**기본 문법**

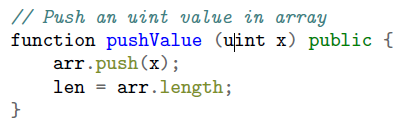
* **event**



event 는 블록체인 네트워크 블록에 특정 값을 기록하는 것

event 이름 (타입과 이름)

* **function**



function 함수명 (parameter) 접근제어자 {

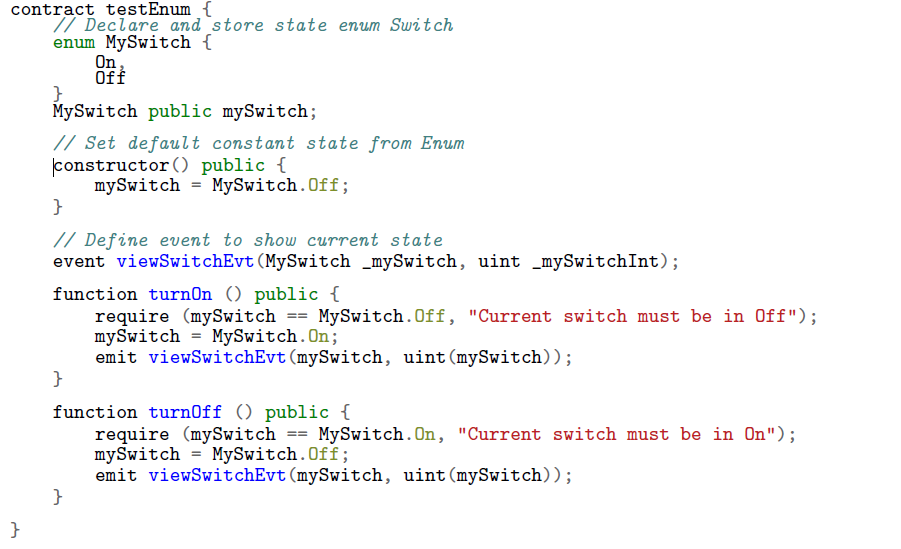
내용}

형식은(return 유무에 따라 여러 형식 존재) 이러하고 접근제어자는 public, private, internal, external 이 있음.

* **modifier**



Modifier는 함수의 동작을 변경시키기 위해 사용된다.



* **1번- enum**

enum은 사람이 읽을 수 있게 사용자/개발자에 의해 정의된 상수세트 타입. 한 개의 enum당 0부터 255까지 부여됨. 위 예시에서는 on – 0 / off – 1로 부여된 것.

* **2번- constructor**

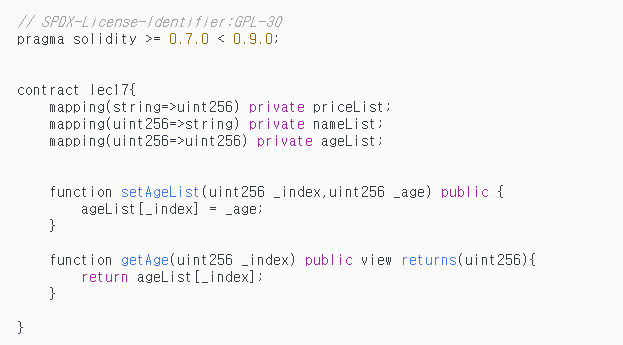
constructor는 컨트랙트 실행 시 처음 동작하는 값을 셋팅해 주는 것.

위 예시에서는 enum의 기본값을 off로 정해준 것.

* **3번-emit**

함수를 실행하는 것?

* **mapping**

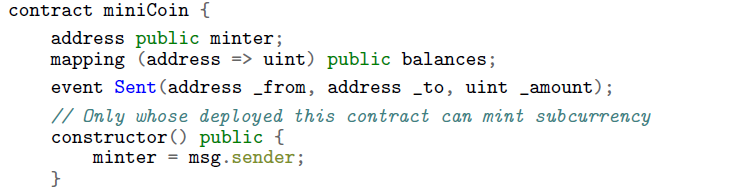


매핑은 키-값 쌍으로 이루어진 해시 테이블(그냥 키값으로 이루어진 테이블 저장소로 생각). 솔리디티에서 구조화된 데이터를 저장하는 또다른 방법

mapping (키의 타입 => 값의 타입) 접근제한자 변수 이름; 이렇게 정의(접근제한자 없을 수 있음)

위 예시에서 setAgeList 함수로 agelist 매핑 테이블에 값을 저장 가능하고, getAge 함수로 저장된 값을 불러오기 가능

* **msg.sender**



Msg.sender는 모든 함수에서 이용 가능한 전역 변수이고, 현재 함수를 호출한 사람(혹은 스마트 컨트랙트)의 주소를 가르킴

우리가 remix로 실습할 때 맨위에 주소 설정하고 함수 누를 때 생각해보면, 설정한 주소가 msg.sender임.

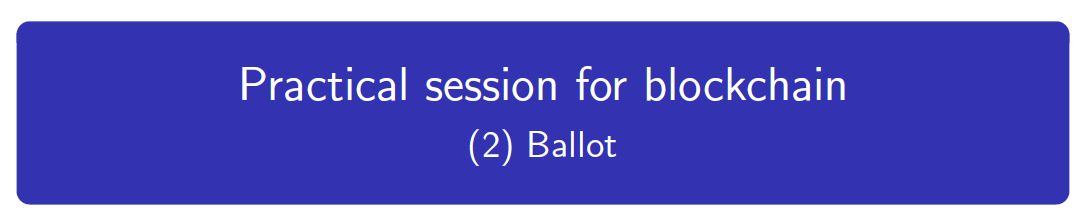
* **require**



특정 조건이 참이 아닐 때 함수가 에러 메시지를 발생하고 실행을 멈춤.

