**Technial Documentation**

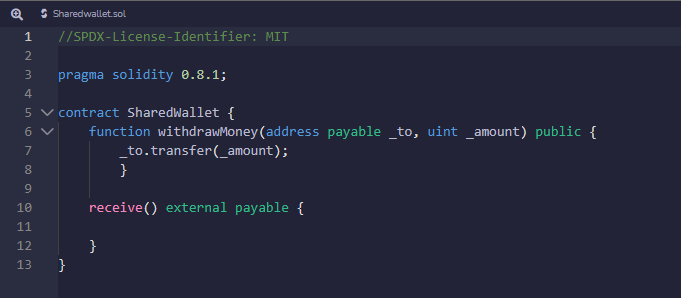
**LAB 2 : Shared Wallet**

By : Fajri Nurfauzan | 1103180184

Pada lab kedua untuk UTS ini akan mempelajari cara sebuah Wallet yang dapat menyimpan dana dan memungkinan penggunanya untuk menarik kembali dananya dengan contoh kasus seperti :

* Tunjangan Anak per hari/minggu/bulan untuk dapat membelanjakan dana dalam jumlah tertentu.
* Majikan memberi karyawan tunjangan untuk biaya perjalanan mereka.
* Bisnis memberi kontraktor tunjangan untuk menghabiskan anggaran tertentu.

Setelah memahami teori penggunaan dan contoh studi kasus bisa langsung mengikuti langkah berikut :



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

Ini adalah kontrak pintar (*Smart Contract*) yang sangat mendasar yang dapat menerima Eter dan dimungkinkan untuk menarik Eter, tetapi secara keseluruhan, sebenarnya ini tidak terlalu berguna atau cukup belum digunakan secara kasus.



Pada langkah ini dapat dilihat code akan membatasi penarikan kepada pemilik dompet. Bagaimana cara menentukan pemiliknya? Itu adalah dengan pengguna yang menyebarkan kontrak pintar (*Smart Contract*).

Ketahuilah bahwa code yang sudah diketik perlu menambahkan pengubah "onlyOwner" ke fungsi withdrawMoney!



Untuk memiliki logika seperti pemilik secara langsung dalam satu kontrak pintar (*Smart Contract*) merupakan hal yang tidak mudah untuk diaudit. Maka dari itu perlu diuraikan menjadi bagian-bagian yang lebih kecil dan akan digunakan kembali kontrak pintar (*Smart Contract*) yang sudah diaudit dari OpenZeppelin untuk itu. Kontrak OpenZeppelin terbaru tidak memiliki

isOwner() fungsi lagi, jadi kita harus membuat sendiri. Perhatikan bahwa pemilik() adalah fungsi dari kontrak Ownable.sol.

