```
1 // SPDX-License-Identifier: MIT
                                                              1 // SPDX-License-Identifier: MIT
 2 pragma solidity 0.8.11;
                                                              2 pragma solidity 0.8.11;
 4 interface erc20 {
                                                              4 interface erc20 {
       function totalSupply() external view returns (u
                                                                    function totalSupply() external view returns (u
   int256);
                                                                int256);
       function transfer(address recipient, uint amoun
                                                                    function transfer(address recipient, uint amoun
                                                                t) external returns (bool);
   t) external returns (bool);
       function decimals() external view returns (uint
                                                                    function decimals() external view returns (uint
       function symbol() external view returns (string
                                                                    function symbol() external view returns (string
   memory);
                                                                memory);
       function balanceOf(address) external view retur
                                                                    function balanceOf(address) external view retur
                                                                ns (uint);
   ns (uint);
       function transferFrom(address sender, address r
                                                                    function transferFrom(address sender, address r
   ecipient, uint amount) external returns (bool);
                                                                ecipient, uint amount) external returns (bool);
      function approve(address spender, uint value) e
                                                                    function approve(address spender, uint value) e
   xternal returns (bool);
                                                                xternal returns (bool);
12 }
14 library Math {
                                                             14 library Math {
       function min(uint a, uint b) internal pure retu
                                                                    function min(uint a, uint b) internal pure retu
   rns (uint) {
                                                                rns (uint) {
           return a < b ? a : b;
                                                                        return a < b ? a : b:
       function sqrt(uint y) internal pure returns (ui
                                                                    function sqrt(uint y) internal pure returns (ui
                                                                nt z) {
   nt z) {
           if (y > 3) {
                                                                        if (y > 3) {
               z = y;
                                                                            z = y;
               uint x = y / 2 + 1;
                                                                            uint x = y / 2 + 1;
               while (x < z) {
                                                                            while (x < z) {
                   z = x;
                                                                                z = x;
                   x = (y / x + x) / 2;
                                                                                x = (y / x + x) / 2;
           } else if (y != 0) {
                                                                        } else if (y != 0) {
               z = 1;
                                                                            z = 1;
           }
       }
                                                                    }
                                                             30 }
   interface IBaseV1Callee {
                                                             32 interface IBaseV1Callee {
       function hook(address sender, uint amount0, uin
                                                                    function hook(address sender, uint amount0, uin
   t amount1, bytes calldata data) external;
                                                                t amount1, bytes calldata data) external;
                                                             34 }
36 // Base V1 Fees contract is used as a 1:1 pair rela
                                                             36 // Base V1 Fees contract is used as a 1:1 pair rela
   tionship to split out fees, this ensures that the c
                                                                tionship to split out fees, this ensures that the c
   urve does not need to be modified for LP shares
                                                                urve does not need to be modified for LP shares
  contract BaseV1Fees {
                                                                contract BaseV1Fees {
       address internal immutable pair; // The pair it
                                                                    address internal immutable pair; // The pair it
   is bonded to
                                                                is bonded to
       address internal immutable token0; // token0 of
                                                                    address internal immutable token0; // token0 of
   pair, saved localy and statically for gas optimizat
                                                                pair, saved localy and statically for gas optimizat
       address internal immutable token1; // Token1 of
                                                                    address internal immutable token1; // Token1 of
   pair, saved localy and statically for gas optimizat
                                                                pair, saved localy and statically for gas optimizat
   ion
                                                                ion
       constructor(address _token0, address _token1) {
                                                                    constructor(address _token0, address _token1) {
           pair = msg.sender;
                                                                        pair = msg.sender;
           token0 = _token0;
                                                                        token0 = _token0;
           token1 = _token1;
                                                                        token1 = _token1;
```

```
function _safeTransfer(address token,address t
                                                                    function _safeTransfer(address token,address t
                                                                o,uint256 value) internal {
   o,uint256 value) internal {
           require(token.code.length > 0);
                                                                        require(token.code.length > 0);
           (bool success, bytes memory data) =
                                                                        (bool success, bytes memory data) =
           token.call(abi.encodeWithSelector(erc20.tra
                                                                        token.call(abi.encodeWithSelector(erc20.tra
   nsfer.selector, to, value));
                                                                nsfer.selector, to, value));
           require(success && (data.length == 0 || ab
                                                                        require(success && (data.length == 0 || ab
   i.decode(data, (bool))));
                                                                i.decode(data, (bool)));
       // Allow the pair to transfer fees to users
                                                                    // Allow the pair to transfer fees to users
       function claimFeesFor(address recipient, uint a
                                                                    function claimFeesFor(address recipient, uint a
   mount0, uint amount1) external {
                                                                mount0, uint amount1) external {
           require(msg.sender == pair);
                                                                        require(msg.sender == pair);
           if (amount0 > 0) _safeTransfer(token0, reci
                                                                        if (amount0 > 0) _safeTransfer(token0, reci
   pient, amount0);
                                                                pient, amount0);
           if (amount1 > 0) _safeTransfer(token1, reci
                                                                        if (amount1 > 0) _safeTransfer(token1, reci
   pient, amount1);
                                                                pient, amount1);
65 // The base pair of pools, either stable or volatil
                                                             65 // The base pair of pools, either stable or volatil
66 contract BaseV1Pair {
                                                             66 contract BaseV1Pair {
       string public name;
                                                                    string public name;
       string public symbol;
                                                                    string public symbol;
       uint8 public constant decimals = 18;
                                                                    uint8 public constant decimals = 18;
       // Used to denote stable or volatile pair, not
                                                                    // Used to denote stable or volatile pair, not
    immutable since construction happens in the initia
                                                                 immutable since construction happens in the initia
   lize method for CREATE2 deterministic addresses
                                                                lize method for CREATE2 deterministic addresses
       bool public immutable stable;
                                                                    bool public immutable stable;
       uint public totalSupply = 0;
                                                                    uint public totalSupply = 0;
       mapping(address => mapping (address => uint)) p
                                                                    mapping(address => mapping (address => uint)) p
   ublic allowance:
                                                                ublic allowance:
       mapping(address => uint) public balanceOf;
                                                                    mapping(address => uint) public balanceOf;
       bytes32 internal DOMAIN_SEPARATOR;
                                                                    bytes32 internal DOMAIN_SEPARATOR;
       // keccak256("Permit(address owner,address spen
                                                                    // keccak256("Permit(address owner,address spen
   der,uint256 value,uint256 nonce,uint256 deadlin
                                                                der,uint256 value,uint256 nonce,uint256 deadlin
   e)");
                                                                e)");
       bytes32 internal constant PERMIT TYPEHASH = 0x6
                                                                    bytes32 internal constant PERMIT TYPEHASH = 0x6
   e71edae12b1b97f4d1f60370fef10105fa2faae0126114a169c
                                                                e71edae12b1b97f4d1f60370fef10105fa2faae0126114a169c
   64845d6126c9;
                                                                64845d6126c9;
                                                                    mapping(address => uint) public nonces;
       mapping(address => uint) public nonces;
       uint internal constant MINIMUM_LIQUIDITY = 10**
                                                                    uint internal constant MINIMUM_LIQUIDITY = 10**
       address public immutable token0;
                                                                    address public immutable token0;
       address public immutable token1;
                                                                    address public immutable token1;
       address public immutable fees;
                                                                    address public immutable fees;
       address immutable factory;
                                                                    address immutable factory;
       // Structure to capture time period obervations
                                                                    // Structure to capture time period obervations
   every 30 minutes, used for local oracles
                                                                every 30 minutes, used for local oracles
       struct Observation {
                                                                    struct Observation {
           uint timestamp;
                                                                        uint timestamp:
           uint reserve0Cumulative;
                                                                        uint reserve0Cumulative;
           uint reserve1Cumulative;
                                                                        uint reserve1Cumulative;
