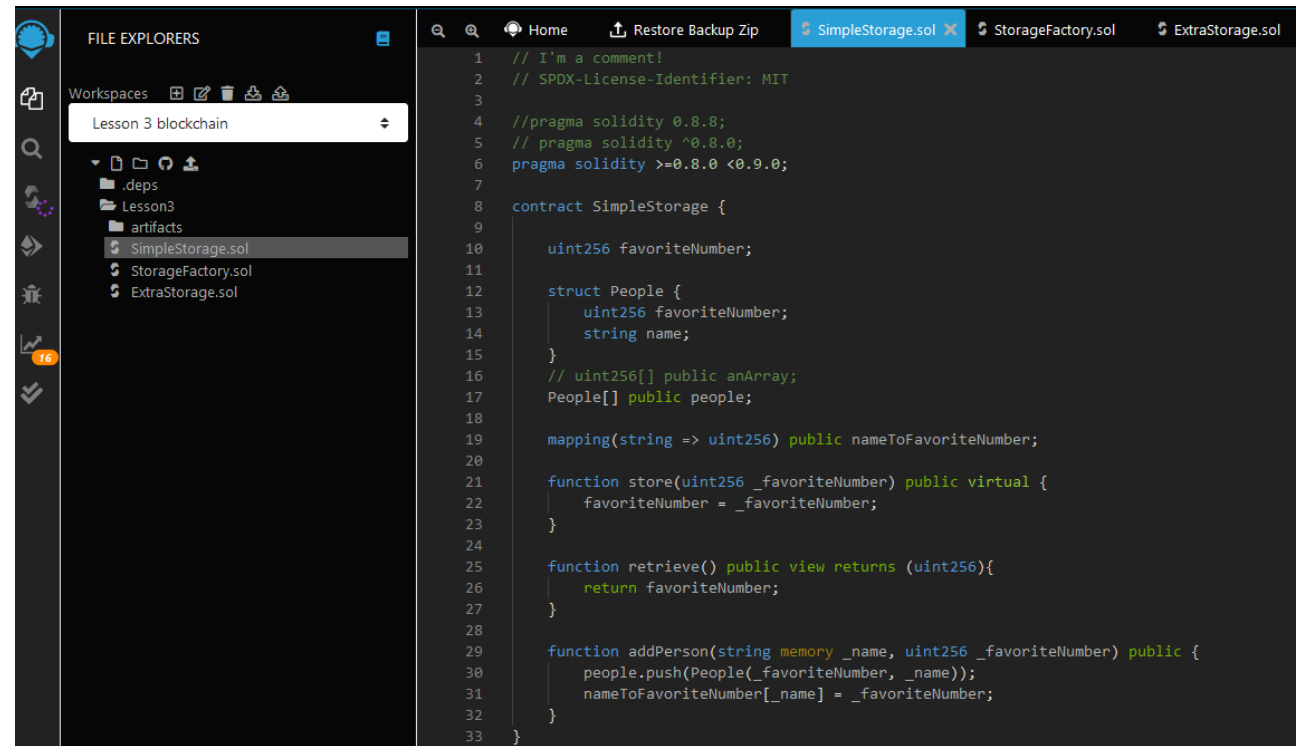
lalu yang kita dapat lakukan adalah membuat tiga file yang berekstensi “.sol” yang di antara lainnya adalah :

SimpleStorage.sol

StorageFactory.sol

ExtraStorage.sol

semua file ini kita akan gunakan seiring waktu.



The screenshot shows the Remix IDE interface. On the left, the 'FILE EXPLORERS' panel displays a workspace named 'Lesson 3 blockchain'. It contains a folder 'Lesson3' with subfolders '.deps' and 'artifacts', and three files: 'SimpleStorage.sol', 'StorageFactory.sol', and 'ExtraStorage.sol'. The 'SimpleStorage.sol' file is selected. The main editor area shows the content of 'SimpleStorage.sol', which includes a comment, a Solidity pragma statement, and a contract definition for 'SimpleStorage'.

```
1 // I'm a comment!  
2 // SPDX-License-Identifier: MIT  
3  
4 //pragma solidity 0.8.8;  
5 // pragma solidity ^0.8.0;  
6 pragma solidity >=0.8.0 <0.9.0;  
7  
8 contract SimpleStorage {  
9  
10     uint256 favoriteNumber;  
11  
12     struct People {  
13         uint256 favoriteNumber;  
14         string name;  
15     }  
16     // uint256[] public anArray;  
17     People[] public people;  
18  
19     mapping(string => uint256) public nameToFavoriteNumber;  
20  
21     function store(uint256 _favoriteNumber) public virtual {  
22         favoriteNumber = _favoriteNumber;  
23     }  
24  
25     function retrieve() public view returns (uint256){  
26         return favoriteNumber;  
27     }  
28  
29     function addPerson(string memory _name, uint256 _favoriteNumber) public {  
30         people.push(People(_favoriteNumber, _name));  
31         nameToFavoriteNumber[_name] = _favoriteNumber;  
32     }  
33 }
```

Mempersiapkan Simple Storage

lalu kita akan mempersiapkan code kita pada SimpleStorage. Pada code disamping saya mengguna compiler solidity lebih dari 0.7.0 hingga kurang dari versi 0.9.0. Lalu kita pun membuat nama kontrak dengan SimpleStorage dengan menyimpan variable uint256, mapping variable, dan fungsi dari smart contract yang kita buat

```
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 disinmpn 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     //Fungsi retrieve yang gunanya adalah mengambil nilai value dari favorite number
28     function retrieve() public view returns (uint256){
29         return favoriteNumber;
30     }
31     //Fungsi addPersin yang gunanya adalah menyimpan data orang baru beserta nomor favoritnya
32     function addPerson(string memory _name, uint256 _favoriteNumber) public {
33         people.push(People(_favoriteNumber, _name));
34         nameToFavoriteNumber[_name] = _favoriteNumber;
35     }
36 }
37
```

