EXAMPLE #4

pragma solidity 0.6.0;

// Defining a Contract

contract escrow{

    // Declaring the state variables

    address payable public buyer;

    address payable public seller;

    address payable public arbiter;

    mapping(address => uint) TotalAmount;

    // Defining a enumerator 'State'

    enum State{

        // Following are the data members

        awate\_payment, awate\_delivery, complete

    }

    // Declaring the object of the enumerator

    State public state;

    // Defining function modifier 'instate'

    modifier instate(State expected\_state){

        require(state == expected\_state);

        \_;

    }

   // Defining function modifier 'onlyBuyer'

    modifier onlyBuyer(){

        require(msg.sender == buyer ||

                msg.sender == arbiter);

        \_;

    }

    // Defining function modifier 'onlySeller'

    modifier onlySeller(){

        require(msg.sender == seller);

        \_;

    }

    // Defining a constructor

    constructor(address payable \_buyer,

                address payable \_sender) public{

        // Assigning the values of the

        // state variables

        arbiter = msg.sender;

        buyer = \_buyer;

        seller = \_sender;

        state = State.awate\_payment;

    }

    // Defining function to confirm payment

    function confirm\_payment() onlyBuyer instate(

      State.awate\_payment) public payable{

        state = State.awate\_delivery;

    }

    // Defining function to confirm delivery

    function confirm\_Delivery() onlyBuyer instate(

      State.awate\_delivery) public{

        seller.transfer(address(this).balance);

        state = State.complete;

    }

    // Defining function to return payment

    function ReturnPayment() onlySeller instate(

      State.awate\_delivery)public{

       buyer.transfer(address(this).balance);

    }

}

EXAMPLE #5

pragma solidity ^0.4.0;

// Creating a contract

contract smartcontract

{

    // Declaring the state variable

    uint x;

    // Mapping of addresses to their balances

    mapping(address => uint) balance;

    // Creating a constructor

    constructor() public

    {

        // Set x to default

        // value of 10

        x=10;

    }

    // Creating a function

    function SetX(uint \_x) public returns(bool)

    {

        // Set x to the

        // value sent

        x=\_x;

        return true;

    }

    // This fallback function

    // will keep all the Ether

    function() public payable

    {

        balance[msg.sender] += msg.value;

    }

}

// Creating the sender contract

contract Sender

{

  function transfer() public payable

  {

      // Address of GeeksForGeeks contract

      address \_receiver =

              0xbcc0185441de06F0452D45AEd6Ad8b98017796fb;

      // Transfers 100 Eth to above contract

      \_receiver.transfer(100);

  }

}