 // SPDX-License-Identifier: MIT LICENSE

 // imported required sol files for implementation for token

pragma solidity 0.8.4;

import "@openzeppelin/contracts/access/Ownable.sol"; owner methods to control account imported

import "@openzeppelin/contracts/utils/math/SafeMath.sol"; mathfunctions imported

import "@openzeppelin/contracts/token/ERC20/ERC20.sol"; erc20 token imported for supply

import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol"; it is I used to destruction of token

import "@openzeppelin/contracts/token/ERC20/IERC20.sol"; to ckeck total supply and balance of token

contract Tapas is ERC20, ERC20Burnable, Ownable {

variables declareations

  using SafeMath for uint256;// contract Tapas

  mapping(address => uint256) private \_balances; // mapped address as private balance

  mapping(address => bool) controllers; // set controller true or false

  uint256 private \_totalSupply; // total supply of currency

  uint256 private MAXSUP; // max to add

  uint256 constant MAXIMUMSUPPLY=1000000\*10\*\*18; // no supply beyond this limit

  constructor() ERC20("Tapas", "Taps") {

      \_mint(msg.sender, 1000000 \* 10 \*\* 18); // standard etherium token of ERC20 supply maximum can.

  }

// this function mint to generate coins on certain grounds

  function mint(address to, uint256 amount) external { // address of controller to mint.

    require(controllers[msg.sender], "Only controllers can mint");

    require((MAXSUP+amount)<=MAXIMUMSUPPLY,"Maximum supply has been reached");// checking maximum supply is less than maxup in current

    \_totalSupply = \_totalSupply.add(amount);

    MAXSUP=MAXSUP.add(amount);// adding amount to it.

    \_balances[to] = \_balances[to].add(amount);// balance account to add amount

    \_mint(to, amount); // mint to add coins and done

  }

//This function burn exces coin which is pulled or in circulation to check money in circulation.

  function burnFrom(address account, uint256 amount) public override {

      if (controllers[msg.sender]) { // if controller instructs to burn then its burns or destroys token

          \_burn(account, amount);

      }

      else {

          super.burnFrom(account, amount); // same does

      }

  }

// function to add controller who will mint or burn

  function addController(address controller) external onlyOwner {

    controllers[controller] = true;

  }

// function to remove controller by owner

  function removeController(address controller) external onlyOwner {

    controllers[controller] = false;

  }

  // function to see total current supply in circle

  function totalSupply() public override view returns (uint256) {

    return \_totalSupply;

  }

//function what max supply should be

  function maxSupply() public  pure returns (uint256) {

    return MAXIMUMSUPPLY;

  }

}