Nama : Putri Mellia Zahrani

NIM : 1103190143

**LAB 2 : Shared Wallet**

1. **Project Shared Wallet**
2. **Real-World Use-Case for this Project**

Kasus nyata penggunaan projek ini adalah :

* Tunjangan untuk anak-anak per hari/minggu/bulan untuk dapat membelanjakan sejumlah dana tertentu.
* Memberi tunjangan karyawan untuk biaya perjalanan
* Bisnis memberi kelonggaran untuk membelanjakan anggaran tertentu.

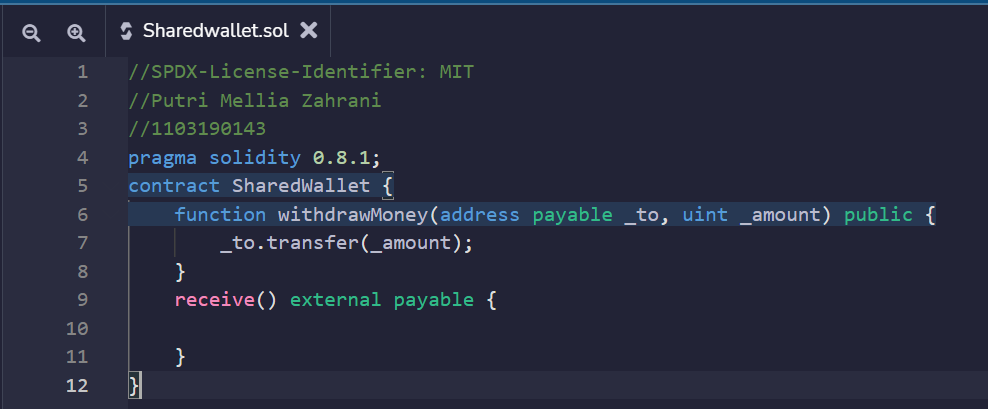
1. **Development Goal**

Tujuan pengembangan :

* Memiliki Smart Contract dompet on-chain
* Wallet Contract dapat menyimpan dana dan memungkinkan pengguna menarik kembali
* Dapat memberikan tunjangan ke alamat pengguna tertentu lainnya
* Batasi fungsi untuk peran pengguna tertentu (pemilik, pengguna)
* Menggunakan kembali Smart Contract yang ada yang sudah diaudit secara maksimal.

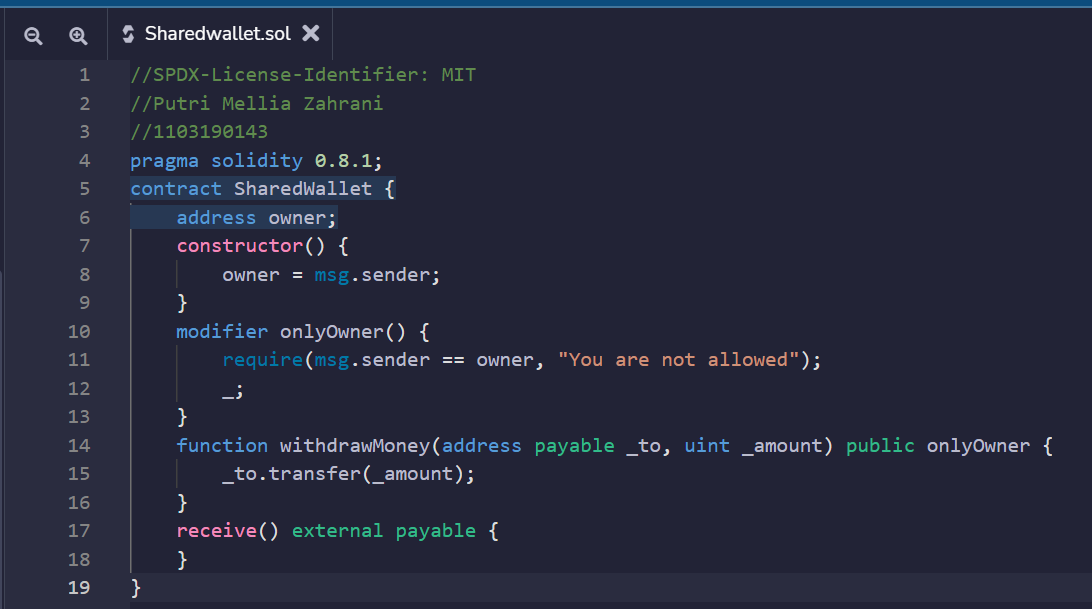
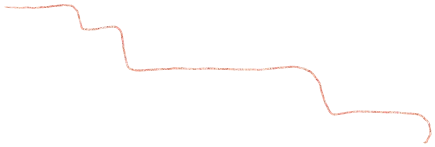
1. **Define the Basic Smart Contract**

Kode dibawah ini merupakan basic Smart contract. Smart contract ini dapat menerima ether dan bisa juga melakukan withdraw ether. Namun, tidak begitu berguna.



