Technial Documentation

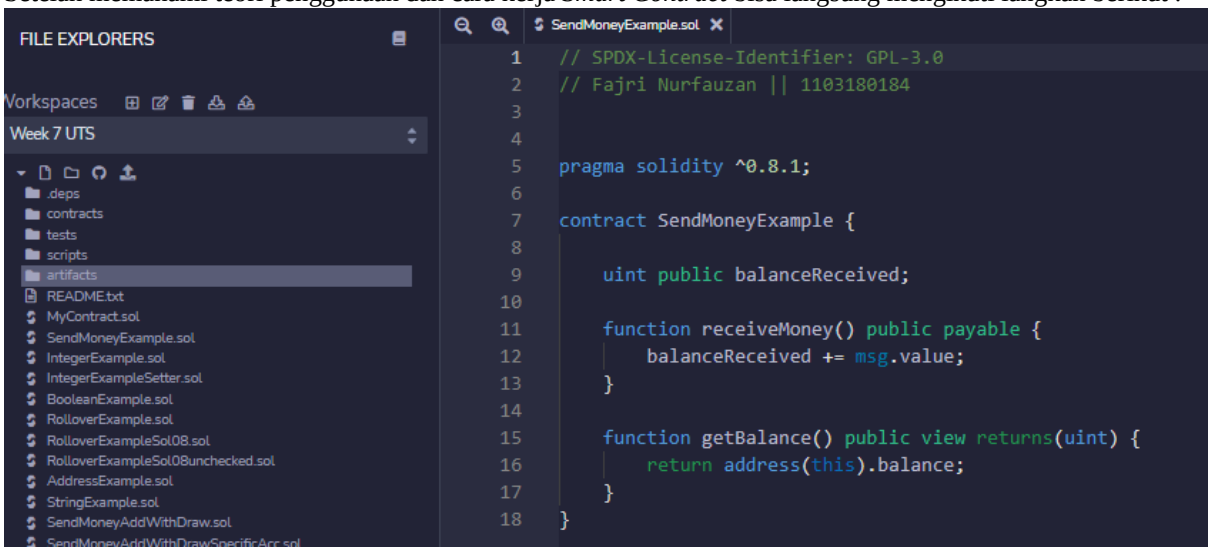
LAB 1 : Deposit/Withdraw Ether

By : Fajri Nurfauzan | 1103180184

Pada lab pertama untuk UTS ini akan mempelajari cara sebuah Kontrak Cerdas (*Smart Contract*) mengelola dananya sendiri. Langkah selanjutnya akan mengirim *Eter* ke Kontrak Cerdas (*Smart Contract*) yang sudah dibuat pada sebelumnya. Kemudian Kontrak Cerdas (*Smart Contract*) akan mengelola *Eter*nya sendiri dan dapat menyampaikannya kepada orang lain. Cara kerja ini sepeti layaknya rekening bank dengan kode pemrograman yang melekat pada sebuah sistem bank tersebut.

Ini juga dapat digunakan untuk *escrow Eter* ke dalam Kontrak Cerdas (*Smart Contract*). Contoh pertama akan dilakukan deposit/penarikan yang sangat sederhana, lalu bagaimana Kontrak Cerdas (*Smart Contract*) dapat mengunci dana menggunakan fungsi penarikan yang diaktifkan oleh waktu.

Setelah memahami teori penggunaan dan cara kerja *Smart Contract* bisa langsung mengikuti langkah berikut :



Buat file remix sederhana seperti berikut dan beri nama SendMoneyExample.sol

Jika sudah kita perlu mengenali fungsi codenya terlebih dahulu, sepeti :

