SOLIDITY.*FINANCE*

Bees.finance Fund - Smart Contract Audit Report

SUMMARY

Bees Finance is an upcoming ecosystem to empower simple and secure DeFi.

The Bees.finance consists of 3 Smart Contracts: Two token contracts and one Chef contract to support staking and rewards functions. The token contracts include all the standard ERC20 functions as defined in the ERC20 protocol by the Ethereum Foundation.

> - *Bees token's total supply has been minted. No accessible mint functions exist, though there is a burn function to reduce supply.*
> - *HoneyJar token is minted by the Chef contract. The deployer has transfered ownership of this token to the Chef Contract in this transaction.*
> - *$20,000 in liquidity locked until 2023.*
> - *Ownership - Some functions are protected and can only be called by the contract owner. The deployer and future owners can transfer ownership to any address.*
> - *Utilization of SafeMath to prevent overflows.*
>
> *Audit Findings Summary*
>
> - *No security issues were identified.*
> - *Date: October 29th, 2020*

COMBINED AUDIT RESULTS

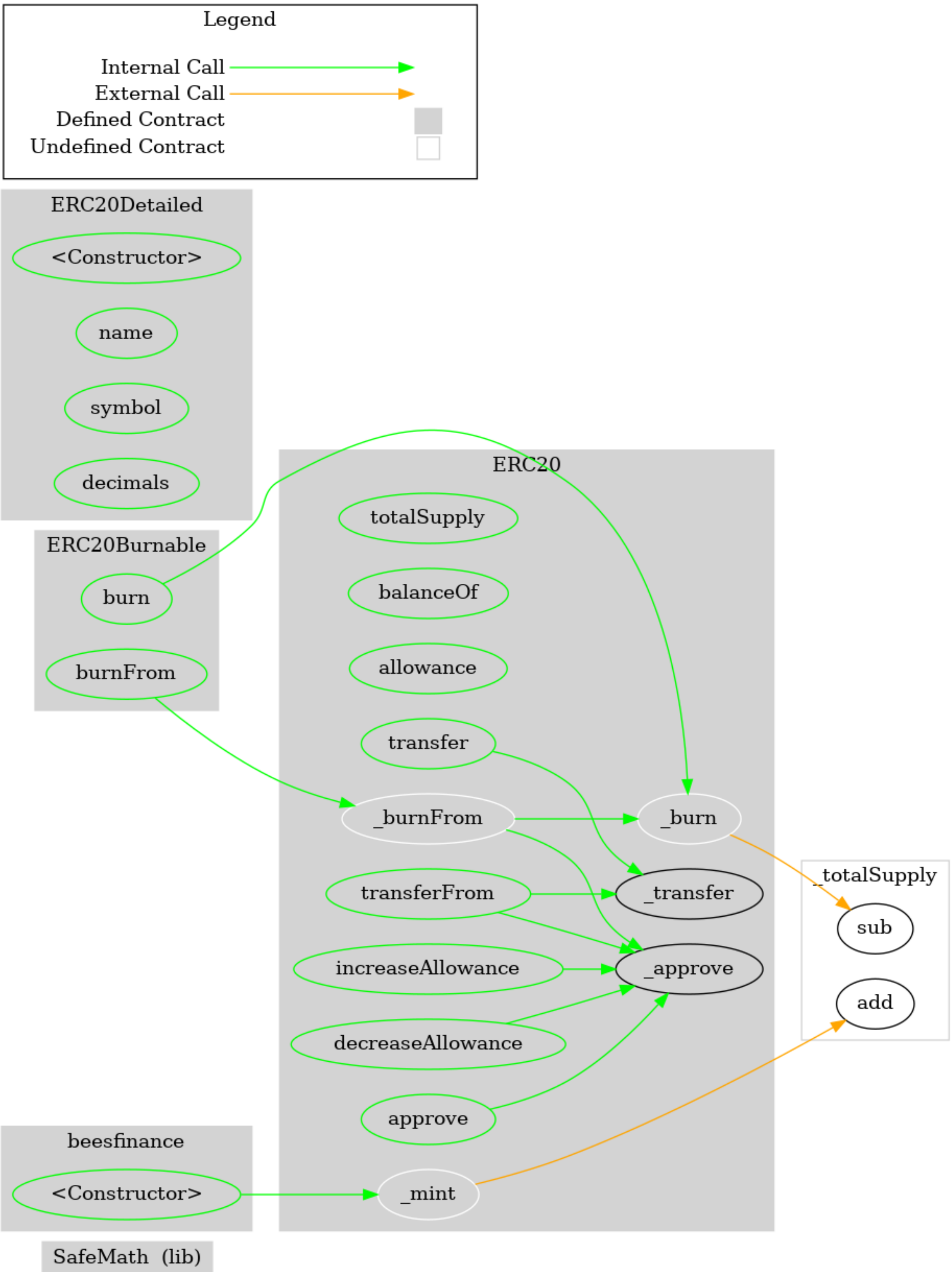
> *We ran over 400,000 transactions interacting with this suite of contracts on a test blockchain to determine these results.*
>
> *Date: October 4th, 2020*

Vulnerability Category	***Notes***	***Result***
Arbitrary Storage Write	*N/A*	*PASS*

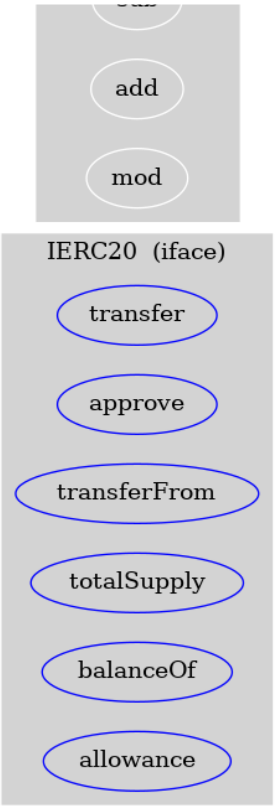
Vulnerability Category	*Notes*	*Result*
Delegate Call to Untrusted Contract	*N/A*	*PASS*
Dependence on Predictable Variables	*N/A*	*PASS*
Deprecated Opcodes	*N/A*	*PASS*
Ether Thief	*N/A*	*PASS*
Exceptions	*N/A*	*PASS*
External Calls	*N/A*	*PASS*
Integer Over/Underflow	*N/A*	*PASS*
Multiple Sends	*N/A*	*PASS*
Suicide	*N/A*	*PASS*
State Change External Calls	*N/A*	*PASS*
Unchecked Retval	*N/A*	*PASS*
User Supplied Assertion	*N/A*	*PASS*
Critical Solidity Compiler	*N/A*	*PASS*

DETAILS: BEESFINANCE TOKEN

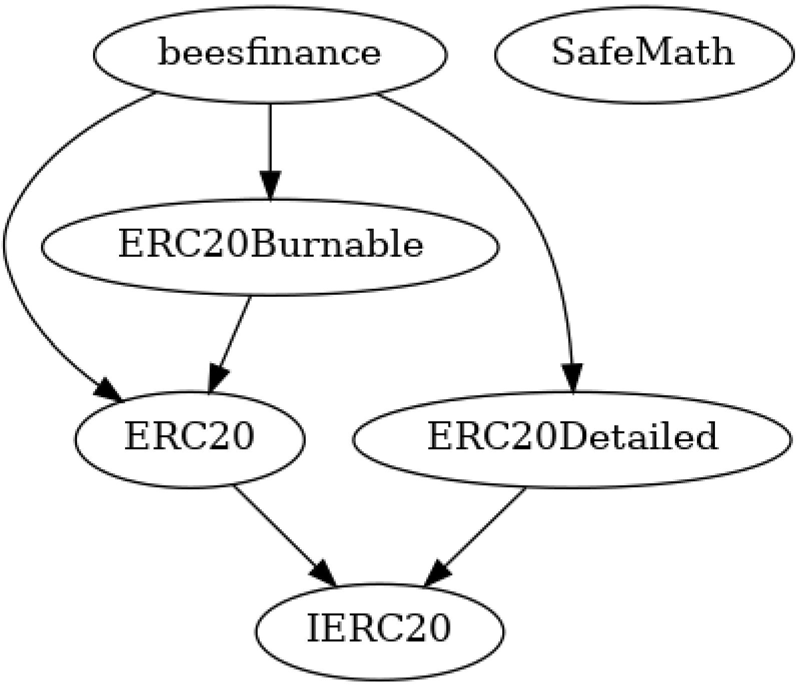
FUNCTION GRAPH



Please review our Terms & Conditions, Privacy Policy, and other legal information here. By using this site, you explicitly agree to these terms.



