FlightSuretyData.sol

<<Contract>>

FlightSuretyData

- + address: contractOwner
- + bool: operational
- + struct: Airline
- + struct: Insurance
- + struct: Flight
- + uint: airlineCount
- + uint: next insurance
- + mapping(bytes32 => Flight) flights
- + bytes32[]: flight_keys
- + mapping(address => Airline) airlines
- + mapping(uint => Insurance) insurances
- + address[]: insurees
- + mapping(address => uint256) payouts
- + mapping(address => uint256) funds
- + mapping(address => uint256) authorizedContracts
- + uint256: MAX INSURANCE POLICY
- + uint256: AIRLINE MIN FUNDS
- + event: InsureeCredited
- + event debug(uint i, address insuree, bytes32 id, address airline, string flight, uint256 ftimestamp, uint256 value);
- + modifier requireIsOperational()
- + modifier requireContractOwner()
- + modifier isCallerAuthorized()
- + modifier paidInRange()
- + modifier checkAndRefund(address insuree)
- + function isOperational()
- + function setOperatingStatus(bool mode) external requireContractOwner
- + function authorizeCaller(address caller) public requireContractOwner
- + function deauthorizeCaller(address caller) public requireContractOwner
- + function registerAirline(string calldata name, address wallet) external isCallerAuthorized
- + function getAirlineCount() public view returns (uint256)
- + function getFlights() external view returns (string[] memory, address[] memory, uint256[] memory)
- + function isAirlineRegistered(address wallet) external view returns (bool)
- + function isAirlineFunded(address wallet) external view returns (bool)
- + function isFlightRegistered(string memory name, uint256 timestamp, address airline) public view returns (bool)
- + function registerFlight(string calldata name, uint256 timestamp, address airline) external isCallerAuthorized
- + function buy(string calldata flight, uint256 timestamp, address airline, address insuree) external payable isCallerAuthorized paidInRange checkAndRefund(insuree)
- + function creditInsurees(string calldata flight, uint256 timestamp, address airline)external
- + function checkFunds(address insuree) external view returns (uint)
- + function pay(address payable insuree) external isCallerAuthorized
- + function fundForwarded(address sender) external payable isCallerAuthorized
- + function getFlightKey(address airline, string memory flight, uint256 timestamp) pure internal returns (bytes32)
- + function()external payable

+ uint8 p

+ addres

+ mappir

+ addres

+ uint8 p

+ uint256

+ struct (

+ struct I + mappir

+ mappii

+ modifie

+ modifie

+ modifie

+ modifie + event l

+ event

+ event (

+ functio

+functio

+functio

+functio

+functio

+functio

+functio

+functio

+functio

<<Contract> FlightSuretyApp

```
rivate constant STATUS CODE UNKNOWN = 0;
rivate constant STATUS_CODE_ON_TIME = 10;
rivate constant STATUS_CODE_LATE_AIRLINE = 20;
rivate constant STATUS_CODE_LATE_WEATHER = 30;
rivate constant STATUS_CODE_LATE_TECHNICAL = 40;
rivate constant STATUS_CODE_LATE_OTHER = 50;
s private contractOwner
ng(address => bool) multiCalls
s[] multiCallKeys = new address[](0);
rivate nonce = 0;
public constant REGISTRATION_FEE = 1 ether
5 public constant MIN_RESPONSES = 3;
Oracle
ResponseInfo
ng(address => Oracle) private oracles
ng(bytes32 => ResponseInfo) private oracleResponses
er requireIsOperational()
er requireContractOwner()
er isFunded(address wallet)
er isAllowedToRegisterAirline()
FlightStatusInfo(address airline, string flight, uint256 timestamp, uint8 status)
OracleReport(address airline, string flight, uint256 timestamp, uint8 status)
OracleRequest(uint8 index, address airline, string flight, uint256 timestamp)
n isOperational() public view returns (bool)
n registerAirline(string calldata name, address wallet) external isAllowedToRegisterAirline
returns (bool success, uint256 votes)
n getFunds() external
n registerFlight(string calldata name, uint256 timestamp, address airline) external
n processFlightStatus(address airline, string memory flight, uint256 timestamp, uint8 statusCode) internal
n fetchFlightStatus(address airline, string calldata flight, uint256 timestamp) external
n registerOracle() external payable
```

- n getMyIndexes() view external returns (uint8[3] memory)
- n submitOracleResponse(uint8 index, address airline, string calldata flight, uint256 timestamp, uint8 statusCode) external
- n buylnsurance(string calldata flight, uint256 timestamp, address airline) external payable
- n getFlightKey(address airline, string memory flight, uint256 timestamp) pure internal returns (bytes32)
- n generateIndexes(address account)internal returns (uint8[3] memory)
- n getRandomIndex (address account)internal returns (uint8)
- n() external payable

<<Contract>

FlightSuretyData

- n isOperational() external view returns (bool);
- n setOperatingStatus(bool mode) external
- n isAirlineRegistered(address wallet) external view returns (bool);
- n isAirlineFunded(address wallet) external view returns (bool);
- n registerAirline(string calldata name, address wallet) external;
- n getAirlineCount() external view returns (uint256);
- n isFlightRegistered(string calldata name, uint256 timestamp, address airline) external view returns (bool):
- n registerFlight(string calldata name, uint256 timestamp, address airline) external;
- n buy(string calldata flight, uint256 timestamp, address airline, address insuree) external payable;
- n creditInsurees(string calldata flight, uint256 timestamp, address airline) external;
- n pay(address payable insuree) external;
- n fundForwarded(address sender) external payable;