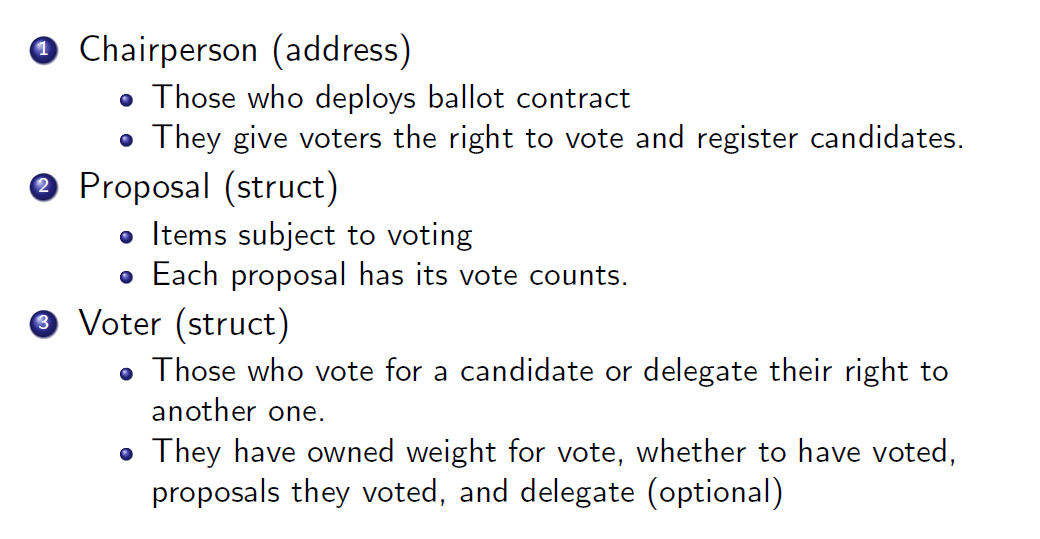
위 예시에서는 호출한 사람이 한번만 호출이 가능한 의미. 만약 이전에 해당 함수를 호출했다면 require 조건에 거짓이므로 에러발생 후 실행 멈춤

