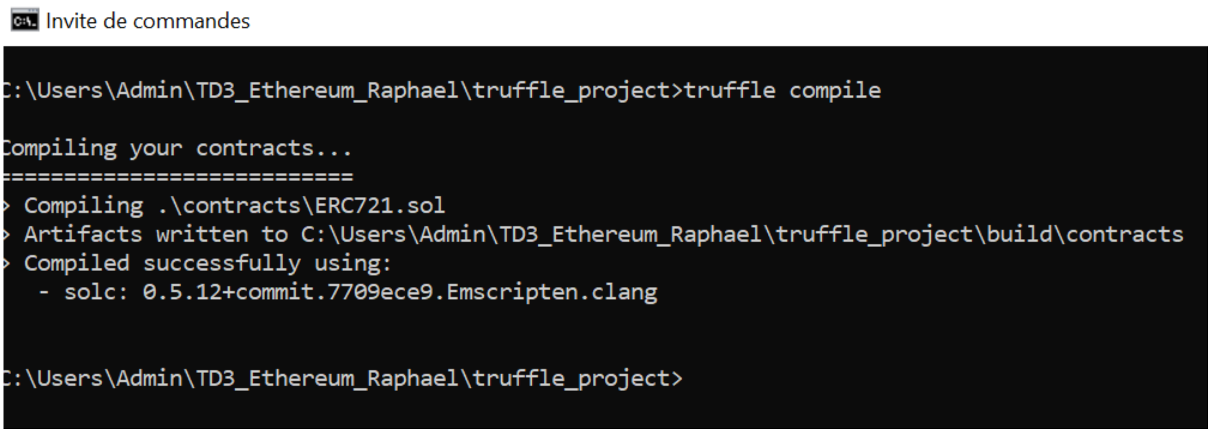
Mardi 03 Décembre 2019,

Monnaies Numériques

Computer Science & Engineering
9329020182
Raphael STIEFFATRE

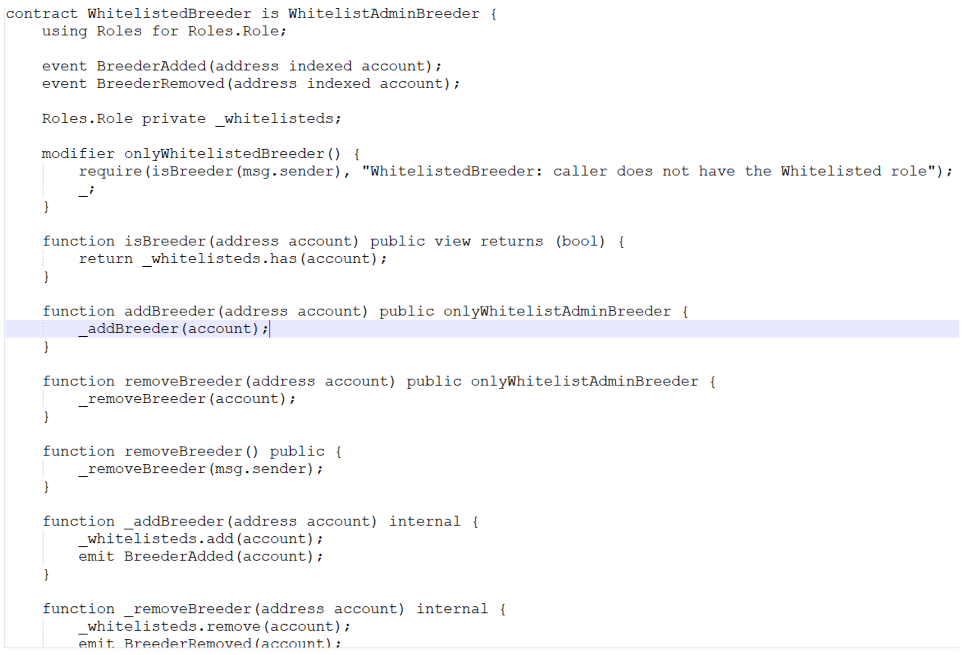
1) <u>ERC721 token contract :</u>

ERC20 : implementation of OpenZepelin

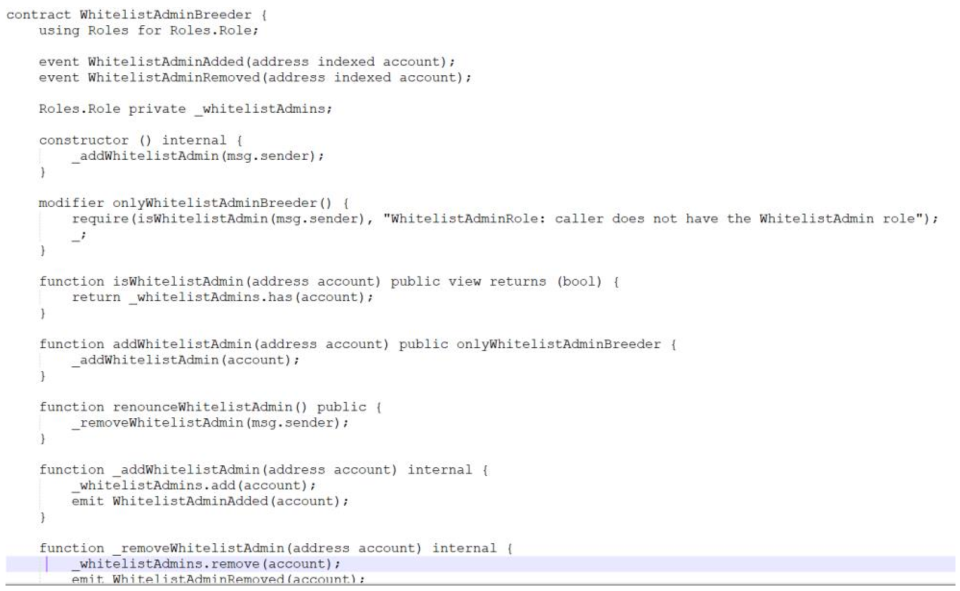


```
Invite de commandes

C:\Users\Admin\TD3_Ethereum_Raphael\truffle_project>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\ERC721.sol
> Artifacts written to C:\Users\Admin\TD3_Ethereum_Raphael\truffle_project\build\contracts
> Compiled successfully using:
   - solc: 0.5.12+commit.7709ece9.Emscripten.clang


C:\Users\Admin\TD3_Ethereum_Raphael\truffle_project>
```

2) <u>registerBreeder :</u>



```
contract WhitelistedBreeder is WhitelistAdminBreeder {
    using Roles for Roles.Role;

    event BreederAdded(address indexed account);
    event BreederRemoved(address indexed account);

    Roles.Role private _whitelisteds;

    modifier onlyWhitelistedBreeder() {
        require(isBreeder(msg.sender), "WhitelistedBreeder: caller does not have the Whitelisted role");
        _;
    }

    function isBreeder(address account) public view returns (bool) {
        return _whitelisteds.has(account);
    }

    function addBreeder(address account) public onlyWhitelistAdminBreeder {
        _addBreeder(account);
    }

    function removeBreeder(address account) public onlyWhitelistAdminBreeder {
        _removeBreeder(account);
    }

    function removeBreeder() public {
        _removeBreeder(msg.sender);
    }

    function _addBreeder(address account) internal {
        _whitelisteds.add(account);
        emit BreederAdded(account);
    }

    function _removeBreeder(address account) internal {
        _whitelisteds.remove(account);
        emit BreederRemoved(account);
```



```
contract WhitelistAdminBreeder {
    using Roles for Roles.Role;

    event WhitelistAdminAdded(address indexed account);
    event WhitelistAdminRemoved(address indexed account);

    Roles.Role private _whitelistAdmins;

    constructor () internal {