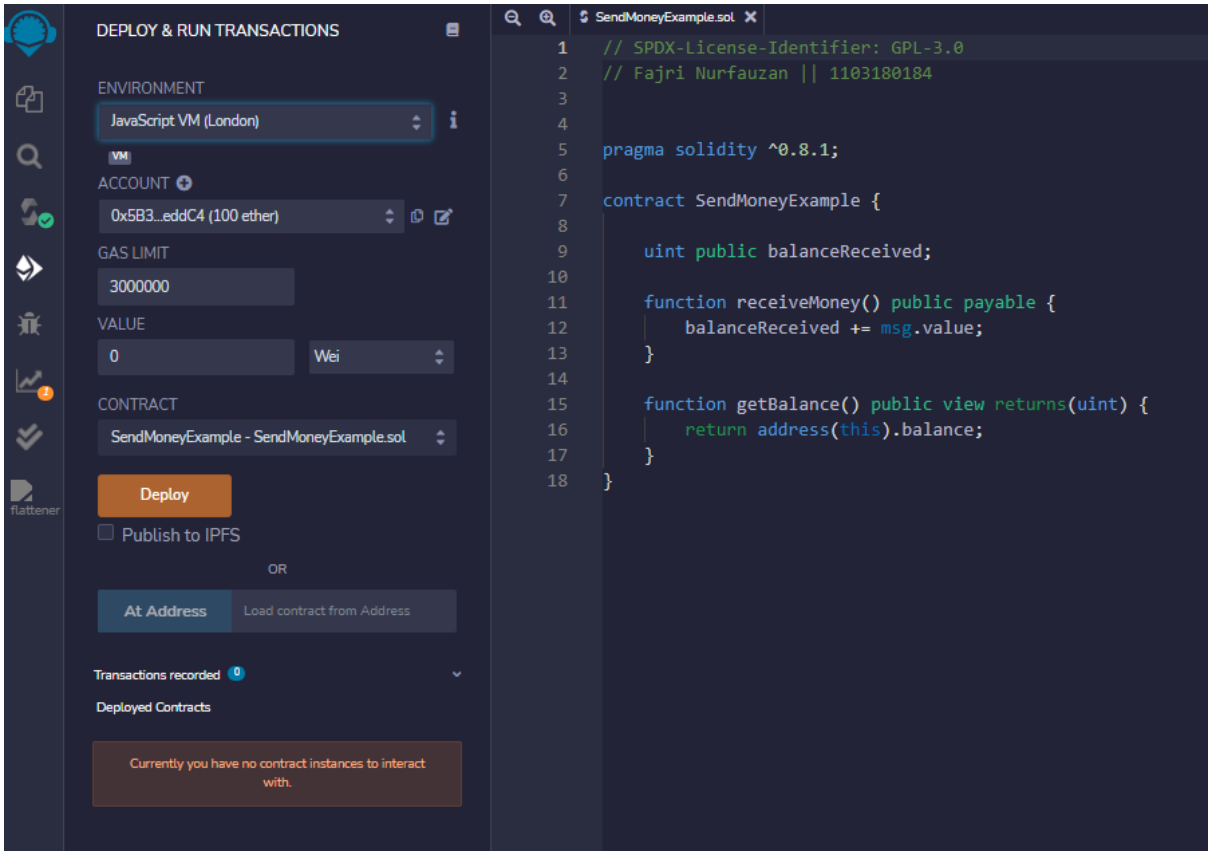
`uint public balanceReceived ;` adalah variabel penyimpanan publik. Variabel publik akan membuat fungsi pengambil secara otomatis di Kepadatan. Jadi kami selalu dapat menanyakan konten variabel ini saat ini.

`balanceReceived += msg.value ;` Objek msg adalah objek global yang selalu ada yang berisi beberapa informasi tentang transaksi yang sedang berlangsung. Dua properti yang paling penting adalah *.value* dan *.sender*. sebelum berisi jumlah Wei yang dulu dikirim ke *Smart Contract*. Yang terakhir berisi alamat yang disebut Kontrak Cerdas (*Smart Contract*).

