Développeur blockchain

Solidity

Aller plus loin

ALYRA

cyril@alyra.fr / benjamin.brucher@alyra.fr

Promo Buterin

Topo du live

- Corrections des exercices
- Généralisation systeme notation
- Comprendre les ficelles de l'héritage
- Reprendre les enums
- Faire votre propre token
- comprendre le random

Et ce sera la fin de solidity pour cette semaine, pour les bases

Cela restera en toile de fond toute la formation

Parler tirage au sort

ALYRA

Solidity

Exercice 1:

Faire un "deviner c'est gagné!"

Un administrateur va placer un mot, et un indice sur le mot
Les joueurs vont tenter de découvrir ce mot en faisant un essai

Le jeu doit donc
1) instancier un owner
2) permettre a l'owner de mettre un mot et un indice
3) les autres joueurs vont avoir un getter sur l'indice
4) ils peuvent proposer un mot, qui sera comparé au mot référence, return un boolean
5) les joueurs seront inscrit dans un mapping qui permet de savoir si il a déjà joué
6) avoir un getter, qui donne si il existe le gagnant.
7) facultatif (nécessite un array): faire un reset du jeu pour relancer une instance

Corrections

"Devinez c'est gagner "

Comment faire cet exercice?

- Rechercher comment faire de la comparaison de string,
- Maitriser le If,

Pour la version avec reset:

- comprendre l'utilisation d'arrays, leurs length, le pop
- utiliser une fonction dans une fonction
- comprendre que c'est pas optimal

Image 03

```solidity
// SPDX-License-Identifier: GPL-3.0

pragma solidity 0.8.12;

import "@openzeppelin/contracts/access/Ownable.sol";

contract epargne is Ownable {

    string private mot;
    string public indice;
    address public gagnant;
    mapping(address⇒bool) played;

    function setMot(string calldata _mot, string calldata _indice)  public onlyOwner{
        mot=_mot;
        indice=_indice;
    }

    function getWinner() view public returns (string memory){
        if(gagnant==address(0)){
            return "Pas encore de gagnant!";
        }
        else {
            return "Il y a un gagnant!";
        }
    }

    function playWord(string calldata _try) public returns (bool) {
        require(played[msg.sender]==false, "t'as deja joue");
        if (keccak256(abi.encodePacked(_try)) == keccak256(abi.encodePacked(mot))){
            gagnant=msg.sender;
            played[msg.sender]=true;
            return true;
        }
        else {
            played[msg.sender]=true;
            return false;
        }
    }
}
```

ALYRA

Image 04

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity 0.8.12;

import "@openzeppelin/contracts/access/Ownable.sol";

contract epargne is Ownable {

    string private mot;
    string public indice;
    address public gagnant;
    mapping(address=>bool) played;
    address[] public players;

    function setMot(string memory _mot, string memory
_indice)  public onlyOwner{
        mot=_mot;
        indice=_indice;
    }

    function getWinner() view public returns (string
memory){
        if(gagnant==address(0)){
            return "Pas encore de gagnant!";
        }
        else {
            return "Il y a un gagnant!";
        }
    }

    function playWord(string memory _try) public
returns (bool) {
 require(played[msg.sender]==false, "t'as deja
joue");
        if (
            keccak256(abi.encodePacked(_try)) ==
keccak256(abi.encodePacked(mot))
        ){
            gagnant=msg.sender;
            played[msg.sender]=true;
            players.push(msg.sender);
            return true;
        }
        else {
            played[msg.sender]=true;
            players.push(msg.sender);
            return false;
        }
    }
    function reset(string memory _mot, string memory
_indice) public onlyOwner{
        uint tailleTableau=players.length;
        address  tempPlayers;
        for (uint i=tailleTableau-1; i>0; i--){
            tempPlayers=players[i];
            played[tempPlayers]=false;
            players.pop();
        }
        gagnant = address(0);
        setMot(_mot, _indice);
    }
}
```

Solidity

Exercice 2:système de notation élève

Ecrire un smart contract qui gère un système de notation d'une classe d'étudiants avec addNote, getNote, setNote. Un élève est défini par une structure de données :

