Gustavo's Project

OpenZeppelin Solidity Framework v1.9.0

Ownable
+owner: address
<<event>> OwnershipTransferred(owner:address, newOwner:address)
<<modifier>> onlyOwner()
+Ownable()
+transferOwnership(newOwner:address)
onlyOwner

<<library>>
SafeMath
#mul(a:uint,b:uint): uint const
#div(a:uint,b:uint): uint const
#sub(a:uint,b:uint): uint const
#add(a:uint,b:uint): uint const

Whitelist
+isWhitelisted(_account:address): bool const

actually no realization but duck typing

<<interface>>
WhitelistInterface
+isWhitelisted(_account:address): bool const

Crowdsale
+token: ERC20
+wallet: address
+rate: uint
+weiRaised: uint
<<event>> TokenPurchase(purchaser:address, beneficiary:address, value:uint,amount:uint)
+Crowdsale(_rate:uint,_wallet:address,_token:ERC20)
+()
payable
+buyTokens(_beneficiary:address)
payable
#_preValidatePurchase(_beneficiary:address, _weiAmount:uint)
#_postValidatePurchase(_beneficiary:address, _weiAmount:uint)
#_deliverTokens(_beneficiary:address,_tokenAmount:uint)
#_processPurchase(_beneficiary:address,_tokenAmount:uint)
#_updatePurchasingState(_beneficiary:address, _weiAmount:uint)
#_getTokenAmount(_weiAmount): uint const
#_forwardFunds()

1

ERC20Basic
<<event>> Transfer(from:address,to:address, value:uint)
+totalSupply(): uint const
+balanceOf(who:address): uint const
+transfer(to:address,value:uint): bool

ERC20
<<event>> Approval(owner:address,spender:address, value:uint)
+allowance(owner:address,spender:address): uint const
+transferFrom(from:address,to:address,value:uint): bool
+approve(spender:address,value:uint): bool

BasicToken
+balances: {address=>uint}
+totalSupply_: uint
+totalSupply(): uint const
+transfer(_to:address,_value:uint): bool
+balanceOf(_owner:address): uint const

Pausable
+paused: bool = false
<<event>> Pause()
<<event>> Unpause()
<<modifier>> whenNotPaused()
<<modifier>> whenPaused()
+pause()
onlyOwner
whenNotPaused
+unpause()
onlyOwner
whenPaused

StandardToken
#allowed: {address=>{address=>uint}}
+transferFrom(_from:address,_to:address, _value:uint): bool
+approve(_spender:address,_value:uint): bool
+allowance(_owner:address,_spender:address): uint const
+increaseApproval(_spender:address,_addedValue:uint): bool
+decreaseApproval(_spender:address,_subtractedValue:uint): bool

BurnableToken
<<event>> Burn(burner:address,value:uint)
+burn(_value:uint)
#_burn(_who:address,_value:uint)

MintedCrowdsale
#_deliverTokens(_beneficiary:address,_tokenAmount:uint)

TimedCrowdsale
+openingTime: uint
+closingTime: uint
<<modifier>> onlyWhileOpen()
+TimedCrowdsale(_openingTime:uint,_closingTime:uint)
+hasClosed(): bool const
+_preValidatePurchase(_beneficiary:address, _weiAmount:uint)
onlyWhileOpen

MintableToken
+mintingFinished: bool = false
<<event>> Mint(to:address,amount:uint)
<<event>> MintFinished()
<<modifier>> canMint()
+mint(_to:address,_amount:uint): bool
onlyOwner
canMint
+finishMinting(): bool
onlyOwner
canMint

PausableToken
+transfer(_to:address,_value:uint): bool
whenNotPaused
+transferFrom(_from:address,_to:address, _value:uint): bool
whenNotPaused
+approve(_spender:address,_value:uint): bool
whenNotPaused
+increaseApproval(_spender:address,_addedValue:uint): bool
whenNotPaused
+decreaseApproval(_spender:address,_subtractedValue:uint): bool
whenNotPaused