       // Capture oracle reading every 30 minutes
                                                                    // Capture oracle reading every 30 minutes
       uint constant periodSize = 1800;
                                                                    uint constant periodSize = 1800;
```

```
Observation[] public observations;
                                                                       Observation[] public observations;
          uint internal immutable decimals0:
                                                                       uint internal immutable decimals0:
         uint internal immutable decimals1;
                                                                       uint internal immutable decimals1;
         uint public reserve0;
                                                                       uint public reserve0;
         uint public reserve1;
                                                                       uint public reserve1;
         uint public blockTimestampLast;
                                                                       uint public blockTimestampLast;
         uint public reserve0CumulativeLast;
                                                                       uint public reserve0CumulativeLast;
         uint public reserve1CumulativeLast:
                                                                       uint public reserve1CumulativeLast;
          // index0 and index1 are used to accumulate fee
                                                                       // index0 and index1 are used to accumulate fee
     s, this is split out from normal trades to keep the
                                                                   s, this is split out from normal trades to keep the
     swap "clean"
                                                                   swap "clean"
          // this further allows LP holders to easily cla
                                                                       // this further allows LP holders to easily cla
     im fees for tokens they have/staked
                                                                   im fees for tokens they have/staked
         uint public index0 = 0;
                                                                       uint public index0 = 0;
         uint public index1 = 0;
                                                                       uint public index1 = 0;
          // position assigned to each LP to track their
                                                                       // position assigned to each LP to track their
      current index0 & index1 vs the global position
                                                                    current index0 & index1 vs the global position
                                                                       mapping(address => uint) public supplyIndex0;
         mapping(address => uint) public supplyIndex0;
         mapping(address => uint) public supplyIndex1;
                                                                       mapping(address => uint) public supplyIndex1;
          // tracks the amount of unclaimed, but claimabl
                                                                       // tracks the amount of unclaimed, but claimabl
     e tokens off of fees for token0 and token1
                                                                   e tokens off of fees for token0 and token1
                                                                       mapping(address => uint) public claimable0;
         mapping(address => uint) public claimable0;
          mapping(address => uint) public claimable1;
                                                                       mapping(address => uint) public claimable1;
         event Fees(address indexed sender, uint amount
                                                                       event Fees(address indexed sender, uint amount
     0, uint amount1);
                                                                   0, uint amount1);
         event Mint(address indexed sender, uint amount
                                                                       event Mint(address indexed sender, uint amount
     0, uint amount1);
                                                                   0, uint amount1);
129
          event Burn(address indexed sender, uint amount
                                                                       event Burn(address indexed sender, uint amount
     0, uint amount1, address indexed to);
                                                                   0, uint amount1, address indexed to);
         event Swap(
                                                                       event Swap(
             address indexed sender,
                                                                           address indexed sender,
             uint amount0In,
                                                                           uint amount@In,
             uint amount1In,
                                                                           uint amount1In,
                                                                           uint amount00ut
             uint amount00ut,
                                                              134
             uint amount10ut,
                                                                           uint amount10ut,
              address indexed to
                                                                           address indexed to
         event Sync(uint reserve0, uint reserve1);
                                                                       event Sync(uint reserve0, uint reserve1);
         event Claim(address indexed sender, address ind
                                                                       event Claim(address indexed sender, address ind
     exed recipient, uint amount0, uint amount1);
                                                                   exed recipient, uint amount0, uint amount1);
         event Transfer(address indexed from, address in
                                                                       event Transfer(address indexed from, address in
     dexed to, uint amount);
                                                                   dexed to, uint amount);
          event Approval(address indexed owner, address i
                                                                       event Approval(address indexed owner, address i
     ndexed spender, uint amount);
                                                                   ndexed spender, uint amount);
         constructor() {
                                                                       constructor() {
              factory = msg.sender;
                                                                           factory = msg.sender;
 146
              (address _token0, address _token1, bool _st
                                                                           (address _token0, address _token1, bool _st
     able) = BaseV1Factory(msg.sender).getInitializable
                                                                   able) = BaseV1Factory(msg.sender).getInitializable
     ();
              (token0, token1, stable) = (_token0, _token
                                                                           (token0, token1, stable) = (_token0, _token
     1, _stable);
                                                                   1, _stable);
              fees = address(new BaseV1Fees(_token0, _tok
                                                                           fees = address(new BaseV1Fees(_token0, _tok
     en1));
                                                                   en1));
              if (_stable) {
                                                                           if (_stable) {
                 name = string(abi.encodePacked("StableV
                                                                               name = string(abi.encodePacked("StableV
     1 AMM - ", erc20(_token0).symbol(), "/", erc20(_tok
                                                                   1 AMM - ", erc20(\_token0).symbol(), "/", erc20(\_token0)
     en1).symbol()));
                                                                   en1).symbol()));
                  symbol = string(abi.encodePacked("sAMM-
                                                                               symbol = string(abi.encodePacked("sAMM-
     ", erc20(_token0).symbol(), "/", erc20(_token1).sym
                                                                   ", erc20(_token0).symbol(), "/", erc20(_token1).sym
     bol()));
                                                                   bol()));
```

```
} else {
                                                                     } else {
           name = string(abi.encodePacked("Volatil
                                                                         name = string(abi.encodePacked("Volatil
                                                             eV1 AMM - ", erc20(_token0).symbol(), "/", erc20(_t
eV1 AMM - ", erc20(_token0).symbol(), "/", erc20(_t
oken1).symbol()));
                                                             oken1).symbol()));
            symbol = string(abi.encodePacked("vAMM-
                                                                         symbol = string(abi.encodePacked("vAMM-
", erc20(_token0).symbol(), "/", erc20(_token1).sym
                                                             ", erc20(_token0).symbol(), "/", erc20(_token1).sym
bol()));
                                                             bol()));
        decimals0 = 10**erc20( token0).decimals();
                                                                     decimals0 = 10**erc20(_token0).decimals();
        decimals1 = 10**erc20(_token1).decimals();
                                                                     decimals1 = 10**erc20(_token1).decimals();
        observations.push (\tt Observation(block.timesta
                                                                     observations.push(Observation(block.timesta
mp, 0, 0));
                                                             mp, 0, 0));
    // simple re-entrancy check
                                                                 // simple re-entrancy check
    uint internal _unlocked = 1;
                                                                 uint internal _unlocked = 1;
    modifier lock() {
                                                                 modifier lock() {
        require(_unlocked == 1);
                                                                     require(_unlocked == 1);
        _unlocked = 2;
                                                                     _unlocked = 2;
        _unlocked = 1;
                                                                     _unlocked = 1;
    function observationLength() external view retu
                                                                 function observationLength() external view retu
rns (uint) {
                                                             rns (uint) {
        return observations.length;
                                                                     return observations.length;
    function lastObservation() public view returns
                                                                 function lastObservation() public view returns
                                                              (Observation memory) {
 (Observation memory) {
        return observations[observations.length-1];
                                                                     return observations[observations.length-1]:
    function metadata() external view returns (uint
                                                                 function metadata() external view returns (uint
dec0, uint dec1, uint r0, uint r1, bool st, address
                                                             dec0, uint dec1, uint r0, uint r1, bool st, address
t0, address t1) {
                                                             t0, address t1) {
        return (decimals0, decimals1, reserve0, res
                                                                     return (decimals0, decimals1, reserve0, res
erve1, stable, token0, token1);
                                                             erve1, stable, token0, token1);
    function tokens() external view returns (addres
                                                                 function tokens() external view returns (addres
s, address) {
                                                             s, address) {
        return (token0, token1);
                                                                     return (token0, token1);
    // claim accumulated but unclaimed fees (viewab
                                                                 // claim accumulated but unclaimed fees (viewab
le via claimable0 and claimable1)
                                                             le via claimable0 and claimable1)
   function claimFees() external returns (uint cla
                                                                 function claimFees() external returns (uint cla
imed0, uint claimed1) {
                                                             imed0, uint claimed1) {
        return claimFeesFor(msg.sender);