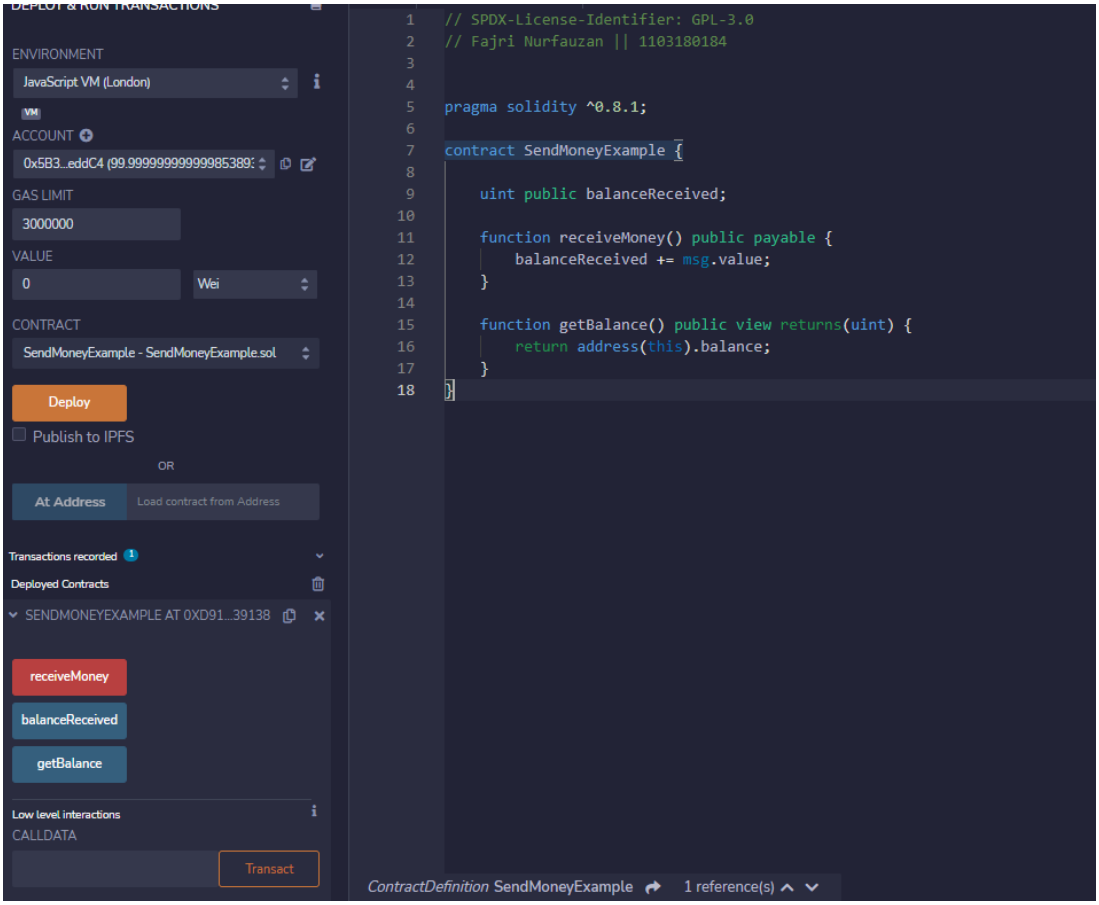
`function getBalance() public view returns(uint) ;` fungsi tampilan adalah fungsi yang tidak mengubah penyimpanan (*Read-only*) dan dapat mengembalikan informasi. Itu tidak perlu ditambang dan hampir gratis.

`address(this).balance ;` Variabel bertipe address selalu memiliki properti bernama .balance yang memberi Anda jumlah eter disimpan di alamat itu. Itu tidak berarti Anda dapat mengaksesnya, itu hanya memberi tahu Anda berapa banyak yang disimpan di sana. Ingat, itu semua informasi publik. `address(this)` mengonversi instans Kontrak Cerdas (*Smart Contract*) menjadi sebuah alamat. Jadi, baris ini pada dasarnya mengembalikan jumlah *Eter* yang disimpan di Kontrak Cerdas (*Smart Contract*) itu sendiri.

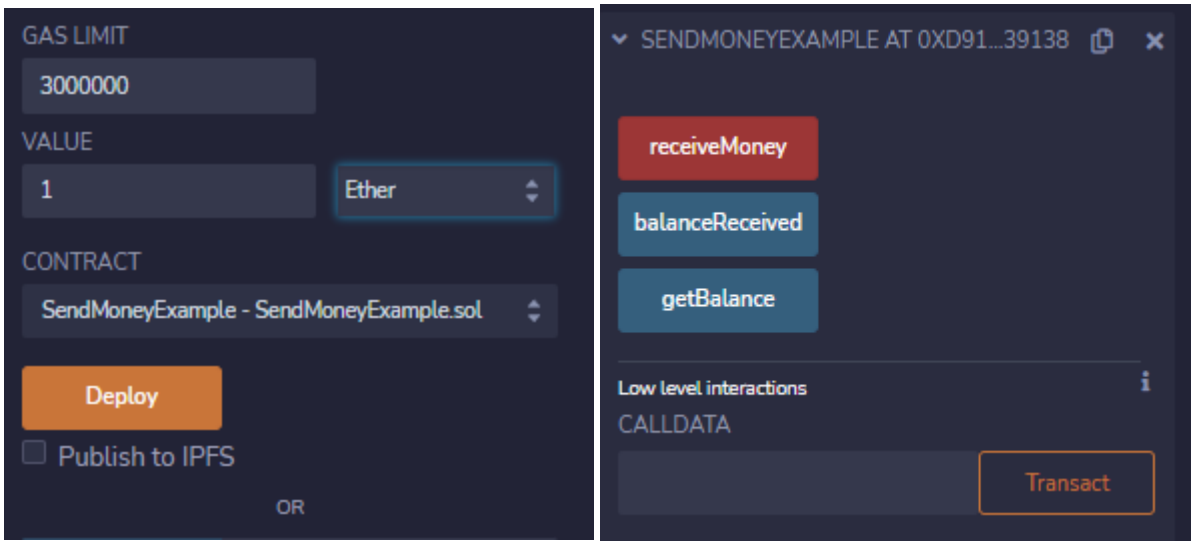
Setelah memahami fungsi diatas maka hal yang perlu dilakukan adalah mendeploynya dengan cara Buka Plugin Deploy and Run Transactions dan terapkan Smart Contract ke dalam JavaScript VM :