**이제 2강부터 대충 중요해 보이는 것만 정리해보자**

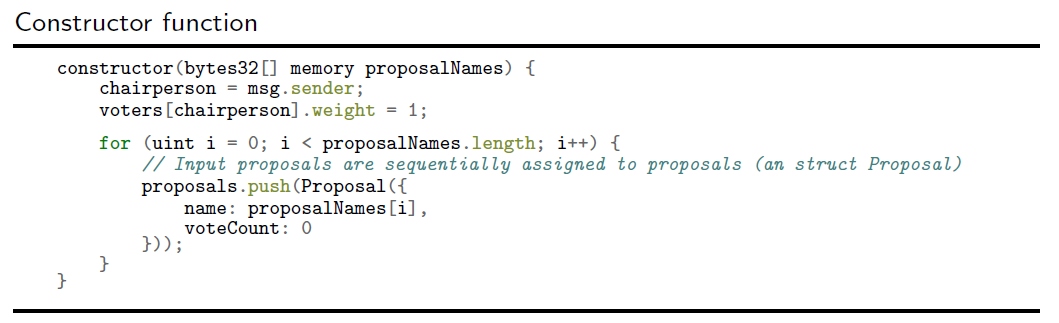




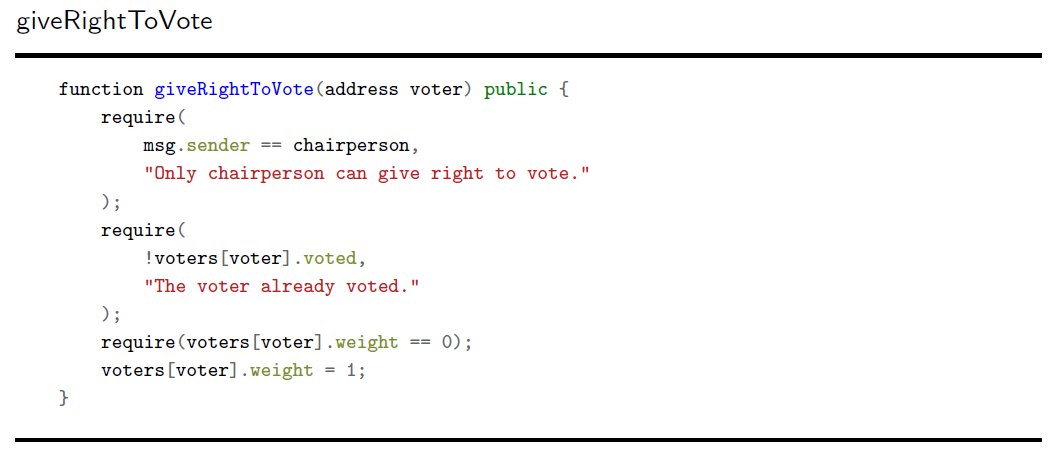
Ballot은 투표하는 기능



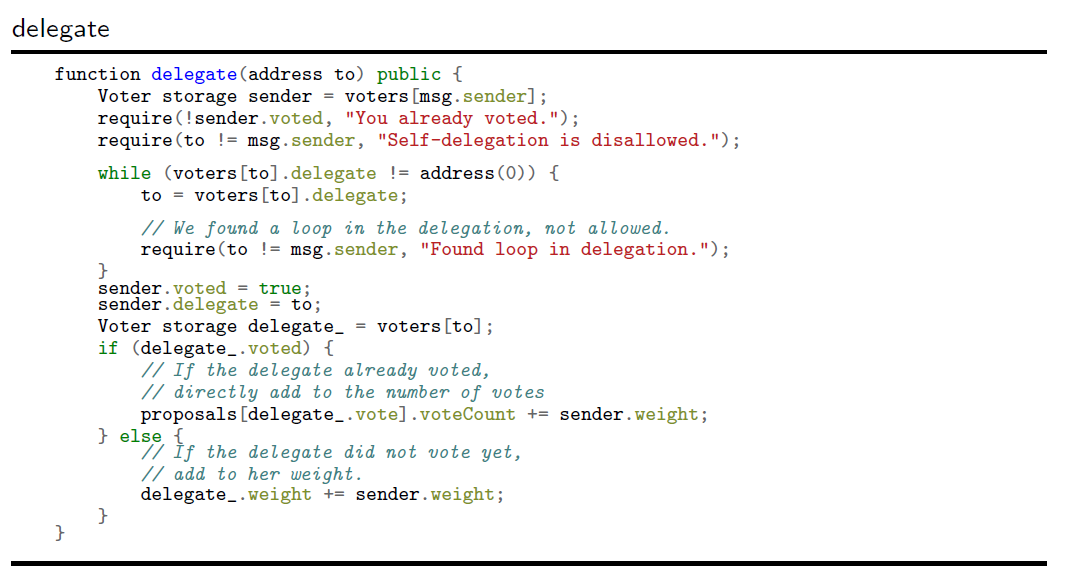
Chairperson, proposal, voter 구성됨



constructor에서 투표 후보인 제안들을 저장한다.(아닐수도있음 느낌만 보자)



Require 보면 체어펄슨만 가능한걸 알 수 있음. 이 함수는 참가자에게(파라미터가 보터의 주소임) 투표권을 주는것



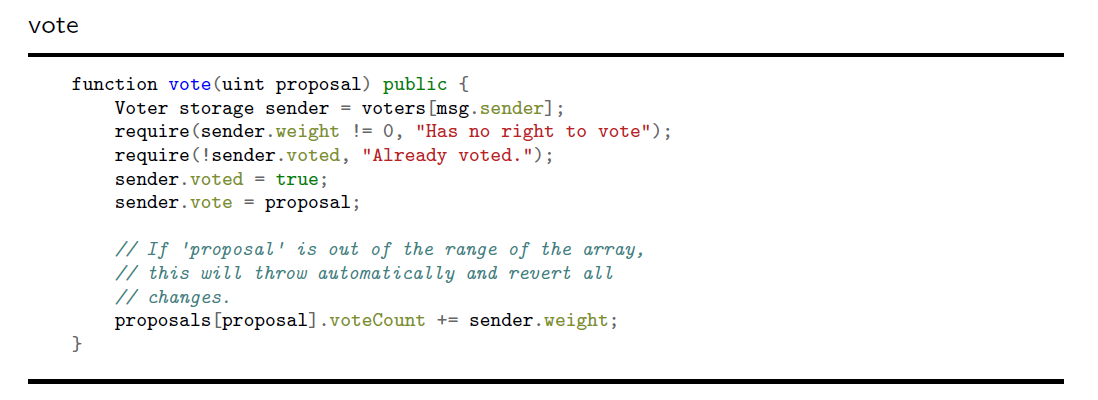
투표권을 위임하는 함수 2강중 이게 제일 어려운듯

Require 에서 확인하는 것은 - 이미 투표를 했는지? / 스스로에게 위임하는지?(무한루프로 오류 발생 가능하기 때문)

이 함수에서 파라미터를 보면 알 수 있다. to는 위임을 하고싶은 대상 주소임.

If 에서는 위임 대상이 투표를 했으면 to가 투표한 대상의 카운트를 1 증가

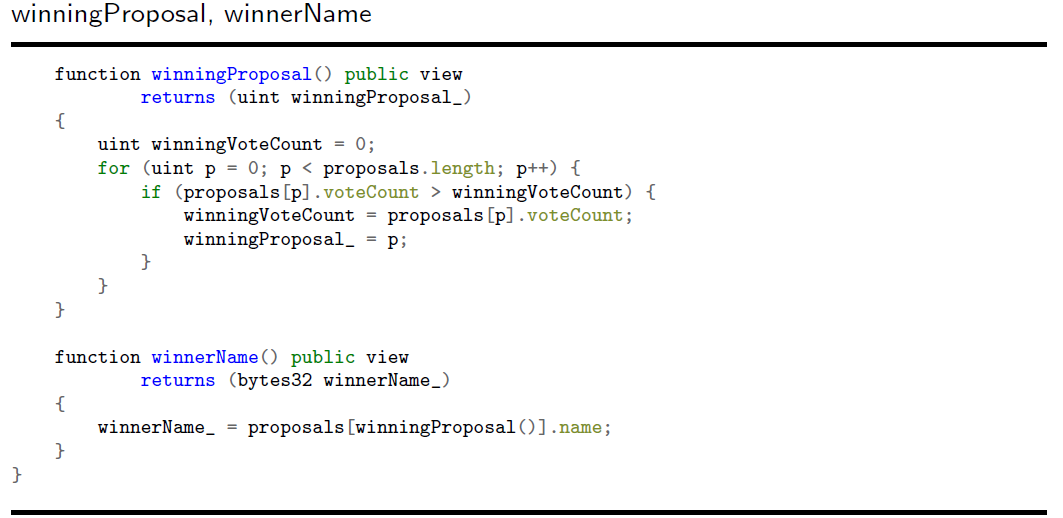
Else는 위임 대상의 weight를 자기가 갖고있는 투표권 만큼 증가시켜줌



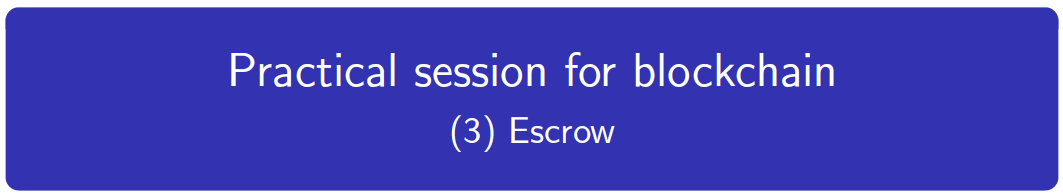
이 함수는 안건 중 하나의 카운트를 증가시키는것.

require에서 sender의 weight 즉, 투표권이 0이 아닌지 / 이미 투표를 했는지 확인을 한다.