                                                                     return claimFeesFor(msg.sender);
    function claimFeesFor(address recipient) public
                                                                 function claimFeesFor(address recipient) public
lock returns (uint claimed0, uint claimed1) {
                                                             lock returns (uint claimed0, uint claimed1) {
        _updateFor(recipient);
                                                                     _updateFor(recipient);
        claimed0 = claimable0[recipient];
                                                                     claimed0 = claimable0[recipient];
        claimed1 = claimable1[recipient];
                                                                     claimed1 = claimable1[recipient];
        claimable0[recipient] = 0;
                                                                     claimable0[recipient] = 0;
        claimable1[recipient] = 0;
                                                                     claimable1[recipient] = 0;
        BaseV1Fees(fees).claimFeesFor(recipient, cl
                                                                     BaseV1Fees(fees).claimFeesFor(recipient, cl
aimed0, claimed1);
                                                             aimed0, claimed1);
        emit Claim(msg.sender, recipient, claimed0,
                                                                     emit Claim(msg.sender, recipient, claimed0,
claimed1);
                                                             claimed1);
```

```
// Accrue fees on token0
                                                                 // Accrue fees on token0
    function _update0(uint amount) internal {
                                                                 function _update0(uint amount) internal {
        _safeTransfer(token0, fees, amount); // tra
                                                                     _safeTransfer(token0, fees, amount); // tra
nsfer the fees out to BaseV1Fees
                                                             nsfer the fees out to BaseV1Fees
       uint256 _ratio = amount * 1e18 / totalSuppl
                                                                     uint256 _ratio = amount * 1e18 / totalSuppl
y; // 1e18 adjustment is removed during claim
                                                             y; // 1e18 adjustment is removed during claim
        if (_ratio > 0) {
                                                                     if (_ratio > 0) {
            index0 += _ratio;
                                                                         index0 += _ratio;
        emit Fees(msg.sender, amount, 0);
                                                                     emit Fees(msg.sender, amount, 0);
    // Accrue fees on token1
                                                                 // Accrue fees on token1
    function _update1(uint amount) internal {
                                                                 function _update1(uint amount) internal {
        _safeTransfer(token1, fees, amount);
                                                                     _safeTransfer(token1, fees, amount);
        uint256 _ratio = amount * 1e18 / totalSuppl
                                                                     uint256 _ratio = amount * 1e18 / totalSuppl
у;
                                                             ٧;
        if (_ratio > 0) {
                                                                     if (_ratio > 0) {
            index1 += _ratio;
                                                                         index1 += _ratio;
        emit Fees(msg.sender, 0, amount);
                                                                     emit Fees(msg.sender, 0, amount);
    // this function MUST be called on any balance
                                                                 // this function MUST be called on any balance
changes, otherwise can be used to infinitely claim
                                                              changes, otherwise can be used to infinitely claim
    // Fees are segregated from core funds, so fees
                                                                 // Fees are segregated from core funds, so fees
can never put liquidity at risk
                                                             can never put liquidity at risk
    function _updateFor(address recipient) internal
                                                                 function _updateFor(address recipient) internal
       uint _supplied = balanceOf[recipient]; // g
                                                                     uint _supplied = balanceOf[recipient]; // g
et LP balance of `recipient`
                                                             et LP balance of `recipient`
        if (_supplied > 0) {
                                                                     if (_supplied > 0) {
            uint _supplyIndex0 = supplyIndex0[recip
                                                                         uint _supplyIndex0 = supplyIndex0[recip
ient]; // get last adjusted index0 for recipient
                                                             ient]; // get last adjusted index0 for recipient
           uint _supplyIndex1 = supplyIndex1[recip
                                                                         uint _supplyIndex1 = supplyIndex1[recip
ientl;
                                                             ient];
            uint _index0 = index0; // get global in
                                                                         uint _index0 = index0; // get global in
dex0 for accumulated fees
                                                             dex0 for accumulated fees
           uint _index1 = index1;
                                                                        uint _index1 = index1;
            supplyIndex0[recipient] = _index0; // u
                                                                         supplyIndex0[recipient] = _index0; // u
pdate user current position to global position
                                                             pdate user current position to global position
            supplyIndex1[recipient] = _index1;
                                                                         supplyIndex1[recipient] = _index1;
            uint _delta0 = _index0 - _supplyIndex0;
                                                                         uint _delta0 = _index0 - _supplyIndex0;
// see if there is any difference that need to be a
                                                             // see if there is any difference that need to be a
ccrued
                                                             ccrued
            uint _delta1 = _index1 - _supplyIndex1;
                                                                         uint _delta1 = _index1 - _supplyIndex1;
            if (_delta0 > 0) {
                                                                         if (_delta0 > 0) {
               uint _share = _supplied * _delta0 /
                                                                             uint _share = _supplied * _delta0 /
1e18; // add accrued difference for each supplied t
                                                             1e18; // add accrued difference for each supplied t
oken
                                                             oken
                claimable0[recipient] += _share;
                                                                             claimable0[recipient] += share;
                                                                         3
            if ( delta1 > 0) {
                                                                         if ( delta1 > 0) {
                                                        244
                uint _share = _supplied * _delta1 /
                                                                             uint _share = _supplied * _delta1 /
1e18;
                                                             1e18;
                claimable1[recipient] += _share;
                                                                             claimable1[recipient] += _share;
                                                                         }
        } else {
                                                                     } else {
            supplyIndex0[recipient] = index0; // ne
                                                                         supplyIndex0[recipient] = index0; // ne
w users are set to the default global state
                                                             w users are set to the default global state
            supplyIndex1[recipient] = index1;
                                                                         supplyIndex1[recipient] = index1;
    function getReserves() public view returns (uin
                                                                 function getReserves() public view returns (uin
t _reserve0, uint _reserve1, uint _blockTimestampLa
                                                             t _reserve0, uint _reserve1, uint _blockTimestampLa
st) {
                                                             st) {
```

```
reserve0 = reserve0;
                                                                     reserve0 = reserve0;
        reserve1 = reserve1;
                                                                     reserve1 = reserve1;
        _blockTimestampLast = blockTimestampLast;
                                                                     blockTimestampLast = blockTimestampLast;
   }
   // update reserves and, on the first call per b
                                                                 // update reserves and, on the first call per b
lock, price accumulators
                                                             lock, price accumulators
    function _update(uint balance0, uint balance1,
                                                                 function _update(uint balance0, uint balance1,
uint _reserve0, uint _reserve1) internal {
                                                              uint _reserve0, uint _reserve1) internal {
       uint blockTimestamp = block.timestamp;
                                                                     uint blockTimestamp = block.timestamp;
       uint timeElapsed = blockTimestamp - blockTi
                                                                     uint timeElapsed = blockTimestamp - blockTi
mestampLast; // overflow is desired
                                                             mestampLast; // overflow is desired
       if (timeElapsed > 0 && _reserve0 != 0 && _r
                                                                     if (timeElapsed > 0 && _reserve0 != 0 && _r
eserve1 != 0) {
            reserve0CumulativeLast += _reserve0 * t
                                                                         reserve0CumulativeLast += _reserve0 * t
imeElapsed;
                                                             imeElapsed;
            reserve1CumulativeLast += reserve1 * t
                                                                         reserve1CumulativeLast += reserve1 * t
imeElapsed;
                                                             imeElapsed;
        Observation memory _point = lastObservation
                                                                     Observation memory _point = lastObservation
        timeElapsed = blockTimestamp - _point.times
                                                                     timeElapsed = blockTimestamp - _point.times
tamp; // compare the last observation with current
                                                             tamp; // compare the last observation with current
timestamp, if greater than 30 minutes, record a ne
                                                              timestamp, if greater than 30 minutes, record a ne
w event
                                                             w event
        if (timeElapsed > periodSize) {
                                                                     if (timeElapsed > periodSize) {
            observations.push(Observation(blockTime
                                                                         observations.push(Observation(blockTime
stamp, reserve0CumulativeLast, reserve1CumulativeLa
                                                             \verb|stamp|, reserve0CumulativeLast|, reserve1CumulativeLa|\\
                                                             st));
       reserve0 = balance0:
                                                                     reserve0 = balance0:
        reserve1 = balance1;
                                                                     reserve1 = balance1;
        blockTimestampLast = blockTimestamp;
                                                                     blockTimestampLast = blockTimestamp;
        emit Sync(reserve0, reserve1);
                                                                     emit Sync(reserve0, reserve1);
    // produces the cumulative price using counterf
                                                                 // produces the cumulative price using counterf
actuals to save gas and avoid a call to sync.
