# Mobius Finance DeFi 协议 Solidity 开发负责功能

在 Mobius Finance 固定利率借贷协议中，Solidity 作为核心合约开发语言，承担 “借贷业务逻辑链上实现、资产安全管控、合约权限治理” 三大核心职能，是保障固定利率、固定期限规则落地，以及用户资产安全的底层技术载体。具体负责以下功能模块设计与开发：

## 一、核心借贷合约体系设计与实现

围绕 “固定利率 + 固定期限” 核心特性，设计分层合约架构（借贷池合约、利率计算合约、抵押品管理合约），实现借贷全生命周期的链上逻辑闭环：

### 1. 借贷池合约（MobiusLendingPool）—— 核心业务入口

作为用户交互的核心合约，封装 “创建借贷订单、偿还贷款、清算抵押品” 等核心功能，严格遵循固定利率与期限约束：

* **固定利率借贷订单创建**：
  + 定义标准化借贷订单结构体（Loan），包含关键字段：borrower（借款人地址）、lender（出借人地址，初始为借贷池）、borrowAmount（借款金额，支持 ETH/ERC20）、fixedRate（固定年利率，精确到 18 位小数）、loanTerm（借款期限，单位：秒）、maturityTime（到期时间，创建时计算：block.timestamp + loanTerm）、collateralInfo（抵押品信息，含合约地址、tokenID / 数量、风险系数）、status（订单状态：未激活 / 已生效 / 已偿还 / 已清算 / 已过期）。
  + 实现createFixedRateLoan函数：用户发起借贷请求时，先校验 “借款期限在协议允许范围（7 天≤loanTerm≤365 天）”“固定利率不低于协议最低利率（由治理参数控制）”“抵押品价值≥借款金额 ×（1 + 利息）×1.2（安全系数）”；校验通过后，锁定用户抵押品（ERC20 通过transferFrom，NFT 通过safeTransferFrom转入合约），从借贷池划拨借款资金至借款人地址，同时生成唯一订单 ID（基于keccak256(abi.encodePacked(borrower, block.timestamp, borrowAmount))），将订单信息存入mapping(uint256 => Loan)（订单 ID - 订单映射）。
* **贷款偿还逻辑**：
  + 实现repayLoan(uint256 loanId)函数：借款人偿还时，计算 “应还总额 = 借款金额 + 利息（按固定利率 × 借款天数计算，精确到 18 位小数）”，支持 ETH 直接转账或 ERC20 授权支付；资金到账后，先划拨利息至借贷池（用于分配给出借人），剩余本金归还给借贷池，最后解锁抵押品并划转回借款人地址，同时更新订单状态为 “已偿还”。
  + 支持 “提前还款”：允许借款人在到期前还款，利息按 “实际借款天数” 计算（不收取违约金），仅需在函数中动态计算actualLoanDays = block.timestamp - loanCreationTime，确保利率固定性不受影响。
* **过期订单处理**：
  + 实现markLoanAsExpired(uint256 loanId)函数：仅允许合约管理员或定时触发合约调用，当block.timestamp > maturityTime且订单未偿还时，将订单状态更新为 “已过期”，同时触发抵押品清算流程（调用抵押品管理合约的initiateLiquidation函数）。

### 2. 固定利率计算合约（MobiusRateCalculator）—— 利率管控核心

独立封装利率计算逻辑，确保固定利率的公平性与合理性，避免市场操纵：

* **动态固定利率生成**：
  + 实现calculateFixedRate(uint256 borrowAmount, uint256 loanTerm, address asset)函数：基于 “借贷资产类型、借款金额、期限” 三维度计算基础利率 —— 例如 ETH 作为借款资产，借款金额 10-100 ETH、期限 30 天，基础利率为 5%；借款金额＞100 ETH、期限＞90 天，基础利率下调至 4.2%（鼓励长期大额借款），利率计算参数（如金额分段阈值、期限利率系数）通过治理合约动态调整。
  + 引入 “市场供需调节因子”：当借贷池某资产 “已借总额 / 总余额”＞70%（供不应求），在基础利率上上浮 10%-20%；当该比例＜30%（供过于求），下浮 5%-10%，调节因子每 24 小时更新一次，确保利率反映市场真实供需。
* **利息精确计算**：
  + 实现calculateInterest(uint256 principal, uint256 fixedRate, uint256 loanDays)函数：采用 “单利计算模型”，公式为interest = principal × fixedRate × loanDays / 365，使用SafeMath库（或 Solidity 0.8 + 内置溢出检查）确保计算无溢出，同时支持 18 位小数精度（如 1 ETH 本金、5% 利率、30 天期限，利息 = 1×5%×30/365=0.004109589 ETH）。