StandardBurnableToken
+burnFrom(_from:address,_value:uint)

FinalizableCrowdsale
+isFinalized: bool = false
<<event>> Finalized()
+finalize()
onlyOwner
#finalization()

CappedToken
+cap: uint
+CappedToken(_cap:uint)
+mint(_to:address,_amount:uint): bool
onlyOwner
canMint

VreoTokenSale
+TOKEN_CAP: uint = 450000000e18
= 450.000.000 e18
+whitelist: WhitelistInterface
<<event>> RateChanged(newRate:uint)
+constructor(_token:VreoToken,_openingTime:uint, _closingTime:uint,_rate:uint, _wallet:address)
Crowdsale(_rate, _wallet, _token)
TimedCrowdsale(_openingTime, _closingTime)
+setRate(_newRate:uint)
onlyOwner
+buyTokens(_beneficiary:address)
payable
#_preValidatePurchase(_beneficiary:address, _weiAmount:uint)
#_postValidatePurchase(_beneficiary:address

VreoTokenSale
+TOTAL_TOKEN_CAP: uint = 450000000e18
= 450.000.000 e18
+TEAM_TOKEN_SHARE: uint = 85000000e18
= 85.000.000 e18
+ADVISORS_TOKEN_SHARE: uint = 58000000e18
= 58.000.000 e18
+LEGAL_TOKEN_SHARE: uint = 57000000e18
= 57.000.000 e18
+BOUNTY_TOKEN_SHARE: uint = 50000000e18
= 50.000.000 e18
+bounty: VreoTokenBounty
+investors: {address=>Investor}
<<event>> RateChanged(newRate:uint)
<<event>> InvestorValidated(investor:address)
<<event>> InvestorInvalidated(investor:address)