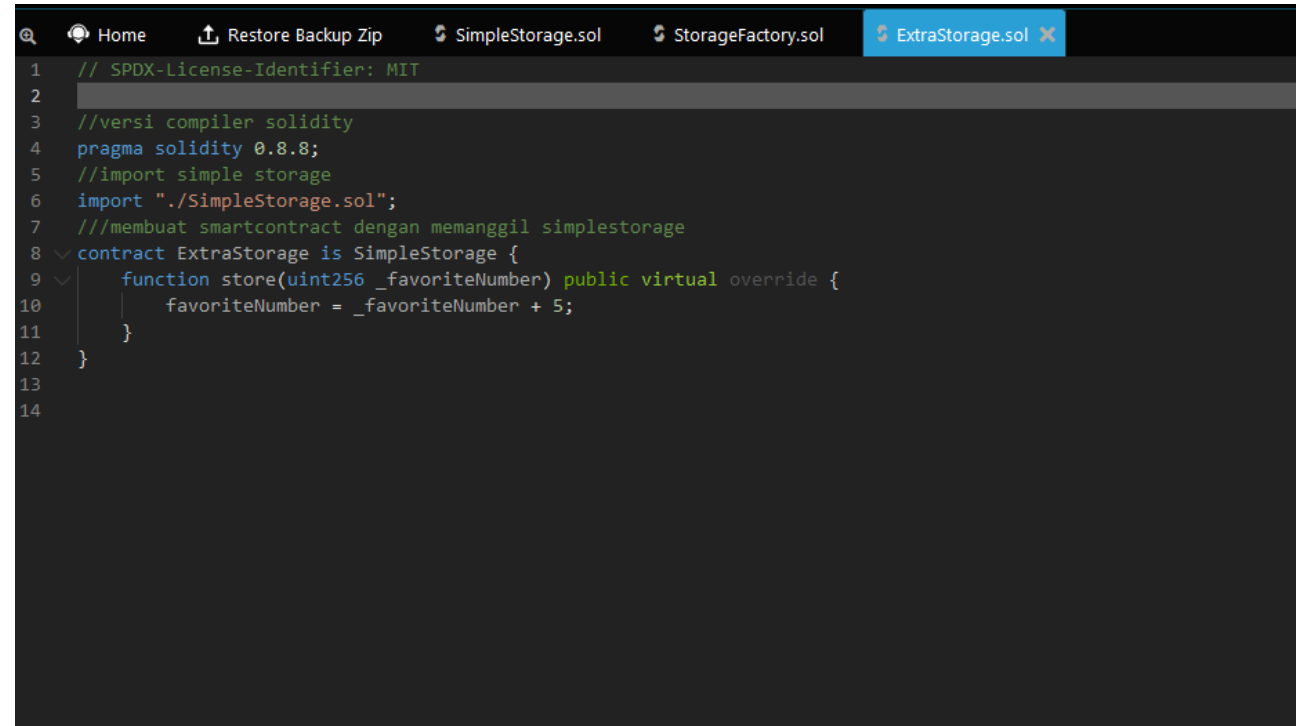
Mempersiapkan Storage Factory

lalu kita akan mempersiapkan code kita pada StorageFactory. Pada code disamping saya mengguna compiler solidity lebih dari lebih dari 0.8.0. Lalu kita pun membuat nama kontrak dengan StorageFactory dengan menyimpan SimpleStorageArray ke SimpleStorage dan bersifat public, selain itu kita membuat fungsi pada kontrak seperti fungsi createsimplestoragecontract yang bersifat public dengan fungsinya adalah 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 nialinya
27     function sfGet(uint256 _simpleStorageIndex) public view returns (uint256) {
28         // return SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).retrieve();
29         return simpleStorageArray[_simpleStorageIndex].retrieve();
30     }
31 }
32
```

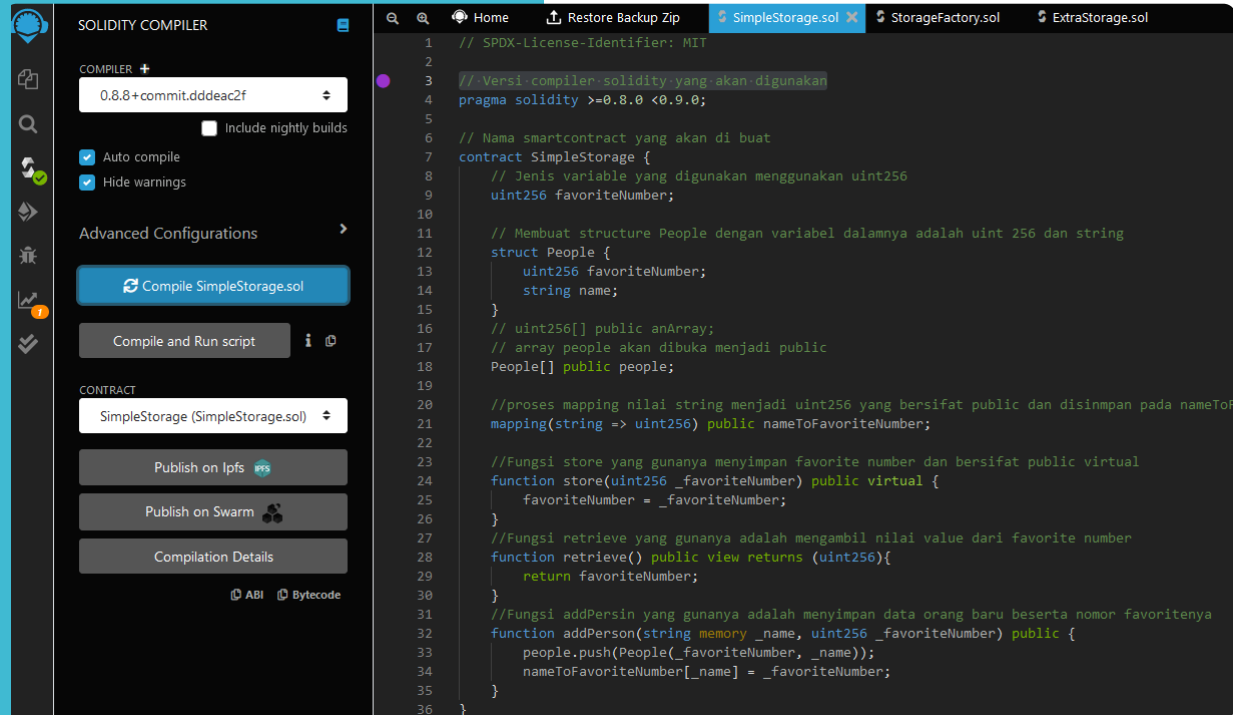
Mempersiapkan Extra Storage

lalu kita akan mempersiapkan code kita pada ExtraStorage. Pada code disamping saya mengguna compiler solidity lebih dari lebih dari 0.8.8. Lalu kita pun membuat nama kontrak dengan ExtraStorage dengan memanggil SimpleStorage terlebih dahulu. lalu pada smart contract nya sendiri akan mengoverride nilai favoritenumbr dengan menambahkan lima pada nilai favoritenumbr sebelumnya



```
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
```

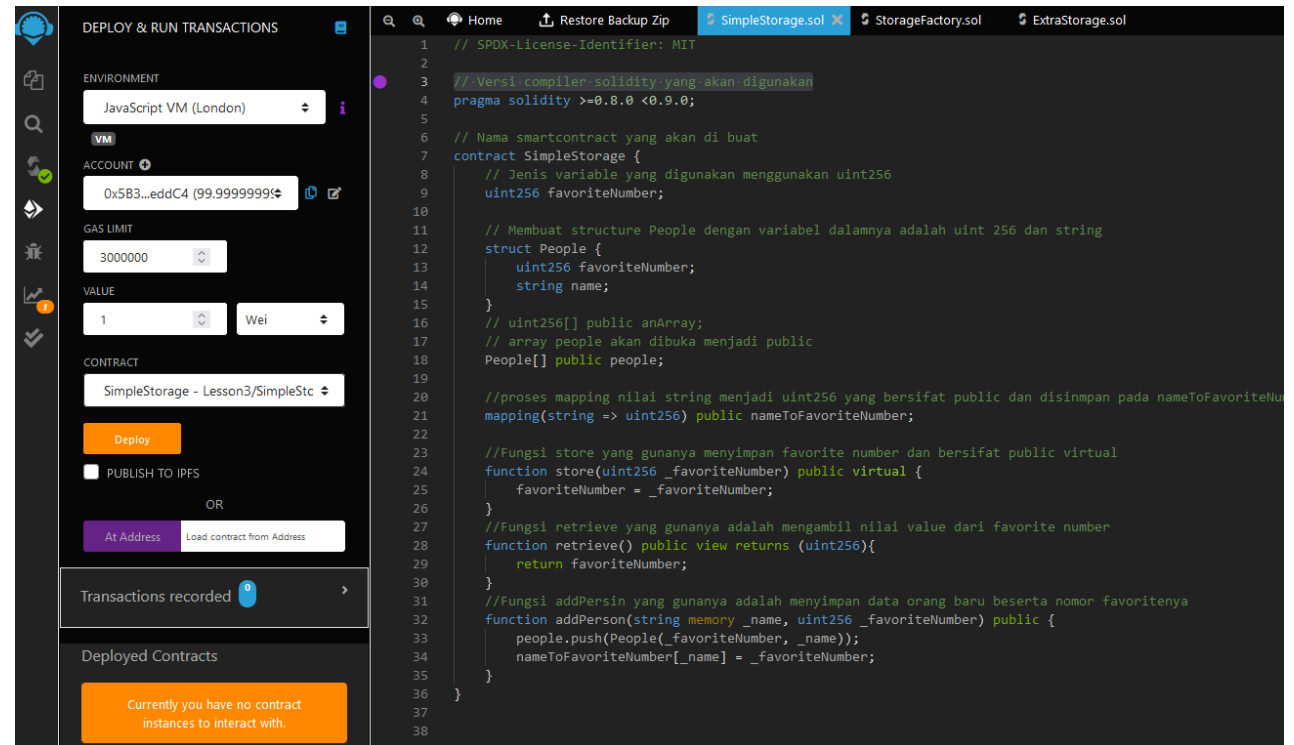
Mengcompile Simple Storage



pada tahapan ini kita akan meng compile terlebih dahulu code kita yang telah dibuat sebelumnya

