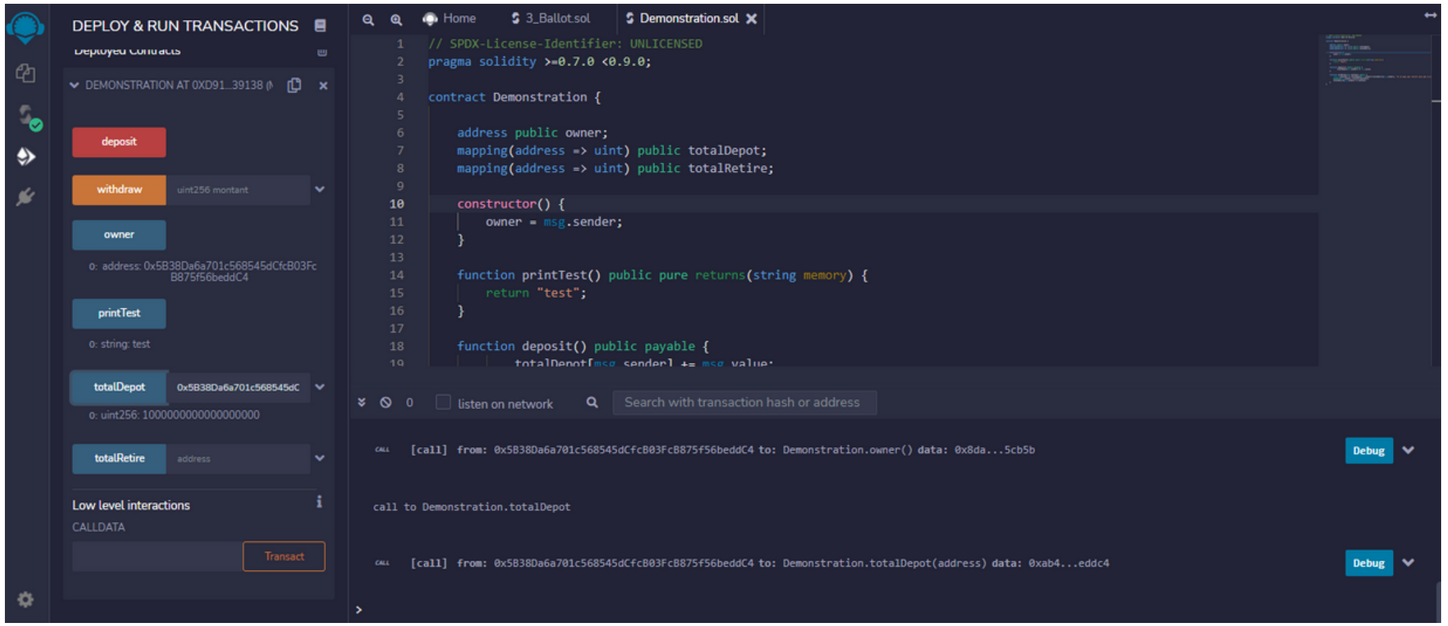
SOLIDITY

Solidity - Cours 1

Environnement de développement

Comme pour tout langage de programmation on peut développer sur n'importe quel éditeur de texte, mais le plus approprié est Remix

Remix va nous permettre de tester nos contrats dans une VM et de déployer nos contrats sur la blockchain (qui supporte l'EVM) de notre choix



Solidity : Un langage orienté objet

Elle consiste en la définition et l'interaction de briques logicielles appelées objets ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre. Il possède une structure interne et un comportement, et il sait interagir avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations ; l'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, de mieux résoudre le ou les problèmes.

Montrer Exemple MonkeFlip

Premier contrat

```
pragma solidity >=0.4.0 <0.6.0;
contract SimpleStorage {
   uint storedData;
   function set(uint x) public {
      storedData = x;
   }
   function get() public view returns (uint) {
      return storedData;
   }
}
```

Ce contrat sert à définir une valeur et à la retourner

Une fonction set(uint x) pour assigner un nombre à 'x'
Une fonction get() pour retourner 'x'

Exo :

Refaire le programme et le tester
Faire une assignation à 'x' au déploiement du contrat

```
constructor() {

}
```

Faire la même chose que 'x' mais pour une chaîne (string)

(On va attendre que tout le monde ait pu tester cette implémentation (C'est normal que ce soit pas facile))

Solution :

```
string chaine;

function setString(string memory _chaine) public {
    chaine = _chaine;
}

function getString() public view returns(string memory) {
    return chaine;
}
```

Certains types en Solidity demande à ce que l'on précise le mode de stockage des données dans l'exécution d'une fonction d'un contrat

Memory : Une variable qui sera stockée en mémoire (Supprimée à la fin de l'éxécution)

Storage : Une variable qui sera stockée durablement dans le contrat

Exemple :
Ici j'ajoute un nouveau Round dans un tableau contenant des Rounds.
J'initialise les valeurs grâce à la nouvelle variable qui est liée à un index de mon tableau

```
function createRound() public onlyAdmin isRoundFinished {
    Round storage round = rounds[actualRound++];
    round.totalAmount = 0;
    round.closed = false;
    round.winSide = Bet.None;
}
```

Deuxieme Contrat

```
contract MiniBanque {

    mapping(address => uint) public totalDepot;
    mapping(address => uint) public totalRetire;

    function depot() public payable {
            totalDepot[msg.sender] += msg.value;
    }

    function retirer(uint montant) public {
        require(montant <= totalDepot[msg.sender]-totalRetire[msg.sender], "Tu ne peux pas retirer plus que ce que tu as depose");

        payable(msg.sender).transfer(montant);

        totalRetire[msg.sender] += montant;
    }
}
```

Ce contrat permet de déposer de l'ETH
Puis de retirer cet ETH mais pas plus que ce que l'on a déjà déposé

Une fonction depot() qui permet de déposer de l'ETH et qui met à jour notre montant total déposé.

Une fonction retirer(uint montant) qui vérifie que l'on puisse retirer, et qui nous transfère le montant demandé

Exo :
Créer une structure Utilisateur contenant le nombre de retraits et le nombre de dépots
Indexer les mapping avec ces Utilisateurs
Supprimer les mappings totalDepot et totalRetire et placer ces variables dans la structure
Faire les changements nécessaires au bon fonctionnement du contrat

Troisième contrat

Gros exo pour refaire tout ce qu'on a vu

Télécharger le fichier Salaire.sol sur github

Exo : Cet exo consiste en la création d'un contrat permettant le dépot sans limite de cryptos, afin de payer le salaire des employés, salaire que les employés doivent réclamer
Faire les fonctions nouvelEmploye(), depot(), et retrait() ainsi que le modifier estEmploye()
Faire une fonction virerEmploye()

Et après on sera bon pour ce cours là
Pour aller plus loin sur cet exercice vous pouvez implémenter un système de hiérarchie, où les plus hauts dans la hiérarchie gagne plus que les autres.
Créer un administrateur qui va pouvoir modifier les employés, retirer toutes les cryptos du contrat