INHERITENCE CHART



```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+ [Int] IERC20
   - [Ext] transfer #
   - [Ext] approve #
   - [Ext] transferFrom #
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] allowance

+ [Lib] SafeMath
   - [Int] mul
   - [Int] div
   - [Int] sub
   - [Int] add
   - [Int] mod

+  ERC20 (IERC20)
   - [Pub] totalSupply
   - [Pub] balanceOf
   - [Pub] allowance
   - [Pub] transfer #
   - [Pub] approve #
   - [Pub] transferFrom #
   - [Int] _mint #
   - [Int] _burn #
   - [Int] _approve #
   - [Int] _burnFrom #

+  ERC20Detailed (IERC20)
   - [Pub]  #
   - [Pub] name
   - [Pub] symbol
   - [Pub] decimals

+  ERC20Burnable (ERC20)
   - [Pub] burn #
   - [Pub] burnFrom #

+  beesfinance (ERC20, ERC20Detailed, ERC20Burnable
   - [Pub]  #
      - modifiers: ERC20Detailed
```

SOURCE CODE

Click here to download the source code as a .sol file.

```
/**
 *Submitted for verification at Etherscan.io on 2020
```

```
// File: openzeppelin-solidity/contracts/token/ERC20

/**
 * https://bees.finance/
 */

interface IERC20 {
    function transfer(address to, uint256 value) ext

    function approve(address spender, uint256 value)

    function transferFrom(address from, address to,

    function totalSupply() external view returns (ui

    function balanceOf(address who) external view re

    function allowance(address owner, address spende

    event Transfer(address indexed from, address ind

    event Approval(address indexed owner, address in
}

// File: openzeppelin-solidity/contracts/math/SafeMa

pragma solidity ^0.5.2;

/**
 * @title SafeMath
```

```
    /**
     * @dev Multiplies two unsigned integers, revert
     */
    function mul(uint256 a, uint256 b) internal pure
        // Gas optimization: this is cheaper than re
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/open
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
     * @dev Integer division of two unsigned integer
     */
    function div(uint256 a, uint256 b) internal pure
        // Solidity only automatically asserts when 
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is n

        return c;
    }

    /**
     * @dev Subtracts two unsigned integers, reverts
```

```
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Adds two unsigned integers, reverts on o
     */
    function add(uint256 a, uint256 b) internal pure
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
     * @dev Divides two unsigned integers and return
     * reverts when dividing by zero.
     */
    function mod(uint256 a, uint256 b) internal pure
        require(b != 0);
        return a % b;
    }
}

// File: openzeppelin-solidity/contracts/token/ERC20

pragma solidity ^0.5.2;

/**
 * @title Standard ERC20 token
 *
 * This implementation emits additional Approval eve
 * all accounts just by listening to said events. No
 * compliant implementations may not do it.
 */
contract ERC20 is IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)

    uint256 private _totalSupply;

    /**
     * @dev Total number of tokens in existence
     */
    function totalSupply() public view returns (uint
        return _totalSupply;
    }

    /**
     * @dev Gets the balance of the specified addres
     * @param owner The address to query the balance
     * @return A uint256 representing the amount own
     */
    function balanceOf(address owner) public view re
        return _balances[owner];
    }

    /**
```

```
     * @return A uint256 specifying the amount of to
     */
    function allowance(address owner, address spende
        return _allowed[owner][spender];
    }

    /**
     * @dev Transfer token to a specified address
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    function transfer(address to, uint256 value) pub
        _transfer(msg.sender, to, value);
        return true;
    }

    /**
     * @dev Approve the passed address to spend the 
     * Beware that changing an allowance with this m
     * and the new allowance by unfortunate transact
     * race condition is to first reduce the spender
     * https://github.com/ethereum/EIPs/issues/20#is
     * @param spender The address which will spend t
     * @param value The amount of tokens to be spent
     */
    function approve(address spender, uint256 value)
        _approve(msg.sender, spender, value);
        return true;
    }

    /**
```

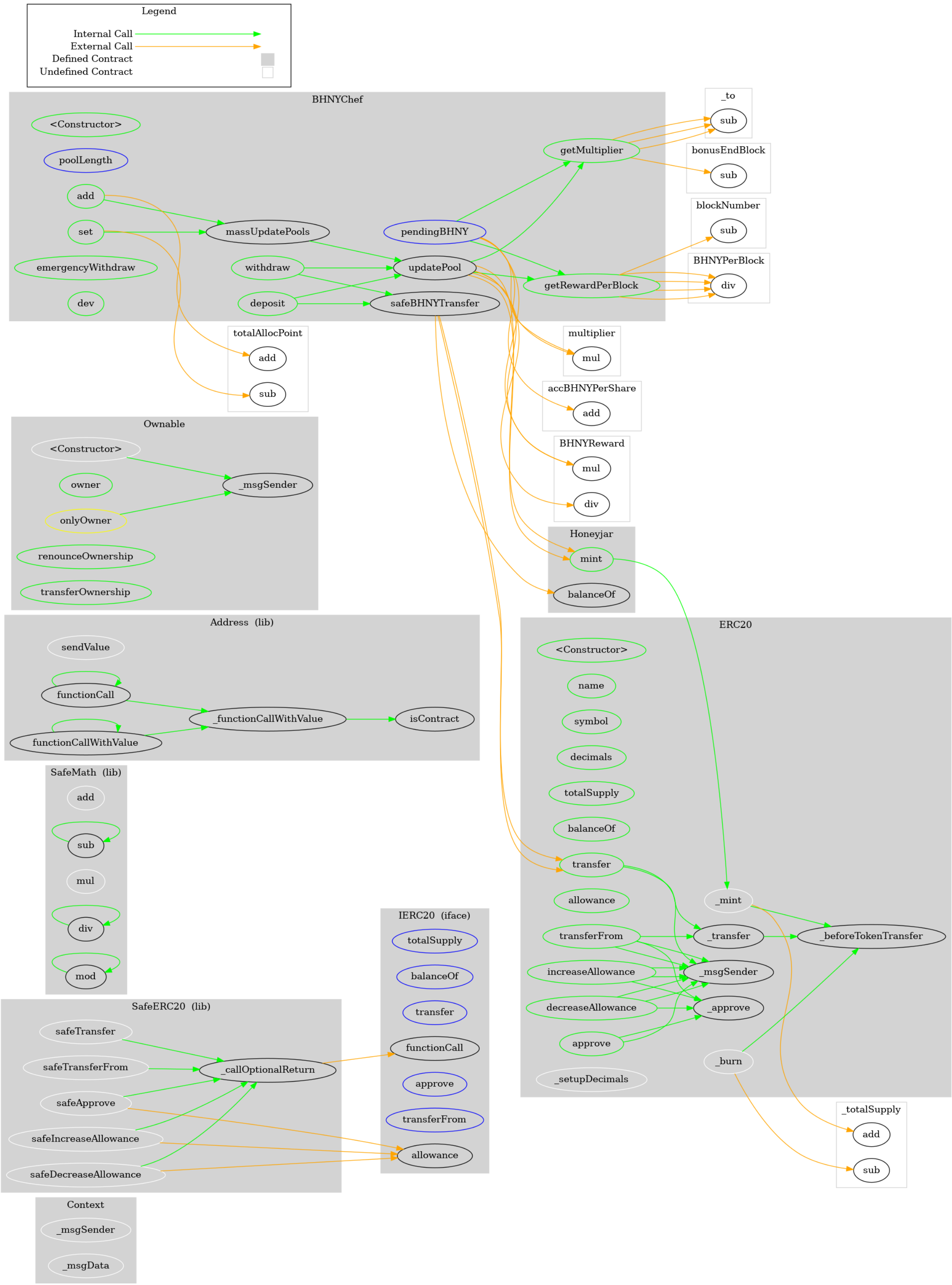
```
     * @param from address The address which you wan
     * @param to address The address which you want 
     * @param value uint256 the amount of tokens to 
     */
    function transferFrom(address from, address to, 
        _transfer(from, to, value);
        _approve(from, msg.sender, _allowed[from][ms
        return true;
    }

    /**
     * @dev Increase the amount of tokens that an ow
     * approve should be called when _allowed[msg.se
     * allowed value is better to use this function 
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend t
     * @param addedValue The amount of tokens to inc
     */
    function increaseAllowance(address spender, uint
        _approve(msg.sender, spender, _allowed[msg.s
        return true;
    }

    /**
     * @dev Decrease the amount of tokens that an ow
     * approve should be called when _allowed[msg.se
     * allowed value is better to use this function 
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     */
    function decreaseAllowance(address spender, uint
        _approve(msg.sender, spender, _allowed[msg.s
        return true;
    }

    /**
     * @dev Transfer token for a specified addresses
     * @param from The address to transfer from.
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    function _transfer(address from, address to, uin
        require(to != address(0));

        _balances[from] = _balances[from].sub(value)
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }

    /**
     * @dev Internal function that mints an amount o
     * an account. This encapsulates the modificatio
     * proper events are emitted.
     * @param account The account that will receive
     * @param value The amount that will be created.
     */
    function _mint(address account, uint256 value) i
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
    /**
     * @dev Internal function that burns an amount o
     * account.
     * @param account The account whose tokens will
     * @param value The amount that will be burnt.
     */
    function _burn(address account, uint256 value) i
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(
        emit Transfer(account, address(0), value);
    }

    /**
     * @dev Approve an address to spend another addr
     * @param owner The address that owns the tokens
     * @param spender The address that will spend th
     * @param value The number of tokens that can be
     */
    function _approve(address owner, address spender
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }

    /**
     * @dev Internal function that burns an amount o
     * @param account The account whose tokens will
     * @param value The amount that will be burnt.
     */
    function _burnFrom(address account, uint256 valu
        _burn(account, value);
        _approve(account, msg.sender, _allowed[accou
    }
}

// File: openzeppelin-solidity/contracts/token/ERC20

pragma solidity ^0.5.2;



/**
 * @title ERC20Detailed token
 * @dev The decimals are only for visualization purp
 * All the operations are done using the smallest an
 * just as on Ethereum all the operations are done i
 */
contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor (string memory name, string memory s
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }

    function name() public view returns (string memo
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
    function symbol() public view returns (string me
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8)
        return _decimals;
    }
}

// File: openzeppelin-solidity/contracts/token/ERC20

pragma solidity ^0.5.2;



/**
 * @title Burnable Token
 * @dev Token that can be irreversibly burned (destr
 */
contract ERC20Burnable is ERC20 {
    /**
     * @dev Burns a specific amount of tokens.
```

```
        _burn(msg.sender, value);
    }

    /**
     * @dev Burns a specific amount of tokens from t
     * @param from address The account whose tokens 
     * @param value uint256 The amount of token to b
     */
    function burnFrom(address from, uint256 value) p
        _burnFrom(from, value);
    }
}

// File: contracts/bees.finance.sol

pragma solidity ^0.5.0;

contract beesfinance is ERC20, ERC20Detailed, ERC20B
    constructor() ERC20Detailed('bees.finance', 'BZZ
        _mint(msg.sender, 80000 * 10**18);
    }
}
```

