pragma solidity ^0.4.19;

import "./Ownable.sol";

import "./Manager.sol";

import "./LandTransport.sol";

import "./Distrubuter.sol";

import "./Escrow.sol";

import "./Customs.sol";

import "./PortAuthority.sol";

import "./Shipping.sol";

contract Manufacture is Ownable, main, LandTransport, Distrubuter, Customs, portauthority, ShipmentTracking {

event OrderConfirmed(uint \_orderId);

event OrderManifactured(uint \_orderId);

event OrderDispached(uint \_orderId);

event DelayInManufacturing(uint \_orderId, uint \_delayTime);

function TotaltimeRequired(uint orderId, uint \_TotaltimeRequired, uint \_expectedTimeOfDeparture){

itemMap[orderId].totalTimeRequired = now + \_TotaltimeRequired; // Took time required for whole process.

statsMap[orderId].timeToNextEntity = now + \_expectedTimeOfDeparture; // Took time required to transfer to next entity.

}

function AddMoney(address AccountHolder, uint Amount){

BalanceOfMoney[AccountHolder]=BalanceOfMoney[AccountHolder]+Amount;

}

function orderConfirmed(uint orderId) {

require(msg.sender==flowOfObject[orderId].Addresses[currentaddress[orderId]]);

require(bankConfirmation[orderId]==true);

statsMap[orderId].checkPoint="OrderConfirmed"; // Updates currentStatusOfOrder.

statsMap[orderId].timeTheEventCalled=now;

OrderConfirmed(orderId); // Event of OrderConfirmed.

}

function orderManifactured(uint orderId){

require(keccak256(statsMap[orderId].checkPoint)== keccak256("OrderConfirmed"));

statsMap[orderId].checkPoint="OrderManifactured"; // Updates currentStatusOfOrder.

statsMap[orderId].timeTheEventCalled=now;

emit OrderManifactured(orderId); // Event of OrderManifactured.

}

function orderDispached(uint orderId, uint Weight) {

require(keccak256(statsMap[orderId].checkPoint)== keccak256("OrderManifactured"));

statsMap[orderId].checkPoint="OrderDispached";

statsMap[orderId].timeTheEventCalled=now; // Updates currentStatusOfOrder.

itemMap[orderId].weight=Weight;

OrderDispached(orderId); // Event of OrderDispached.

transferPossesion(orderId);

}

function delayInManufacturing(uint orderId, uint \_delayTime) {

uint delayTime = \_delayTime;

statsMap[orderId].checkPoint="OrderDelayed";

statsMap[orderId].timeTheEventCalled=now; // Updates currentStatusOfOrder.

DelayInManufacturing(orderId, delayTime); // Event of DelayInManufacturing.

}

function currentStatusOfOrder(uint \_orderId) public returns(string, uint){ // Function to check currentStatusOfOrder.

return(statsMap[\_orderId].checkPoint, statsMap[\_orderId].timeTheEventCalled);

}

}