1. **Permissions : Allow only the Owner to Withdraw Ether**

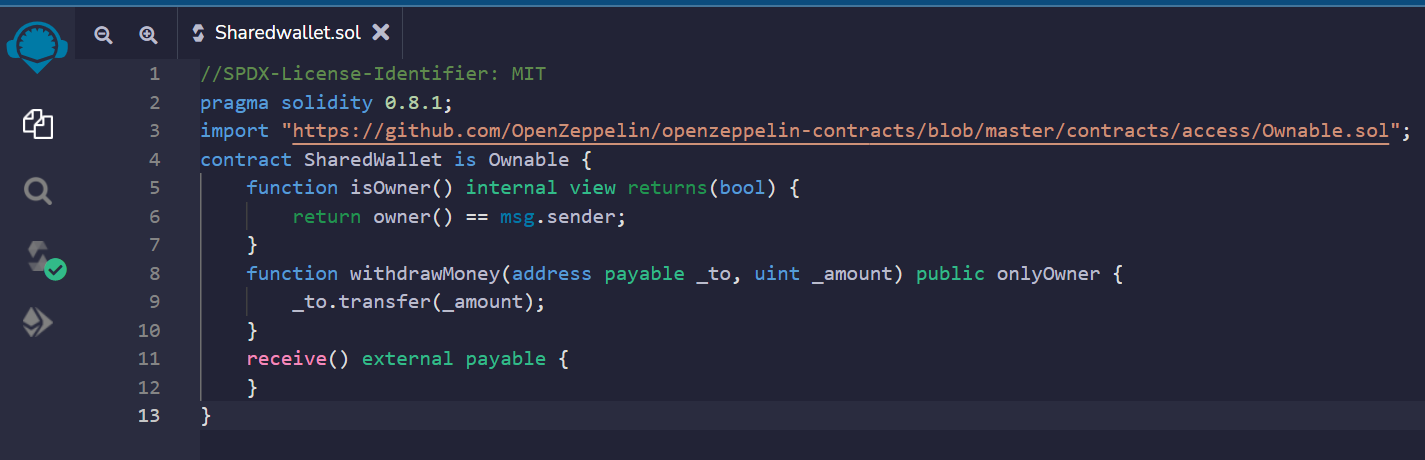
Pada langkah ini kita membatasi penarikan kepada pemilik dari wallet. Pemilik adalah pengguna yang mengembangkan Smart Contract.



Dapat kita lihat bahwa kita juga menambahkan modifier “onlyOwner” ke dalam fungsi “withdrawMoney”.