DETAILS: BHNYCHEF (BEESKEEPER)

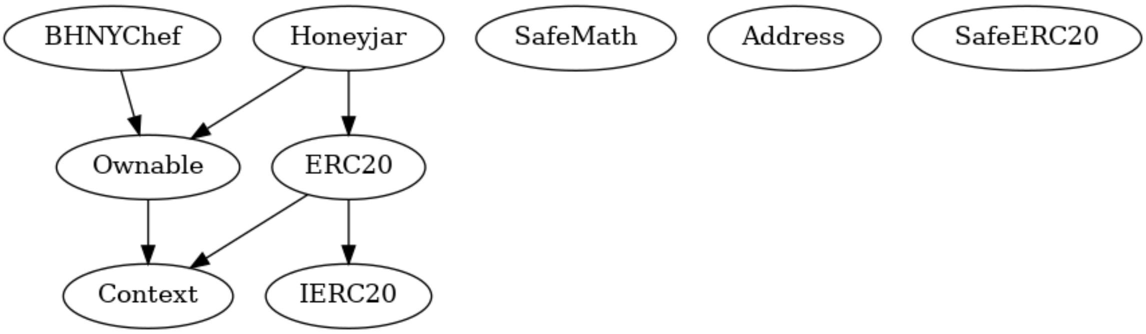
FUNCTION GRAPH

Please review our Terms & Conditions, Privacy Policy, and other legal information here. By using this site, you explicitly agree to these terms.



INHERITENCE CHART

Please review our Terms & Conditions, Privacy Policy, and other legal information here. By using this site, you explicitly agree to these terms.



FUNCTIONS OVERVIEW

```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+  Context
   - [Int] _msgSender
   - [Int] _msgData

+ [Int] IERC20
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #
   - [Int] sub
   - [Int] mul
   - [Int] div
   - [Int] div
   - [Int] mod
   - [Int] mod

+ [Lib] Address
   - [Int] isContract
   - [Int] sendValue #
   - [Int] functionCall #
   - [Int] functionCall #
   - [Int] functionCallWithValue #
   - [Int] functionCallWithValue #
   - [Prv] _functionCallWithValue #

+ [Lib] SafeERC20
   - [Int] safeTransfer #
   - [Int] safeTransferFrom #
   - [Int] safeApprove #
   - [Int] safeIncreaseAllowance #
   - [Int] safeDecreaseAllowance #
   - [Prv] _callOptionalReturn #

+  Ownable (Context)
   - [Int]  #
   - [Pub] owner
   - [Pub] renounceOwnership #
      - modifiers: onlyOwner
   - [Pub] transferOwnership #
      - modifiers: onlyOwner
   - [Pub] name
   - [Pub] symbol
   - [Pub] decimals
   - [Pub] totalSupply
   - [Pub] balanceOf
   - [Pub] transfer #
   - [Pub] allowance
   - [Pub] approve #
   - [Pub] transferFrom #
   - [Pub] increaseAllowance #
   - [Pub] decreaseAllowance #
   - [Int] _transfer #
   - [Int] _mint #
   - [Int] _burn #
   - [Int] _approve #
   - [Int] _setupDecimals #
   - [Int] _beforeTokenTransfer #

+  Honeyjar (ERC20, Ownable)
   - [Pub] mint #
      - modifiers: onlyOwner

+  BHNYChef (Ownable)
   - [Pub]  #
   - [Ext] poolLength
   - [Pub] add #
      - modifiers: onlyOwner
   - [Pub] set #
      - modifiers: onlyOwner
   - [Pub] getMultiplier
   - [Pub] getRewardPerBlock
```

```
    - [Pub] deposit #
    - [Pub] withdraw #
    - [Pub] emergencyWithdraw #
    - [Int] safeBHNYTransfer #
    - [Pub] dev #
```

