// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract Flats{

    struct flat{

        address owner;

        uint squaer;

        uint lifetime;

        bool owns; // Владеет квартирой сейчас или нет

    }

    struct Request{

        address owner;

        uint idFlat;

        uint []prices;

        uint saleTerm; // дата продажи

        bool statusRequest; // закрыта заявка или нет

        address [] buyers;

        bool [] statusBuyer;

    }

    // struct buyer{

    //     address buyer;

    //     uint priceBuyer;

    // }

    address admin;

    mapping (address => flat[]) bazaOwners;

    Request[] public requests;

    // buyer[]massBuyer;

    constructor(){

        admin = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;

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

    }

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

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

        require(squaer>0, "squaer is not 0");

        require(lifetime>0, "lifetime is not 0");

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

    }

    function CreateRequest(uint idFlat, uint [] memory prices, address [] memory buyers)public{

        require(bazaOwners[msg.sender].length > idFlat, "flat not found");

        require(bazaOwners[msg.sender][idFlat].owns == true, "you are no longer the owner of the flat");

        // requestsprices[prices.length] = price;

        // buyers.push(0x0000000000000000000000000000000000000000);

        bool [] memory statusBuyer;

        requests.push(Request(msg.sender, idFlat, prices, block.timestamp, false, buyers, statusBuyer));

    }

    function purchase (uint idRequest)public payable{

        uint [] memory prices = requests[idRequest].prices;

        require(msg.value >= prices[prices.length-1], "you offered a low price");

        requests[idRequest].prices.push(msg.value);

        requests[idRequest].buyers.push(msg.sender);

    }

    function SellFlat(uint idRequest)public payable{

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

        require(requests[idRequest].statusRequest == false, "You have already sold flat");

        address [] memory buyers = requests[idRequest].buyers;

        uint [] memory prices = requests[idRequest].prices;

        address owner = requests[idRequest].owner;

        uint idFlat = requests[idRequest].idFlat;

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

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

        payable(owner).transfer(prices[prices.length-1]);

        bazaOwners[buyers[buyers.length-1]].push(flat(buyers[buyers.length-1],squaer, lifetime,true));

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

        requests[idRequest].statusRequest = true;

    }

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

        return bazaOwners[owner];

    }

}