        _addWhitelistAdmin(msg.sender);
    }

    modifier onlyWhitelistAdminBreeder() {
        require(isWhitelistAdmin(msg.sender), "WhitelistAdminRole: caller does not have the WhitelistAdmin role");
        _;
    }

    function isWhitelistAdmin(address account) public view returns (bool) {
        return _whitelistAdmins.has(account);
    }

    function addWhitelistAdmin(address account) public onlyWhitelistAdminBreeder {
        _addWhitelistAdmin(account);
    }

    function renounceWhitelistAdmin() public {
        _removeWhitelistAdmin(msg.sender);
    }

    function _addWhitelistAdmin(address account) internal {
        _whitelistAdmins.add(account);
        emit WhitelistAdminAdded(account);
    }

    function _removeWhitelistAdmin(address account) internal {
        _whitelistAdmins.remove(account);
        emit WhitelistAdminRemoved(account);
```

3) <u>declareAnimal et deadAnimal :</u>

```
// Mapping from animal to address of breeders
mapping(address => Animal) registeredAnimal;
address[] public registeredAnimalAccts;
```

```
// Animal
struct Animal {
    string race;
    string color;
    bool isCarnivorous;
    int legs;
    int size;
}
```

```
/**
* @dev declareAnimal()
*
*/
function declareAnimal(address to, string memory race, string memory color, bool isCarnivorous, int legs, int size) onlyWhitelistedBreeder public {
    Animal memory animal = registeredAnimal[to];
    animal.race = race;
    animal.color = color;
    animal.isCarnivorous = isCarnivorous;
    animal.legs = legs;
    animal.size = size;
    registeredAnimalAccts.push(to) -1;
}

/**
* @dev deadAnimal()
*
*/
function deadAnimal(address to) onlyWhitelistedBreeder public {
    delete registeredAnimal[to];
}
```

```
function transfer(address to, uint256 value)
onlyWhitelisted
public
returns (bool)
{
  require(value <= _balances[msg.sender]);
  require(to != address(0));

  _balances[msg.sender] = _balances[msg.sender].sub(value);
  _balances[to] = _balances[to].add(value);
  emit Transfer(msg.sender, to, value);
  return true;
}
```