SOURCE CODE

Click here to download the source code as a .sol file.

```
/**
 *Submitted for verification at Etherscan.io on 2020
*/

pragma solidity ^0.6.12;
// SPDX-License-Identifier: MIT
/*
 * @dev Provides information about the current execu
 * sender of the transaction and its data. While the
 * via msg.sender and msg.data, they should not be a
 * manner, since when dealing with GSN meta-transact
 * paying for execution may not be the actual sender
 * is concerned).
 *
 * This contract is only required for intermediate,
 */
```

```
    }

    function _msgData() internal view virtual return
        this; // silence state mutability warning wi
        return msg.data;
    }
}

/**
 * @dev Interface of the ERC20 standard as defined i
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existenc
     */
    function totalSupply() external view returns (ui

    /**
     * @dev Returns the amount of tokens owned by `a
     */
    function balanceOf(address account) external vie

    /**
     * @dev Moves `amount` tokens from the caller's 
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amo
     * zero by default.
     *
     * This value changes when {approve} or {transfe
     */
    function allowance(address owner, address spende

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * Returns a boolean value indicating whether th
     *
     * IMPORTANT: Beware that changing an allowance 
     * that someone may use both the old and the new
     * transaction ordering. One possible solution t
     * condition is to first reduce the spender's al
     * desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#is
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount

    /**
     * @dev Moves `amount` tokens from `sender` to `
     * allowance mechanism. `amount` is then deducte
     * allowance.
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
     */
     * @dev Emitted when `value` tokens are moved fr
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address ind

    /**
     * @dev Emitted when the allowance of a `spender
     * a call to {approve}. `value` is the new allow
     */
    event Approval(address indexed owner, address in
}

/**
 * @dev Wrappers over Solidity's arithmetic operatio
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflo
 * in bugs, because programmers usually assume that 
 * error, which is the standard behavior in high lev
 * `SafeMath` restores this intuition by reverting t
 * operation overflows.
 *
 * Using this library instead of the unchecked opera
 * class of bugs, so it's recommended to use it alwa
 */
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned int
     * overflow.
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow

        return c;
    }

    /**
     * @dev Returns the subtraction of two unsigned
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b) internal pure
        return sub(a, b, "SafeMath: subtraction over
    }

    /**
     * @dev Returns the subtraction of two unsigned
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     */
    function sub(uint256 a, uint256 b, string memory
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Returns the multiplication of two unsign
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
    function mul(uint256 a, uint256 b) internal pure
        // Gas optimization: this is cheaper than re
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/open
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplicatio

        return c;
    }
```

```
     * division by zero. The result is rounded towar
     *
     * Counterpart to Solidity's `/` operator. Note:
     * `revert` opcode (which leaves remaining gas u
     * uses an invalid opcode to revert (consuming a
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure
        return div(a, b, "SafeMath: division by zero
    }

    /**
     * @dev Returns the integer division of two unsi
     * division by zero. The result is rounded towar
     *
     * Counterpart to Solidity's `/` operator. Note:
     * `revert` opcode (which leaves remaining gas u
     * uses an invalid opcode to revert (consuming a
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b, string memory
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is n
    /**
     * @dev Returns the remainder of dividing two un
     * Reverts when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This 
     * opcode (which leaves remaining gas untouched)
     * invalid opcode to revert (consuming all remai
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure
        return mod(a, b, "SafeMath: modulo by zero")
    }

    /**
     * @dev Returns the remainder of dividing two un
     * Reverts with custom message when dividing by 
     *
     * Counterpart to Solidity's `%` operator. This 
     * opcode (which leaves remaining gas untouched)
     * invalid opcode to revert (consuming all remai
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b, string memory
        require(b != 0, errorMessage);
        return a % b;
```

```
/**
 * @dev Collection of functions related to the addre
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for wh
     * false is an externally-owned account (EOA) an
     *
     * Among others, `isContract` will return false 
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be create
     *  - an address where a contract lived, but was
     * ====
     */
    function isContract(address account) internal vi
        // According to EIP-1052, 0x0 is the value r
        // and 0xc5d2460186f7233c927e7db2dcc703c0e50
        // for accounts without code, i.e. `keccak25
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e
        // solhint-disable-next-line no-inline-assem
        assembly { codehash := extcodehash(account) 
        return (codehash != accountHash && codehash 
     * @dev Replacement for Solidity's `transfer`: s
     * `recipient`, forwarding all available gas and
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP18
     * of certain opcodes, possibly making contracts
     * imposed by `transfer`, making them unable to
     * `transfer`. {sendValue} removes this limitati
     *
     * https://diligence.consensys.net/posts/2019/09
     *
     * IMPORTANT: because control is transferred to
     * taken to not create reentrancy vulnerabilitie
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.5.11/se
     */
    function sendValue(address payable recipient, ui
        require(address(this).balance >= amount, "Ad

        // solhint-disable-next-line avoid-low-level
        (bool success, ) = recipient.call{ value: am
        require(success, "Address: unable to send va
    }

    /**
     * @dev Performs a Solidity function call using
     * plain`call` is an unsafe replacement for a fu
     * function instead.
     *
     * If `target` reverts with a revert reason, it
     * function (like regular Solidity function call
     *
     * Requirements:
     *
     * - `target` must be a contract.
     * - calling `target` with `data` must not rever
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memo
      return functionCall(target, data, "Address: lo
    }

    /**
     * @dev Same as {xref-Address-functionCall-addre
     * `errorMessage` as a fallback revert reason wh
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memo
        return _functionCallWithValue(target, data, 
    }

    /**
     * @dev Same as {xref-Address-functionCall-addre
     * but also transferring `value` wei to `target`
     *
     * Requirements:
     *
     * - the calling contract must have an ETH balan
     * - the called Solidity function must be `payab
     *
     * _Available since v3.1._
    }

    /**
     * @dev Same as {xref-Address-functionCallWithVa
     * with `errorMessage` as a fallback revert reas
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target, b
        require(address(this).balance >= value, "Add
        return _functionCallWithValue(target, data,
    }

    function _functionCallWithValue(address target,
        require(isContract(target), "Address: call t

        // solhint-disable-next-line avoid-low-level
        (bool success, bytes memory returndata) = ta
        if (success) {
            return returndata;
        } else {
            // Look for revert reason and bubble it
            if (returndata.length > 0) {
                // The easiest way to bubble the rev

                // solhint-disable-next-line no-inli
                assembly {
                    let returndata_size := mload(ret
                    revert(add(32, returndata), retu
                }
            } else {
    }
}

/**
 * @title SafeERC20
 * @dev Wrappers around ERC20 operations that throw
 * contract returns false). Tokens that return no va
 * throw on failure) are also supported, non-reverti
 * successful.
 * To use this library you can add a `using SafeERC2
 * which allows you to call the safe operations as `
 */
library SafeERC20 {
    using SafeMath for uint256;
    using Address for address;

    function safeTransfer(IERC20 token, address to,
        _callOptionalReturn(token, abi.encodeWithSel
    }

    function safeTransferFrom(IERC20 token, address
        _callOptionalReturn(token, abi.encodeWithSel
    }

    /**
     * @dev Deprecated. This function has issues sim
     * {IERC20-approve}, and its usage is discourage
     *
     * Whenever possible, use {safeIncreaseAllowance
     * {safeDecreaseAllowance} instead.
     */
```

```
        // 'safeIncreaseAllowance' and 'safeDecrease
        // solhint-disable-next-line max-line-length
        require((value == 0) || (token.allowance(add
            "SafeERC20: approve from non-zero to non
        );
        _callOptionalReturn(token, abi.encodeWithSel
    }

    function safeIncreaseAllowance(IERC20 token, add
        uint256 newAllowance = token.allowance(addre
        _callOptionalReturn(token, abi.encodeWithSel
    }

    function safeDecreaseAllowance(IERC20 token, add
        uint256 newAllowance = token.allowance(addre
        _callOptionalReturn(token, abi.encodeWithSel
    }

    /**
     * @dev Imitates a Solidity high-level call (i.e
     * on the return value: the return value is opti
     * @param token The token targeted by the call.
     * @param data The call data (encoded using abi.
     */
    function _callOptionalReturn(IERC20 token, bytes
        // We need to perform a low level call here,
        // we're implementing it ourselves. We use {
        // the target address contains contract code

        bytes memory returndata = address(token).fun
        if (returndata.length > 0) { // Return data
    }
}



/**
 * @dev Contract module which provides a basic acces
 * there is an account (an owner) that can be grante
 * specific functions.
 *
 * By default, the owner account will be the one tha
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will 
 * `onlyOwner`, which can be applied to your functio
 * the owner.
 */
contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previ

    /**
     * @dev Initializes the contract setting the dep
     */
    constructor () internal {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0), msgSen
    }
    function owner() public view returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if called by any account other th
     */
    modifier onlyOwner() {
        require(_owner == _msgSender(), "Ownable: ca
        _;
    }

    /**
     * @dev Leaves the contract without owner. It wi
     * `onlyOwner` functions anymore. Can only be ca
     *
     * NOTE: Renouncing ownership will leave the con
     * thereby removing any functionality that is on
     */
    function renounceOwnership() public virtual only
        emit OwnershipTransferred(_owner, address(0)
        _owner = address(0);
    }

    /**
     * @dev Transfers ownership of the contract to a
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) pub
        require(newOwner != address(0), "Ownable: ne
        emit OwnershipTransferred(_owner, newOwner);
```

```
/**
 * @dev Implementation of the {IERC20} interface.
 *
 * This implementation is agnostic to the way tokens
 * that a supply mechanism has to be added in a deri
 * For a generic mechanism see {ERC20PresetMinterPau
 *
 * TIP: For a detailed writeup see our guide
 * https://forum.zeppelin.solutions/t/how-to-impleme
 * to implement supply mechanisms].
 *
 * We have followed general OpenZeppelin guidelines:
 * of returning `false` on failure. This behavior is
 * and does not conflict with the expectations of ER
 *
 * Additionally, an {Approval} event is emitted on c
 * This allows applications to reconstruct the allow
 * by listening to said events. Other implementation
 * these events, as it isn't required by the specifi
 *
 * Finally, the non-standard {decreaseAllowance} and
 * functions have been added to mitigate the well-kn
 * allowances. See {IERC20-approve}.
 */
contract ERC20 is Context, IERC20 {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _balances;
    uint256 private _totalSupply;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    /**
     * @dev Sets the values for {name} and {symbol},
     * a default value of 18.
     *
     * To select a different value for {decimals}, u
     *
     * All three of these values are immutable: they
     * construction.
     */
    constructor (string memory name, string memory s
        _name = name;
        _symbol = symbol;
        _decimals = 18;
    }

    /**
     * @dev Returns the name of the token.
     */
    function name() public view returns (string memo
        return _name;
    }

    /**
     * @dev Returns the symbol of the token, usually
     * name.
```

```
}

/**
 * @dev Returns the number of decimals used to g
 * For example, if `decimals` equals `2`, a bala
 * be displayed to a user as `5,05` (`505 / 10 *
 *
 * Tokens usually opt for a value of 18, imitati
 * Ether and Wei. This is the value {ERC20} uses
 * called.
 *
 * NOTE: This information is only used for _disp
 * no way affects any of the arithmetic of the c
 * {IERC20-balanceOf} and {IERC20-transfer}.
 */
function decimals() public view returns (uint8)
    return _decimals;
}

/**
 * @dev See {IERC20-totalSupply}.
 */
function totalSupply() public view override retu
    return _totalSupply;
}

/**
 * @dev See {IERC20-balanceOf}.
 */
function balanceOf(address account) public view
    return _balances[account];
```

```
     * @dev See {IERC20-transfer}.
     *
     * Requirements:
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least
     */
    function transfer(address recipient, uint256 amo
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    /**
     * @dev See {IERC20-allowance}.
     */
    function allowance(address owner, address spende
        return _allowances[owner][spender];
    }

    /**
     * @dev See {IERC20-approve}.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function approve(address spender, uint256 amount
        _approve(_msgSender(), spender, amount);
        return true;
    }
```

```
     * Emits an {Approval} event indicating the upda
     * required by the EIP. See the note at the begi
     *
     * Requirements:
     * - `sender` and `recipient` cannot be the zero
     * - `sender` must have a balance of at least `a
     * - the caller must have allowance for ``sender
     * `amount`.
     */
    function transferFrom(address sender, address re
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[s
        return true;
    }

    /**
     * @dev Atomically increases the allowance grant
     *
     * This is an alternative to {approve} that can
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the upda
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint
        _approve(_msgSender(), spender, _allowances[
        return true;
    }
```

```
     *
     * This is an alternative to {approve} that can
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the upda
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the calle
     * `subtractedValue`.
     */
    function decreaseAllowance(address spender, uint
        _approve(_msgSender(), spender, _allowances[
        return true;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `
     *
     * This is internal function is equivalent to {t
     * e.g. implement automatic token fees, slashing
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `sender` cannot be the zero address.
     * - `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `a
     */
```