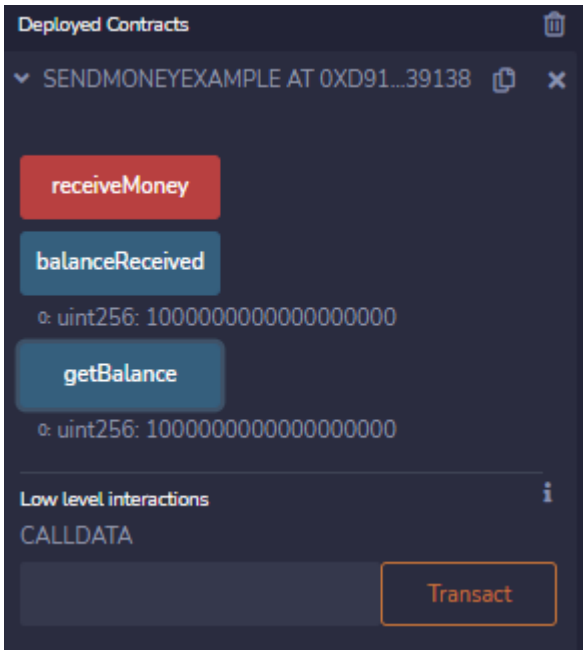
Setelah mengklik deploy akan muncul sebuaha output pada *deployed contracts* dengan tampilan sebagai berikut:



Setelah itu Isi Value dengan 1 dan ubah mata uangnya menjadi *Ether* jika sudah langsung klik *receive money*

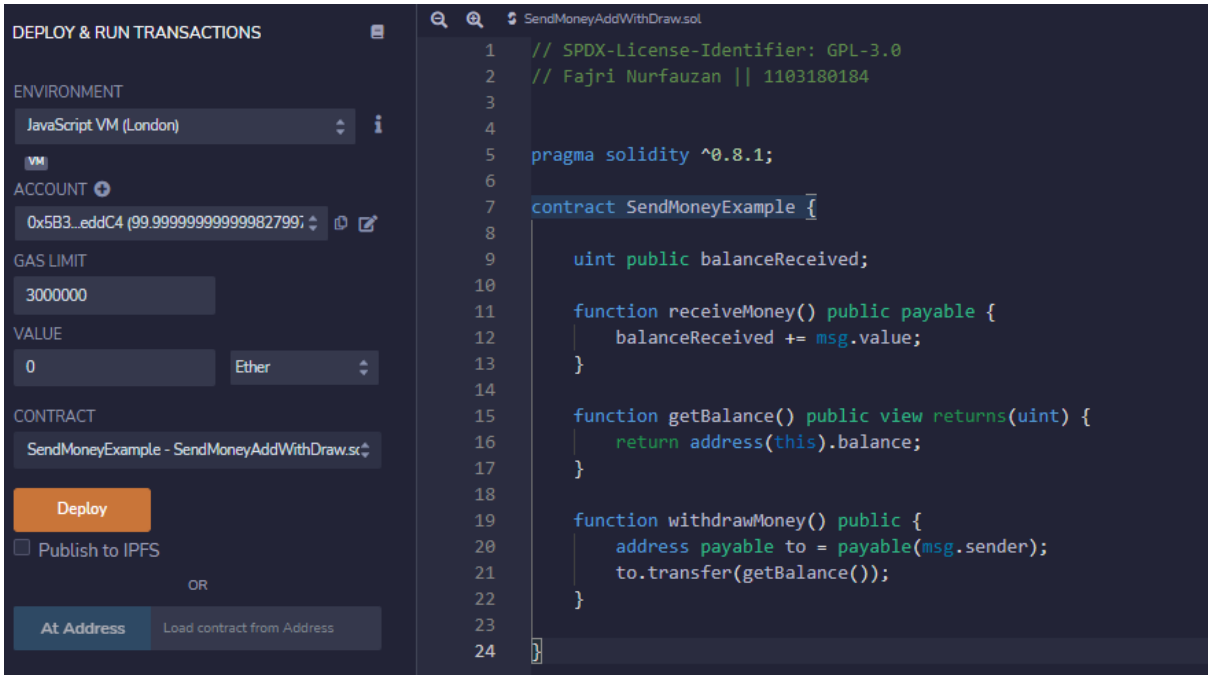


Jika sudah disini akan melakukan pengecekan *balance* bisa mengklik tombol *balanceReceived* dan *getBalance*, dari hasil nilai dari kedua funsi tersebut harus memiliki nilai yang sama seperti Digambar ini :

