pragma solidity ^0.4.0;

contract main {

struct item { // Structure to tell about the order item which is placed;

uint product\_Id;

string name;

string description;

uint quantity;

uint totalCost;

uint weight;

uint shipmentId;

uint totalTimeRequired;

uint shipagent;

address manufacturer;

address distrubuter;

}

mapping(uint => item)public itemMap;

mapping(uint => bool)public bankConfirmation;

mapping(address => uint)BalanceOfMoney;

mapping(address => uint)commisionAmount;

mapping(uint => uint)currentaddress;

// 0xca35b7d915458ef540ade6068dfe2f44e8fa733c is Manager.

// 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c is manufacture.

// 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db is exlandtrasport.

// 0x583031d1113ad414f02576bd6afabfb302140225 is excustoms.

// 0xdd870fa1b7c4700f2bd7f44238821c26f7392148 is exportAuthority.

// 0x67a9f0601ff6a156e97d864d8854efb3526336c1 is shipping.

// 0xed7bdba6147756c5b009bac835c944d2e4f9982f is importAuthority.

// 0x0bd9496f73ac71f9061974e17e9a6413cfdd2d2d is imcustoms.

// 0x410acb3f8df43a9134ef769fe981a8b7c0e4d9c4 is imlandtransport.

// 0x0063e14162a52e762257a737a0c96aa79e1202cf is distrubuter.

// 0xcddb5e49b709b2e38171e36263c656e1c3bcf047 is escrow.

// 0x0000000000000000000000000000000000000000 is notavailable.

struct stats {

string checkPoint;

uint timeTheEventCalled;

uint timeToNextEntity;

}

struct AddressStruct{

address[] Addresses;

}

mapping(uint =>AddressStruct) flowOfObject;

mapping(uint => stats)public statsMap;

function setOrder(uint p\_Id,string name,string Description,uint Quantity, uint \_totalCost) public returns(uint ) {

//Function to set or take the order from the Customer;

require(msg.sender==0xca35b7d915458ef540ade6068dfe2f44e8fa733c);

uint orderId = uint(keccak256(p\_Id + Quantity));

itemMap[orderId].product\_Id = p\_Id;

itemMap[orderId].name = name;

itemMap[orderId].description = Description;

itemMap[orderId].quantity = Quantity ;

itemMap[orderId].totalCost = \_totalCost;

commisionAmount[0x14723a09acff6d2a60dcdf7aa4aff308fddc160c]=70;

commisionAmount[0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db]=30;

return orderId;

}

function setflowoforder( uint orderId,

address \_manufacturer,

address \_exlandtrasport,

address \_excustoms,

address \_exportAuthority,

address \_shipping,

address \_importAuthority,

address \_imcustoms,

address \_imlandtransport,

address \_distrubuter){

require(msg.sender==0xca35b7d915458ef540ade6068dfe2f44e8fa733c);

if( \_manufacturer!=0){

flowOfObject[orderId].Addresses.push(\_manufacturer);

itemMap[orderId].manufacturer=\_manufacturer;

}

if( \_exlandtrasport!=0){

flowOfObject[orderId].Addresses.push(\_exlandtrasport);

}

if( \_excustoms!=0){

flowOfObject[orderId].Addresses.push(\_excustoms);

}

if( \_exportAuthority!=0){

flowOfObject[orderId].Addresses.push(\_exportAuthority);

}

if( \_shipping!=0){

flowOfObject[orderId].Addresses.push(\_shipping);

}

if( \_importAuthority!=0){

flowOfObject[orderId].Addresses.push(\_importAuthority);

}

if( \_imcustoms!=0){

flowOfObject[orderId].Addresses.push(\_imcustoms);

}

if( \_imlandtransport!=0){

flowOfObject[orderId].Addresses.push(\_imlandtransport);

}

if( \_distrubuter!=0){

flowOfObject[orderId].Addresses.push(\_distrubuter);

itemMap[orderId].distrubuter=\_distrubuter;

}

flowOfObject[orderId].Addresses.push(0);

}

event PossesionTransferred(address previousOwner, address newOwner);

function transferPossesion(uint orderId) internal {

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

address currentEntity=msg.sender;

currentaddress[orderId]++;

address newentity=flowOfObject[orderId].Addresses[currentaddress[orderId]];

require(newentity != address(0));

emit PossesionTransferred(currentEntity, newentity);

}

}