Denaro token smart contract source code

https://etherscan.io/address/0x10b35b348fd49966f2baf81df35a511c18bd1f80#code

```
pragma solidity 0.4.18;

/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
  address public owner;

  /**
   * @dev The Ownable constructor sets the original `owner` of the contract to the sender account.
   */
  function Ownable() public {
    owner = msg.sender;
  }

  /**
   * @dev Throws if called by any account other than the owner.
   */
  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }

  /**
   * @dev Allows the current owner to transfer control of the contract to a newOwner.
   * @param newOwner The address to transfer ownership to.
   */
  function transferOwnership(address newOwner) public onlyOwner {
    if (newOwner != address(0)) {
      owner = newOwner;
    }
  }

}


/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
 */
```

```
contract Pausable is Ownable {
  event Pause();
  event Unpause();

  bool public paused = false;

  function Pausable() public {}

  /**
   * @dev modifier to allow actions only when the contract IS paused
   */
  modifier whenNotPaused() {
    require(!paused);
    _;
  }

  /**
   * @dev modifier to allow actions only when the contract IS NOT paused
   */
  modifier whenPaused {
    require(paused);
    _;
  }

  /**
   * @dev called by the owner to pause, triggers stopped state
   */
  function pause() public onlyOwner whenNotPaused {
    paused = true;
    Pause();
  }

  /**
   * @dev called by the owner to unpause, returns to normal state
   */
  function unpause() public onlyOwner whenPaused {
    paused = false;
    Unpause();
  }
}


/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
contract SafeMath {
```

```
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a * b;
    assert(a == 0 || c / a == b);
    return c;
  }

  function div(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a / b;
    return c;
  }

  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    assert(b <= a);
    return a - b;
  }

  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}


contract Denaro is Pausable, SafeMath {

  uint256 public totalSupply;

  mapping(address => uint) public balances;
  mapping (address => mapping (address => uint)) public allowed;

  // ERC20 properties
  string public constant name = "Denaro";
  string public constant symbol = "DNO";
  uint8 public constant decimals = 7;

  // custom properties
  bool public mintingFinished = false;
  uint256 public constant MINTING_LIMIT = 100000000 * (uint256(10) ** decimals);

  event Transfer(address indexed from, address indexed to, uint256 value);
  event Approval(address indexed owner, address indexed spender, uint256 value);

  event Mint(address indexed to, uint256 amount);
  event MintFinished();

  function Denaro() public {}
```

```
function() public payable {
  revert();
}

function balanceOf(address _owner) public view returns (uint256) {
  return balances[_owner];
}

function transfer(address _to, uint _value) public whenNotPaused returns (bool) {

  balances[msg.sender] = sub(balances[msg.sender], _value);
  balances[_to] = add(balances[_to], _value);

  Transfer(msg.sender, _to, _value);
  return true;
}

function transferFrom(address _from, address _to, uint _value) public whenNotPaused returns (bool) {
  var _allowance = allowed[_from][msg.sender];

  balances[_to] = add(balances[_to], _value);
  balances[_from] = sub(balances[_from], _value);
  allowed[_from][msg.sender] = sub(_allowance, _value);

  Transfer(_from, _to, _value);
  return true;
}

function approve(address _spender, uint _value) public whenNotPaused returns (bool) {
  //https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
  require((_value == 0) || (allowed[msg.sender][_spender] == 0));

  allowed[msg.sender][_spender] = _value;
  Approval(msg.sender, _spender, _value);
  return true;
}

function allowance(address _owner, address _spender) public view returns (uint256) {
  return allowed[_owner][_spender];
}

modifier canMint() {
  require(!mintingFinished);
  _;
}
```

```
  function mint(address _to, uint256 _amount) public onlyOwner canMint {
    totalSupply = add(totalSupply, _amount);
    require(totalSupply <= MINTING_LIMIT);

    balances[_to] = add(balances[_to], _amount);
    Mint(_to, _amount);
  }

  function finishMinting() public onlyOwner {
    require(!mintingFinished);
    mintingFinished = true;
    MintFinished();
  }

}
```