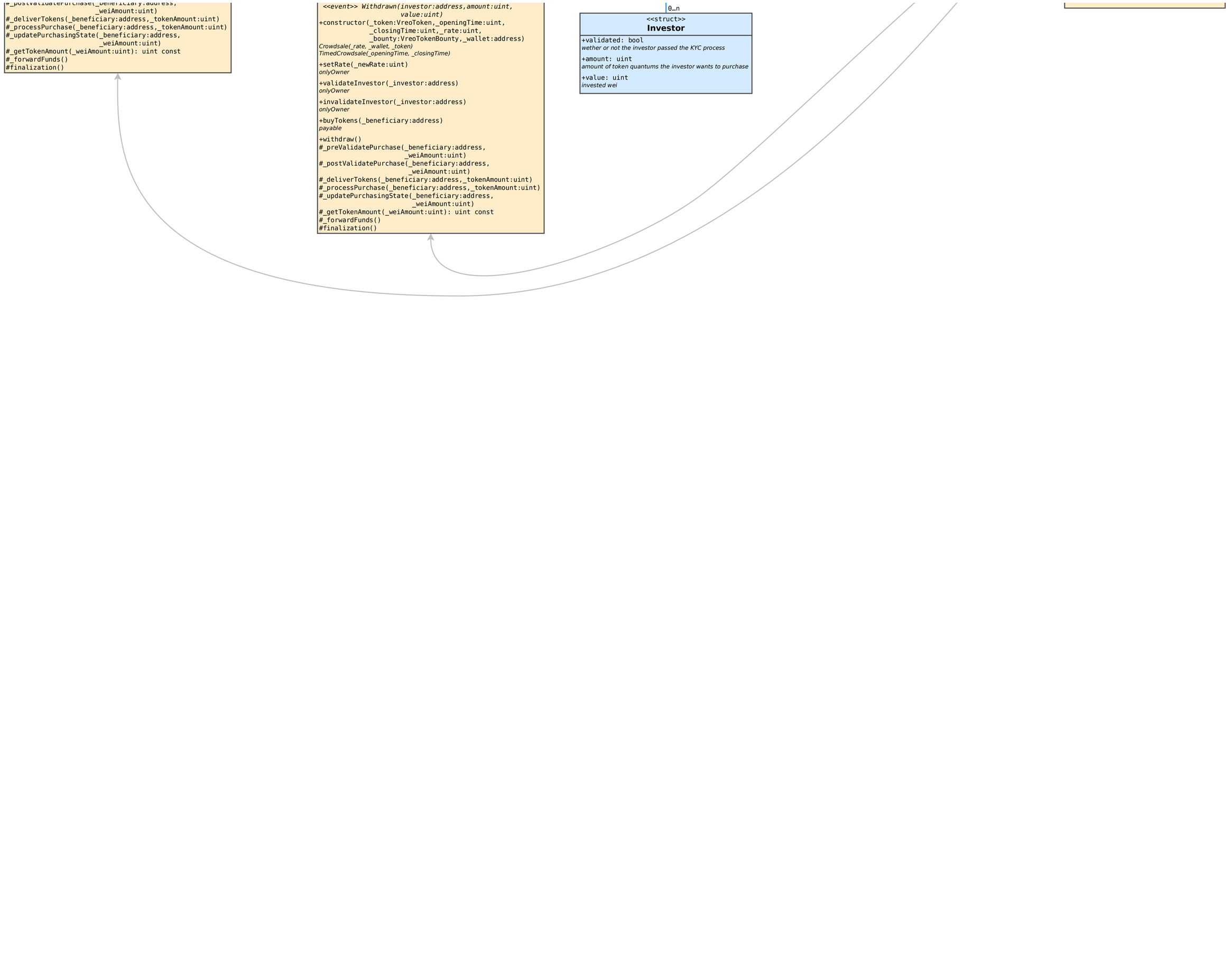
1

VreoTokenBounty
+token: VreoToken
<<event>> TokenDistributed(recipient:address, amount:uint)
+constructor(_token:VreoToken)
+distributeTokens(_recipients:address[], _amounts:uint[])
onlyOwner

1

VreoToken
+name: string = "Vreo MTC"
+symbol: string = "MTC"
+decimals: uint8 = 18
+minters: {address=>bool}
<<event>> MinterAdded(minter:address)
<<event>> MinterRemoved(minter:address)
<<modifier>> onlyMinter()
+constructor()
CappedToken(700000000e18)
+addMinter(_minter:address)
onlyOwner
+removeMinter(_minter:address)
onlyOwner
+mint(_to:address,_amount:uint): bool
onlyMinter
canMint

minters


_weiAmount:uint)
#_deliverTokens(_beneficiary:address,_tokenAmount:uint)
#_processPurchase(_beneficiary:address,_tokenAmount:uint)
#_updatePurchasingState(_beneficiary:address,
_weiAmount:uint)
#_getTokenAmount(_weiAmount:uint): uint const
#_forwardFunds()
#finalization()
<<event>> Withdrawn(investor:address,amount:uint,
value:uint)
+constructor(_token:VreoToken,_openingTime:uint,
_closingTime:uint,_rate:uint,
_bounty:VreoTokenBounty,_wallet:address)
Crowdsale(_rate, _wallet, _token)
TimedCrowdsale(_openingTime, _closingTime)
+setRate(_newRate:uint)
onlyOwner
+validateInvestor(_investor:address)
onlyOwner
+invalidateInvestor(_investor:address)
onlyOwner
+buyTokens(_beneficiary:address)
payable
+withdraw()
#_preValidatePurchase(_beneficiary:address,
_weiAmount:uint)
#_postValidatePurchase(_beneficiary:address,
_weiAmount:uint)
#_deliverTokens(_beneficiary:address,_tokenAmount:uint)
#_processPurchase(_beneficiary:address,_tokenAmount:uint)
#_updatePurchasingState(_beneficiary:address,
_weiAmount:uint)
#_getTokenAmount(_weiAmount:uint): uint const
#_forwardFunds()
#finalization()
0..n
<<struct>>
Investor
+validated: bool
wether or not the investor passed the KYC process
+amount: uint
amount of token quantums the investor wants to purchase
+value: uint
invested wei