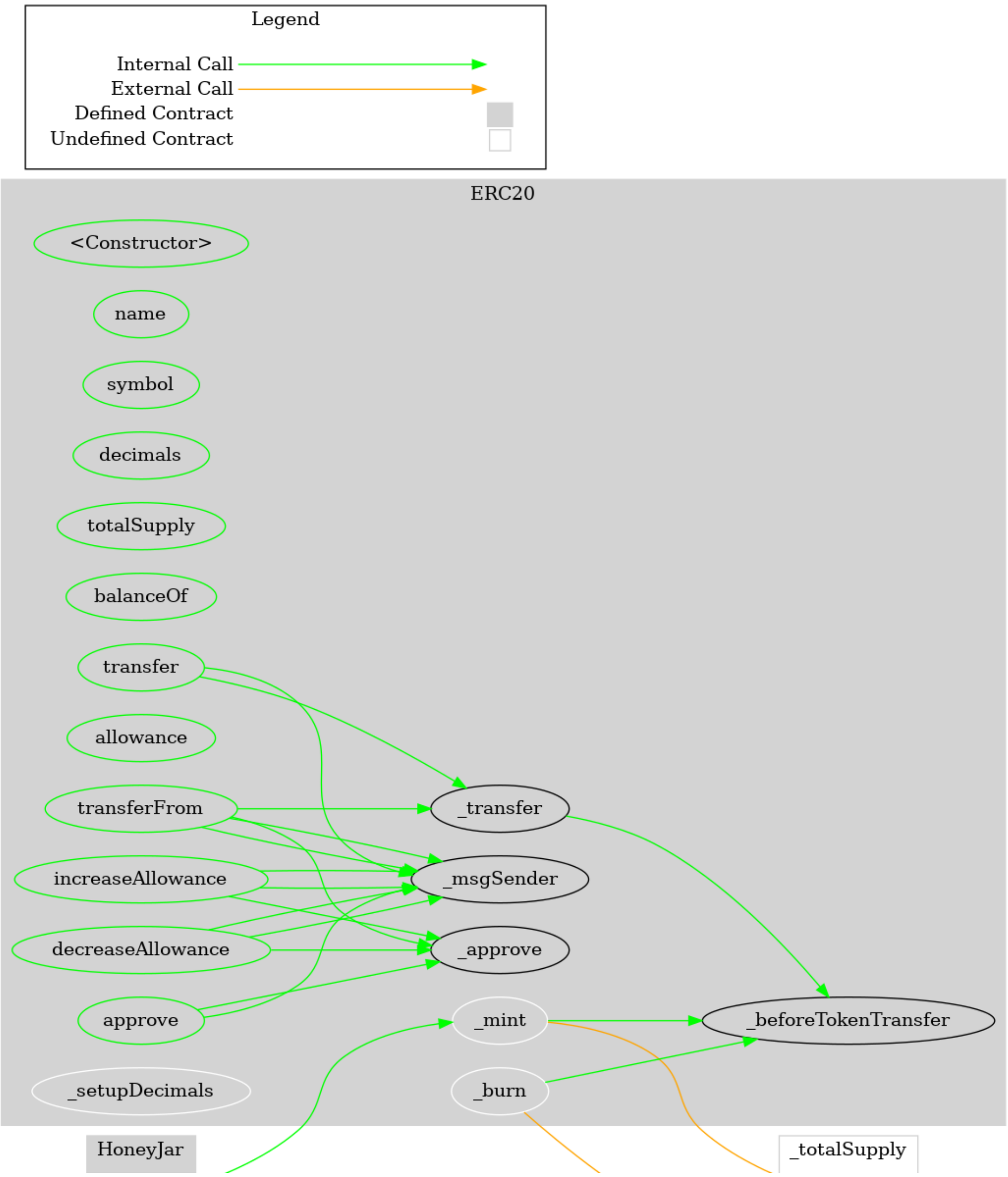
```
        _beforeTokenTransfer(sender, recipient, amou

        _balances[sender] = _balances[sender].sub(am
        _balances[recipient] = _balances[recipient].
        emit Transfer(sender, recipient, amount);
    }

    /** @dev Creates `amount` tokens and assigns the
     * the total supply.
     *
     * Emits a {Transfer} event with `from` set to t
     *
     * Requirements
     *
     * - `to` cannot be the zero address.
     */
    function _mint(address account, uint256 amount)
        require(account != address(0), "ERC20: mint

        _beforeTokenTransfer(address(0), account, am

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(
        emit Transfer(address(0), account, amount);
    }

    /**
     * @dev Destroys `amount` tokens from `account`,
     * total supply.
     *
     *
     * - `account` cannot be the zero address.
     * - `account` must have at least `amount` token
     */
    function _burn(address account, uint256 amount)
        require(account != address(0), "ERC20: burn

        _beforeTokenTransfer(account, address(0), am

        _balances[account] = _balances[account].sub(
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * This is internal function is equivalent to `a
     * e.g. set automatic allowances for certain sub
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
     */
    function _approve(address owner, address spender
        require(owner != address(0), "ERC20: approve
        require(spender != address(0), "ERC20: appro
    /**
     * @dev Sets {decimals} to a value other than th
     *
     * WARNING: This function should only be called 
     * applications that interact with token contrac
     * {decimals} to ever change, and may work incor
     */
    function _setupDecimals(uint8 decimals_) interna
        _decimals = decimals_;
    }

    /**
     * @dev Hook that is called before any transfer 
     * minting and burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero, `am
     * will be to transferred to `to`.
     * - when `from` is zero, `amount` tokens will b
     * - when `to` is zero, `amount` of ``from``'s t
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:
     */
    function _beforeTokenTransfer(address from, addr
}

// Honeyjar with Governance.
contract Honeyjar is ERC20("Honey", "BHNY"), Ownable
```