Mendeploy Simple Storage

pada tahapan ini kita akan mendeploy smart contract kita ke dalam jaringan blockchain tetapi hanya masih pada test network saja. dan pada environment kita dapat menggunakan yang tersedia pada remix, tetapi pada test ini saya menggunakan environment london



The screenshot displays the Remix IDE interface, which is used for developing and deploying smart contracts. The interface is divided into two main sections: the left sidebar for deployment settings and the right pane for the Solidity code editor.

Left Sidebar (Deploy & Run Transactions):

- ENVIRONMENT:** Set to "JavaScript VM (London)".
- ACCOUNT:** Set to "0x5B3...eddC4 (99.99999999)".
- GAS LIMIT:** Set to "3000000".
- VALUE:** Set to "1" Wei.
- CONTRACT:** Set to "SimpleStorage - Lesson3/SimpleStc".
- Deploy Button:** An orange button labeled "Deploy".
- PUBLISH TO IPFS:** A checkbox that is currently unchecked.
- OR:** A separator between the "Deploy" button and the "At Address" option.
- At Address:** A button labeled "At Address" with a sub-label "Load contract from Address".
- Transactions recorded:** A section showing "0" transactions recorded.
- Deployed Contracts:** A section showing "Currently you have no contract instances to interact with."

Right Pane (Solidity Code Editor):

The code editor shows the Solidity code for the "SimpleStorage" contract. The code is as follows:

