day70-sol-struct



포인터에 대해 알아보기

```
# 구조체

struct 구조체명 {

타입 변수명,

타입 변수명,

.........
}
```

어떠한 변수들을 모아서 쓰기 떄문에 편리하다.

자기가 타입을 정의해서 쓸 수 있다. (사용자 정의 타입)

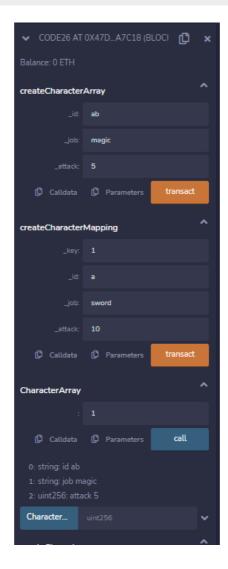
다른 언어의 구조체와 비슷하나 솔리디티만의 방식이 있다.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.0 <0.9.0;
// # 구조체
// struct 구조체명 {
// 타입 변수명,
// 타입 변수명,
// 어떠한 변수들을 모아서 쓰기 떄문에 편리하다.
// 자기가 타입을 정의해서 쓸 수 있다. ( 사용자 정의 타입)
// 다른 언어의 구조체와 비슷하나 솔리디티만의 방식이 있다.
contract code26{
             struct Character{
                          string id;
                          string job;
                          uint256 attack;
             Character[] public CharacterArray;
             mapping (uint256=>Character) public CharacterMapping;
             function createCharacter(string memory _id, string memory _job, uint256 _attack) pure public returns(Character memory){
                          return Character(_id, _job, _attack);
             function\ create Character Mapping (uint 256 \_key,\ string\ memory\ \_id,\ string\ memory\ \_job,\ uint 256\ \_attack)\ public \{ action between the context of the context o
                          CharacterMapping[_key] = Character(_id, _job, _attack);
             function \ getCharacter \texttt{Mapping}(\texttt{uint256} \ \_\texttt{key}) \ public \ \texttt{view} \ returns(\texttt{Character} \ \texttt{memory}) \ \{
                          return CharacterMapping[_key];
```

day70-sol-struct 1

```
function createCharacterArray(string memory _id, string memory _job, uint256 _attack) public {
    CharacterArray.push(Character( _id, _job, _attack));
}

function getCharacterArray(uint256 _index) public view returns(Character memory) {
    return CharacterArray[_index];
}
```



day70-sol-struct 2