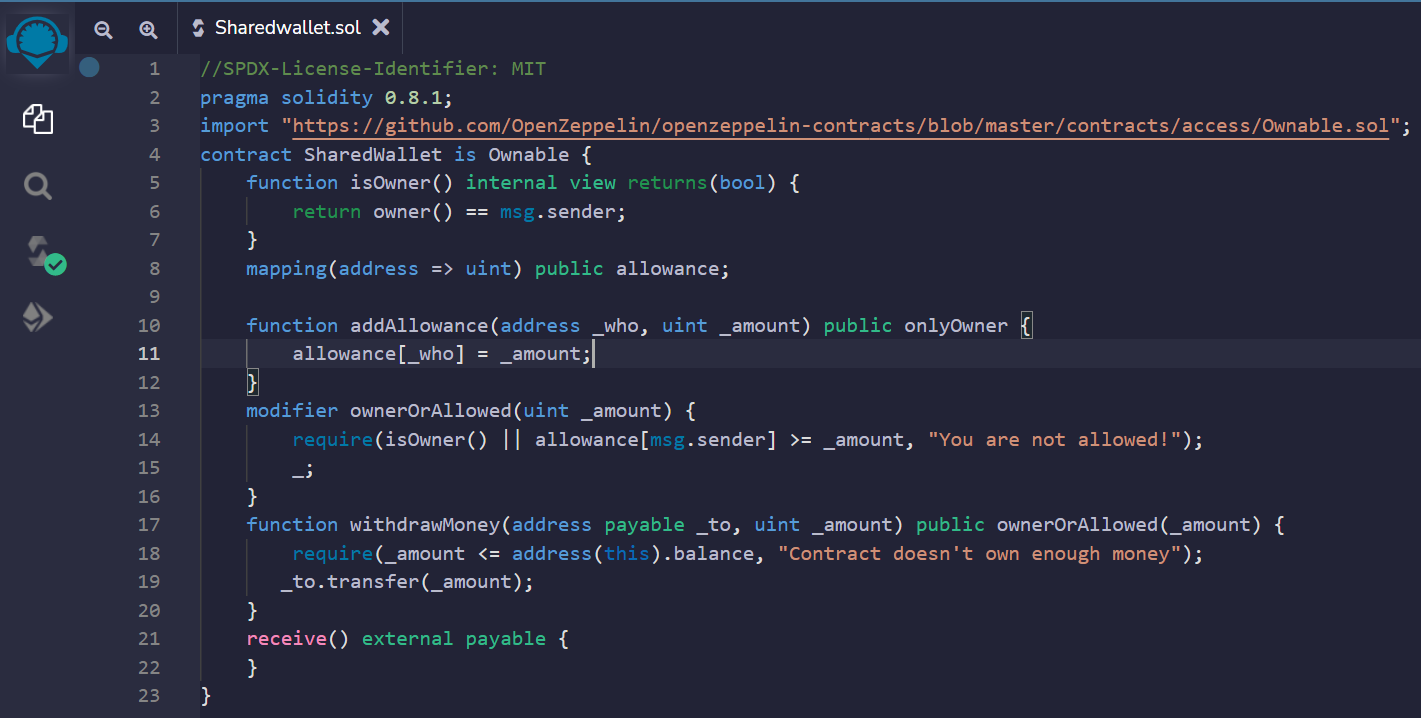
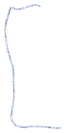
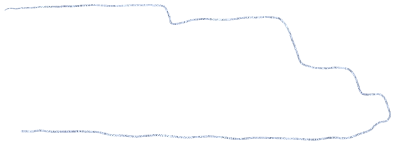
1. **Use Re-Useble Smart Contract from OpenZeppelin**

Memiliki logika owner secara langsung dalam satu Smart Contract tidak mudah untuk diaudit. Mari kita uraikan menjadi bagian-bagian yang lebih kecil dan gunakan kembali Smart Contract yang sudah diaudit dari OpenZeppelin untuk itu. Kontrak OpenZeppelin terbaru tidak memiliki isOwner() fungsi lagi, jadi kita harus membuat sendiri. Perhatikan bahwa owner() adalah fungsi dari kontrak Ownable.sol.



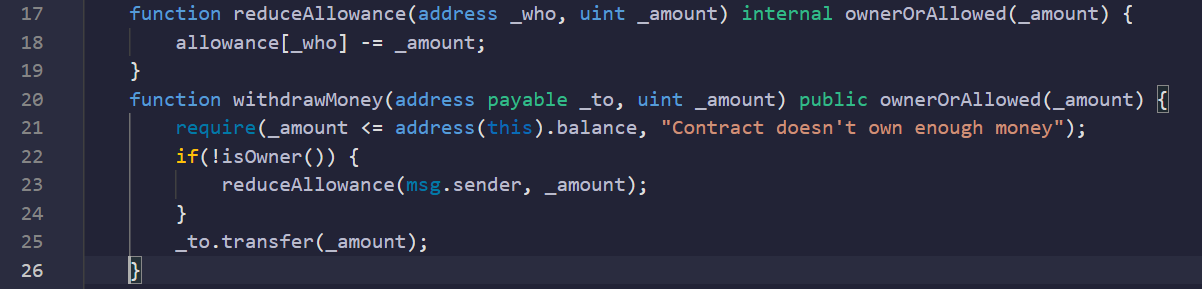
1. **Permissions : Add Allowances for External Roles**

Pada langkah ini kita menambahkan mapping sehingga kita dapat menyimpan alamat=>uint amounts. Ini akan seperti array yang menyimpan alamat [0x123456..] ke nomor tertentu. Sehingga, kita tahu berapa banyak orang yang bisa melakukan penarikan atau withdraw. Kita juga menambahkan modifier baru yang memeriksa apakah owner atau hanya seseorang dengan allowance (tunjangan).