```
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 addPersin 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

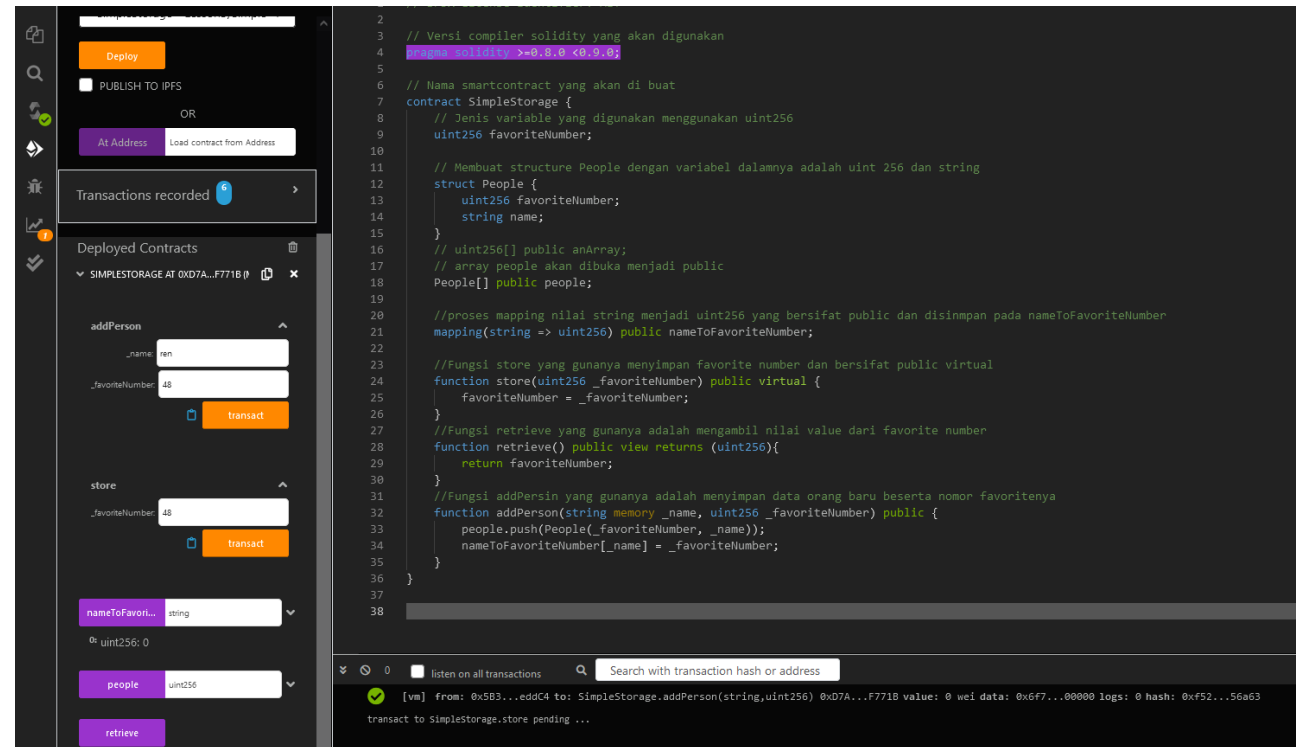
pertama-tama kita akan mencoba menyimpan value nama orang ke dalam blockchain melalui smart contract kita

The screenshot displays the Remix IDE interface. On the left, the 'Deployed Contracts' panel shows the 'SIMPLESTORAGE' contract at address 0xD7A...F7718. The 'addPerson' function is selected, with input fields for '_name' (ren) and '_favoriteNumber' (48). The 'transact' button is visible. Below, the 'store' function is also shown with its input field. The 'retrieve' function is listed at the bottom. The right panel shows the Solidity source code for the 'SimpleStorage' contract, which includes a 'People' struct, an array 'people', and functions for adding, storing, and retrieving data. The bottom status bar indicates a pending transaction to 'SimpleStorage.addPerson'.

```
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 addPersin 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

lalu kita akan mencoba menyimpan value nomor favorite orang tadi ke blockchain melalui smart contract



The screenshot displays the Remix IDE interface for the SimpleStorage smart contract. On the left, the 'Deployed Contracts' panel shows the contract instance at address 0xD7A...F771B. It includes input fields for 'addPerson' (name: 'ren', favoriteNumber: 48) and 'store' (favoriteNumber: 48), each with a 'transact' button. Below these, the 'nameToFavoriteNumber' variable is shown as a string, and the 'people' array is shown as a uint256. A 'retrieve' button is also present.

On the right, the Solidity code for the SimpleStorage contract is visible. The code defines a contract with a uint256 favoriteNumber variable, a People struct, and functions for adding, storing, and retrieving data.