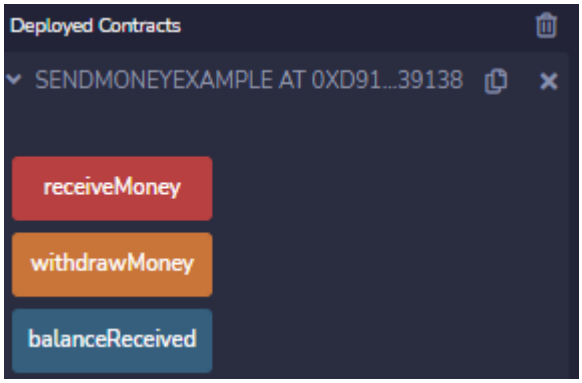


Langkah selanjutnya kita coba menambahkan fungsi `withdrawMoney()` dibawah `getbalace()`:

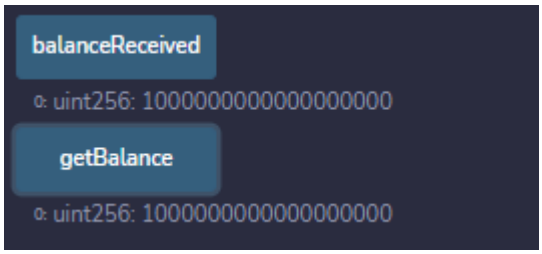
```
function withdrawMoney() public {
        address payable to = payable(msg.sender);
        to.transfer(getBalance());
    }
```



Jika sudah lakukan deploy Kembali dan lihat output di *Deployed Contract* dan lakukan Langkah sebelumnya :



