// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract InsuranceContract {

    // Structs

    struct Policy {

        uint premium;

        uint coverage;

        uint remainingCoverage;

        uint startTime;

        uint endTime;

        uint duration;

    }

    struct Claim {

        address claimant;

        uint amount;

        string reason;

        bool isApproved;

        bool isPaid;

    }

    // State variables

    mapping(address => Policy) public policies;

    mapping(uint => Claim) public claims;

    mapping(address => bool) public insurers;

    uint public claimIdCounter; // Changed to public

    // Events (updated to include claimId)

    event PolicyIssued(address indexed policyholder, uint premium, uint coverage);

    event PremiumPaid(address indexed policyholder, uint amount);

    event ClaimSubmitted(uint indexed claimId, address indexed claimant, uint amount, string reason); // Updated

    event ClaimApproved(address indexed claimant, uint amount);

    event ClaimPaid(address indexed claimant, uint amount);

    modifier onlyInsurer() {

        require(insurers[msg.sender], "Only insurer can call");

        \_;

    }

    constructor() {

        insurers[msg.sender] = true;

    }

    function addInsurer(address \_newInsurer) external onlyInsurer {

        insurers[\_newInsurer] = true;

    }

    function issuePolicy(

        address \_policyholder,

        uint \_premium,

        uint \_coverage,

        uint \_duration

    ) external onlyInsurer {

        policies[\_policyholder] = Policy({

            premium: \_premium,

            coverage: \_coverage,

            remainingCoverage: \_coverage,

            startTime: block.timestamp,

            endTime: block.timestamp + \_duration,

            duration: \_duration

        });

        emit PolicyIssued(\_policyholder, \_premium, \_coverage);

    }

    function payPremium() external payable {

        Policy storage policy = policies[msg.sender];

        require(block.timestamp <= policy.endTime, "Policy expired");

        require(msg.value == policy.premium, "Incorrect premium");

        policy.endTime = block.timestamp + policy.duration;

        emit PremiumPaid(msg.sender, msg.value);

    }

    function submitClaim(uint \_amount, string memory \_reason) external {

        Policy storage policy = policies[msg.sender];

        require(block.timestamp <= policy.endTime, "Policy expired");

        require(\_amount <= policy.remainingCoverage, "Claim exceeds coverage");

        claims[claimIdCounter] = Claim({

            claimant: msg.sender,

            amount: \_amount,

            reason: \_reason,

            isApproved: false,

            isPaid: false

        });

        // Emit claimId in the event (NEW)

        emit ClaimSubmitted(claimIdCounter, msg.sender, \_amount, \_reason);

        claimIdCounter++;

    }

    function approveClaim(uint \_claimId) external onlyInsurer {

        Claim storage claim = claims[\_claimId];

        require(!claim.isApproved, "Claim already approved");

        Policy storage policy = policies[claim.claimant];

        require(block.timestamp <= policy.endTime, "Policy expired");

        require(claim.amount <= policy.remainingCoverage, "Claim exceeds coverage");

        claim.isApproved = true;

        emit ClaimApproved(claim.claimant, claim.amount);

    }

    function payClaim(uint \_claimId) external onlyInsurer {

        Claim storage claim = claims[\_claimId];

        require(claim.isApproved, "Claim not approved");

        require(!claim.isPaid, "Claim already paid");

        require(address(this).balance >= claim.amount, "Insufficient funds");

        Policy storage policy = policies[claim.claimant];

        policy.remainingCoverage -= claim.amount;

        claim.isPaid = true;

        (bool sent, ) = payable(claim.claimant).call{value: claim.amount}("");

        require(sent, "Payment failed");

        emit ClaimPaid(claim.claimant, claim.amount);

    }

    // Public getter to track claimIdCounter (NEW)

    function getCurrentClaimId() public view returns (uint) {

        return claimIdCounter;

    }

    receive() external payable {}

}