그리고 require 두개가 참이면, sender의 투표 여부를 트루로 바꾸고. 투표한 번호의 카운트를 증가.

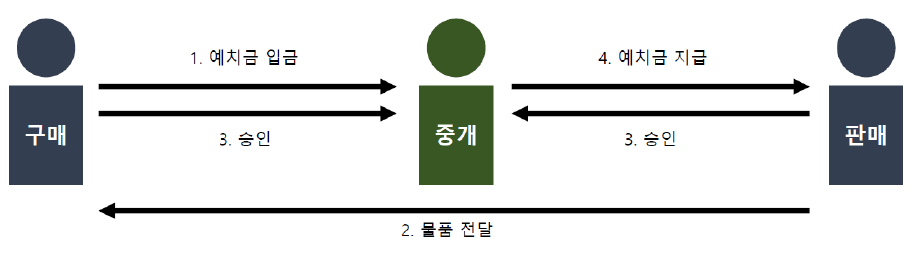


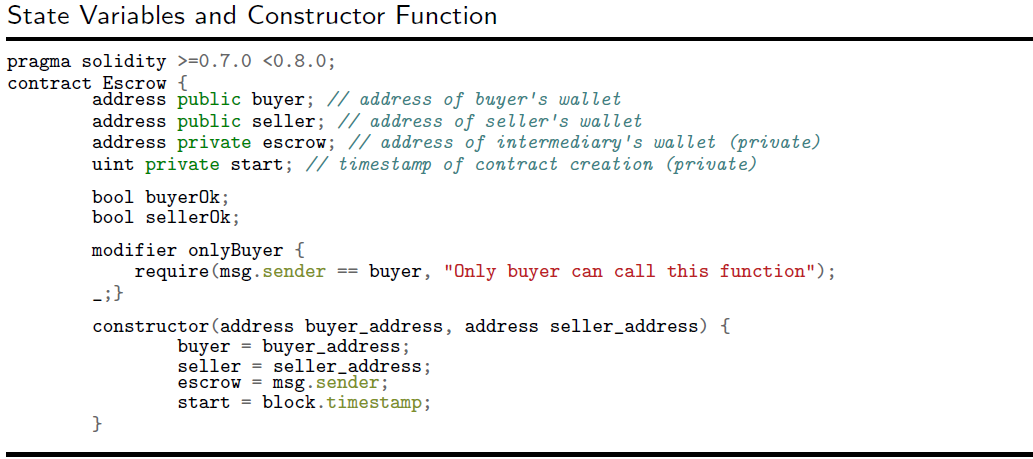
넘어가자





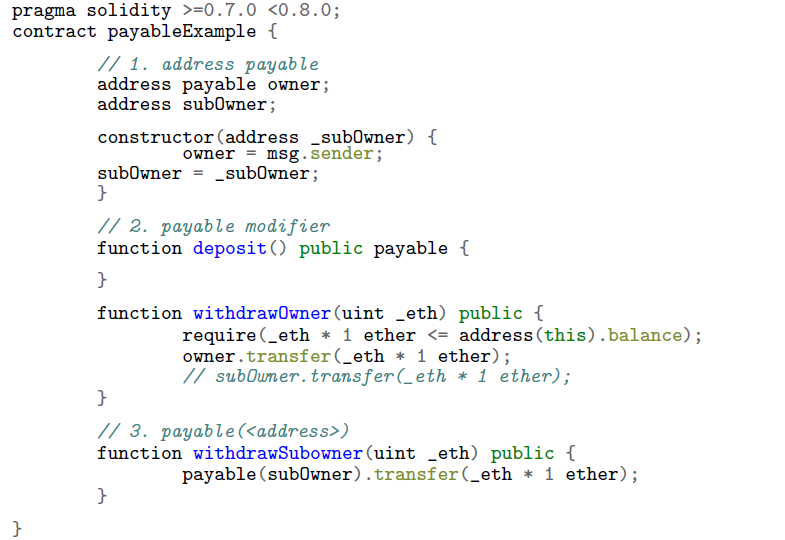
Escrow – 안전한 거래를 위한 방법(중개 거래?)