Dan cek Kembali *balancenya*

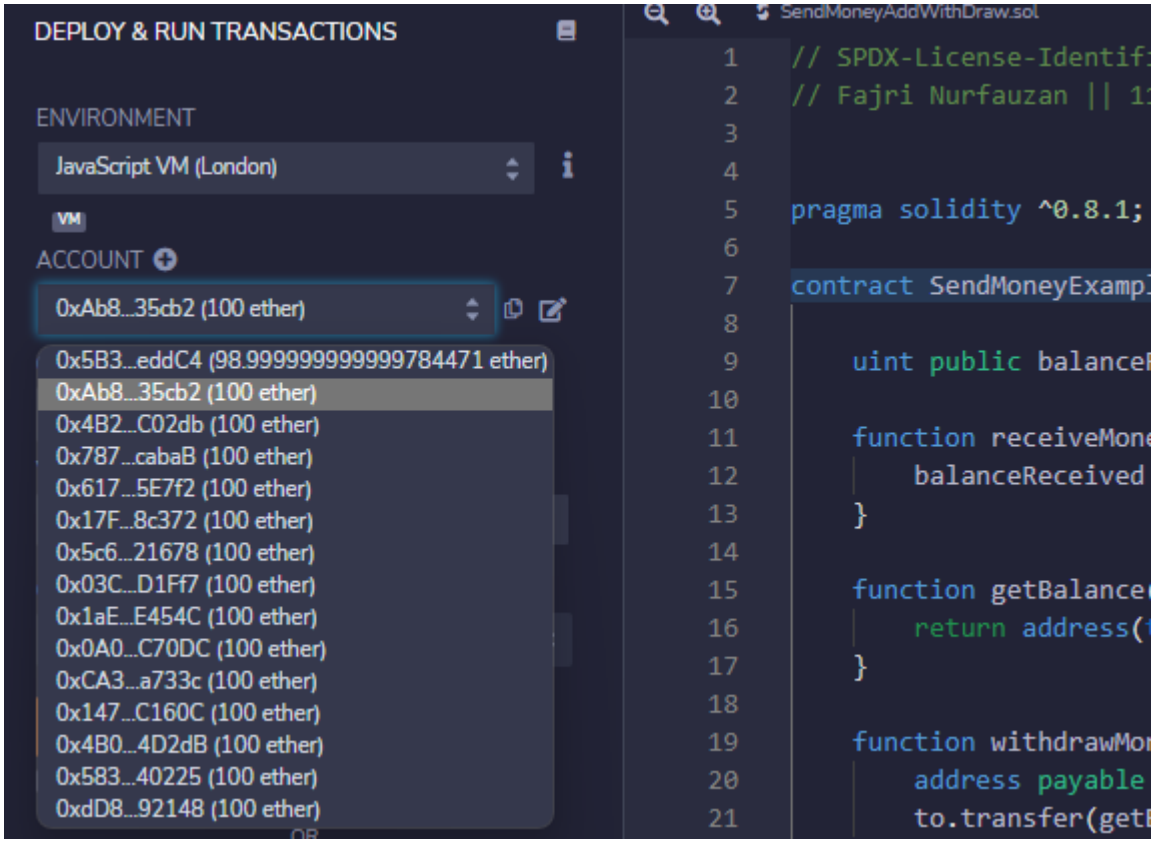


Saldo yang ditampilkan harus 1 Eter lagi: Bukan 1 Eter?

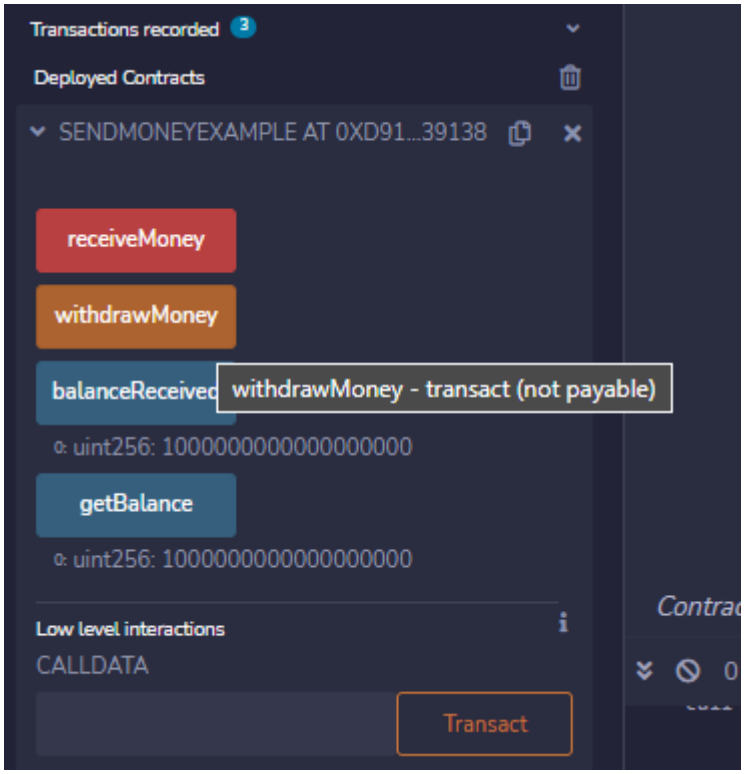
Jika saldo 0, maka periksa kembali kolom nilai

Jika saldo adalah 2 Eter, periksa kembali Instance kontrak yang berinteraksi!

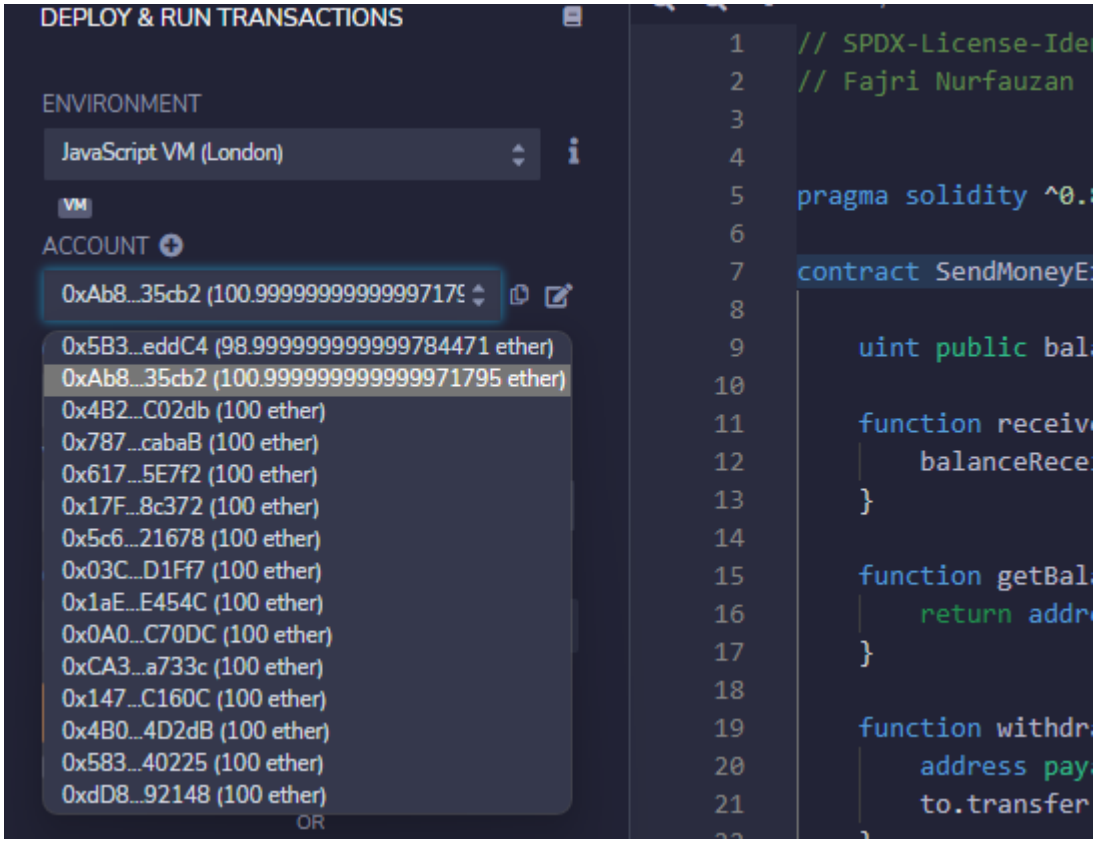
Sekarang saatnya kita menggunakan fungsi baru yang sudah dibuat dan akan mencoba ke akun yang berbeda dengan cara mengklik dropdown pada account !



