// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract Flats{

    struct flat{

        address owner;

        uint squer;

        uint lifetime;

        bool statuseOwner;

        bool statusRent;

    }

    struct Request{

        address owner;

        uint idFlat;

        uint price;

        uint saleTerm;

        bool statusRequest;

        address buyer;

        bool statusBuyer;

        bool statuseAnswer;

    }

    struct Buyer{

        address buyer;

        uint priceBuyer;

    }

    address admin;

    mapping (address => flat[]) bazaOwners;

    Request[]requests;

    Buyer[]massBuyer;

    uint nowMoment = block.timestamp;

    constructor(){

        admin = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;

        bazaOwners[0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2].push(flat(0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2, 50, 7, true, false));

    }

    //Создание запроса на продажу

    function CreateRequest(uint idFlat, uint price, uint TimeActual)public{

        require(bazaOwners[msg.sender].length > idFlat, "Error");

        requests.push(Request(msg.sender, idFlat, price, nowMoment + (TimeActual \* 24 \*60\*\*2), false, 0x0000000000000000000000000000000000000000,false, false));

    }

    //Добавление квартиры

    function AddFlat(address owner, uint squer,uint lifetime)public{

        require(msg.sender == admin, "You not admin");

        bazaOwners[owner].push(flat(owner, squer, lifetime,true,false));

    }

    //Заявка на покупку квартиры

    function purchase (uint idRequest)public payable{

        require(msg.sender != requests[idRequest].owner, "you cannot sell the flat to yourself");

        require(requests[idRequest].statusRequest == false, "Application closed");

        require(requests[idRequest].saleTerm > nowMoment, "Application is not valid");

        require(msg.value >= requests[idRequest].price, "you offered a low price");

        requests[idRequest].statusBuyer = true;

        requests[idRequest].buyer = msg.sender;

        requests[idRequest].price = msg.value;

    }

    //Продажа квартиры

    function SellFlat(uint idRequest)public payable{

        require(requests[idRequest].owner == msg.sender,"You not owner");

        require(requests[idRequest].saleTerm > nowMoment, "Application is not valid");

        require(requests[idRequest].statusRequest == false, "Application closed");

        require(requests[idRequest].statuseAnswer == true);

        require(requests[idRequest].statusBuyer == true, "You has not buyer");

        // require(bazaOwners[msg.sender][requests[idRequest].idFlat].statuseOwner == false, "You are no longer the owner of this apartment");

        address buyer = requests[idRequest].buyer;

        address owner = msg.sender;

        uint idFlat = requests[idRequest].idFlat;

        uint squer = bazaOwners[owner][idFlat].squer;

        uint lifetime = bazaOwners[owner][idFlat].lifetime;

        payable(owner).transfer(requests[idRequest].price);

        bazaOwners[buyer].push(flat(buyer,squer, lifetime, true,false));

        bazaOwners[owner][idFlat].statuseOwner = false;

        requests[idRequest].statusRequest = true;

    }

    //Отказ от продажи

    function dpropRequest (uint idRequest)public payable{

        require(requests[idRequest].owner == msg.sender,"You not owner");

        require(requests[idRequest].statusRequest == false, "Application closed");

        require(requests[idRequest].saleTerm > nowMoment, "Application is not valid");

        if(requests[idRequest].statusBuyer == true){

            payable(requests[idRequest].buyer).transfer(requests[idRequest].price);

        }

        requests[idRequest].statusRequest = true;

    }

    //Соглашение пользлвателя

    function answerOwner(uint idRequest, bool answer)public{

        require(requests[idRequest].owner == msg.sender,"You not owner");

        require(requests[idRequest].statusBuyer == true, "You has not buyer");

        if(answer){

            requests[idRequest].statuseAnswer = true;

        }

        else{

            requests[idRequest].statusBuyer = false;

            requests[idRequest].buyer = 0x0000000000000000000000000000000000000000;

            payable(requests[idRequest].buyer).transfer(requests[idRequest].price);

        }

    }

    //Отмена покупки квартиры

    function cancelRequest(uint idRequest) public {

        require(requests[idRequest].saleTerm >= nowMoment, "Time is low");

        require(requests[idRequest].buyer == msg.sender,"You not buyer");

        require(requests[idRequest].statusRequest == false, "Application closed");

        require(requests[idRequest].statuseAnswer == true, "Status is not true");

        uint priceCancel = requests[idRequest].price;