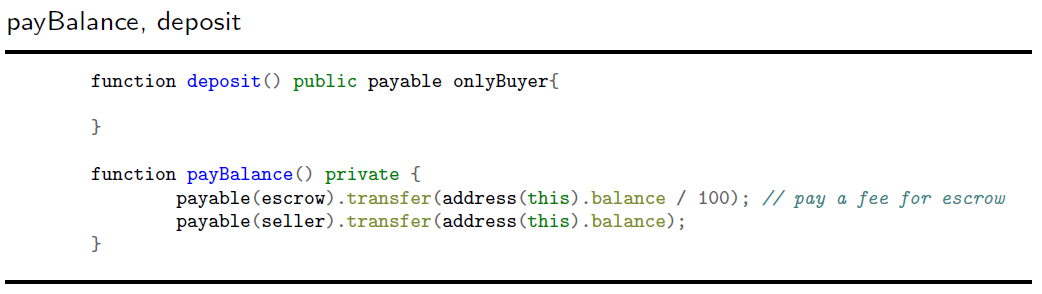
Escrow 컨트랙트

변수 선언과, modifier, constructor 정의

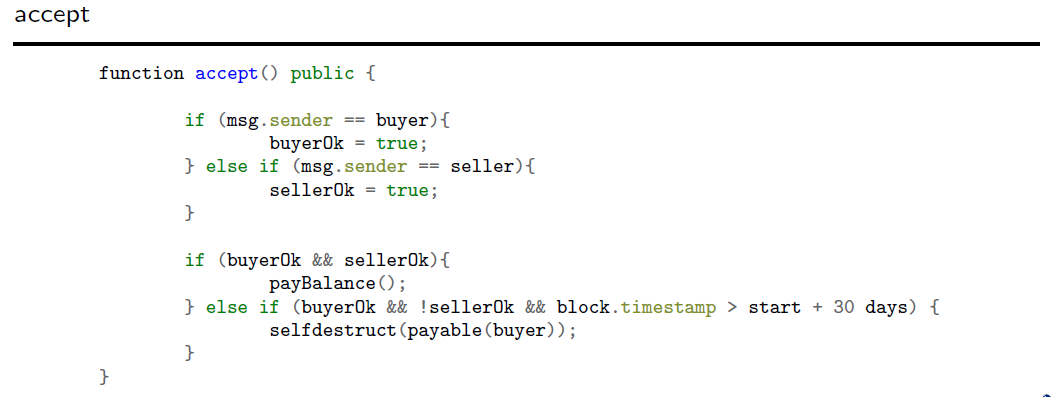


Payable 컨트랙트

Deposit , withdrawOwner, withdrawSubowner 별거 없어보임



payBalacne() 함수는 escrow에게 수수료 지급하고, seller에게 입금



Accept는 거래를 승인한다는 함수 구매자/판매자를 if/elif로 나눠서 한번에 처리해줌

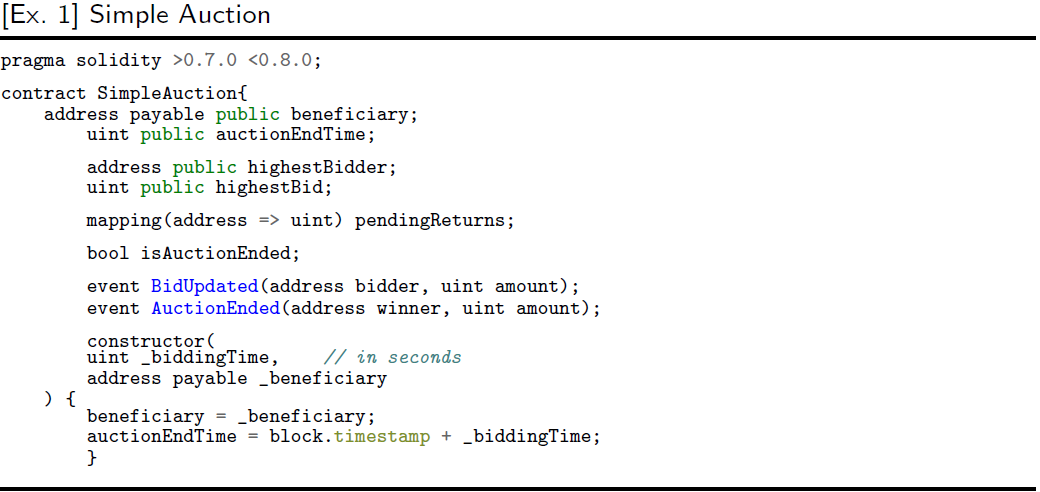
그리고 buyer/seller가 둘다 ok이면 돈 지급 둘다 ok 아니면 셀프디스트럭트(배포된 계약 파기)



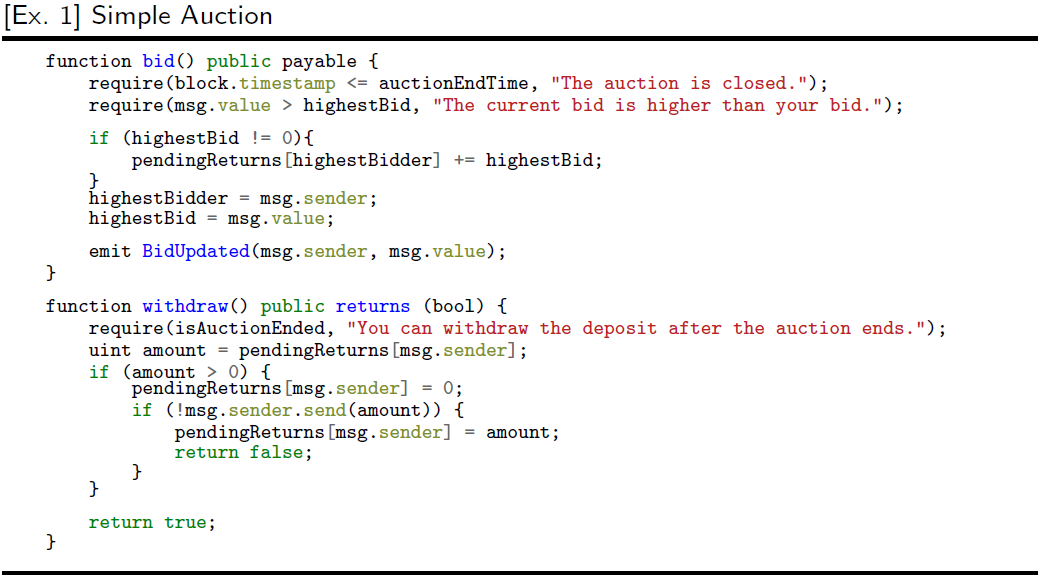


경매

심플옥션 and 블라인드 옥션



변수 선언, 매핑, 이벤트, constructor 정의함.