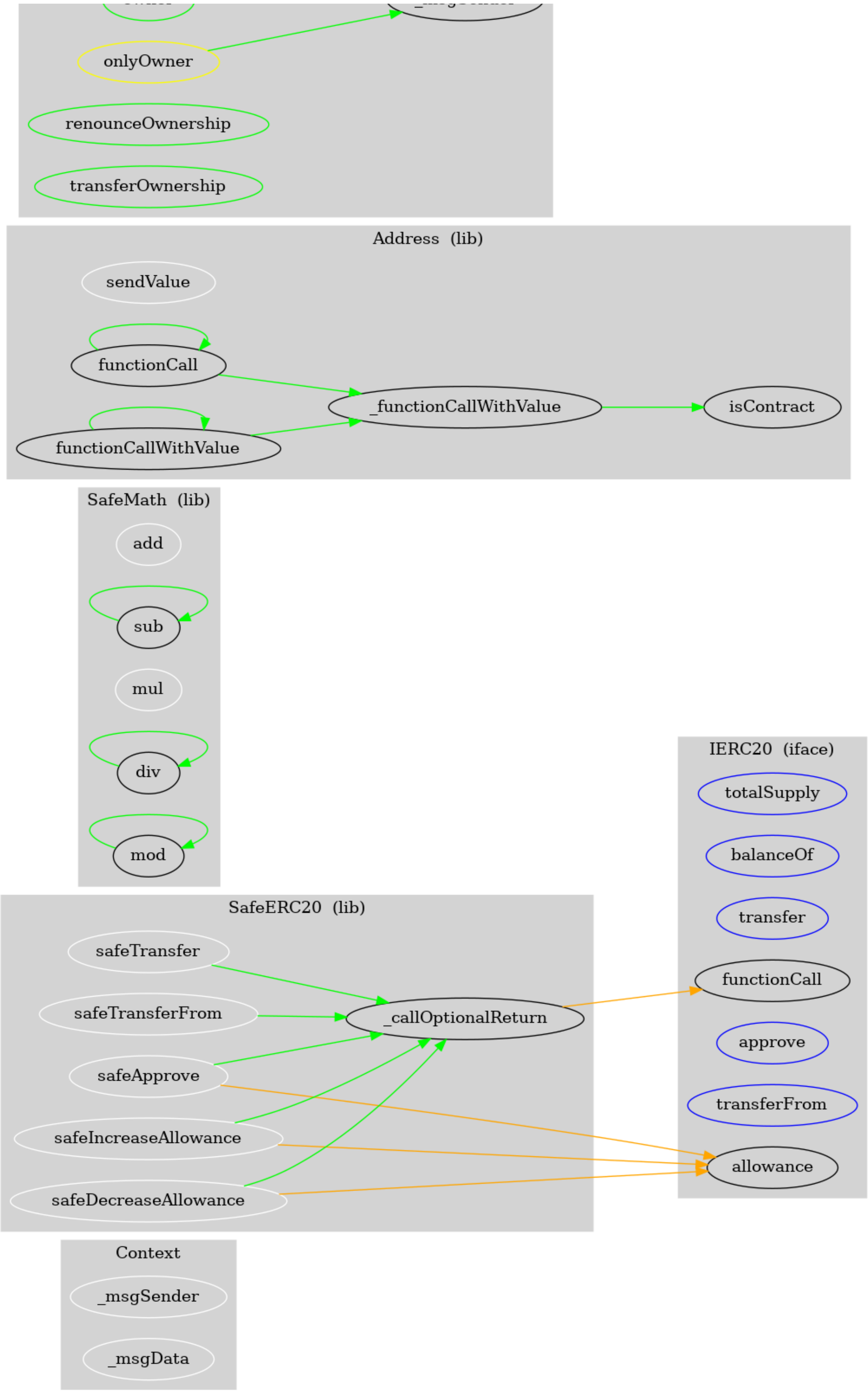
```
    }
}

contract BHNYChef is Ownable {
    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    // Info of each user.
    struct UserInfo {
        uint256 amount;     // How many LP tokens th
        uint256 rewardDebt; // Reward debt. See expl
        //
        // We do some fancy math here. Basically, an
        // entitled to a user but is pending to be d
        //
        //   pending reward = (user.amount * pool.ac
        //
        // Whenever a user deposits or withdraws LP 
        //   1. The pool's `accBHNYPerShare` (and `l
        //   2. User receives the pending reward sen
        //   3. User's `amount` gets updated.
        //   4. User's `rewardDebt` gets updated.
    }

    // Info of each pool.
    struct PoolInfo {
        IERC20 lpToken;           // Address of LP t
        uint256 allocPoint;       // How many alloca
        uint256 lastRewardBlock;  // Last block numb
        uint256 accBHNYPerShare; // Accumulated BHNY
    }
    // Dev address.
    address public devaddr;
    // Block number when bonus BHNY period ends.
    uint256 public bonusEndBlock;
    // BHNY tokens created per block.
    uint256 public BHNYPerBlock;
    // Bonus muliplier for early BHNY makers.
    uint256 public constant BONUS_MULTIPLIER = 1; //

    // No of blocks in a day  - 7000
    uint256 public constant perDayBlocks = 7000; //

    // Info of each pool.
    PoolInfo[] public poolInfo;
    // Info of each user that stakes LP tokens.
    mapping (uint256 => mapping (address => UserInfo
    // Total allocation poitns. Must be the sum of a
    uint256 public totalAllocPoint = 0;
    // The block number when BHNY mining starts.
    uint256 public startBlock;

    event Deposit(address indexed user, uint256 inde
    event Withdraw(address indexed user, uint256 ind
    event EmergencyWithdraw(address indexed user, ui

    constructor(
        Honeyjar _BHNY,
        address _devaddr,
        uint256 _BHNYPerBlock,
        uint256 _startBlock,
        uint256 _bonusEndBlock
        BHNYPerBlock = _BHNYPerBlock;
        bonusEndBlock = _bonusEndBlock;
        startBlock = _startBlock;
    }

    function poolLength() external view returns (uin
        return poolInfo.length;
    }

    // Add a new lp to the pool. Can only be called 
    // XXX DO NOT add the same LP token more than on
    function add(uint256 _allocPoint, IERC20 _lpToke
        if (_withUpdate) {
            massUpdatePools();
        }
        uint256 lastRewardBlock = block.number > sta
        totalAllocPoint = totalAllocPoint.add(_alloc
        poolInfo.push(PoolInfo({
            lpToken: _lpToken,
            allocPoint: _allocPoint,
            lastRewardBlock: lastRewardBlock,
            accBHNYPerShare: 0
        }));
    }

    // Update the given pool's BHNY allocation point
    function set(uint256 _pid, uint256 _allocPoint, 
        if (_withUpdate) {
            massUpdatePools();
        }
        totalAllocPoint = totalAllocPoint.sub(poolIn
    // Return reward multiplier over the given _from
    function getMultiplier(uint256 _from, uint256 _t
        if (_to <= bonusEndBlock) {
            return _to.sub(_from).mul(BONUS_MULTIPLI
        } else if (_from >= bonusEndBlock) {
            return _to.sub(_from);
        } else {
            return bonusEndBlock.sub(_from).mul(BONU
                _to.sub(bonusEndBlock)
            );
        }
    }

     // reward prediction at specific block
    function getRewardPerBlock(uint blockNumber) pub
        if (blockNumber >= startBlock){

            uint256 blockDaysPassed = (blockNumber.s

            if(blockDaysPassed <= 0){
                 return BHNYPerBlock;
            }
            else if(blockDaysPassed > 0 && blockDays
                 return BHNYPerBlock.div(2);
            }
            else if(blockDaysPassed > 7 && blockDays
                 return BHNYPerBlock.div(4);
            }
            else if(blockDaysPassed > 30 && blockDay
                return BHNYPerBlock.div(10);
            }

        } else {
            return 0;
        }
    }

    // View function to see pending BHNYs on fronten
    function pendingBHNY(uint256 _pid, address _user
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][_user
        uint256 accBHNYPerShare = pool.accBHNYPerSha
        uint256 lpSupply = pool.lpToken.balanceOf(ad
        if (block.number > pool.lastRewardBlock && l
            uint256 multiplier = getMultiplier(pool.
            uint256 rewardThisBlock = getRewardPerBl
            uint256 BHNYReward = multiplier.mul(rewa
            accBHNYPerShare = accBHNYPerShare.add(BH
        }
        return user.amount.mul(accBHNYPerShare).div(
    }

    // Update reward vairables for all pools. Be car
    function massUpdatePools() public {
        uint256 length = poolInfo.length;
        for (uint256 pid = 0; pid < length; ++pid) {
            updatePool(pid);
        }
    }
```

```
        PoolInfo storage pool = poolInfo[_pid];
        if (block.number <= pool.lastRewardBlock) {
            return;
        }
        uint256 lpSupply = pool.lpToken.balanceOf(ad
        if (lpSupply == 0) {
            pool.lastRewardBlock = block.number;
            return;
        }
        uint256 multiplier = getMultiplier(pool.last
        uint256 rewardThisBlock = getRewardPerBlock(
        uint256 BHNYReward = multiplier.mul(rewardTh
        BHNY.mint(devaddr, BHNYReward.div(25)); // 4
        BHNY.mint(address(this), BHNYReward);
        pool.accBHNYPerShare = pool.accBHNYPerShare.
        pool.lastRewardBlock = block.number;
    }

    // Deposit LP tokens to MasterChef for BHNY allo
    function deposit(uint256 _pid, uint256 _amount)
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.s
        updatePool(_pid);
        if (user.amount > 0) {
            uint256 pending = user.amount.mul(pool.a
            safeBHNYTransfer(msg.sender, pending);
        }
        pool.lpToken.safeTransferFrom(address(msg.se
        user.amount = user.amount.add(_amount);
        user.rewardDebt = user.amount.mul(pool.accBH
        emit Deposit(msg.sender, _pid, _amount);
    function withdraw(uint256 _pid, uint256 _amount)
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.s
        require(user.amount >= _amount, "withdraw: n
        updatePool(_pid);
        uint256 pending = user.amount.mul(pool.accBH
        safeBHNYTransfer(msg.sender, pending);
        user.amount = user.amount.sub(_amount);
        user.rewardDebt = user.amount.mul(pool.accBH
        pool.lpToken.safeTransfer(address(msg.sender
        emit Withdraw(msg.sender, _pid, _amount);
    }

    // Withdraw without caring about rewards. EMERGE
    function emergencyWithdraw(uint256 _pid) public
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.s
        pool.lpToken.safeTransfer(address(msg.sender
        emit EmergencyWithdraw(msg.sender, _pid, use
        user.amount = 0;
        user.rewardDebt = 0;
    }

    // Safe BHNY transfer function, just in case if
    function safeBHNYTransfer(address _to, uint256 _
        uint256 BHNYBal = BHNY.balanceOf(address(thi
        if (_amount > BHNYBal) {
            BHNY.transfer(_to, BHNYBal);
        } else {
            BHNY.transfer(_to, _amount);
        }
    function dev(address _devaddr) public {
        require(msg.sender == devaddr, "dev: wut?");
        devaddr = _devaddr;
    }
```