```
1 // SimpleStorage.sol
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 addPensin 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 }
```

The bottom panel shows the transaction log, indicating that the 'addPerson' function was successfully executed on the SimpleStorage contract.

Explore Simple Storage SC

lalu kita coba memanggil hasil return jika kita menginputkan nama orang tadi ke dalam blockchain, apakah nilai yang dibalikin sama seperti nilai yang kita inputkan? jika benar berarti smart contract yang telah dibuat sebelumnya sudah benar

The image shows a web interface for a Simple Storage smart contract and its corresponding Solidity code. The interface is divided into two main sections: a left sidebar for interacting with the contract and a right pane for viewing the code.

Left Sidebar (Contract Interaction):

- Deployed Contracts:** Shows a contract named "SIMPLESTORAGE AT 0xD7A...F771B".
- addPerson:** A form with a text input for "name" (value: "ren") and a numeric input for "favoriteNumber" (value: "48"). A "transact" button is present.
- store:** A form with a numeric input for "favoriteNumber" (value: "48"). A "transact" button is present.
- nameToFavoriteNumber:** A form with a text input for "name" (value: "ren") and a "call" button.
- Output:** Below the "nameToFavoriteNumber" call, it shows "uint256: 48".
- people:** A dropdown menu showing "uint256".
- retrieve:** A button to retrieve data.
- Low level interactions:** A section for viewing transaction details, showing "CALLDATA".

Right Pane (Solidity Code):

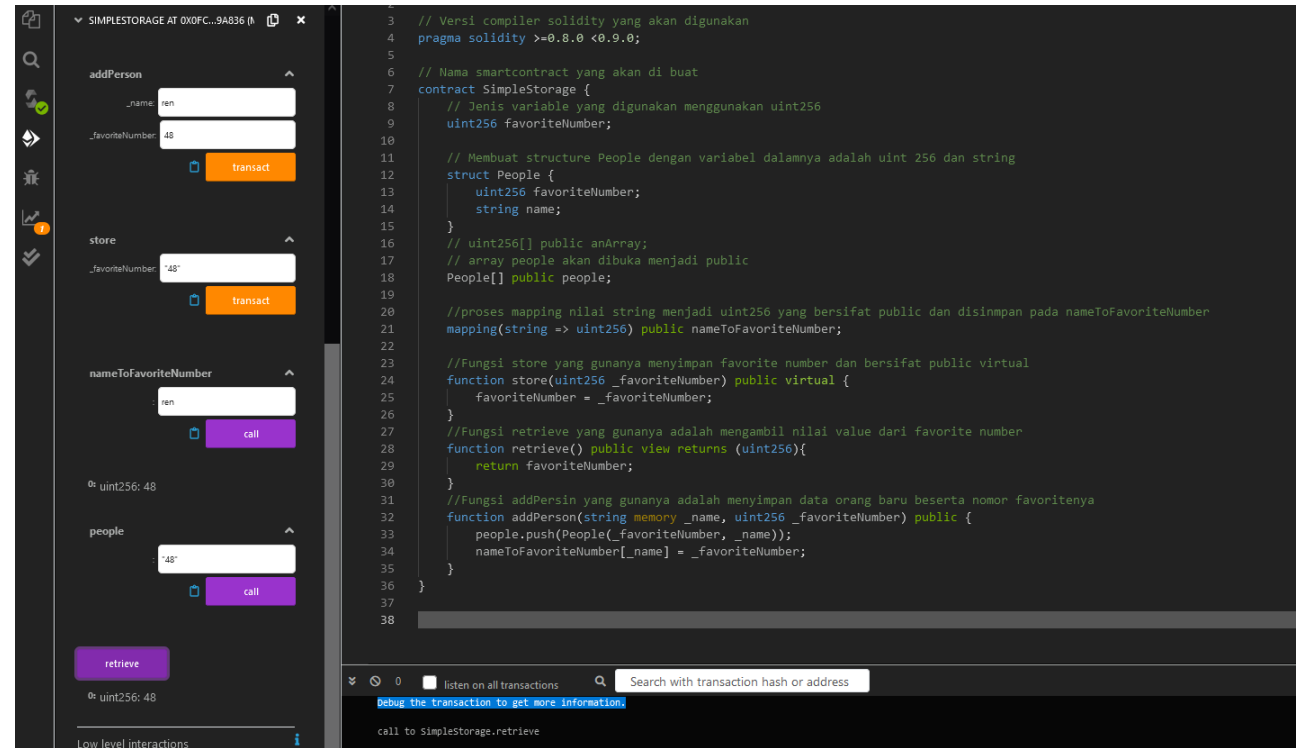
```
1 // SPDX-License-Identifier: MIT
2
3 // Versi compiler solidity yang akan digunakan
4 // solidity >=0.8.0 <0.9.0
5
6 // Nama smartcontract yang akan di buat
7 contract SimpleStorage {
8     // Jenis variable yang digunakan menggunakan uint256
9     uint256 favoriteNumber;
10
11     // Membuat structure 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 addPersin 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 }
```

Bottom Bar (Transaction Log):

- listen on all transactions:** A checkbox to enable transaction monitoring.
- Search with transaction hash or address:** A search bar.
- Transaction Log:** Shows a transaction with the following details:
 - [call] from:** 0x58380a6a701c568545dcfc803fc8075f56beddc4
 - to:** SimpleStorage.nameToFavoriteNumber(string) data: 0x8ba...00000
 - call to:** SimpleStorage.nameToFavoriteNumber

Explore Simple Storage SC

lalu kita coba memanggil hasil return jika kita menginputkan nomor favorite orang, jika outputnya adalah yang kita inputkan sebelumnya maka code kita sudah benar



The image shows a web interface on the left and Solidity code on the right. The web interface is for a smart contract named 'SIMPLESTORAGE AT 0X0FC...9A836'. It has four main sections: 'addPerson' with inputs for '_name' (ren) and '_favoriteNumber' (48) and a 'transact' button; 'store' with an input for '_favoriteNumber' (48) and a 'transact' button; 'nameToFavoriteNumber' with an input for '_name' (ren) and a 'call' button; and 'people' with an input for '_name' (48) and a 'call' button. Below these is a 'retrieve' button. The bottom of the interface shows 'uint256: 48' and 'Low level interactions'. The Solidity code on the right is for a contract named 'SimpleStorage'. It includes a pragma statement for Solidity version, a contract definition with a 'uint256 favoriteNumber' variable, a 'struct People' with 'uint256 favoriteNumber' and 'string name' fields, an array 'People[] public people', a 'mapping' for 'nameToFavoriteNumber', and three functions: 'store' (public virtual), 'retrieve' (public view returns uint256), and 'addPerson' (public).

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