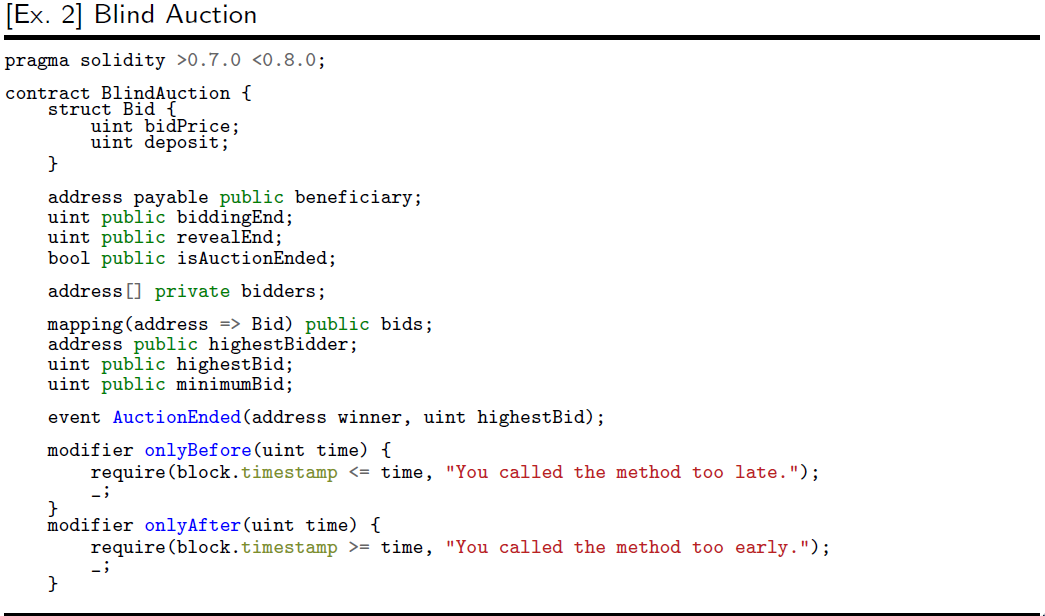


Bid 함수는 require에서 옥션시간 확인 / 현재 입찰 가격보다 낮은지를 체크

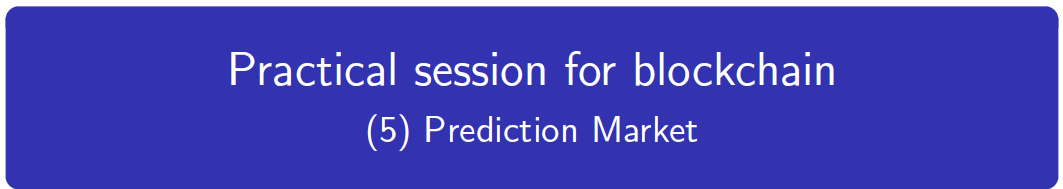
If문에서 최고입찰가가 0이 아니면 처음에 선언한 pendingReturns 매핑에 이름과 가격을 저장.

그리고 bidupdated 실행

Withdraw 함수는 경매가 끝난 후 실행 가능(require문) – 경매를 끝내는 함수

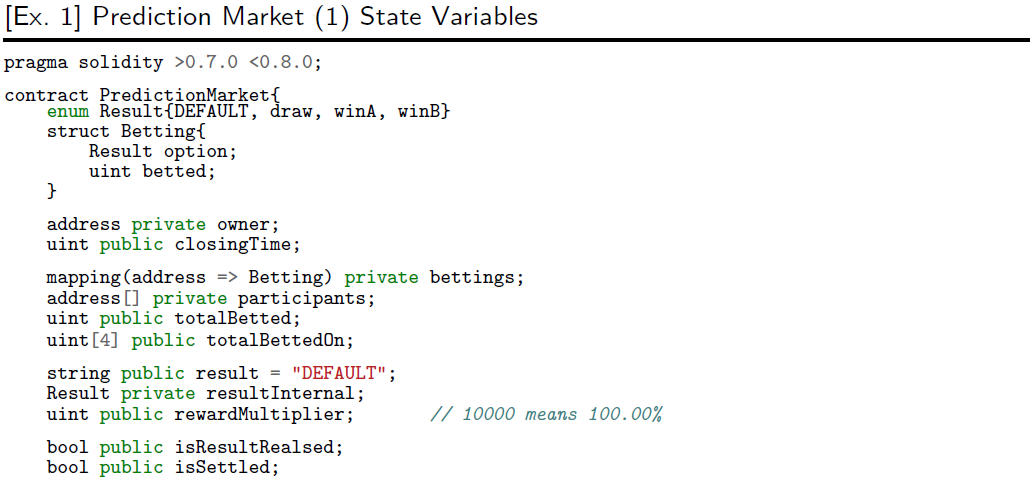


Blind 옥션 이건 **private**가 중요해보임(bidders!!!)

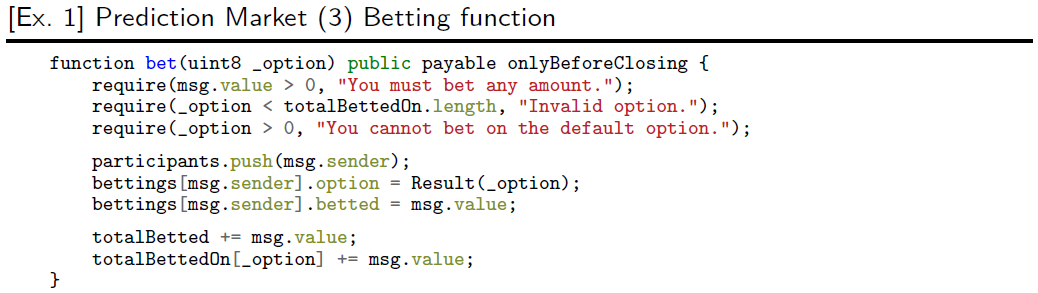




탈 중앙화 defi/ prediction market 예측? 도박?