                                                             actuals to save gas and avoid a call to sync.
    function currentCumulativePrices() public view
                                                                 function currentCumulativePrices() public view
returns (uint reserveOCumulative, uint reserve1Cum
                                                              returns (uint reserve0Cumulative, uint reserve1Cum
ulative, uint blockTimestamp) {
                                                             ulative, uint blockTimestamp) {
        blockTimestamp = block.timestamp;
                                                                     blockTimestamp = block.timestamp;
        reserve0Cumulative = reserve0CumulativeLas
                                                                     reserve0Cumulative = reserve0CumulativeLas
        reserve1Cumulative = reserve1CumulativeLas
                                                        284
                                                                     reserve1Cumulative = reserve1CumulativeLas
        // if time has elapsed since the last updat
                                                                     // if time has elapsed since the last updat
e on the pair, mock the accumulated price values
                                                             e on the pair, mock the accumulated price values
        (uint reserve0, uint reserve1, uint bloc
                                                                     (uint reserve0, uint reserve1, uint bloc
kTimestampLast) = getReserves();
                                                             kTimestampLast) = getReserves();
        if (_blockTimestampLast != blockTimestamp)
                                                                     if (_blockTimestampLast != blockTimestamp)
            // subtraction overflow is desired
                                                                         // subtraction overflow is desired
            uint timeElapsed = blockTimestamp - _bl
                                                                         uint timeElapsed = blockTimestamp - _bl
ockTimestampLast;
                                                             ockTimestampLast;
            reserve0Cumulative += _reserve0 * timeE
                                                                         reserve0Cumulative += _reserve0 * timeE
lapsed;
                                                             lapsed;
            reserve1Cumulative += _reserve1 * timeE
                                                                         reserve1Cumulative += _reserve1 * timeE
lansed:
                                                             lansed:
                                                        294
    // gives the current twap price measured from a
                                                                 // gives the current twap price measured from a
                                                             mountIn * tokenIn gives amountOut
mountIn * tokenIn gives amountOut
    function current(address tokenIn, uint amountI
                                                                 function current(address tokenIn, uint amountI
n) external view returns (uint amountOut) {
                                                             n) external view returns (uint amountOut) {
```

```
Observation memory _observation = lastObser
                                                                          Observation memory observation = lastObser
    vation():
                                                                  vation():
                                                                          (uint reserve0Cumulative, uint reserve1Cumu
            (uint reserve0Cumulative, uint reserve1Cumu
    lative,) = currentCumulativePrices();
                                                                  lative,) = currentCumulativePrices();
            if (block.timestamp == _observation.timesta
                                                                          if (block.timestamp == _observation.timesta
    mp) {
                                                                  mp) {
                _observation = observations[observation
                                                                              _observation = observations[observation
    s.length-2];
                                                                  s.length-2];
304
            uint timeElapsed = block.timestamp - obser
                                                             304
                                                                          uint timeElapsed = block.timestamp - obser
    vation.timestamp;
                                                                  vation.timestamp;
            uint _reserve0 = (reserve0Cumulative - _obs
                                                                          uint _reserve0 = (reserve0Cumulative - _obs
    ervation.reserve0Cumulative) / timeElapsed;
                                                                  ervation.reserve0Cumulative) / timeElapsed;
            uint _reserve1 = (reserve1Cumulative - _obs
                                                                          uint _reserve1 = (reserve1Cumulative - _obs
    ervation.reserve1Cumulative) / timeElapsed;
                                                                  ervation.reserve1Cumulative) / timeElapsed;
            amountOut = _getAmountOut(amountIn, tokenI
                                                                          amountOut = _getAmountOut(amountIn, tokenI
       _reserve0, _reserve1);
                                                                     _reserve0, _reserve1);
        // as per `current`, however allows user config
                                                                      // as per `current`, however allows user config
    ured granularity, up to the full window size
                                                                  ured granularity, up to the full window size
        function quote(address tokenIn, uint amountIn,
                                                                      function quote(address tokenIn, uint amountIn,
     uint granularity) external view returns (uint amou
                                                                   uint granularity) external view returns (uint amou
    ntOut) {
                                                                  ntOut) {
            uint [] memory _prices = sample(tokenIn, am
                                                                          uint [] memory _prices = sample(tokenIn, am
    ountIn, granularity, 1);
                                                                  ountIn, granularity, 1);
            uint priceAverageCumulative;
                                                                          uint priceAverageCumulative;
            for (uint i = 0; i < _prices.length; i++) {</pre>
                                                                          for (uint i = 0; i < \_prices.length; i++) {
                priceAverageCumulative += _prices[i];
                                                                              priceAverageCumulative += _prices[i];
            return priceAverageCumulative / granularit
                                                                          return priceAverageCumulative / granularit
    у;
        // returns a memory set of twap prices
                                                                      // returns a memory set of twap prices
        function prices(address tokenIn, uint amountIn,
                                                                      function prices(address tokenIn, uint amountIn,
    uint points) external view returns (uint[] memory)
                                                                  uint points) external view returns (uint[] memory)
     {
                                                                   {
            return sample(tokenIn, amountIn, points,
                                                                          return sample(tokenIn, amountIn, points,
        }
                                                             324
        function sample(address tokenIn, uint amountIn,
                                                                      function sample(address tokenIn, uint amountIn,
    uint points, uint window) public view returns (uint
                                                                  uint points, uint window) public view returns (uint
    [] memory) {
                                                                  [] memory) {
            uint[] memory _prices = new uint[](points);
                                                                          uint[] memory _prices = new uint[](points);
            uint length = observations.length-1;
                                                                          uint length = observations.length-1;
            uint i = length - (points * window);
                                                                          uint i = length - (points * window);
            uint nextIndex = 0;
                                                                          uint nextIndex = 0;
            uint index = 0:
                                                                          uint index = 0:
331
            for (; i < length; i+=window) {</pre>
                                                                          for (; i < length; i+=window) {</pre>
                nextIndex = i + window;
                                                                              nextIndex = i + window;
                                                              334
                uint timeElapsed = observations[nextInd
                                                                              uint timeElapsed = observations[nextInd
    ex].timestamp - observations[i].timestamp;
                                                                  ex].timestamp - observations[i].timestamp;
                uint _reserve0 = (observations[nextInde
                                                                              uint _reserve0 = (observations[nextInde
    x].reserve0Cumulative - observations[i].reserve0Cum
                                                                  x].reserve0Cumulative - observations[i].reserve0Cum
    ulative) / timeElapsed;
                                                                  ulative) / timeElapsed;
                uint _reserve1 = (observations[nextInde
                                                                              uint _reserve1 = (observations[nextInde
    x].reserve1Cumulative - observations[i].reserve1Cum
                                                                  \verb|x||.reserve1Cumu| a tive - observations[i].reserve1Cum|
    ulative) / timeElapsed;
                                                                  ulative) / timeElapsed;
                _prices[index] = _getAmountOut(amountI
                                                                              _prices[index] = _getAmountOut(amountI
    n, tokenIn, _reserve0, _reserve1);
                                                                  n, tokenIn, _reserve0, _reserve1);
                index = index + 1;
                                                                              index = index + 1;
                                                                          return _prices;
            return _prices;
        }
```

```
// this low-level function should be called fro
                                                                      // this low-level function should be called fro
     m a contract which performs important safety checks
                                                                  m a contract which performs important safety checks
         // standard uniswap v2 implementation
                                                                      // standard uniswap v2 implementation
          function mint(address to) external lock returns
                                                                      function mint(address to) external lock returns