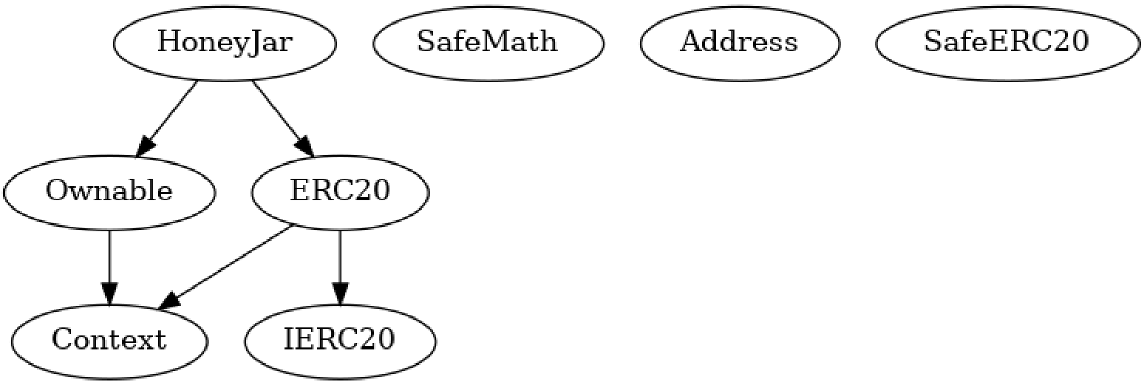
DETAILS: HONEYJAR TOKEN

FUNCTION GRAPH





INHERITENCE CHART



FUNCTIONS OVERVIEW

```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+  Context
   - [Int] _msgSender
   - [Int] _msgData

+ [Int] IERC20
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #
   - [Int] sub
   - [Int] sub
   - [Int] mul
   - [Int] div
   - [Int] div
   - [Int] mod
   - [Int] mod

+ [Lib] Address
   - [Int] isContract
   - [Int] sendValue #
   - [Int] functionCall #
   - [Int] functionCall #
   - [Int] functionCallWithValue #
   - [Int] functionCallWithValue #
   - [Prv] _functionCallWithValue #

+ [Lib] SafeERC20
   - [Int] safeTransfer #
   - [Int] safeTransferFrom #
   - [Int] safeApprove #
   - [Int] safeIncreaseAllowance #
   - [Int] safeDecreaseAllowance #
   - [Prv] _callOptionalReturn #

+  Ownable (Context)
   - [Int]  #
   - [Pub] owner
   - [Pub] renounceOwnership #
      - modifiers: onlyOwner
   - [Pub] transferOwnership #
```

```
      - [Pub]  #
      - [Pub] name
      - [Pub] symbol
      - [Pub] decimals
      - [Pub] totalSupply
      - [Pub] balanceOf
      - [Pub] transfer #
      - [Pub] allowance
      - [Pub] approve #
      - [Pub] transferFrom #
      - [Pub] increaseAllowance #
      - [Pub] decreaseAllowance #
      - [Int] _transfer #
      - [Int] _mint #
      - [Int] _burn #
      - [Int] _approve #
      - [Int] _setupDecimals #
      - [Int] _beforeTokenTransfer #

   +  HoneyJar (ERC20, Ownable)
      - [Pub] mint #
         - modifiers: onlyOwner
```