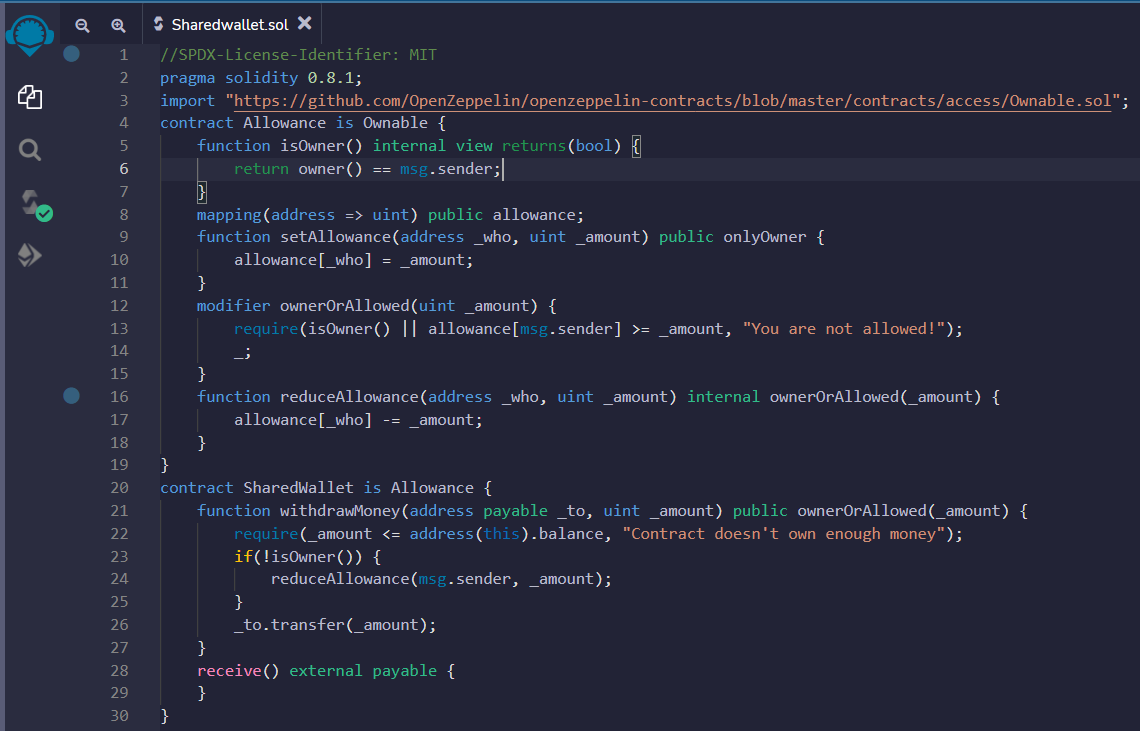
1. **Improve/Fix Allowance to avoid Double-Spending**

Tanpa mengurangi tunjangan penarikan, seseorang dapat terus menerus menarik jumlah yang sama berulang-ulang. Kami harus mengurangi tunjangan untuk semua orang selain pemilik.