     (uint liquidity) {
                                                                   (uint liquidity) {
             (uint _reserve0, uint _reserve1) = (reserve
                                                                           (uint _reserve0, uint _reserve1) = (reserve
     0, reserve1);
                                                                  0, reserve1);
             uint _balance0 = erc20(token0).balance0f(ad
                                                                          uint _balance0 = erc20(token0).balance0f(ad
     dress(this));
                                                                  dress(this));
             uint _balance1 = erc20(token1).balance0f(ad
                                                                          uint _balance1 = erc20(token1).balance0f(ad
     dress(this));
                                                                  dress(this));
             uint _amount0 = _balance0 - _reserve0;
                                                                          uint _amount0 = _balance0 - _reserve0;
             uint _amount1 = _balance1 - _reserve1;
                                                                          uint _amount1 = _balance1 - _reserve1;
             uint _totalSupply = totalSupply; // gas sav
                                                                          uint _totalSupply = totalSupply; // gas sav
     ings, must be defined here since totalSupply can up
                                                                  ings, must be defined here since totalSupply can up
     date in _mintFee
                                                                  date in _mintFee
             if (_totalSupply == 0) {
                                                                          if (_totalSupply == 0) {
                  liquidity = Math.sqrt(_amount0 * _amoun
                                                                              liquidity = Math.sqrt(_amount0 * _amoun
     t1) - MINIMUM_LIQUIDITY;
                                                                  t1) - MINIMUM_LIQUIDITY;
                                                                              _mint(address(0), MINIMUM_LIQUIDITY);
                  _mint(address(0), MINIMUM_LIQUIDITY);
      // permanently lock the first MINIMUM_LIQUIDITY to
                                                                   // permanently lock the first MINIMUM_LIQUIDITY to
     kens
                                                                  kens
             } else {
                                                                           } else {
                  liquidity = Math.min(_amount0 * _totalS
                                                                              liquidity = Math.min(_amount0 * _totalS
     upply / _reserve0, _amount1 * _totalSupply / _reser
                                                                  upply / _reserve0, _amount1 * _totalSupply / _reser
     ve1);
                                                                   ve1);
             require(liquidity > 0, 'ILM'); // BaseV1: I
                                                                           require(liquidity > 0, 'ILM'); // BaseV1: I
     NSUFFICIENT_LIQUIDITY_MINTED
                                                                  NSUFFICIENT LIQUIDITY MINTED
             _mint(to, liquidity);
                                                                          _mint(to, liquidity);
363
             _update(_balance0, _balance1, _reserve0, _r
                                                                           _update(_balance0, _balance1, _reserve0, _r
     eserve1);
                                                                   eserve1);
             emit Mint(msg.sender, _amount0, _amount1);
                                                                           emit Mint(msg.sender, _amount0, _amount1);
          // this low-level function should be called fro
                                                                       // this low-level function should be called fro
     m a contract which performs important safety checks
                                                                  m a contract which performs important safety checks
          // standard uniswap v2 implementation
                                                                       // standard uniswap v2 implementation
          function burn(address to) external lock returns
                                                                       function burn(address to) external lock returns
     (uint amount0, uint amount1) {
                                                                   (uint amount0, uint amount1) {
             (uint _reserve0, uint _reserve1) = (reserve
                                                                           (uint _reserve0, uint _reserve1) = (reserve
     0, reserve1);
                                                                  0, reserve1);
             (address token0, address token1) = (token
                                                                           (address _token0, address _token1) = (token
     0, token1);
                                                                  0, token1);
             uint _balance0 = erc20(_token0).balance0f(a
                                                                           uint _balance0 = erc20(_token0).balance0f(a
     ddress(this));
                                                                  ddress(this));
             uint _balance1 = erc20(_token1).balance0f(a
                                                                           uint _balance1 = erc20(_token1).balance0f(a
     ddress(this));
                                                                  ddress(this));
             uint _liquidity = balanceOf[address(this)];
                                                                          uint _liquidity = balanceOf[address(this)];
             uint _totalSupply = totalSupply; // gas sav
                                                                          uint _totalSupply = totalSupply; // gas sav
     ings, must be defined here since totalSupply can up
                                                                  ings, must be defined here since totalSupply can up
     date in _mintFee
                                                                  date in _mintFee
             amount0 = _liquidity * _balance0 / _totalSu
                                                                          amount0 = _liquidity * _balance0 / _totalSu
     pply; // using balances ensures pro-rata distributi
                                                                  pply; // using balances ensures pro-rata distributi
             amount1 = _liquidity * _balance1 / _totalSu
                                                                          amount1 = _liquidity * _balance1 / _totalSu
     pply; // using balances ensures pro-rata distributi
                                                                  pply; // using balances ensures pro-rata distributi
             require(amount0 > 0 && amount1 > 0, 'ILB');
                                                                           require(amount0 > 0 && amount1 > 0, 'ILB');
     // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
                                                                   // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
             _burn(address(this), _liquidity);
                                                                           _burn(address(this), _liquidity);
             _safeTransfer(_token0, to, amount0);
                                                                          _safeTransfer(_token0, to, amount0);
             _safeTransfer(_token1, to, amount1);
                                                                          _safeTransfer(_token1, to, amount1);
```

```
s(this));
                                                                   s(this));
              _balance1 = erc20(_token1).balance0f(addres
                                                                           _balance1 = erc20(_token1).balance0f(addres
                                                              384
      s(this));
                                                                   s(this));
              _update(_balance0, _balance1, _reserve0, _r
                                                                           _update(_balance0, _balance1, _reserve0, _r
      eserve1);
                                                                   eserve1);
             emit Burn(msg.sender, amount0, amount1, t
                                                                           emit Burn(msg.sender, amount0, amount1, t
          // this low-level function should be called fro
                                                                       // this low-level function should be called fro
      m a contract which performs important safety checks
                                                                   m a contract which performs important safety checks
          function swap(uint amount00ut, uint amount10ut,
                                                                       function swap(uint amount00ut, uint amount10ut,
      address to, bytes calldata data) external lock {
                                                                   address to, bytes calldata data) external lock {
              require(!BaseV1Factory(factory).isPaused
                                                                           require(!BaseV1Factory(factory).isPaused
      ());
                                                                   ());
              require(amount00ut > 0 || amount10ut > 0,
                                                                           require(amount00ut > 0 || amount10ut > 0,
       'IOA'); // BaseV1: INSUFFICIENT_OUTPUT_AMOUNT
                                                                    'IOA'); // BaseV1: INSUFFICIENT_OUTPUT_AMOUNT
              (uint _reserve0, uint _reserve1) = (reserv
                                                                           (uint _reserve0, uint _reserve1) = (reserv
      e0, reserve1);
                                                                   e0, reserve1);
                                                                           require(amount00ut < _reserve0 && amount10u
              require(amount00ut < _reserve0 && amount10u</pre>
      t < _reserve1, 'IL'); // BaseV1: INSUFFICIENT_LIQUI
                                                                   t < _reserve1, 'IL'); // BaseV1: INSUFFICIENT_LIQUI
      DITY
                                                                   DTTY
             uint _balance0;
                                                                           uint _balance0;
             uint _balance1;
                                                                           uint _balance1;
              { // scope for _token{0,1}, avoids stack to
                                                                           { // scope for _token{0,1}, avoids stack to
      o deep errors
                                                                   o deep errors
              (address _token0, address _token1) = (token
                                                                           (address _token0, address _token1) = (token
      0, token1);
                                                                   0, token1);
              require(to != token0 && to != token1, 'I
                                                                           require(to != token0 && to != token1, 'I
      T'); // BaseV1: INVALID_TO
                                                                   T'); // BaseV1: INVALID_TO
402
              if (amount00ut > 0) _safeTransfer(_token0,
                                                                           if (amount00ut > 0) _safeTransfer(_token0,
       to, amount00ut); // optimistically transfer tokens
                                                                    to, amount00ut); // optimistically transfer tokens