SOURCE CODE

Click here to download the source code as a .sol file.

```
*/

pragma solidity ^0.6.12;
// SPDX-License-Identifier: MIT
/*
 * @dev Provides information about the current execu
 * sender of the transaction and its data. While the
 * via msg.sender and msg.data, they should not be a
 * manner, since when dealing with GSN meta-transact
 * paying for execution may not be the actual sender
 * is concerned).
 *
 * This contract is only required for intermediate,
 */
abstract contract Context {
    function _msgSender() internal view virtual retu
        return msg.sender;
    }

    function _msgData() internal view virtual return
        this; // silence state mutability warning wi
        return msg.data;
    }
}

/**
 * @dev Interface of the ERC20 standard as defined i
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existenc
    /**
     * @dev Returns the amount of tokens owned by `a
     */
    function balanceOf(address account) external vie

    /**
     * @dev Moves `amount` tokens from the caller's
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amo

    /**
     * @dev Returns the remaining number of tokens t
     * allowed to spend on behalf of `owner` through
     * zero by default.
     *
     * This value changes when {approve} or {transfe
     */
    function allowance(address owner, address spende

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * Returns a boolean value indicating whether th
     *
     * IMPORTANT: Beware that changing an allowance
     * that someone may use both the old and the new
     * transaction ordering. One possible solution t
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount

    /**
     * @dev Moves `amount` tokens from `sender` to `
     * allowance mechanism. `amount` is then deducte
     * allowance.
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(address sender, address re

    /**
     * @dev Emitted when `value` tokens are moved fr
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address ind

    /**
     * @dev Emitted when the allowance of a `spender
     * a call to {approve}. `value` is the new allow
     */
    event Approval(address indexed owner, address in
}
```

```
 *
 * Arithmetic operations in Solidity wrap on overflo
 * in bugs, because programmers usually assume that
 * error, which is the standard behavior in high lev
 * `SafeMath` restores this intuition by reverting t
 * operation overflows.
 *
 * Using this library instead of the unchecked opera
 * class of bugs, so it's recommended to use it alwa
 */
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned int
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow

        return c;
    }

    /**
     * @dev Returns the subtraction of two unsigned
     * overflow (when the result is negative).
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b) internal pure
        return sub(a, b, "SafeMath: subtraction over
    }

    /**
     * @dev Returns the subtraction of two unsigned
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b, string memory
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Returns the multiplication of two unsign
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     */
    function mul(uint256 a, uint256 b) internal pure
        // Gas optimization: this is cheaper than re
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/open
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplicatio

        return c;
    }

    /**
     * @dev Returns the integer division of two unsi
     * division by zero. The result is rounded towar
     *
     * Counterpart to Solidity's `/` operator. Note:
     * `revert` opcode (which leaves remaining gas u
     * uses an invalid opcode to revert (consuming a
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure
        return div(a, b, "SafeMath: division by zero
    }
```

```
     *
     * Counterpart to Solidity's `/` operator. Note:
     * `revert` opcode (which leaves remaining gas u
     * uses an invalid opcode to revert (consuming a
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b, string memory
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is n

        return c;
    }

    /**
     * @dev Returns the remainder of dividing two un
     * Reverts when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This
     * opcode (which leaves remaining gas untouched)
     * invalid opcode to revert (consuming all remai
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure
        return mod(a, b, "SafeMath: modulo by zero")
     * @dev Returns the remainder of dividing two un
     * Reverts with custom message when dividing by
     *
     * Counterpart to Solidity's `%` operator. This
     * opcode (which leaves remaining gas untouched)
     * invalid opcode to revert (consuming all remai
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b, string memory
        require(b != 0, errorMessage);
        return a % b;
    }
}



/**
 * @dev Collection of functions related to the addre
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for wh
     * false is an externally-owned account (EOA) an
     *
     * Among others, `isContract` will return false
     *  - a contract in construction
     *  - an address where a contract will be create
     *  - an address where a contract lived, but was
     * ====
     */
    function isContract(address account) internal vi
        // According to EIP-1052, 0x0 is the value r
        // and 0xc5d2460186f7233c927e7db2dcc703c0e50
        // for accounts without code, i.e. `keccak25
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e
        // solhint-disable-next-line no-inline-assem
        assembly { codehash := extcodehash(account)
        return (codehash != accountHash && codehash
    }

    /**
     * @dev Replacement for Solidity's `transfer`: s
     * `recipient`, forwarding all available gas and
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP18
     * of certain opcodes, possibly making contracts
     * imposed by `transfer`, making them unable to
     * `transfer`. {sendValue} removes this limitati
     *
     * https://diligence.consensys.net/posts/2019/09
     *
     * IMPORTANT: because control is transferred to
     * taken to not create reentrancy vulnerabilitie
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.5.11/se
```

```
        // solhint-disable-next-line avoid-low-level
        (bool success, ) = recipient.call{ value: am
        require(success, "Address: unable to send va
    }

    /**
     * @dev Performs a Solidity function call using
     * plain`call` is an unsafe replacement for a fu
     * function instead.
     *
     * If `target` reverts with a revert reason, it
     * function (like regular Solidity function call
     *
     * Returns the raw returned data. To convert to
     * use https://solidity.readthedocs.io/en/latest
     *
     * Requirements:
     *
     * - `target` must be a contract.
     * - calling `target` with `data` must not rever
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memo
      return functionCall(target, data, "Address: lo
    }

    /**
     * @dev Same as {xref-Address-functionCall-addre
     * `errorMessage` as a fallback revert reason wh
    function functionCall(address target, bytes memo
        return _functionCallWithValue(target, data,
    }

    /**
     * @dev Same as {xref-Address-functionCall-addre
     * but also transferring `value` wei to `target`
     *
     * Requirements:
     *
     * - the calling contract must have an ETH balan
     * - the called Solidity function must be `payab
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target, b
        return functionCallWithValue(target, data, v
    }

    /**
     * @dev Same as {xref-Address-functionCallWithVa
     * with `errorMessage` as a fallback revert reas
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target, b
        require(address(this).balance >= value, "Add
        return _functionCallWithValue(target, data,
    }

    function _functionCallWithValue(address target,
        (bool success, bytes memory returndata) = ta
        if (success) {
            return returndata;
        } else {
            // Look for revert reason and bubble it 
            if (returndata.length > 0) {
                // The easiest way to bubble the rev

                // solhint-disable-next-line no-inli
                assembly {
                    let returndata_size := mload(ret
                    revert(add(32, returndata), retu
                }
            } else {
                revert(errorMessage);
            }
        }
    }
}

/**
 * @title SafeERC20
 * @dev Wrappers around ERC20 operations that throw 
 * contract returns false). Tokens that return no va
 * throw on failure) are also supported, non-reverti
 * successful.
 * To use this library you can add a `using SafeERC2
 * which allows you to call the safe operations as `
 */
library SafeERC20 {
    using SafeMath for uint256;
```

```
        _callOptionalReturn(token, abi.encodeWithSel
    }

    function safeTransferFrom(IERC20 token, address 
        _callOptionalReturn(token, abi.encodeWithSel
    }

    /**
     * @dev Deprecated. This function has issues sim
     * {IERC20-approve}, and its usage is discourage
     *
     * Whenever possible, use {safeIncreaseAllowance
     * {safeDecreaseAllowance} instead.
     */
    function safeApprove(IERC20 token, address spend
        // safeApprove should only be called when se
        // or when resetting it to zero. To increase
        // 'safeIncreaseAllowance' and 'safeDecrease
        // solhint-disable-next-line max-line-length
        require((value == 0) || (token.allowance(add
            "SafeERC20: approve from non-zero to non
        );
        _callOptionalReturn(token, abi.encodeWithSel
    }

    function safeIncreaseAllowance(IERC20 token, add
        uint256 newAllowance = token.allowance(addre
        _callOptionalReturn(token, abi.encodeWithSel
    }

    function safeDecreaseAllowance(IERC20 token, add
    /**
     * @dev Imitates a Solidity high-level call (i.e
     * on the return value: the return value is opti
     * @param token The token targeted by the call.
     * @param data The call data (encoded using abi.
     */
    function _callOptionalReturn(IERC20 token, bytes
        // We need to perform a low level call here,
        // we're implementing it ourselves. We use {
        // the target address contains contract code

        bytes memory returndata = address(token).fun
        if (returndata.length > 0) { // Return data 
            // solhint-disable-next-line max-line-le
            require(abi.decode(returndata, (bool)), 
        }
    }
}




/**
 * @dev Contract module which provides a basic acces
 * there is an account (an owner) that can be grante
 * specific functions.
 *
 * By default, the owner account will be the one tha
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will 
```

```
contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previ

    /**
     * @dev Initializes the contract setting the dep
     */
    constructor () internal {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0), msgSen
    }

    /**
     * @dev Returns the address of the current owner
     */
    function owner() public view returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if called by any account other th
     */
    modifier onlyOwner() {
        require(_owner == _msgSender(), "Ownable: ca
        _;
    }

    /**
     * @dev Leaves the contract without owner. It wi
     * thereby removing any functionality that is on
     */
    function renounceOwnership() public virtual only
        emit OwnershipTransferred(_owner, address(0)
        _owner = address(0);
    }

    /**
     * @dev Transfers ownership of the contract to a
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) pub
        require(newOwner != address(0), "Ownable: ne
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}



/**
 * @dev Implementation of the {IERC20} interface.
 *
 * This implementation is agnostic to the way tokens
 * that a supply mechanism has to be added in a deri
 * For a generic mechanism see {ERC20PresetMinterPau
 *
 * TIP: For a detailed writeup see our guide
 * https://forum.zeppelin.solutions/t/how-to-impleme
 * to implement supply mechanisms].
 *
 * We have followed general OpenZeppelin guidelines:
```

```
 * Additionally, an {Approval} event is emitted on c
 * This allows applications to reconstruct the allow
 * by listening to said events. Other implementation
 * these events, as it isn't required by the specifi
 *
 * Finally, the non-standard {decreaseAllowance} and
 * functions have been added to mitigate the well-kn
 * allowances. See {IERC20-approve}.
 */
contract ERC20 is Context, IERC20 {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    /**
     * @dev Sets the values for {name} and {symbol},
     * a default value of 18.
     *
     * To select a different value for {decimals}, u
     *
     * All three of these values are immutable: they
     * construction.
```

```
        _symbol = symbol;
        _decimals = 18;
    }

    /**
     * @dev Returns the name of the token.
     */
    function name() public view returns (string memo
        return _name;
    }

    /**
     * @dev Returns the symbol of the token, usually
     * name.
     */
    function symbol() public view returns (string me
        return _symbol;
    }

    /**
     * @dev Returns the number of decimals used to g
     * For example, if `decimals` equals `2`, a bala
     * be displayed to a user as `5,05` (`505 / 10 *
     *
     * Tokens usually opt for a value of 18, imitati
     * Ether and Wei. This is the value {ERC20} uses
     * called.
     *
     * NOTE: This information is only used for _disp
     * no way affects any of the arithmetic of the c
     * {IERC20-balanceOf} and {IERC20-transfer}.
    }

    /**
     * @dev See {IERC20-totalSupply}.
     */
    function totalSupply() public view override retu
        return _totalSupply;
    }

    /**
     * @dev See {IERC20-balanceOf}.
     */
    function balanceOf(address account) public view 
        return _balances[account];
    }

    /**
     * @dev See {IERC20-transfer}.
     *
     * Requirements:
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least 
     */
    function transfer(address recipient, uint256 amo
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    /**
     * @dev See {IERC20-allowance}.
    }

    /**
     * @dev See {IERC20-approve}.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function approve(address spender, uint256 amount
        _approve(_msgSender(), spender, amount);
        return true;
    }

    /**
     * @dev See {IERC20-transferFrom}.
     *
     * Emits an {Approval} event indicating the upda
     * required by the EIP. See the note at the begi
     *
     * Requirements:
     * - `sender` and `recipient` cannot be the zero
     * - `sender` must have a balance of at least `a
     * - the caller must have allowance for ``sender
     * `amount`.
     */
    function transferFrom(address sender, address re
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[s
        return true;
    }
```

```
     *
     * This is an alternative to {approve} that can
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the upda
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint
        _approve(_msgSender(), spender, _allowances[
        return true;
    }

    /**
     * @dev Atomically decreases the allowance grant
     *
     * This is an alternative to {approve} that can
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the upda
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the calle
     * `subtractedValue`.
     */
    function decreaseAllowance(address spender, uint
        _approve(_msgSender(), spender, _allowances[
    /**
     * @dev Moves tokens `amount` from `sender` to `
     *
     * This is internal function is equivalent to {t
     * e.g. implement automatic token fees, slashing
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `sender` cannot be the zero address.
     * - `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `a
     */
    function _transfer(address sender, address recip
        require(sender != address(0), "ERC20: transf
        require(recipient != address(0), "ERC20: tra

        _beforeTokenTransfer(sender, recipient, amou

        _balances[sender] = _balances[sender].sub(am
        _balances[recipient] = _balances[recipient].
        emit Transfer(sender, recipient, amount);
    }

    /** @dev Creates `amount` tokens and assigns the
     * the total supply.
     *
     * Emits a {Transfer} event with `from` set to t
     *
     * Requirements
    function _mint(address account, uint256 amount)
        require(account != address(0), "ERC20: mint

        _beforeTokenTransfer(address(0), account, am

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(
        emit Transfer(address(0), account, amount);
    }

    /**
     * @dev Destroys `amount` tokens from `account`,
     * total supply.
     *
     * Emits a {Transfer} event with `to` set to the
     *
     * Requirements
     *
     * - `account` cannot be the zero address.
     * - `account` must have at least `amount` token
     */
    function _burn(address account, uint256 amount)
        require(account != address(0), "ERC20: burn

        _beforeTokenTransfer(account, address(0), am

        _balances[account] = _balances[account].sub(
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }
```

```
     * This is internal function is equivalent to `a
     * e.g. set automatic allowances for certain sub
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
     */
    function _approve(address owner, address spender
        require(owner != address(0), "ERC20: approve
        require(spender != address(0), "ERC20: appro

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    /**
     * @dev Sets {decimals} to a value other than th
     *
     * WARNING: This function should only be called
     * applications that interact with token contrac
     * {decimals} to ever change, and may work incor
     */
    function _setupDecimals(uint8 decimals_) interna
        _decimals = decimals_;
    }

    /**
     * @dev Hook that is called before any transfer
     *
     * - when `from` and `to` are both non-zero, `am
     * will be to transferred to `to`.
     * - when `from` is zero, `amount` tokens will b
     * - when `to` is zero, `amount` of ``from``'s t
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:
     */
    function _beforeTokenTransfer(address from, addr
}

// Honey Token with Governance.
contract HoneyJar is ERC20("Honey", "BHNY"), Ownable
    /// @notice Creates `_amount` token to `_to`. Mu
    function mint(address _to, uint256 _amount) publ
        _mint(_to, _amount);
    }
}
```

PRINT EXPANDED SECTIONS

GO HOME

Please review our Terms & Conditions, Privacy Policy, and other legal information here. By using this site, you explicitly agree to these terms.