Mendeploy Storage Factory

jika berhasil kita akan melihat smart contract yang kita telah buat pada tab deployed contracts. pada deployed contracts kita pun dapat melihat fitur-fitur yang kita buat pada smart contract

The screenshot displays the Remix IDE interface during the deployment of a smart contract named 'StorageFactory'.

Left Panel (Contract Explorer):

- CONTRACT:** Shows 'SimpleStorage - Lesson3/Simple'.
- Deploy:** A button to deploy the contract.
- PUBLISH TO IPFS:** A checkbox.
- OR:** A separator.
- At Address:** A button to load a contract from a specific address.
- Transactions recorded:** A button to view recorded transactions.
- Deployed Contracts:** A list of deployed contracts, including 'SIMPLESTORAGE AT 0X7EF...8CB47' and 'STORAGEFACTORY AT 0XDA0...42B53'.
- Interactions:** A section for interacting with the deployed contracts, showing functions like 'createSimple...', 'sfStore', 'sfGet', and 'simpleStorage...' with input fields and a 'Transact' button.

Right Panel (Code Editor):

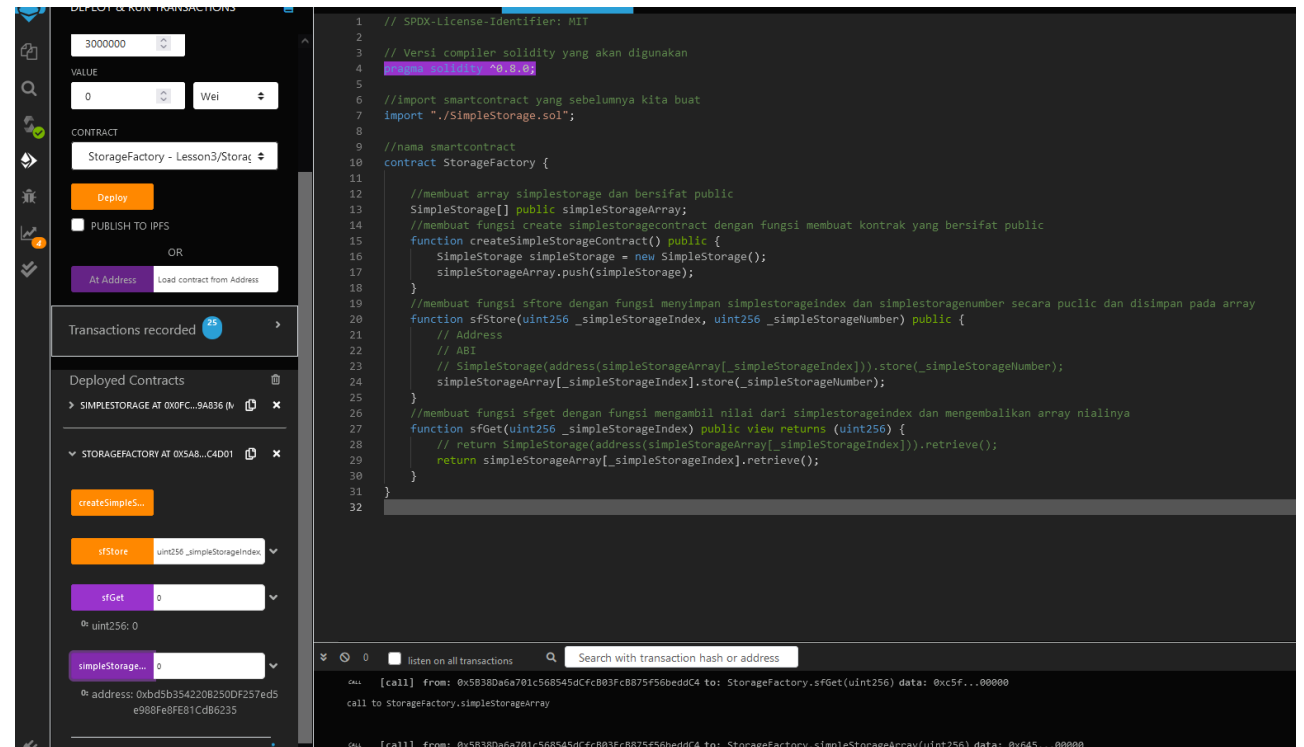
```
2
3 // Versi compiler solidity yang akan digunakan
4 pragma solidity ^0.8.0;
5
6 //import smartcontract yang sebelumnya kita buat
7 import "./SimpleStorage.sol";
8
9 //nama smartcontract
10 contract StorageFactory {
11
12     //membuat array simplestorage dan bersifat public
13     SimpleStorage[] 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
20     //membuat fungsi sfStore dengan fungsi menyimpan simplestorageindex dan simplestoragenumber secara public dan disimpan
21     function sfStore(uint256 _simpleStorageIndex, uint256 _simpleStorageNumber) public {
22         // Address
23         // ABI
24         SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).store(_simpleStorageNumber);
25         simpleStorageArray[_simpleStorageIndex].store(_simpleStorageNumber);
26     }
27
28     //membuat fungsi sfGet dengan fungsi mengambil nilai dari simplestorageindex dan mengembalikan array nialinya
29     function sfGet(uint256 _simpleStorageIndex) public view returns (uint256) {
30         // return SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).retrieve();
31         return simpleStorageArray[_simpleStorageIndex].retrieve();
32     }
33 }
```

Bottom Panel (Console):

- listen on all transactions:** A checkbox.
- Search with transaction hash or address:** A search bar.
- Log:** A log entry showing the successful deployment of the 'StorageFactory' contract: `[vm] from: 0x5B3...eddC4 to: StorageFactory.(constructor) value: 0 wei data: 0x608...80033 logs: 0 hash: 0xd3a...b1ca0 creation of Extrastorage pending...`

Explore Storage Factory

pada smart contract storage factory, kita membuat kode yang dimana kita dapat membuat SC lalu menyimpan kan nilai ke blockchain. pada langkah pertama ini kita mencoba untuk membuat SC dengan index yang dimulai dengan nol



The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the deployment of the 'StorageFactory' contract. The 'VALUE' field is set to 3000000, and the 'CONTRACT' dropdown is set to 'StorageFactory - Lesson3/Storage'. The 'Deploy' button is highlighted. Below the deployment panel, the 'Deployed Contracts' section shows the 'STORAGEFACTORY AT 0x5AB...CAD01' contract. The 'createSimpleStorage' function is visible, with input fields for 'stStore' (uint256 _simpleStorageIndex) and 'stGet' (uint256). The 'simpleStorage...' field is also visible, with an input field for '0'.

On the right, the Solidity code for the 'StorageFactory' contract is shown. The code includes a license identifier, compiler version, and imports. The contract defines a public array 'SimpleStorage' and a public function 'createSimpleStorageContract()'. It also defines a public function 'stStore()' and a public view function 'stGet()'.

