**周亚男 2020131062**

## 完善合约代码

ZYN.sol

// SPDX-License-Identifier: SEE LICENSE IN LICENSE

pragma solidity ^0.8.12;

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

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

import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";

contract ZYN is ERC20, Ownable, ERC20Burnable {

    constructor() ERC20("ZYN", "ZYN") {

    }

    function mint(address reciever, uint256 amount) public onlyOwner {

        \_mint(reciever, amount);

    }

    function \_burn(uint256 amount) public onlyOwner {

        burn(amount);

    }

}

Pricefeed.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract PriceFeed {

    constructor() {}

    /\*\*

     \* Returns the latest price.

     \*/

    // 抵押品的价格，比如1 ZYN = 2 USD

    function getLatestPrice() public pure returns (int price) {

        return 2 \* 1e18;

    }

}

02\_CollateralStableCoin\_start.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

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

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";

import "@openzeppelin/contracts/utils/math/SafeMath.sol";

import "./defi-practices/PriceFeed.sol";

import "./defi-practices/ZYN.sol";

// 此处补全

contract CollateralStableCoin is ERC20 {

    using SafeMath for uint256;

    IERC20 public collateralToken; // 要抵押的币 ZYN

    PriceFeed public priceFeed; // 价格预言机 返回当前token的价格

    uint256 public amountOfCollateralToken; // 抵押币的总量

    uint256 public constant COLLATERAL\_RATIO\_PRECISION = 1e18;

    constructor(

        address \_collateralToken,

        address \_priceFeed

    ) ERC20("DAI", "DAI") {

        collateralToken = IERC20(\_collateralToken);

        priceFeed = PriceFeed(\_priceFeed);

    }

    function getCollateralPrice() public view returns (uint256) {

        return uint256(priceFeed.getLatestPrice());

    }

    function calculateCollateralAmount(

        uint256 \_stablecoinAmount

    ) public view returns (uint256) {

        // 150% 超额抵押 得到换\_stablecoinAmount个稳定币需要抵押的币

        // uint256\*getCollateralPrice().mul(100).div(150)==\_stablecoinAmount;

        return

            \_stablecoinAmount

                .mul(COLLATERAL\_RATIO\_PRECISION)

                .mul(150)

                .div(100)

                .div(getCollateralPrice());

    }

    function getzyn() public view returns (uint256) {

        return collateralToken.balanceOf(msg.sender);

    }

    function mint(uint256 \_stablecoinAmount) external {

        require(\_stablecoinAmount > 0);

        uint256 collateralToStablecoin = calculateCollateralAmount(

            \_stablecoinAmount

        );

        require(

            collateralToken.balanceOf(msg.sender) >= collateralToStablecoin

        );

        collateralToken.transferFrom(

            msg.sender,

            address(this),

            collateralToStablecoin

        );

        amountOfCollateralToken = amountOfCollateralToken.add(

            collateralToStablecoin

        );

        \_mint(msg.sender, \_stablecoinAmount);

    }

    function burn(uint256 \_stablecoinAmount) external {

        uint256 collateralToStablecoin = calculateCollateralAmount(

            \_stablecoinAmount

        );

        require(\_stablecoinAmount > 0);

        require(amountOfCollateralToken >= collateralToStablecoin);

        require(balanceOf(msg.sender) >= \_stablecoinAmount);

        collateralToken.transfer(msg.sender, collateralToStablecoin);