```
struct Student {
  string name
  uint noteBiology;
  uint noteMath;
  uint noteFr;
}
```

Les professeurs adéquats (rentrés en "dur") peuvent rajouter des notes. Chaque élève est stocké de manière pertinente. On doit pouvoir récupérer:

- la moyenne générale d'un élève
- la moyenne de la classe sur une matière
- la moyenne générale de la classe au global

On doit avoir un getter pour savoir si l'élève valide ou non son année.

Corrections

"Système de note"

Comment faire cet exercice?

- Mélanger toutes les connaissances acquises
- Comprendre le fonctionnement de la division sans virgule
- double mapping pour les professeurs!

A part sur le projet, ne pas hésiter à optimiser les structures au fur et à mesure des incrémentations

Solidity

Exercice 3: généralisation à un lycée

On récupère l'exercice précédent et on l'incrémente, le but: en faire un système complet de notation dans un lycée:

Système de classes définis par un nom (exemple: 2b)
Les professeurs peuvent etre différents d'une classe à l'autre
La structure student sera modifiée pour avoir les classes des élèves

Indices:
- Vous devrez surement utiliser un double mapping et/ou un tableau dans un mapping
- Modifier les require
Vérifier les visibilités de nos fonctions

ALYRA

Solidity

Arrays

On peut faire des tableaux de

- uint : uint[]
- string: string[]
- booléen: bool[]
- ...
- de structures: struct[]

Déclaré comme suit: uint[] values : tableau dynamique

Si uint[3] values, alors c'est un tableau comprenant 3 uint, taille fixe

Il existe des tableaux multi dimensionnels (uint[][])

Ces tableaux multi dimensionnels (en taille fixe ici) sont écrits de manière inversée:

uint [2][3] est un tableau de taille 3 composé de tableaux de taille 2 en uint

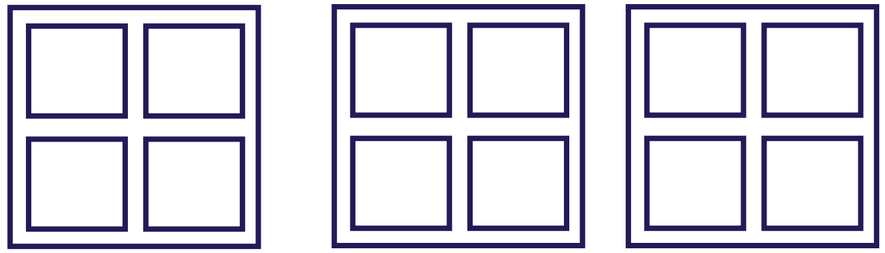
Solidity

Arrays

Premier index 0.

Il possède les attributs:
length, qui retourne la taille du tableau (ne pas oublier que le dernier index est donc length-1),
pop(), qui enlève du tableau le dernier élément,
push(x), qui ajoute x au tableau

uint [4][3]



ALYRA

Solidity

Arrays

- Deux types de tableaux
  - storage (persistant dans la Blockchain) (déclaré en dehors d'une fonction)
  - memory (dans le cadre d'une fonction)

Untitled-1

```
contract Test {
    uint[] public nombres;

    function addNombre(uint _nombre) public {
        nombres.push(_nombre);
    }

    function updateNombre(uint _index, uint _nombre) public {
        nombres[_index] = _nombre;
    }

    function deleteValue(uint _index) public {
        delete nombres[_index];
    }

    function getValue(uint _index) public view returns(uint) {
        return nombres[_index];
    }

    function deleteLastValue() public {
        nombres.pop();
    }

    function getLength() public view returns(uint) {
        return nombres.length;
    }
}
```

Untitled-1

```
pragma solidity 0.8.17;

contract Test {
    uint[] public nombres;

    function addNombre(uint _nombre) public {
        nombres.push(_nombre);
    }

    function getNombreX2() external view returns(uint[] memory) {
        uint longueurTableau = nombres.length;
        uint[] memory nombresX2 = new uint[](longueurTableau);
        for(uint i = 0 ; i < longueurTableau ; i++) {
            nombresX2[i] = nombres[i] * 2;
        }
        return nombresX2;
    }

    function sommeTableau(uint[] memory monTableau) external pure returns(uint) {
        uint longueurTableau = monTableau.length;
        uint somme = 0;
        for(uint  i = 0 ; i < longueurTableau ; i++) {
            somme += monTableau[i];
        }
        return somme;
    }
}
```

Solidity

Untitled-1