              if (amount10ut > 0) _safeTransfer(_token1,
                                                                           if (amount10ut > 0) _safeTransfer(_token1,
       to, amount10ut); // optimistically transfer tokens
                                                                    to, amount10ut); // optimistically transfer tokens
              if (data.length > 0) IBaseV1Callee(to).hook
                                                                           if (data.length > 0) IBaseV1Callee(to).hook
      (msg.sender, amount00ut, amount10ut, data); // call
                                                                   (msg.sender, amount00ut, amount10ut, data); // call
      back, used for flash loans
                                                                   back, used for flash loans
              _balance0 = erc20(_token0).balanceOf(addres
                                                                           _balance0 = erc20(_token0).balanceOf(addres
      s(this));
                                                                   s(this));
              _balance1 = erc20(_token1).balance0f(addres
                                                                           _balance1 = erc20(_token1).balanceOf(addres
      s(this));
                                                                   s(this));
             uint amount0In = balance0 > reserve0 - am
                                                                           uint amount0In = balance0 > reserve0 - am
      ount00ut ? _balance0 - (_reserve0 - amount00ut) :
                                                                   ount00ut ? _balance0 - (_reserve0 - amount00ut) :
             uint amount1In = _balance1 > _reserve1 - am
                                                                           uint amount1In = _balance1 > _reserve1 - am
      ount10ut ? _balance1 - (_reserve1 - amount10ut) :
                                                                   ount10ut ? _balance1 - (_reserve1 - amount10ut) :
       Θ;
              require(amount0In > 0 || amount1In > 0, 'II
                                                                           require(amount0In > 0 || amount1In > 0, 'II
      A'); // BaseV1: INSUFFICIENT_INPUT_AMOUNT
                                                                   A'); // BaseV1: INSUFFICIENT_INPUT_AMOUNT
              { // scope for reserve{0,1}Adjusted, avoids
                                                                           { // scope for reserve{0,1}Adjusted, avoids
      stack too deep errors
                                                                   stack too deep errors
              (address _token0, address _token1) = (token
                                                                           (address _token0, address _token1) = (token
      0, token1);
              if (amount0In > 0) _update0(amount0In / 100
                                                                           if (amount0In > 0) _update0(amount0In / 100
      00); // accrue fees for token0 and move them out of
                                                                   00); // accrue fees for token0 and move them out of
      loog
                                                                   loog
              if (amount1In > 0) update1(amount1In / 100
                                                                           if (amount1In > 0) update1(amount1In / 100
      00); // accrue fees for token1 and move them out of
                                                                   00); // accrue fees for token1 and move them out of
      pool
                                                                   pool
              _balance0 = erc20(_token0).balanceOf(addres
                                                                           _balance0 = erc20(_token0).balanceOf(addres
      s(this)); // since we removed tokens, we need to re
                                                                   s(this)); // since we removed tokens, we need to re
      confirm balances, can also simply use previous bala
                                                                   confirm balances, can also simply use previous bala
      nce - amountIn/ 10000, but doing balanceOf again as
                                                                   nce - amountIn/ 10000, but doing balanceOf again as
```

safety check

_balance0 = erc20(_token0).balanceOf(addres

_balance0 = erc20(_token0).balance0f(addres

safety check

```
_balance1 = erc20(_token1).balanceOf(addres
                                                                           _balance1 = erc20(_token1).balance0f(addres
     s(this));
                                                                  s(this));
             // The curve, either x3y+y3x for stable poo
                                                                          // The curve, either x3y+y3x for stable poo
     ls, or x*y for volatile pools
                                                                   ls, or x*y for volatile pools
             require(_k(_balance0, _balance1) >= _k(_res
                                                                          require(_k(_balance0, _balance1) >= _k(_res
     erve0, _reserve1), 'K'); // BaseV1: K
                                                                   erve0, _reserve1), 'K'); // BaseV1: K
             _update(_balance0, _balance1, _reserve0, _r
                                                                           _update(_balance0, _balance1, _reserve0, _r
     eserve1);
                                                                   eserve1);
             emit Swap(msg.sender, amount0In, amount1In,
                                                                          emit Swap(msg.sender, amount0In, amount1In,
     amount00ut, amount10ut, to);
                                                                   amount00ut, amount10ut, to);
                                                              424
          // force balances to match reserves
                                                                       // force balances to match reserves
          function skim(address to) external lock {
                                                                       function skim(address to) external lock {
             (address _token0, address _token1) = (token
                                                                           (address _token0, address _token1) = (token
     0, token1);
                                                                   0, token1);
             _safeTransfer(_token0, to, erc20(_token0).b
                                                                           _safeTransfer(_token0, to, erc20(_token0).b
     alanceOf(address(this)) - (reserve0));
                                                                   alanceOf(address(this)) - (reserve0));
             _safeTransfer(_token1, to, erc20(_token1).b
                                                                          _safeTransfer(_token1, to, erc20(_token1).b
     alanceOf(address(this)) - (reserve1));
                                                                   alanceOf(address(this)) - (reserve1));
         // force reserves to match balances
                                                                       // force reserves to match balances
         function sync() external lock {
                                                                      function sync() external lock {
             _update(erc20(token0).balanceOf(address(thi
                                                                           _update(erc20(token0).balanceOf(address(thi
     s)), erc20(token1).balanceOf(address(this)), reserv
                                                                   s)), erc20(token1).balanceOf(address(this)), reserv
     e0, reserve1);
                                                                   e0, reserve1);
         function _f(uint x0, uint y) internal pure retu
                                                                       function _f(uint x0, uint y) internal pure retu
     rns (uint) {
                                                                   rns (uint) {
438
             return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e1
                                                                          return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e1
     8*x0/1e18)*y/1e18;
                                                                   8*x0/1e18)*y/1e18;
         function _d(uint x0, uint y) internal pure retu
                                                                       function _d(uint x0, uint y) internal pure retu
     rns (uint) {
                                                                   rns (uint) {
             return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/
                                                                           return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/
     1e18);
                                                                   1e18);
                                                              444
          function _get_y(uint x0, uint xy, uint y) inter
                                                                       function _get_y(uint x0, uint xy, uint y) inter
     nal pure returns (uint) {
                                                                   nal pure returns (uint) {
             for (uint i = 0; i < 255; i++) {
                                                                          for (uint i = 0; i < 255; i++) {
                 uint y_prev = y;
                                                                               uint y_prev = y;
                 uint k = f(x0, y);
                                                                               uint k = f(x0, y);
                 if (k < xy) {
                                                                               if (k < xy) {
                      uint dy = (xy - k)*1e18/_d(x0, y);
                                                                                   uint dy = (xy - k)*1e18/_d(x0, y);
                      y = y + dy;
                                                                                   y = y + dy;
                  } else {
                                                                               } else {
                      uint dy = (k - xy)*1e18/_d(x0, y);
                                                                                   uint dy = (k - xy)*1e18/_d(x0, y);
                      y = y - dy;
                                                                                   y = y - dy;
 454
                  if (y > y_prev) {
                                                                               if (y > y_prev) {
                      if (y - y_prev <= 1) {
                                                                                   if (y - y_prev <= 1) {
                          return y;
                                                                                       return y;
                  } else {
                                                                               } else {
                      if (y_prev - y <= 1) {
                                                                                   if (y_prev - y <= 1) {</pre>
                          return y;
                                                                                       return y;
                      }
                                                                               }
                                                                           return y;
             return y;
```

```
function getAmountOut(uint amountIn, address to
                                                                function getAmountOut(uint amountIn, address to
                                                            kenIn) external view returns (uint) {
kenIn) external view returns (uint) {
       (uint _reserve0, uint _reserve1) = (reserve
                                                                    (uint _reserve0, uint _reserve1) = (reserve
0, reserve1);