```
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 stStore dengan fungsi menyimpan simplestorageindex dan simplestoragenumber secara public dan disimpan pada array
20     function stStore(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 stGet dengan fungsi mengambil nilai dari simplestorageindex dan mengembalikan array nialinya
27     function stGet(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 'TRANSACTIONS' panel shows the execution of the 'stGet' function. The transaction hash is '0x5AB...CAD01'. The transaction details show a call to 'StorageFactory.stGet(uint256)' with data '0x5f...00000'.

Explore Storage Factory

lalu kita dapat menyimpan nilai nomor favorite kita dan disimpan pada index SC tertentu pada blockchain. disini saya menggunakan index nol dan menyimpan nomor favorit delapan belas. jika kita memanggil fungsi retrieve seharusnya kita mendapatkan nilai delapan belas tadi yang kita inputkan sebelumnya.

The screenshot displays the Remix IDE interface. On the left, the 'Deployed Contracts' panel shows the 'StorageFactory' contract at address 0x5A8...C4D01. Below it, the 'Interactions' panel shows the 'createSimpleStorage' function being called with a value of 18. The main editor on the right shows the Solidity code for the 'StorageFactory' contract. The code includes a version comment, an import statement for 'SimpleStorage.sol', and a contract definition with a public array 'simpleStorageArray'. It features a 'createSimpleStorageContract' function that creates a new 'SimpleStorage' instance and pushes it to the array, and an 'sfStore' function that stores a value at a specific index. The bottom panel shows the transaction log with a successful call to 'StorageFactory.sfStore'.

```
// Versi compiler solidity yang akan digunakan
// compiler version: 0.8.0
//import smartcontract yang sebelumnya kita buat
import "./SimpleStorage.sol";

//nama smartcontract
contract StorageFactory {

    //membuat array simplestorage dan bersifat public
    SimpleStorage[] public simpleStorageArray;
    //membuat fungsi create simplestoragecontract dengan fungsi membuat kontrak yang bersifat public
    function createSimpleStorageContract() public {
        SimpleStorage simpleStorage = new SimpleStorage();
        simpleStorageArray.push(simpleStorage);
    }

    //membuat fungsi sfStore dengan fungsi menyimpan simplestorageindex dan simplestoragenumber secara public dan disimpan pada array
    function sfStore(uint256 _simpleStorageIndex, uint256 _simpleStorageNumber) public {
        // ABI
        // SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).store(_simpleStorageNumber);
        simpleStorageArray[_simpleStorageIndex].store(_simpleStorageNumber);
    }

    //membuat fungsi sfGet dengan fungsi mengambil nilai dari simplestorageindex dan mengembalikan array nilainya
    function sfGet(uint256 _simpleStorageIndex) public view returns (uint256) {
        // return SimpleStorage(address(simpleStorageArray[_simpleStorageIndex])).retrieve();
        return simpleStorageArray[_simpleStorageIndex].retrieve();
    }
}
```

