day76-sol-fallback

태그	
날짜	@2023년 1월 16일

fallback

특징

1. 무기명함수, 이름이 없는 함수
2. external 필수
3. payable

왜?

1. 스마트 컨트렌트가 이더를 받을 수 있게 한다.
2. 이더받고 난 후 어떠한 처리를 할 때
3. call함수로 없는 함수가 불려질때 , 후처리

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.0  <0.9.0;

/*
fallback() external payable{

}

*/

contract Bank{
    event JustFallback(address _from, string message);
    event RecevieFallback(address _from,uint256 _value , string message);
    event JustFallbackWithFunds(address _from, uint256 _value, string message);

    receive() external payable{
        emit RecevieFallback(msg.sender, msg.value,"RecevieFallback");
    }

    fallback() external payable{
        emit JustFallbackWithFunds(msg.sender, msg.value,"JustFallbakcWithFunds is called");
    }

}

contract You{
    function DepositWithSend(address payable _to) public payable{
        bool success = _to.send(msg.value);
        require(success,"Failled");
    }

    function DepositwithTransfer(address payable _to)public payable{
        _to.transfer(msg.value);
    }

    //recive
     function DepositWithCall(address payable _to) public payable{
        // ~ 0.7
        // (bool sent, ) = _to.call.value(msg.value)("");
        // require(sent,"Failed to send either");

        //0.7 ~
        (bool sent, ) = _to.call{value: msg.value}("");
        require(sent, "Failled" );
    }


    //fallback
    function JustGiveMessageWithFunds(address payable _to) public payable{
        (bool success,) = _to.call{value:msg.value}("HI");
        require(success,"Failled");
    }

}
```

---

enum

사람이 읽을 수 있게 사용자/ 개발에 의해서 정의되는 상수세트 타입

한개의 enum당 256개 저장 가능 0~255

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.0  <0.9.0;


contract code39{

     enum PlayerStatus{
         IDLE, //0
         MOVE, //1
         ATTACK,//2
         JUMP  //3
     }

     PlayerStatus public playerStatus;

     constructor(){
         playerStatus = PlayerStatus.IDLE;
```

```
    }

    event playerCurrentState(PlayerStatus _playerStatus,uint256 _playerStatusInt);

    function PlayerMove() public{
        //참이면 통과 거짓이면 옆에 에러메세지
        require(playerStatus == PlayerStatus(0),unicode"IDLE 이면통과");

        playerStatus = PlayerStatus.MOVE;
        emit playerCurrentState(playerStatus,uint256(playerStatus));

    }

    function PlayerAttack() public{
         require(playerStatus == PlayerStatus.MOVE,unicode"MOVE면 통과");
         playerStatus = PlayerStatus.ATTACK;
         emit playerCurrentState(playerStatus,uint256(playerStatus));

    }


    function PlayerJump() public{
         require(playerStatus == PlayerStatus.ATTACK,unicode"ATTAK면 통과");
         playerStatus = PlayerStatus.JUMP;
         emit playerCurrentState(playerStatus,uint256(playerStatus));
    }

    function PlayerIdle() public{
          // ATTACK  , JUMP  ->IDLE
          require(playerStatus == PlayerStatus.ATTACK ||
           playerStatus == PlayerStatus.JUMP ,unicode"점프거나 공격이거나 통과");
           playerStatus = PlayerStatus.IDLE;
       emit playerCurrentState(playerStatus,uint256(playerStatus));
    }


    function CheckState() public view returns(PlayerStatus){
        return playerStatus;
    }
```