```
1 pragma solidity ^0.5.16;
                                                            1 pragma solidity ^0.5.16;
3 import "./InterestRateModel.sol";
                                                            3 import "./InterestRateModel.sol";
4 import "./SafeMath.sol";
                                                            4 import "./SafeMath.sol";
6 /**
   * @title Compound's WhitePaperInterestRateModel Co
                                                            7 * @title Compound's WhitePaperInterestRateModel Co
   ntract
                                                              ntract
    * @author Compound
                                                                * @author Compound
8
                                                            8
   * @notice The parameterized model described in sec
                                                            9 * @notice The parameterized model described in sec
   tion 2.4 of the original Compound Protocol whitepape
                                                              tion 2.4 of the original Compound Protocol whitepape
   */
                                                            10 */
10
11 contract WhitePaperInterestRateModel is InterestRate
                                                            11 contract WhitePaperInterestRateModel is InterestRate
   Model {
12
     using SafeMath for uint:
                                                           12
                                                                  using SafeMath for uint:
13
                                                            13
       event NewInterestParams(uint baseRatePerBlock, u
                                                                   event NewInterestParams(uint baseRatePerBlock, u
   int multiplierPerBlock);
                                                               int multiplierPerBlock);
15
                                                            15
16
                                                            16
                                                                  * @notice The approximate number of blocks per
      * @notice The approximate number of blocks per
    year that is assumed by the interest rate model
                                                                year that is assumed by the interest rate model
18
                                                            18
       uint public constant blocksPerYear = 2102400;
                                                                  uint public constant blocksPerYear = 6000000;
19
                                                           19
                                                            20
20
21
                                                            21
       * @notice The multiplier of utilization rate th
                                                                   * @notice The multiplier of utilization rate th
22
                                                            22
                                                               at gives the slope of the interest rate
   at gives the slope of the interest rate
       */
                                                                   */
23
                                                            23
       uint public multiplierPerBlock;
                                                            24
                                                                   uint public multiplierPerBlock;
24
25
                                                            25
26
                                                            26
       * @notice The base interest rate which is the y
                                                                   * @notice The base interest rate which is the y
   -intercept when utilization rate is 0
                                                               -intercept when utilization rate is 0
28
                                                            28
29
      uint public baseRatePerBlock;
                                                            29
                                                                  uint public baseRatePerBlock;
30
                                                            30
                                                            31
31
       * @notice Construct an interest rate model
                                                                   * @notice Construct an interest rate model
32
                                                            32
                                                                  * @param baseRatePerYear The approximate target
      * @param baseRatePerYear The approximate target
   base APR, as a mantissa (scaled by 1e18)
                                                               base APR, as a mantissa (scaled by 1e18)
        * @param multiplierPerYear The rate of increase
                                                                   * @param multiplierPerYear The rate of increase
34
   in interest rate wrt utilization (scaled by 1e18)
                                                              in interest rate wrt utilization (scaled by 1e18)
       */
                                                                   */
     constructor(uint baseRatePerYear, uint multiplie
                                                                 constructor(uint baseRatePerYear, uint multiplie
   rPerYear) public {
                                                               rPerYear) public {
          baseRatePerBlock = baseRatePerYear.div(block
                                                                      baseRatePerBlock = baseRatePerYear.div(block
   sPerYear);
                                                               sPerYear);
          multiplierPerBlock = multiplierPerYear.div(b
                                                                      multiplierPerBlock = multiplierPerYear.div(b
38
                                                            38
   locksPerYear);
                                                               locksPerYear);
39
                                                           39
          emit NewInterestParams(baseRatePerBlock, mul
                                                                      emit NewInterestParams(baseRatePerBlock, mul
40
                                                           40
   tiplierPerBlock);
                                                               tiplierPerBlock);
41
    }
                                                           41 }
                                                           42
42
43
                                                            43
      * @notice Calculates the utilization rate of th
                                                                  * @notice Calculates the utilization rate of th
44
                                                            44
   e market: `borrows / (cash + borrows - reserves)`
                                                               e market: `borrows / (cash + borrows - reserves)`
      * @param cash The amount of cash in the market
                                                                  * @param cash The amount of cash in the market
        * @param borrows The amount of borrows in the m
                                                                   * @param borrows The amount of borrows in the m
```

arket

arket

```
* @param reserves The amount of reserves in the
                                                                   * @param reserves The amount of reserves in the
   market (currently unused)
                                                               market (currently unused)
48
       * @return The utilization rate as a mantissa be
                                                                    * @return The utilization rate as a mantissa be