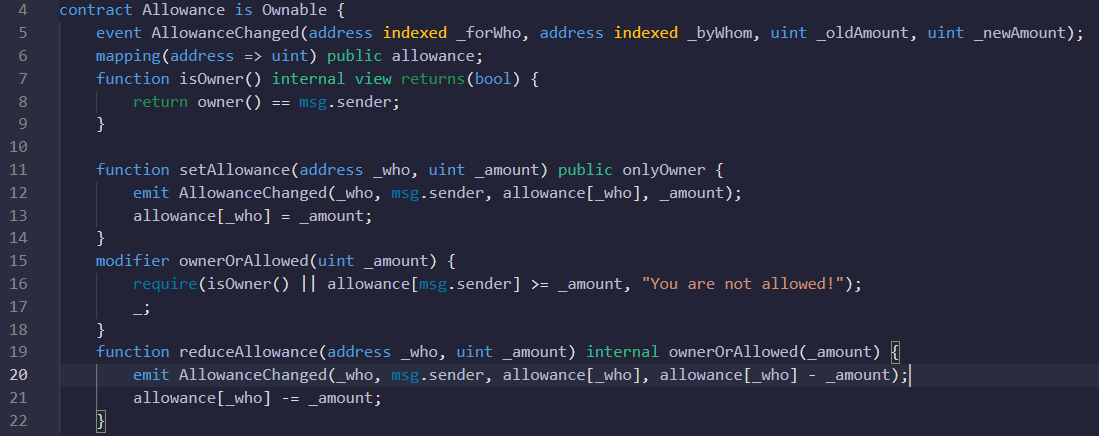
1. **Improve Smart Contract Structure**

Sekarang kita sudah mengetahui fungsional dasar, kita dapat membuat Smart Contract yang berbeda. Untuk membuatnya lebih mudah dibaca, kita dapat membagi 2 fungsionalitas menjadi 2 Smart Contract yang berbeda. Perhatikan bahwa karena Allowance adalah Milik, dan SharedWallet adalah Allowance, oleh karena itu berdasarkan properti komutatif, SharedWallet juga dapat dimiliki.

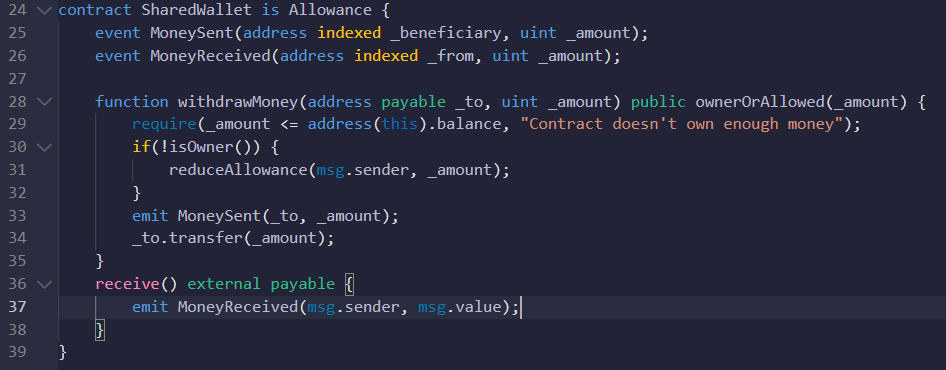


Kedua contract masih tetap berada pada file yang sama sehingga kita tidak perlu melakukan import apapun.

1. **Add Event in the Allowances Smart Contract**



1. **Add Events in the SharedWallet Smart Contract**



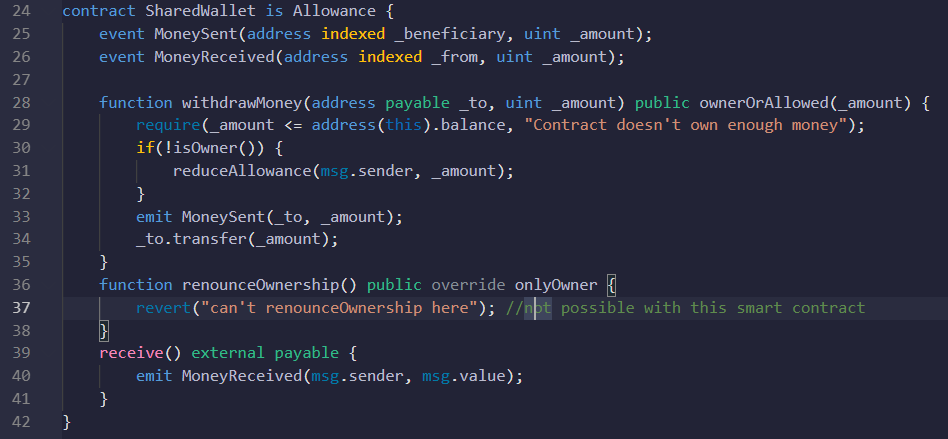
1. **Add the SafeMath Library safeguard Mathematical Operations**

Operasi aritmatika dalam Solidity wrap pada overflow. Ini dapat dengan mudah menghasilkan bug, karena programmer biasanya berasumsi bahwa overflow menimbulkan kesalahan, yang merupakan perilaku standar dalam bahasa pemrograman tingkat tinggi. SafeMath memulihkan intuisi ini dengan mengembalikan transaksi saat operasi overflow.

Pada Solidity dengan update terbaru variable tipe integer tidak dapat mengalami overflow lagi. Sehingga tidak ada bug overflow pada Solidity versi terbaru.