Transaction Log:

```
[vm] from: 0x5B3...eddC4 to: StorageFactory.sfStore(uint256,uint256) 0x5A8...C4D01 value: 0 wei data: 0x156...00012 logs: 0 hash: 0xe9c...02c88
call to StorageFactory.sfStore

[call] from: 0x5B380a6a701c56854dCfcB03fc8875F56beddC4 to: StorageFactory.sfGet(uint256) data: 0xc5f...00000
```

Mendeploy Extra Storage

jika berhasil kita akan melihat smart contract yang kita telah buat pada tab deployed contracts. pada deployed contracts kita pun dapat melihat fitur-fitur yang kita buat pada smart contract

The screenshot displays the Remix IDE interface during the deployment of a smart contract. On the left, the 'Deployed Contracts' tab is active, showing a list of contracts. The 'EXTRASTORAGE AT 0x358...D5EE3 (M)' contract is selected, revealing its functions: 'addPerson' (with inputs 'string _name, uint256 _favorite'), 'store' (with input 'uint256 _favoriteNumber'), 'nameOfFavori...' (with input 'string'), 'people' (with input 'uint256'), and 'retrieve'. The 'Deploy' button is visible at the top of the interface.

On the right, the Solidity code for the 'StorageFactory' contract is shown. The code includes a pragma statement for Solidity version 0.8.0, an import for 'SimpleStorage.sol', and a contract definition for 'StorageFactory'. The contract contains a public array 'SimpleStorage[]', a 'createSimpleStorageContract' function, an 'sfStore' function, and an 'sfGet' function.

```
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
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 nialinya
27     function sfGet(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 console shows a successful deployment message: '[vm] from: 0x583...eddC4 to: StorageFactory.(constructor) value: 0 wei data: 0x608...80033 logs: 0 hash: 0xd3a...bica0 creation of Extrastorage pending...'

Explore Extra Storage

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

The screenshot shows a web interface for exploring a smart contract. On the left, there's a sidebar with icons for contract, transactions, and deployed contracts. The main area is divided into two panels. The top panel, titled 'CONTRACT', shows the contract name 'ExtraStorage - Lesson3/ExtraSto' and a 'Deploy' button. Below it, there's a section for 'Transactions recorded' with a blue circle icon and a right arrow. The bottom panel, titled 'Deployed Contracts', lists three contracts: 'SIMPLESTORAGE AT 0x0FC...9A836', 'STORAGEFACTORY AT 0x5AB...C4D01', and 'EXTRASTORAGE AT 0x38C...24C73'. The 'EXTRASTORAGE' contract is expanded, showing four functions: 'addPerson' (returning 69), 'store' (taking 69 as input), 'nameToFavori...' (returning 'ren'), and 'people' (taking 'uint256' as input). There are also buttons for 'retrieve' and 'load contract from Address'. The right panel shows the Solidity code for the contract, which includes a pragma statement for Solidity 0.8.8, an import for 'SimpleStorage.sol', and a contract definition for 'ExtraStorage' that inherits from 'SimpleStorage'. The 'store' function is overridden to increment the 'favoriteNumber' by 5. The bottom status bar shows a green checkmark and a message: '[vm] from: 0x503...eddC4 to: SimpleStorage.store(uint256) 0x38c...24C73 value: 0 wei data: 0x605...00045 logs: 0 hash: 0x6b8...eafa4 call to ExtraStorage.nameToFavoritNumber'.

```
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
```

[vm] from: 0x503...eddC4 to: SimpleStorage.store(uint256) 0x38c...24C73 value: 0 wei data: 0x605...00045 logs: 0 hash: 0x6b8...eafa4 call to ExtraStorage.nameToFavoritNumber

Explore Extra Storage

tetapi jika retrieve nilai dari SC extra storage kita akan mendapatkan nilai baru yaitu 74 yang berasal dari $69 + 5$

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is open, showing the 'EXTRASTORAGE' contract at address 0x38C...24C73. The 'Deployed Contracts' section lists three contracts: SIMPLESTORAGE, STORAGEFACTORY, and EXTRASTORAGE. The EXTRASTORAGE contract is selected, and its functions are visible: 'addPerson', 'store', 'nameToFavoriteNumber', 'people', and 'retrieve'. The 'retrieve' function is highlighted, showing a return value of 'uint256: 74'.

On the right, the 'ExtraStorage.sol' file is open, showing the Solidity code for the contract. The code defines a 'SimpleStorage' contract and an 'ExtraStorage' contract that inherits from 'SimpleStorage'. The 'ExtraStorage' contract has a 'store' function that increments the 'favoriteNumber' by 5 and a 'retrieve' function that returns the 'favoriteNumber'.

```
1 // SPDX-License-Identifier: MIT
2
3 //version 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
```

At the bottom, the 'Log' panel shows a transaction log with the following entry:

```
[call] from: 0x58380a6a701c56854dcfc803fcb875f56beddC4 to: SimpleStorage.nameToFavoriteNumber(string) data: 0x8ba...00000
call to Extrastorage.retrieve
```