LESSON SC 4 – Enum

University of West Attica
Department of Electrical and Electronics Engineering
Ioannis Christidis
Christoforos Kachris

Support by Ethereum Foundation ESP

What will we accomplish!

This is the third lesson and we will learn about `enums`.

We will continue on our `ICR.sol`.

Enums

Enums in Solidity are *user-defined types that allow you to define a finite set of constant values*.

These `constants` *are internally represented as a* `uint8`*, starting from* `0` *for the first constant*, `1` for the second, and so on.

The default value of an `enum` *is the first constant*.

`Enums` *prevent invalid values by restricting variables to predefined options.*

Let's use an `enum` in our project.

In a previous lesson we mentioned that a car can be occupied or unoccupied. What if we wanted to add more options?

Example: What if the owner of the car wants to do a service on it? In that case we would need a third option other than occupied and unoccupied.

Define an Enum

For the reasons mentioned in the previous slide *we will change* `occupied bool` *to an* `enum` *called* `Status` *and it will have 3 options* that as mentioned internally they will be represented as a uint8:

`(0) => UNAVAILABLE` (*the car cannot be rented*)

`(1) => AVAILABLE` (*the car is available for rent*)

`(2) => OCCUPIED` (*the car is currently rented*)

```
enum Status {
    UNAVAILABLE, AVAILABLE, OCCUPIED
}
```

Remember that the *default value for* `enums` *is the first element so for us it will be UNAVAILABLE*.

Next, we will change the functionality of the SC to accept the new `enum` we created.

Use Enum in Struct and Functions

We will start with the Car struct and instead of the bool occupied we will use our new Status enum. So, we will add a Status called status and remove the bool occupied.

```
struct Car {
    address mc;
    uint256 price;
    Status status; // New status
}
```

Next, we will change the registerCar function. In the line that we define the car, instead of false we will use Status.UNAVAILABLE.

This will make the car unavailable for rent when the car is register and then the owner of the car will have to make it available for rent.

This is good because it gives the owner some control, where previously anyone could rent the car the instance it was registered resulting in possible problems for the owner.

```
function registerCar(address _mc, uint256 _price) public {
    Car memory car = Car(_mc, _price, Status.UNAVAILABLE); // changed false to Status.UNAVAILABLE
    uint256 currentCarId = nextCarId;
    cars[currentCarId] = car;
    nextCarId++;
}
```

Use Enum in Struct and Functions (2)

Now we will change the function changeCarOccupied.

We will start be changing its name to changeCarStatus.

Since we do not have two options anymore, we will make the caller of the function to choose the Status they want to change the car to, by adding it as input.

For example, the owner would want to change the Status of a car to AVAILABLE so it can be rented or UNAVAILBLE to go the car for service. The user who rents the car would probably want to change it to OCCUPIED.

```
    function changeCarStatus(uint256 _carId, Status _status) public {
        if(_carId < nextCarId){
            Car storage car = cars[_carId];
            car.status = _status;
        }
    }
```

Code Complete

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract ICR {
    enum Status {
        UNAVAILABLE, AVAILABLE, OCCUPIED
    }
    struct Car {
        address mc;
        uint256 price;
        Status status;
    }
    mapping (uint256 _carId => Car _car) public cars;
    uint256 public nextCarId;

    function registerCar(address _mc, uint256 _price) public {
        Car memory car = Car(_mc, _price, Status.UNAVAILABLE);
        uint256 currentCarId = nextCarId;
        cars[currentCarId] = car;
        nextCarId++;
    }


    function changeCarStatus(uint256 _carId, Status _status) public {
        if(_carId < nextCarId){
            Car storage car = cars[_carId];
            car.status = _status;
        }
    }
}
```

Outro

You finished this lesson.

Task for home: Deploy the SC on Remix IDE, register a new car and try to change its `Status`.

Next, we will go over the concept of `msg.sender` in solidity.