```
contract Owner {
    address i_owner;

    constructor() {
        i_owner = msg.sender;
    }

    modifier isOwner() {
        require(msg.sender == i_owner, "Not the owner");
        _;
    }
}

contract Test is Owner {
    uint nombre;

    function setNombre(uint _nombre) external isOwner {
        nombre = _nombre;
    }
}
```

Héritage

2) Utilisation de l'héritage:

2 cas possibles :

¤ contract son is contractFather{ }
On indique au contrat utilisé (le fils) qu'on hérite d'un contrat parent.
Ainsi, notre contrat va posséder toutes les fonctions du contrat parent.

¤ Using A for B;
où A est une librairie, B est un type
Exemple, librairie counters de open zeppelin
On indique qu'un type de variable hérite de fonctions déterminées par le contrat dont il hérite

ALYRA

Solidity
Héritage

Exercice 1

Faire trois contrats (dans un seul fichier sol):

- Un contrat parent, ayant une variable, et une fonction agissant sur la variable
- Un contrat enfant, ayant une fonction capable de retourner la valeur de la variable du contrat parent
- Un contrat caller, capable de faire une instance du contrat enfant, et capable de lancer la fonction du contrat parent et retourner la variable

ALYRA

Image 03

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity 0.8.12;

contract parent{

 // Declaring internal
 // state varaiable
 uint internal sum;

 // Defining external function to set value of internal state
variable sum
 function setValue(uint _value) external {
   sum = _value;
 }
}

// Defining child contract
contract child is parent{
  // Defining external function to return value of internal state
variable sum
 function getValue() external view returns(uint) {
   return sum;
 }
}
 // Defining calling contract
contract caller {
 // Creating child contract object
 child cc = new child();
  // Defining function to call setValue and getValue functions
 function testInheritance(uint _value) public returns (uint) {
   cc.setValue(_value);
   return cc.getValue();
 }
}
```

ALYRA

Solidity
Héritage

Exercice 2:

On déploie un Simple Storage sur Goerli

Vous allez réaliser un contrat appelant le contrat déployé dans une instance, en appelant les deux fonctions Get et Set, afin d'intéragir sur le contrat précédent

Attention, il faut récupérer la signature des fonctions (l'en tete), requise par l'ABI

ALYRA

Image 03

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 * @custom:dev-run-script ./scripts/deploy_with_ethers.ts grgr
 */
contract Storage {

    uint256 public number;

    /**
     * @dev Store value in variable
     * @param num value to store
     */
    function store(uint256 num) payable public {
        number = num;
    }

    /**
     * @dev Return value
     * @return value of 'number'
     */
    function retrieve() public view returns (uint256){
        return number;
    }
}
```

ALYRA

Try Pitch

Ressources

Ressources anglophones gratuites :

- La documentation officielle de Solidity
- Les tutoriels de Block Geeks
- Les tutoriels de Coinmonks
- Les tutoriels Dapp university
- Les tutoriels EatTheBlocks

Ressources sur le sujet en français :

- Les nombreux articles et tutoriels d'Ethereum France
- Les tutoriels de Cryptoast

Ressources

Les liens que l'on va utiliser par la suite:

- Consensys
- OpenZeppelin
- Chainlink random

Solidity
ERC20

Exercice: Créez votre propre token!

En utilisant l'erc20 d'openZeppelin,
Vous devrez créer un token
Vous devez etre capable de l'ajouter sur metamask
Transferer le à vos camarades si vous le voulez

ALYRA

Image 03

```
 // SPDX-License-Identifier: GPL-3.0
pragma solidity 0.8.18;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract AlyraToken is ERC20 {
    constructor(uint256 initialSupply) ERC20("AlyraToken", "ATN") {
        _mint(msg.sender, initialSupply);
    }
}
```

ALYRA

Solidity
Random

Le random en informatique classique est quasi impossible à déterminer.

Sur solidity il y a deux méthodes:

- une qui repose sur un hash d'une chaine de caractères sur laquelle nous devons avoir le moins d'incidence possible

- ou utiliser un oracle comme chainlink qui nous propose d'utiliser leurs services pour récuperer des data exogène à la blockchain, comme un nombre random

https://docs.chain.link/docs/get-a-random-number/

ALYRA

Image 03

```
 // SPDX-License-Identifier: GPL-3.0
pragma solidity 0.8.18;


contract parisportif{

  function random() private view returns (uint8) {
    return uint8(uint256(keccak256(block.timestamp, block.difficulty))%100);
  }

}
```

ALYRA

25/05/2023

N'oubliez pas le projet 1 pour lundi soir minuit :)

Merci de votre attention!

ALYRA