1. **Remove the Renounce Ownership functionality**

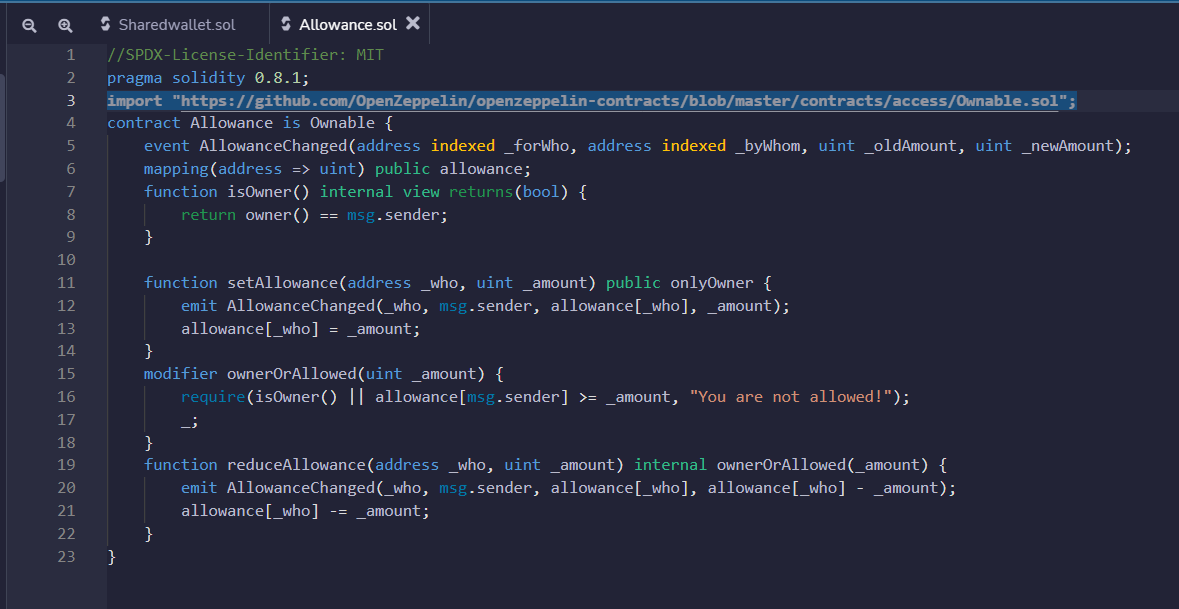
Hapus fungsi untuk menghapus owner. Kami hanya menghentikan ini dengan pengembalian. Tambahkan fungsi berikut ke SharedWallet:



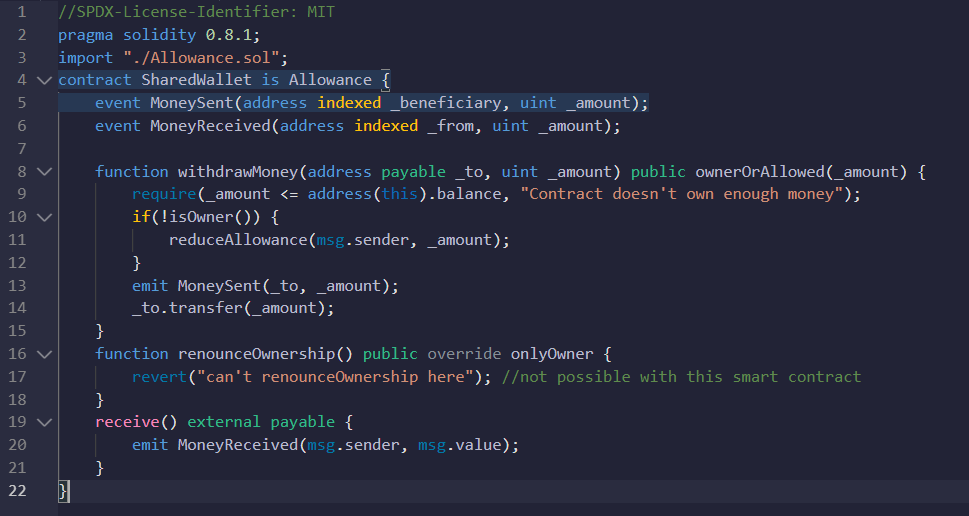
1. **Move the Smart Contract into separate File**

Langkah terakhir, pindahkan Smart Contract ke dalam file terpisah dan gunakan fungsi impor:

1. Allowance.sol



1. Sharedwallet.sol



Pada saat melakukan running jangan lupa untuk memilih akun Smart Contract yang benar.

