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

FlightSuretyApp.sol

<<Contract>>
FlightSuretyApp

+ uint8 private constant STATUS_CODE_UNKNOWN = 0;
+ uint8 private constant STATUS_CODE_ON_TIME = 10;
+ uint8 private constant STATUS_CODE_LATE_AIRLINE = 20;
+ uint8 private constant STATUS_CODE_LATE_WEATHER = 30;
+ uint8 private constant STATUS_CODE_LATE_TECHNICAL = 40;
+ uint8 private constant STATUS_CODE_LATE_OTHER = 50;
+ address private contractOwner
+ mapping(address => bool) multiCalls
+ address[] multiCallKeys = new address[](0);
+ uint8 private nonce = 0;
+ uint256 public constant REGISTRATION_FEE = 1 ether
+ uint256 public constant MIN_RESPONSES = 3;
+ struct Oracle
+ struct ResponseInfo
+ mapping(address => Oracle) private oracles
+ mapping(bytes32 => ResponseInfo) private oracleResponses
+ modifier requireIsOperational()
+ modifier requireContractOwner()
+ modifier isFunded(address wallet)
+ modifier isAllowedToRegisterAirline()
+ event FlightStatusInfo(address airline, string flight, uint256 timestamp, uint8 status)
+ event OracleReport(address airline, string flight, uint256 timestamp, uint8 status)
+ event OracleRequest(uint8 index, address airline, string flight, uint256 timestamp)

+ function isOperational() public view returns (bool)
+ function registerAirline(string calldata name, address wallet) external isAllowedToRegisterAirline
  returns (bool success, uint256 votes)
+ function getFunds() external
+ function registerFlight(string calldata name, uint256 timestamp, address airline) external
+ function processFlightStatus(address airline, string memory flight, uint256 timestamp, uint8 statusCode) internal
+ function fetchFlightStatus(address airline, string calldata flight, uint256 timestamp) external
+ function registerOracle() external payable
+ function getMyIndexes() view external returns (uint8[3] memory)
+ function submitOracleResponse(uint8 index, address airline, string calldata flight, uint256 timestamp, uint8 statusCode) external
+ function buyInsurance(string calldata flight, uint256 timestamp, address airline) external payable
+ function getFlightKey(address airline, string memory flight, uint256 timestamp) pure internal returns (bytes32)
+ function generateIndexes(address account)internal returns (uint8[3] memory)
+ function getRandomIndex (address account)internal returns (uint8)
+ function() external payable

<<Contract>>
FlightSuretyData

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