                                                            0, reserve1);
       amountIn -= amountIn / 10000; // remove fee
                                                                    amountIn -= amountIn / 10000; // remove fee
from amount received
                                                             from amount received
       return _getAmountOut(amountIn, tokenIn, _re
                                                                    return _getAmountOut(amountIn, tokenIn, _re
serve0, _reserve1);
                                                            serve0, _reserve1);
    function _getAmountOut(uint amountIn, address t
                                                                function _getAmountOut(uint amountIn, address t
okenIn, uint _reserve0, uint _reserve1) internal vi
                                                            okenIn, uint _reserve0, uint _reserve1) internal vi
ew returns (uint) {
                                                            ew returns (uint) {
       if (stable) {
                                                                    if (stable) {
           uint xy = _k(_reserve0, _reserve1);
                                                                        uint xy = _k(_reserve0, _reserve1);
           _reserve0 = _reserve0 * 1e18 / decimals
                                                                        _reserve0 = _reserve0 * 1e18 / decimals
0:
                                                            0:
            _reserve1 = _reserve1 * 1e18 / decimals
                                                                        _reserve1 = _reserve1 * 1e18 / decimals
            (uint reserveA, uint reserveB) = tokenI
                                                                         (uint reserveA, uint reserveB) = tokenI
n == token0 ? (_reserve0, _reserve1) : (_reserve1,
                                                            n == token0 ? (_reserve0, _reserve1) : (_reserve1,
 reserve0);
                                                             _reserve0);
           amountIn = tokenIn == token0 ? amountIn
                                                                        amountIn = tokenIn == token0 ? amountIn
 1e18 / decimals0 : amountIn * 1e18 / decimals1;
                                                              1e18 / decimals0 : amountIn * 1e18 / decimals1;
           uint y = reserveB - _get_y(amountIn+res
                                                                        uint y = reserveB - _get_y(amountIn+res
erveA, xy, reserveB);
                                                            erveA, xy, reserveB);
            return y * (tokenIn == token0 ? decimal
                                                                        return y * (tokenIn == token0 ? decimal
s1 : decimals0) / 1e18;
                                                            s1 : decimals0) / 1e18;
       } else {
                                                                    } else {
            (uint reserveA, uint reserveB) = tokenI
                                                                         (uint reserveA, uint reserveB) = tokenI
n == token0 ? (_reserve0, _reserve1) : (_reserve1,
                                                            n == token0 ? (_reserve0, _reserve1) : (_reserve1,
 _reserve0);
                                                              _reserve0);
           return amountIn * reserveB / (reserveA
                                                                        return amountIn * reserveB / (reserveA
 + amountIn);
                                                              + amountIn);
    function _k(uint x, uint y) internal view retur
                                                                 function _k(uint x, uint y) internal view retur
ns (uint) {
                                                            ns (uint) {
       if (stable) {
                                                                    if (stable) {
            uint _x = x * 1e18 / decimals0;
                                                                        uint _x = x * 1e18 / decimals0;
            uint _y = y * 1e18 / decimals1;
                                                                        uint _y = y * 1e18 / decimals1;
            uint _a = (_x * _y) / 1e18;
                                                                        uint _a = (_x * _y) / 1e18;
           uint _b = ((_x * _x) / 1e18 + (_y * _y)
                                                                        uint _b = ((_x * _x) / 1e18 + (_y * _y)
/ 1e18);
                                                             / 1e18);
           return _a * _b / 1e18; // x3y+y3x >= k
                                                                        return _a * _b / 1e18; // x3y+y3x >= k
       } else {
                                                                    } else {
            return x * y; // xy >= k
                                                                        return x * y; // xy >= k
    function _mint(address dst, uint amount) intern
                                                                function _mint(address dst, uint amount) intern
al {
                                                            al {
        _updateFor(dst); // balances must be update
                                                                     _updateFor(dst); // balances must be update
d on mint/burn/transfer
                                                            d on mint/burn/transfer
       totalSupply += amount;
                                                                    totalSupply += amount;
       balanceOf[dst] += amount;
                                                                    balanceOf[dst] += amount;
       emit Transfer(address(0), dst, amount);
                                                                     emit Transfer(address(0), dst, amount);
    function burn(address dst, uint amount) intern
                                                                function burn(address dst, uint amount) intern
al {
                                                            al {
       _updateFor(dst);
                                                                    _updateFor(dst);
       totalSupply -= amount;
                                                                    totalSupply -= amount;
       balanceOf[dst] -= amount;
                                                                    balanceOf[dst] -= amount;
        emit Transfer(dst, address(0), amount);
                                                                     emit Transfer(dst, address(0), amount);
```

```
function approve(address spender, uint amount)
                                                                 function approve(address spender, uint amount)
external returns (bool) {
                                                              external returns (bool) {
                                                                     allowance[msg.sender][spender] = amount;
       allowance[msg.sender][spender] = amount;
        emit Approval(msg.sender, spender, amount);
                                                                     emit Approval(msg.sender, spender, amount);
        return true;
                                                                     return true;
    function permit(address owner, address spender,
                                                                 function permit(address owner, address spender,
uint value, uint deadline, uint8 v, bytes32 r, byte
                                                             uint value, uint deadline, uint8 v, bytes32 r, byte
s32 s) external {
                                                             s32 s) external {
        require(deadline >= block.timestamp, 'BaseV
                                                                     require(deadline >= block.timestamp, 'BaseV
1: EXPIRED');
                                                             1: EXPIRED');
       DOMAIN_SEPARATOR = keccak256(
                                                                     DOMAIN_SEPARATOR = keccak256(
           abi.encode(
                                                                         abi.encode(
                keccak256('EIP712Domain(string nam
                                                                             keccak256('EIP712Domain(string nam
e,string version,uint256 chainId,address verifyingC
                                                             e, string version, uint256 chainId, address verifyingC
ontract)'),
                                                             ontract)'),
                                                                             keccak256(bytes(name)),
                keccak256(bytes(name)),
                keccak256('1')
                                                                             keccak256(bytes('1')),
                block.chainid,
                                                                             block.chainid,
                address(this)
                                                                             address(this)
       bytes32 digest = kec_{cak256}(
                                                                     bytes32 digest = keccak256(
                                                         534
            abi.encodePacked(
                                                                         abi.encodePacked(
                '\x19\x01',
                                                                              '\x19\x01',
                DOMAIN_SEPARATOR,
                                                                             DOMAIN_SEPARATOR,
                keccak256(abi.encode(PERMIT_TYPEHAS
                                                                             keccak256(abi.encode(PERMIT_TYPEHAS
H, owner, spender, value, nonces[owner]++, deadlin
                                                             H, owner, spender, value, nonces[owner]++, deadlin
e))
        address recoveredAddress = ecrecover(diges
                                                                     address recoveredAddress = ecrecover(diges
        require(recoveredAddress != address(0) && r
                                                                     require(recoveredAddress != address(0) && r
ecoveredAddress == owner, 'BaseV1: INVALID_SIGNATUR
                                                             ecoveredAddress == owner, 'BaseV1: INVALID_SIGNATUR
E');
        allowance[owner][spender] = value;
                                                                     allowance[owner][spender] = value;
        emit Approval(owner, spender, value);
                                                                     emit Approval(owner, spender, value);
    function transfer(address dst, uint amount) ext
                                                                 function transfer(address dst, uint amount) ext
ernal returns (bool) {
                                                             ernal returns (bool) {
        _transferTokens(msg.sender, dst, amount);
                                                                     _transferTokens(msg.sender, dst, amount);
        return true:
                                                                     return true:
   }
    function transferFrom(address src, address dst,
                                                                 function transferFrom(address src, address dst,
uint amount) external returns (bool) {
                                                             uint amount) external returns (bool) {
        address spender = msg.sender;
                                                                     address spender = msg.sender;
       uint spenderAllowance = allowance[src][spen
                                                                     uint spenderAllowance = allowance[src][spen
der1:
                                                             der];
        if (spender != src && spenderAllowance != t
                                                                     if (spender != src && spenderAllowance != t
ype(uint).max) {
