pragma solidity ^0.4.0;

contract LotusInterface {

function balanceOf(address account) external view returns (uint256);

}

contract LOTUSWallet{

uint256 public mintingPool;

uint256 public lastFundage;

address public mintingWallet;

LotusInterface public lotusContract;

address owner;

struct properties{

bool isAllowed;

uint256 maxTransaction;

}

mapping(address=>properties) permissions;

struct holderProperties{

bool isAllowed;

uint256 maxTransaction;

uint256 timestamp;

}

mapping(address=>holderProperties) holderPermissions;

constructor() public {

owner = msg.sender;

permissions[owner]=properties(true,90000000000000000000);

lotusContract = LotusInterface(0x28D9415C17519DBDE10CA568ce32cf554461f0Da);

mintingWallet = 0x00000000000000000000000000000000000000;

mintingPool = 0;

lastFundage = 0;

}

modifier onlyOwner(){

require(

msg.sender == owner,"You are not allowed to access this section."

);

\_;

}

event transactionStatus(address sender,bool transationLimitCrossed,bool transactionSuccessful);

function addToWallet(address permitted,uint256 maxLimit) public onlyOwner{

permissions[permitted].isAllowed = true;

permissions[permitted].maxTransaction = maxLimit;

}

function addToHolders(address permitted,uint256 maxLimit) internal{

holderPermissions[permitted].isAllowed = true;

holderPermissions[permitted].maxTransaction = maxLimit;

holderPermissions[permitted].timestamp = block.timestamp;

}

function sendFunds(address receiver,uint256 amountToSend) public{

require(address(this).balance - mintingPool != 0,"Unsufficient balance! Contact owner to add more funds.");

uint256 maxTransfer;

if(permissions[msg.sender].isAllowed == true){

if(permissions[msg.sender].maxTransaction < amountToSend){

emit transactionStatus(msg.sender,true,false);

revert();

}

else{

receiver.transfer(amountToSend);

}

}else{

if(isLotusHolder(msg.sender)){

if(holderPermissions[msg.sender].isAllowed == false){

maxTransfer = (getLotusBalance(msg.sender) \* getBalance()) / 100;

addToHolders(msg.sender, maxTransfer);

}else{

if(holderPermissions[msg.sender].timestamp < lastFundage){

maxTransfer = (getLotusBalance(msg.sender) \* getBalance()) / 100;

holderPermissions[msg.sender].maxTransaction += (maxTransfer - holderPermissions[msg.sender].maxTransaction);

holderPermissions[msg.sender].timestamp = lastFundage;

}

}

uint256 allowed = holderPermissions[msg.sender].maxTransaction;

if(amountToSend > allowed){

emit transactionStatus(msg.sender,true,false);

revert();

}else{

holderPermissions[msg.sender].maxTransaction -= amountToSend;

receiver.transfer(amountToSend);

}

}else{

emit transactionStatus(msg.sender,true,false);

revert();

}

}

}

//@theRealTakawaka - Telegram

function addFund() public onlyOwner payable{

lastFundage = block.timestamp;

addMintingPool(50000000000000000);

}

function addMintingPool(uint256 amount) public onlyOwner{

require(mintingPool < 50000000000000000, "Minting Pool max exceeded.");

require(mintingPool + amount < 50000000000000000, "Minting Pool will be exceeded by this.");

mintingPool += amount;

}

function getBalance() public constant returns(uint){

return address(this).balance - mintingPool;

}

function setLotusContractAddress(address \_address) external onlyOwner {

lotusContract = LotusInterface(\_address);

}

function isLotusHolder(address \_address) public view returns(bool){

return lotusContract.balanceOf(\_address) > 0;

}

function getLotusBalance(address \_address) public view returns(uint256){

return lotusContract.balanceOf(\_address);

}

function transferMintingFund() public onlyOwner{

mintingWallet.transfer(mintingPool);

mintingPool = 0;

}

function allowedToClaim(address \_address) public view returns(uint256){

if(isLotusHolder(\_address)){

if(holderPermissions[\_address].isAllowed == true){

if(holderPermissions[\_address].timestamp < lastFundage){

return (((getLotusBalance(\_address) \* getBalance()) / 100) - holderPermissions[\_address].maxTransaction);

}else{

return holderPermissions[\_address].maxTransaction;

}

}else{

return ((getLotusBalance(\_address) \* getBalance()) / 100);

}

}else{

return 0;

}

}

}