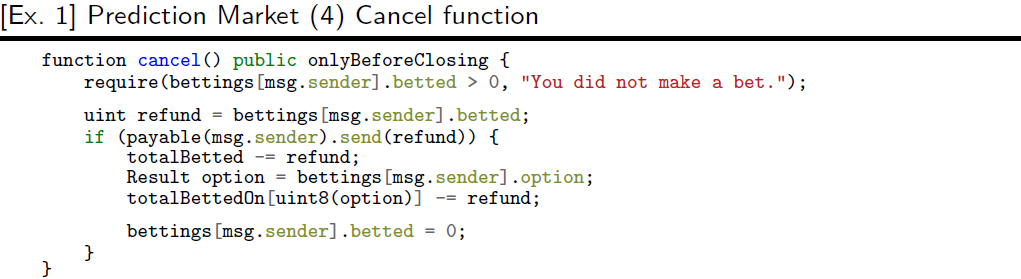


enum으로 default, draw, winA, winB(0, 1, 2, 3 부여), 그리고 변수, 매핑 선언



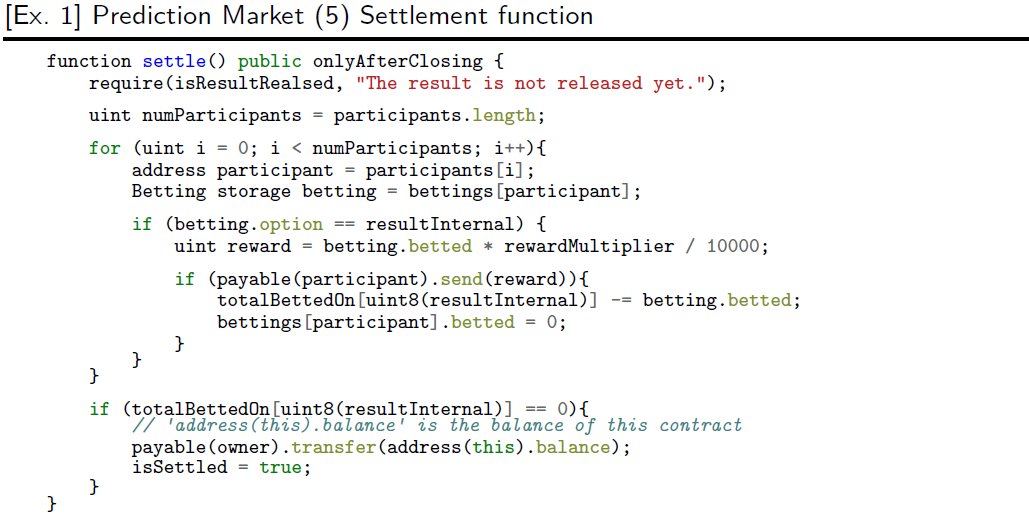
Bet 함수 - require에서 0원 이상을 배팅하고, 옵션이 범위 내에 있는 건지, 기본이 아니라 하나를 선택한지를 확인함.

매핑한 bettings에 옵션과 베팅값을 저장.



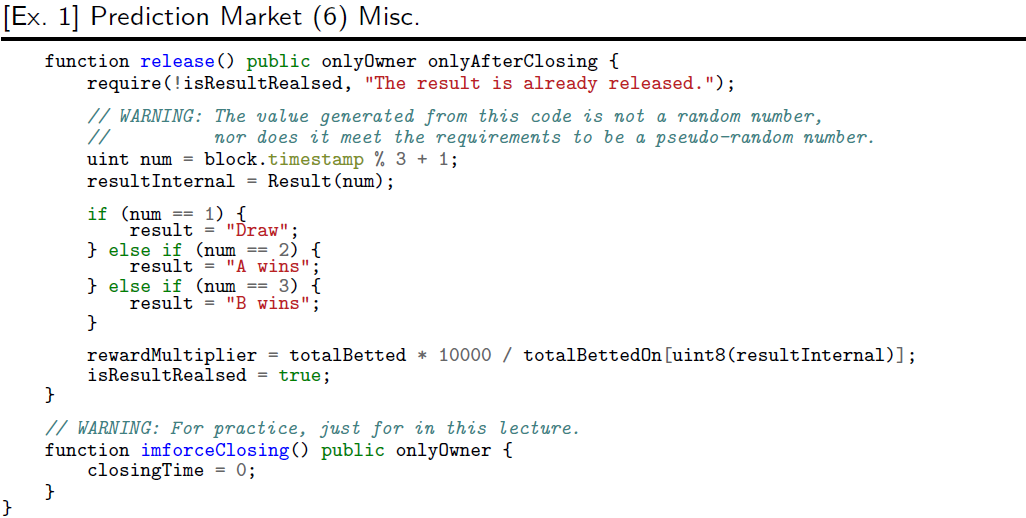
Require에서 베팅 여부를 확인.

bettings에서 refund 할 금액을 가져옴.



Settle은 결과가 나온지를 우선 확인.

for문에서 베팅한 참가자수만큼 반복함. -> 베팅한 옵션이 결과와 일치할 때 돈을 할당?(아닐수도)