Dan klik *withdraw money* pada *Deployed Contract*



Dan BOOOM lihat hasilnya pada account ke dua



Apa yang dapat diamati di sini adalah konsep "Gas" di Ethereum Blockchain. Setiap transaksi di Ethereum sedikit mahal. Dan itu tidak berbeda di sini pada rantai simulasi. Prinsip yang sama berlaku.

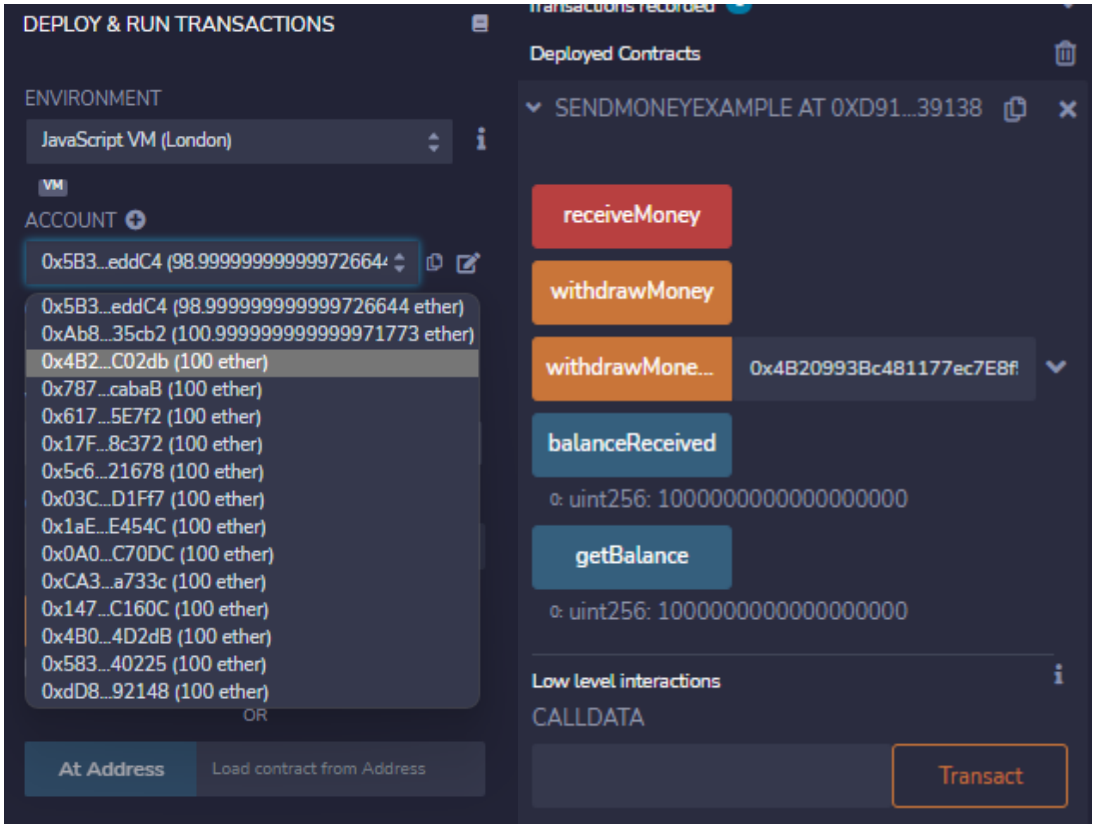
Selanjutnya disini akan mencoba untuk mengirim dana ke rekening atau account tertentu dengan menambahkan funsi berikut :