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

    }

    function showFlat(address owner) public view returns(flat[] memory){

        return bazaOwners[owner];

    }

}

contract presentFlatContract is Flats{

    struct present{

        address owner;

        uint idFlat;

        uint timePresent;

        bool statusePresent;

        bool statuseAnswer;

        address recipient;

    }

    present[] presents;

    function presentFlat(uint idFlat, uint timePresent,address recipient)public{

        uint time = nowMoment + (timePresent \* 24 \*60\*\*2);

        presents.push(present(msg.sender, idFlat, time, false,false,recipient));

    }

    //Ответ на подарок

    function presentAnswer(uint idPresent, bool answer)public{

        require(presents[idPresent].recipient == msg.sender, "This gift is not for you");

        if(answer){

            presents[idPresent].statusePresent = true;

            address recipient = presents[idPresent].recipient;

            address owner = presents[idPresent].owner;

            uint idFlat = presents[idPresent].idFlat;

            uint squer = bazaOwners[owner][idFlat].squer;

            uint lifetime = bazaOwners[owner][idFlat].lifetime;

            bazaOwners[recipient].push(flat(recipient,squer, lifetime, true,false));

            bazaOwners[owner][idFlat].statuseOwner = false;

        }

        presents[idPresent].statuseAnswer = true;

    }

    function dpropPresent(uint idPresent)public{

        require(presents[idPresent].owner == msg.sender,"You not owner");

        require(presents[idPresent].statusePresent == false,"Application closed");

        presents[idPresent].statusePresent = true;

    }

}

contract arenda is Flats{

    struct rent{

        address owner;

        uint idFlat;

        uint price;

        uint term;

        uint termEnd;

        address user;

        bool statusRent; // снимает ли кто-то квартиру

        bool statuseAnswer; // закрыта ли эта заявка(отказ предоставлять квартиру)

    }

    rent[] rents;

    //Создание объявления об аренде

    function createRent(uint idFlat, uint price, uint term)public{

        require(bazaOwners[msg.sender][idFlat].statuseOwner == true, "You not owner");

        require(bazaOwners[msg.sender][idFlat].statusRent == false, "you have already rent");

        rents.push(rent(msg.sender, idFlat, price, term, 0, address(0), false,false));

    }

    //Отклик на предложение

    function requestRent(uint idRent)public payable{

        require(msg.sender != rents[idRent].owner, "you cannot rent the flat to yourself");

        require(rents[idRent].statusRent == false, "apartment already rented");

        require(rents[idRent].statuseAnswer == false, "Application closed");

        require((rents[idRent].price \* rents[idRent].term) == msg.value, "you have low money");

        rents[idRent].user = msg.sender;

    }

    //Соглашение хозяина

    function answerUser(uint idRent, bool answer)public{

        require(rents[idRent].owner == msg.sender,"You not owner");

        require(rents[idRent].user != address(0), "You has not buyer");

        require(rents[idRent].statusRent == false, "You already agreed");

        if(answer){

            rents[idRent].statusRent = true;

            rents[idRent].termEnd = (block.timestamp \* 24 \*60\*\*2) \* rents[idRent].term;

            payable(rents[idRent].owner).transfer(rents[idRent].price \* rents[idRent].term);

        }

        else{

            rents[idRent].user = address(0);

            payable(rents[idRent].user).transfer(rents[idRent].price \* rents[idRent].term);

        }

    }

    //Отказ от аренды

    function dpropRent (uint idRent)public payable{

        require(rents[idRent].owner == msg.sender,"You not owner");

        require(rents[idRent].statuseAnswer == false, "Application closed");

        require(rents[idRent].statusRent == false, "You already agreed");

        if(rents[idRent].user != address(0)){

            payable(rents[idRent].user).transfer(rents[idRent].price \* rents[idRent].term);

            rents[idRent].user = address(0);

        }

        rents[idRent].statuseAnswer = true;

    }

    //Отмена аренды квартиры

    function cancelRent(uint idRent) public payable{

        require(rents[idRent].user == msg.sender,"You not buyer");

        require(rents[idRent].statusRent == false, "You are already renting this apartment.");

        require(rents[idRent].statuseAnswer == false, "Application closed");

        payable(msg.sender).transfer(rents[idRent].price \* rents[idRent].term);

    }

}