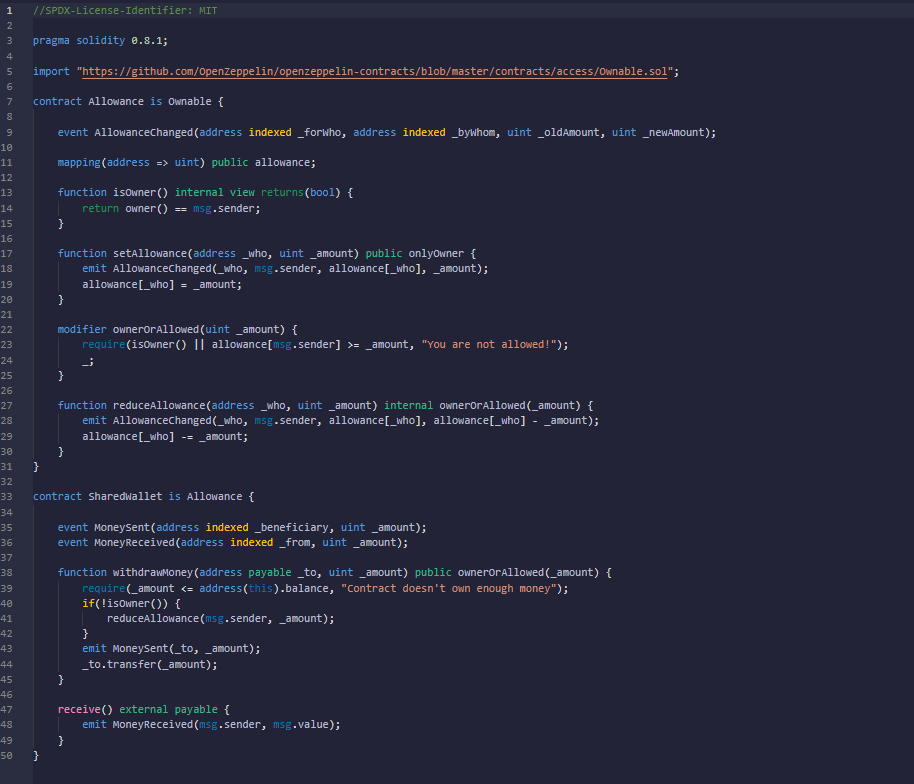


Pada langkah ini perlu menambahkan pemetaan sehingga code dapat menyimpan alamat => jumlah uint. Code ini akan seperti array yang menyimpan [0x123546...] alamat, ke nomor tertentu. Jadi, code selalu tahu berapa banyak yang bisa ditarik seseorang. dan juga menambahkan yang baru modifikator yang akan memeriksa: Apakah pemiliknya sendiri atau hanya seseorang dengan uang saku?

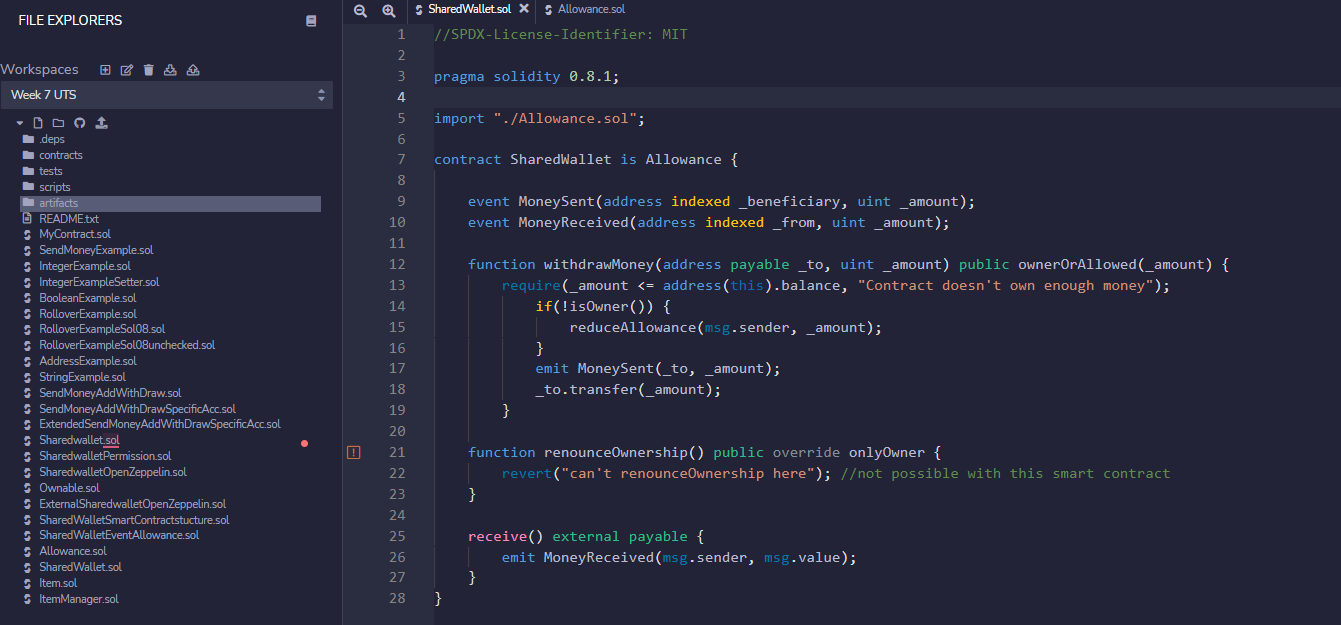
Tanpa mengurangi tunjangan penarikan, seseorang dapat terus menerus menarik jumlah yang sama berulang-ulang



Sekarang setelah mengetahui fungsi dasar hal selanjutnya dapat menyusun kontrak pintar (*Smart Contract*) secara berbeda. Untuk membuatnya lebih mudah dibaca, mungkin dengan bisa mengistirahatkab fungsionalitas menjadi dua kontrak pintar (*Smart Contract*) yang berbeda.



Dan terakhir ini merupakan fungsi penambahan event Allowance dan SharedWallet setelah mengetahui dasar bisa mencoba untuk membuat Smart Contract dengan folder terpisah dan gunakan fungsi impor dan akan menjadi sepeti ini :





Jika sudah membuat dua folder terpisah lalu deploy dan hasilnya akan sepeti ini :