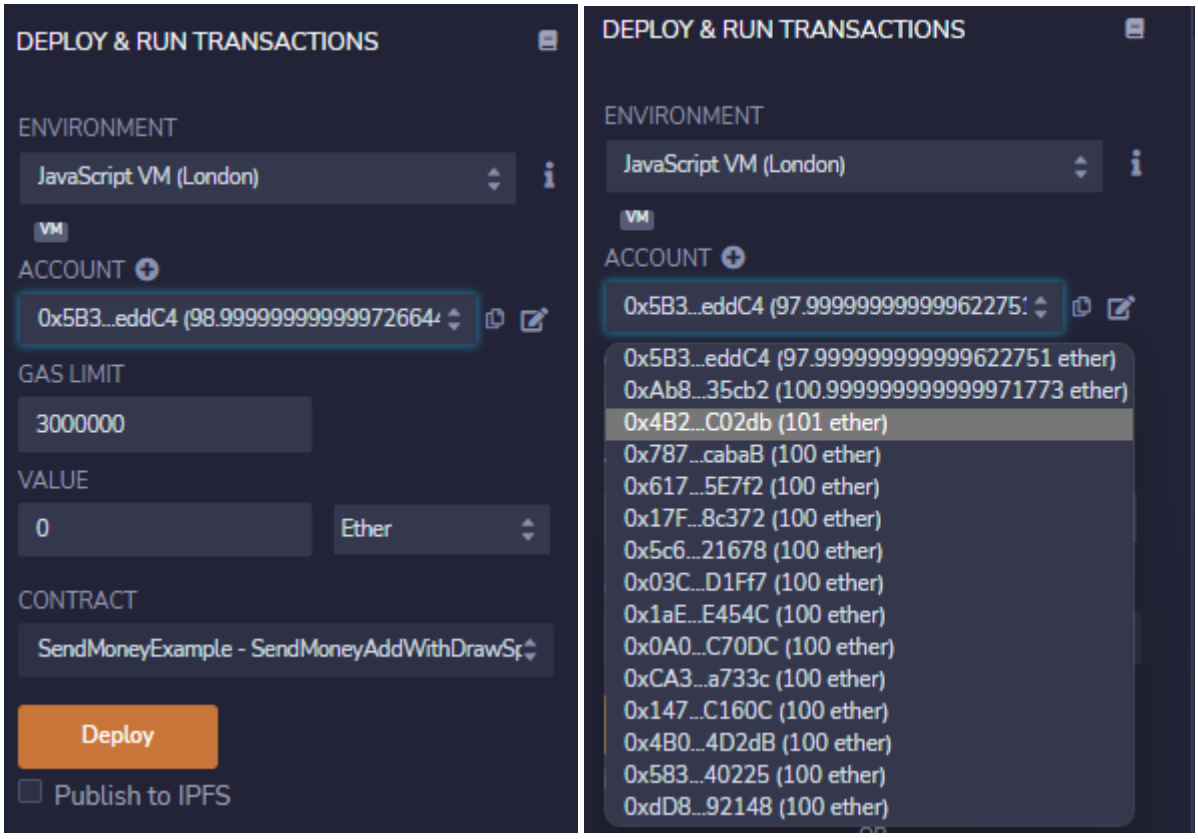
```
function withdrawMoneyTo(address payable _to) public {

    _to.transfer(getBalance());

}
```

SendMoneyAddWithDrawSpecificAcc.sol

```
// SPDX-License-Identifier: GPL-3.0
// Fajri Nurfauzan || 1103180184


pragma solidity ^0.8.1;

contract SendMoneyExample {

    uint public balanceReceived;

    function receiveMoney() public payable {
        balanceReceived += msg.value;
    }

    function getBalance() public view returns(uint) {
        return address(this).balance;
    }

    function withdrawMoney() public {
        address payable to = payable(msg.sender);
        to.transfer(getBalance());
    }


    function withdrawMoneyTo(address payable _to) public {
        _to.transfer(getBalance());
    }
}
```

Dan pilih account ke 3 lalu copy Nomor account tersebut setelah itu simpan di "withdrawMoneyTo" dan kembalikan account ke account 1:





Dan Boom langsung mendapatkan 1 ether penuh karena biaya gas dibayarkan oleh account 1