### 3. 抵押品管理合约（MobiusCollateralManager）—— 资产安全保障

专注抵押品的锁定、估值、清算逻辑，防范借款人违约风险：

* **抵押品估值与风险系数管控**：
  + 定义mapping(address => uint256)（抵押品合约地址 - 风险系数映射），例如 ETH 风险系数 0.7（抵押品价值需≥借款金额 / 0.7）、BTC 风险系数 0.65、小众 ERC20 代币风险系数 0.5（风险越高系数越低），风险系数由治理投票更新。
  + 集成 Chainlink Price Feeds 预言机：实现getCollateralValue(address collateralAddr, uint256 amount, uint256 tokenId)函数，调用 Chainlink latestRoundData接口获取抵押品实时价格（如 ETH/USD 价格），计算抵押品总价值 = 价格 × 数量（ERC20）/ 单个估值（NFT），同时校验价格更新时间（需在 24 小时内，避免使用过期价格）。
* **抵押品清算逻辑**：
  + 实现initiateLiquidation(uint256 loanId)函数：当抵押品价值≤借款金额 ×（1 + 利息）×1.1（清算阈值）或订单过期时，触发清算 —— 允许清算者以 “90% 市场价” 买入抵押品（清算折扣），清算者支付 “借款金额 + 利息” 后，抵押品划转给清算者，超额部分（若有）退还借款人；若清算后仍有未偿还金额，从借贷池风险准备金中抵扣。
  + 支持批量清算：实现batchLiquidate(uint256[] calldata loanIds)函数，允许清算者一次性清算多个符合条件的订单，减少 Gas 消耗（批量清算 Gas 成本比单次清算降低 30% 以上），同时限制单次批量清算订单数≤20（避免 Gas 超限）。

## 二、合约安全与权限管控设计

通过多维度安全机制，防范合约漏洞与恶意攻击，保障用户资产与协议稳定：

### 1. 基础安全防护措施

* **溢出与重入防护**：使用 Solidity 0.8.0 + 版本（内置整数溢出 / 下溢检查），关键函数（如repayLoan、initiateLiquidation）添加nonReentrant修饰符（基于 OpenZeppelin ReentrancyGuard），避免重入攻击。
* **权限控制**：集成 OpenZeppelin Ownable2Step 合约，核心操作（如更新风险系数、调整协议利率参数、触发紧急暂停）仅允许 “管理员 + 多签钱包” 共同调用，避免单点权限滥用；定义onlyAdmin、onlyLiquidator（仅授权清算者）等自定义修饰符，严格限制函数调用者身份。
* **异常场景处理**：在抵押品转账、资金划拨等关键步骤添加try-catch逻辑，例如 NFT 抵押品safeTransferFrom失败时（如接收地址不支持 NFT），自动回滚订单创建流程，退还用户已支付资金，同时触发CollateralTransferFailed事件通知前端。

### 2. 紧急暂停与风险准备金机制

* **紧急暂停功能**：集成 OpenZeppelin Pausable 合约，当协议遭遇重大漏洞（如预言机故障、清算逻辑缺陷）时，管理员可调用pause()函数暂停所有核心功能（创建订单、偿还、清算），避免损失扩大；修复后通过unpause()恢复，暂停期间用户可查询订单但无法操作。
* **风险准备金设计**：在借贷池合约中预留 “风险准备金” 模块，从每笔借贷利息中提取 5% 存入准备金（如 100 ETH 利息提取 5 ETH 准备金），当清算后仍有未覆盖损失时，从准备金中弥补出借人损失；准备金余额不足时，触发治理提案补充，确保出借人资产安全。

## 三、合约交互与事件设计

规范合约间交互逻辑，通过事件机制实现链下数据同步，支撑前端与监控系统：

### 1. 合约间交互规范

* **最小权限原则**：借贷池合约仅向抵押品管理合约开放 “抵押品估值、清算触发” 权限，向利率计算合约开放 “利率计算” 权限，通过interface定义合约交互接口，避免不必要的功能暴露 —— 例如抵押品管理合约仅实现ICollateralManager接口中getCollateralValue、initiateLiquidation等必要函数，其他函数不对外暴露。
* **跨合约调用异常处理**：当借贷池合约调用利率计算合约calculateFixedRate函数时，添加返回值校验（如利率不得为 0 或超过 50%），若返回异常，立即回滚当前操作，同时触发RateCalculationFailed事件，便于链下监控排查问题。

### 2. 标准化事件设计

为每个核心操作定义清晰事件，便于链下系统（如 GO 中间层、前端）监听与数据同步：

* 订单相关事件：LoanCreated(uint256 indexed loanId, address indexed borrower, uint256 borrowAmount, uint256 fixedRate, uint256 maturityTime)、LoanRepaid(uint256 indexed loanId, uint256 repaidAmount, uint256 interest)、LoanLiquidated(uint256 indexed loanId, address indexed liquidator, uint256 collateralValue)。
* 系统相关事件：CollateralRiskFactorUpdated(address indexed collateralAddr, uint256 newRiskFactor)、ProtocolPaused(address indexed operator, uint256 pausedTime)、RiskReserveUpdated(uint256 newReserveAmount)。
* 事件中indexed关键字标记核心字段（如 loanId、borrower），便于链下系统高效筛选特定事件（如查询某用户的所有借贷订单事件）。

## 四、治理与参数配置合约（MobiusGovernance）

支持协议参数的去中心化更新，确保协议长期适应性：

* **治理提案与投票**：实现 “提案 - 投票 - 执行” 治理流程，任何持有协议治理代币（如 MOBI）的用户可发起提案（如更新抵押品风险系数、调整利率计算参数），提案需获得≥20% 治理代币投票且赞成票≥50% 方可通过，投票周期为 7 天。
* **参数动态更新**：定义mapping(string => uint256)（参数名称 - 参数值映射），例如 “minLoanTerm”（最小借款期限）、“maxFixedRate”（最高固定利率）、“liquidationThreshold”（清算阈值），提案通过后，通过updateProtocolParameter(string calldata paramName, uint256 newValue)函数更新参数，更新后触发ParameterUpdated事件，确保全系统参数同步。

（注：文档部分内容可能由 AI 生成）