   tween [0, 1e18]
                                                               tween [0, 1e18]
49
                                                            49
50
       function utilizationRate(uint cash, uint borrow
                                                                   function utilizationRate(uint cash, uint borrow
   s, uint reserves) public pure returns (uint) {
                                                                s, uint reserves) public pure returns (uint) {
           // Utilization rate is 0 when there are no b
                                                                       // Utilization rate is 0 when there are no b
51
   orrows
                                                               orrows
52
           if (borrows == 0) {
                                                            52
                                                                       if (borrows == 0) {
               return 0:
                                                                           return 0:
54
                                                            54
           }
                                                                       }
           return borrows.mul(1e18).div(cash.add(borrow
                                                                       return borrows.mul(1e18).div(cash.add(borrow
56
   s).sub(reserves));
                                                               s).sub(reserves));
57
     }
                                                            57
                                                                  }
58
                                                            58
59
                                                            59
        * @notice Calculates the current borrow rate pe
                                                                    * @notice Calculates the current borrow rate pe
   r block, with the error code expected by the market
                                                               r block, with the error code expected by the market
        * @param cash The amount of cash in the market
                                                                    * @param cash The amount of cash in the market
61
                                                            61
        ^{\star} @param borrows The amount of borrows in the m
                                                                    * @param borrows The amount of borrows in the m
   arket
                                                               arket
                                                                    * @param reserves The amount of reserves in the
        * @param reserves The amount of reserves in the
63
                                                            63
   market
                                                               market
        * @return The borrow rate percentage per block
                                                                    * @return The borrow rate percentage per block
    as a mantissa (scaled by 1e18)
                                                                as a mantissa (scaled by 1e18)
65
                                                            65
       function getBorrowRate(uint cash, uint borrows,
                                                                   function getBorrowRate(uint cash, uint borrows,
    uint reserves) public view returns (uint) {
                                                                uint reserves) public view returns (uint) {
          uint ur = utilizationRate(cash, borrows, res
                                                                       uint ur = utilizationRate(cash, borrows, res
67
                                                            67
                                                                erves);
          return ur.mul(multiplierPerBlock).div(1e18).
                                                                       return ur.mul(multiplierPerBlock).div(1e18).
68
   add(baseRatePerBlock);
                                                               add(baseRatePerBlock);
69
                                                            69
      }
                                                                  }
                                                            70
70
71
                                                            71
       * @notice Calculates the current supply rate pe
                                                                    * @notice Calculates the current supply rate pe
   r block
                                                               r block
        * @param cash The amount of cash in the market
                                                                    * @param cash The amount of cash in the market
        * @param borrows The amount of borrows in the m
                                                                   * @param borrows The amount of borrows in the m
   arket
                                                               arket
        * @param reserves The amount of reserves in the
                                                                    * @param reserves The amount of reserves in the
75
   market
                                                               market
        * @param reserveFactorMantissa The current rese
                                                                    * @param reserveFactorMantissa The current rese
76
   rve factor for the market
                                                                rve factor for the market
        * @return The supply rate percentage per block
                                                                    * @return The supply rate percentage per block
    as a mantissa (scaled by 1e18)
                                                                as a mantissa (scaled by 1e18)
78
                                                            78
79
       function getSupplyRate(uint cash, uint borrows,
                                                            79
                                                                   function getSupplyRate(uint cash, uint borrows,
    uint reserves, uint reserveFactorMantissa) public v
                                                                uint reserves, uint reserveFactorMantissa) public v
   iew returns (uint) {
                                                                iew returns (uint) {
          uint oneMinusReserveFactor = uint(1e18).sub
                                                                       uint oneMinusReserveFactor = uint(1e18).sub
80
                                                            80
   (reserveFactorMantissa);
                                                                (reserveFactorMantissa);
                                                                       uint borrowRate = getBorrowRate(cash, borrow
81
          uint borrowRate = getBorrowRate(cash, borrow
   s, reserves);
                                                                s, reserves);
82
          uint rateToPool = borrowRate.mul(oneMinusRes
                                                            82
                                                                      uint rateToPool = borrowRate.mul(oneMinusRes
   erveFactor).div(1e18);
                                                                erveFactor).div(1e18);
   return utilizationRate(cash, borrows, reserv
                                                            83 return utilizationRate(cash, borrows, reserv
   es).mul(rateToPool).div(1e18);
                                                               es).mul(rateToPool).div(1e18);
   }
                                                            84 }
85 }
                                                            85 }
86
                                                            86
```