        amountOfCollateralToken = amountOfCollateralToken.sub(

            collateralToStablecoin

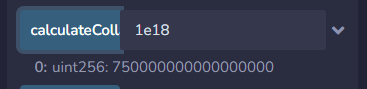
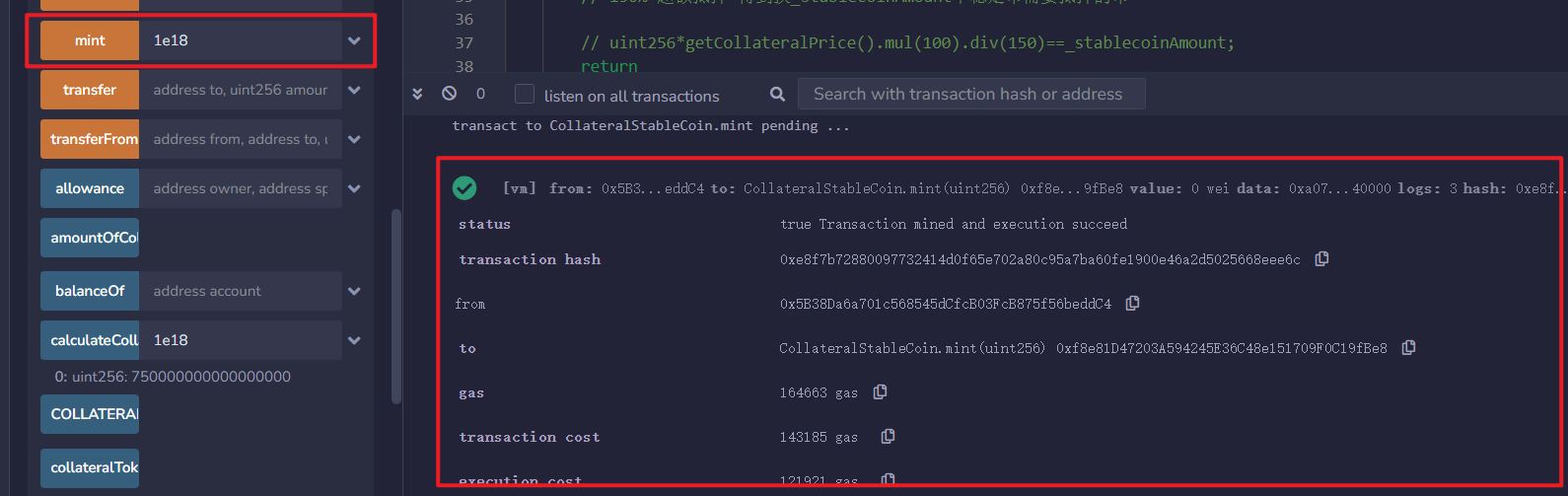
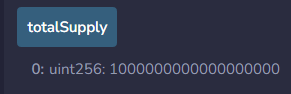
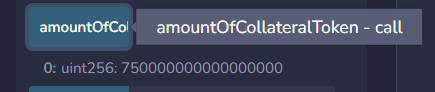
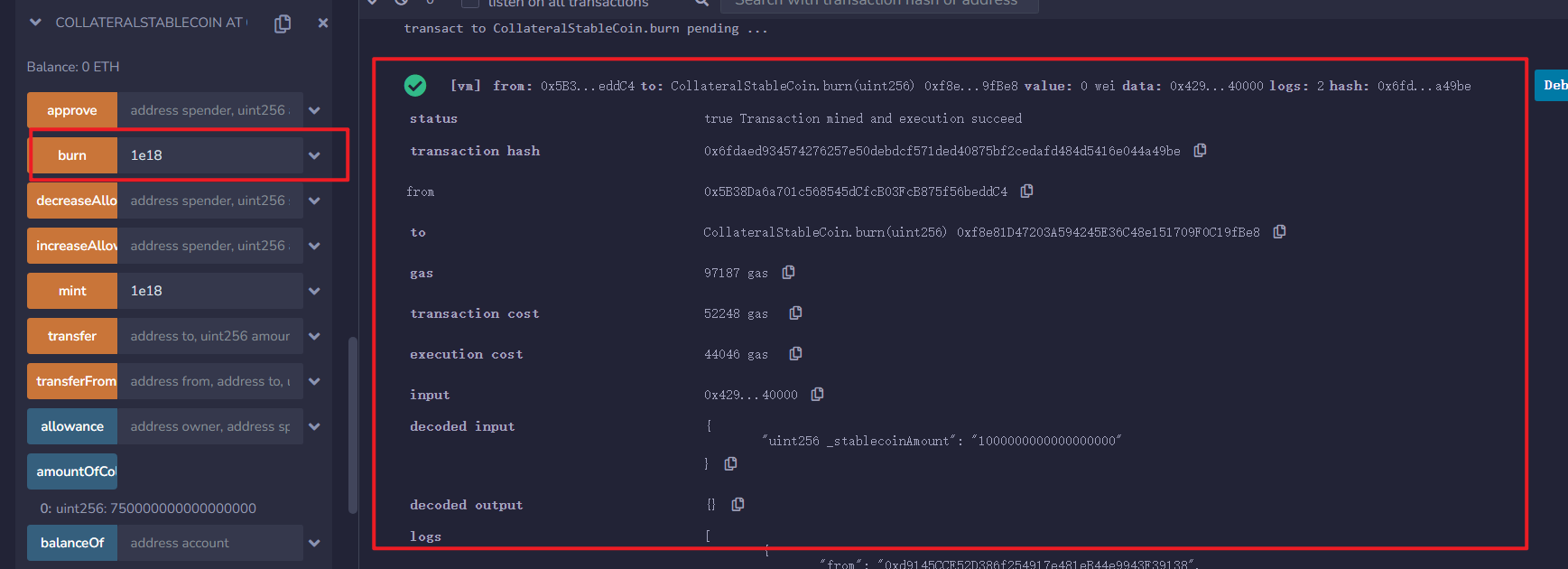
        );

        \_burn(msg.sender, \_stablecoinAmount);

    }

}

## 实验过程

1. 获取一个dai 计算抵押数量  
   
2. 给自己账户mint一个dai  
   
3.  
4. 销毁一个dai  
   
5. 