                                                             ype(uint).max) {
            uint newAllowance = spenderAllowance -
                                                                         uint newAllowance = spenderAllowance -
amount:
                                                              amount:
            allowance[src][spender] = newAllowance;
                                                                         allowance[src][spender] = newAllowance;
            emit Approval(src, spender, newAllowanc
                                                                         emit Approval(src, spender, newAllowanc
        _transferTokens(src, dst, amount);
                                                                     _transferTokens(src, dst, amount);
        return true;
                                                                     return true;
```

```
function _transferTokens(address src, address d
                                                                      function _transferTokens(address src, address d
    st, uint amount) internal {
                                                                 st, uint amount) internal {
            _updateFor(src); // update fee position for
                                                                         _updateFor(src); // update fee position for
    src
                                                                 src
            _updateFor(dst); // update fee position for
                                                                         _updateFor(dst); // update fee position for
    dst
                                                                 dst
            balanceOf[src] -= amount;
                                                                         balanceOf[src] -= amount;
            balanceOf[dst] += amount;
                                                                         balanceOf[dst] += amount;
            emit Transfer(src, dst, amount);
                                                                         emit Transfer(src, dst, amount);
        function _safeTransfer(address token,address t
                                                                     function _safeTransfer(address token,address t
    o,uint256 value) internal {
                                                                 o,uint256 value) internal {
            require(token.code.length > 0);
                                                                         require(token.code.length > 0);
            (bool success, bytes memory data) =
                                                                         (bool success, bytes memory data) =
            token.call(abi.encodeWithSelector(erc20.tra
                                                                         token.call(abi.encodeWithSelector(erc20.tra
    nsfer.selector, to, value));
                                                                 nsfer.selector, to, value));
            require(success && (data.length == 0 || ab
                                                                         require(success && (data.length == 0 \mid \mid ab
    i.decode(data, (bool))));
                                                                  i.decode(data, (bool))));
586 contract BaseV1Factory {
                                                             586 contract BaseV1Factory {
        bool public isPaused;
                                                                     bool public isPaused;
        address public pauser;
                                                                     address public pauser;
        address public pendingPauser;
                                                                     address public pendingPauser;
        mapping(address => mapping(address => mapping(b
                                                                     mapping(address => mapping(address => mapping(b
    ool => address))) public getPair;
                                                                 ool => address))) public getPair;
        address[] public allPairs;
                                                                     address[] public allPairs;
                                                                     mapping(address => bool) public isPair; // simp
        mapping(address => bool) public isPair; // simp
    lified check if its a pair, given that `stable` fla
                                                                 lified check if its a pair, given that `stable` fla
    g might not be available in peripherals
                                                                 g might not be available in peripherals
        address internal _temp0;
                                                                     address internal _temp0;
        address internal _temp1;
                                                                     address internal _temp1;
        bool internal _temp;
                                                                     bool internal _temp;
        event PairCreated(address indexed token0, addre
                                                                      event PairCreated(address indexed token0, addre
    ss indexed token1, bool stable, address pair, uin
                                                                 ss indexed token1, bool stable, address pair, uin
        constructor() {
                                                                     constructor() {
            pauser = msg.sender;
                                                                         pauser = msg.sender;
            isPaused = false;
                                                                         isPaused = false;
                                                             604
        function allPairsLength() external view returns
                                                                      function allPairsLength() external view returns
                                                                 (uint) {
    (uint) {
            return allPairs.length;
                                                                         return allPairs.length;
        function setPauser(address _pauser) external {
                                                                      function setPauser(address _pauser) external {
            require(msg.sender == pauser);
                                                                         require(msg.sender == pauser);
            pendingPauser = _pauser;
                                                                         pendingPauser = _pauser;
        function acceptPauser() external {
                                                                      function acceptPauser() external {
            require(msg.sender == pendingPauser);
                                                                         require(msg.sender == pendingPauser);
            pauser = pendingPauser;
                                                                         pauser = pendingPauser;
        function setPause(bool _state) external {
                                                                      function setPause(bool _state) external {
            require(msg.sender == pauser);
                                                                         require(msg.sender == pauser);
            isPaused = _state;
                                                                         isPaused = _state;
                                                             624
```

```
function pairCodeHash() external pure returns
                                                                     function pairCodeHash() external pure returns
     (bvtes32) {
                                                                  (bvtes32) {
            return keccak256(type(BaseV1Pair).creationC
                                                                         return keccak256(type(BaseV1Pair).creationC
    ode);
                                                                 ode);
        function getInitializable() external view retur
                                                                     function getInitializable() external view retur
    ns (address, address, bool) {
                                                                 ns (address, address, bool) {
            return (_temp0, _temp1, _temp);
                                                                         return (_temp0, _temp1, _temp);
        function createPair(address tokenA, address tok
                                                                     function createPair(address tokenA, address tok
    enB, bool stable) external returns (address pair) {
                                                                 enB, bool stable) external returns (address pair) {
            require(tokenA != tokenB, 'IA'); // BaseV1:
                                                                         require(tokenA != tokenB, 'IA'); // BaseV1:
    IDENTICAL_ADDRESSES
                                                                 IDENTICAL_ADDRESSES
           (address token0, address token1) = tokenA <</pre>
                                                                         (address token0, address token1) = tokenA <</pre>
    tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
                                                                 tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
           require(token0 != address(0), 'ZA'); // Bas
                                                                        require(token0 != address(0), 'ZA'); // Bas
    eV1: ZERO ADDRESS
                                                                 eV1: ZERO ADDRESS
            require(getPair[token0][token1][stable] ==
                                                                         require(getPair[token0][token1][stable] ==
     address(0), 'PE'); // BaseV1: PAIR_EXISTS - single
                                                                  address(0), 'PE'); // BaseV1: PAIR_EXISTS - single
    check is sufficient
                                                                 check is sufficient
            bytes32 salt = keccak256(abi.encodePacked(t
                                                                         bytes32 salt = keccak256(abi.encodePacked(t
    oken0, token1, stable)); // notice salt includes st
                                                                 oken0, token1, stable)); // notice salt includes st
    able as well, 3 parameters
                                                                 able as well, 3 parameters
            (_temp0, _temp1, _temp) = (token0, token1,
                                                                         (_temp0, _temp1, _temp) = (token0, token1,
     stable);
                                                                  stable);
            pair = address(new BaseV1Pair{salt:salt}
                                                                         pair = address(new BaseV1Pair{salt:salt}
    ());
                                                                 ());
            getPair[token0][token1][stable] = pair;
                                                                         getPair[token0][token1][stable] = pair;
            getPair[token1][token0][stable] = pair; //
                                                                         getPair[token1][token0][stable] = pair; //
     populate mapping in the reverse direction
                                                                  populate mapping in the reverse direction
            allPairs.push(pair);
                                                                         allPairs.push(pair);
            isPair[pair] = true;
                                                                         isPair[pair] = true;
            emit PairCreated(token0, token1, stable, pa
                                                                         emit PairCreated(token0, token1, stable, pa
    ir, allPairs.length);
                                                                 ir, allPairs.length);
648 }
                                                             648 }
```