pragma solidity ^0.4.11;

import './IERC20.sol';

import './SafeMath.sol';

contract KPRToken is IERC20

{

using SafeMath for uint256;

//total supply of token

uint public \_totalSupply = 100000000;

//public variables

string public constant symbol = "KPR";

string public constant name = "KPR Token";

uint8 public constant decimals = 18;

//1 ETH = 2,500 KPR

uint56 public constant RATE = 2500;

//where the ETH goes

address public owner;

//map the addresses

mapping(address => uint256) balances;

mapping(address => mapping(address => uint256)) allowed;

//create token function = check

function() payable

{

createTokens();

}

function KPRToken()

{

owner = msg.sender;

}

function createTokens() payable

{

require(msg.value > 0);

uint256 tokens = msg.value.mul(RATE);

//add tokens bought to the customers wallet

balances[msg.sender] = balances[msg.sender].add(tokens);

//add tokens sold to the total \_totalSupply

\_totalSupply = \_totalSupply.add(tokens);

//transfer ETH to the owner of the contract

owner.transfer(msg.value);

}

function balanceOf(address \_owner) constant returns (uint256 balance)

{

return balances[\_owner];

}

function transfer(address \_to, uint256 \_value) returns (bool success)

{

//require is the same as an if statement = checks

require(balances[msg.sender] >= \_value && \_value > 0);

balances[msg.sender] = balances[msg.sender].sub(\_value);

balances[\_to] = balances[\_to].add(\_value);

Transfer(msg.sender, \_to, \_value);

return true;

}

function transferFrom(address \_from, address \_to, uint256 \_value) returns (bool success)

{

//checking if the spender has permission to spend and how much

require(

allowed[\_from][msg.sender] >= \_value

&& balances[\_from] >= \_value

&& \_value > 0);

//updating the spenders balance

balances[\_from] = balances[\_from].sub(\_value);

balances[\_to] = balances[\_to].add(\_value);

allowed[\_from][msg.sender] = allowed[\_from][msg.sender].sub(\_value);

Transfer(\_from, \_to, \_value);

return true;

}

function approve(address \_spender, uint256 \_value) returns (bool success)

{

//if above require is true, approve the spending

allowed[msg.sender][\_spender] = \_value;

Approval(msg.sender, \_spender, \_value);

return true;

}

function allowance(address \_owner, address \_spender) constant returns (uint256 remaining)

{

return allowed[\_owner][\_spender];

}

event Transfer(address indexed \_from, address indexed \_to, uint256 \_value);

event Approval(address indexed \_owner, address indexed \_spender, uint256 \_value);

}