Release는 결과 출력하는 함수 require에서 결과가 안나왔는지 확인

num으로 결과를 결정함 1일시에 draw 2 - winA, 3- winB



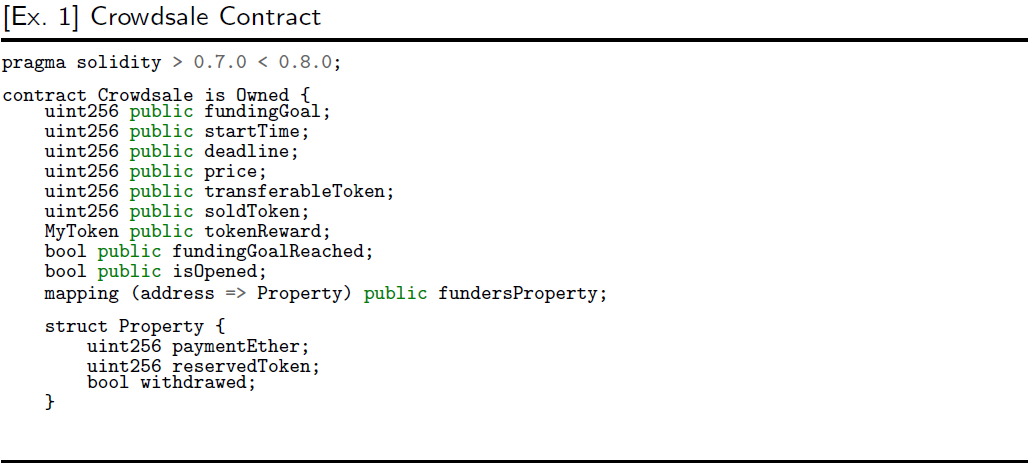
이건 잘 모르겠어서 패스



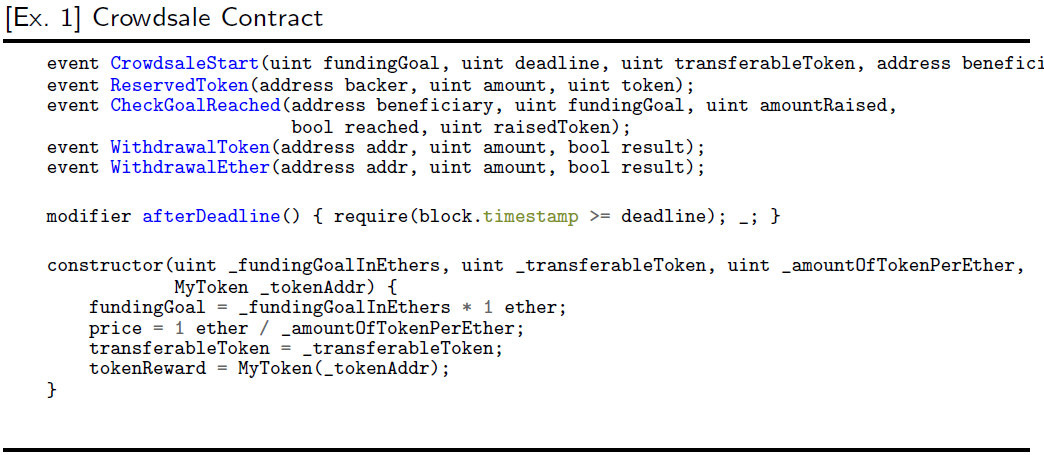
이것도 어렵네 패스



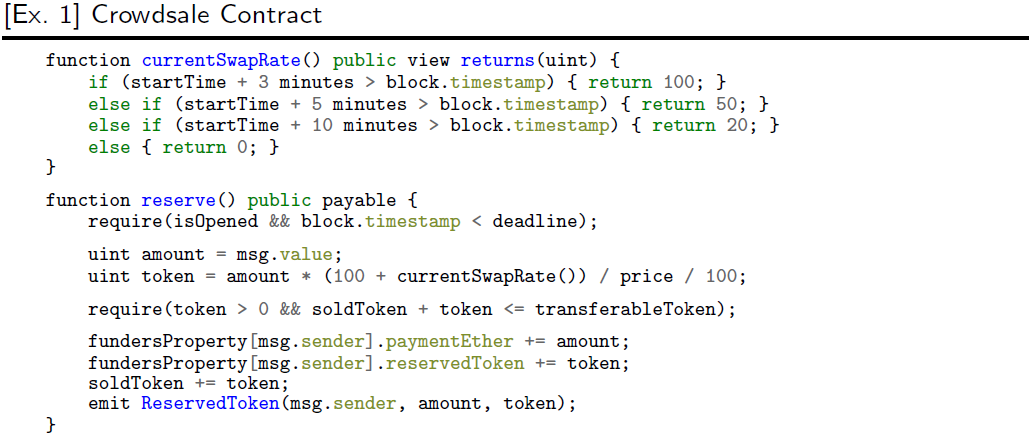
ICO 하는것 – 토큰 공모?



변수, fundersPropoerty 매핑, struct 선언

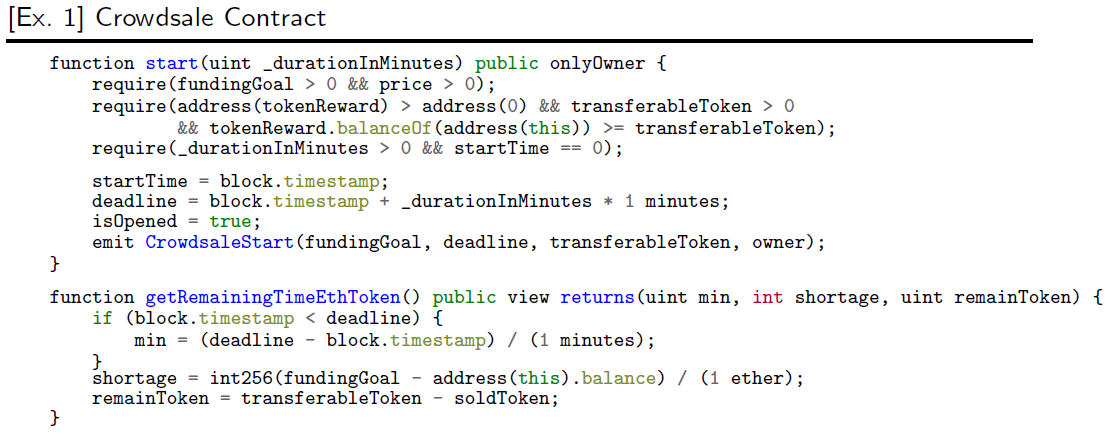


이벤트, modifier, constructor 선언



currentSwapRate는 처음 3분 내에 신청하면 100만큼, 3-5분내에는 50 만큼, 5분 이후는 20만큼 나머지는 0으로 제공하는 함수.

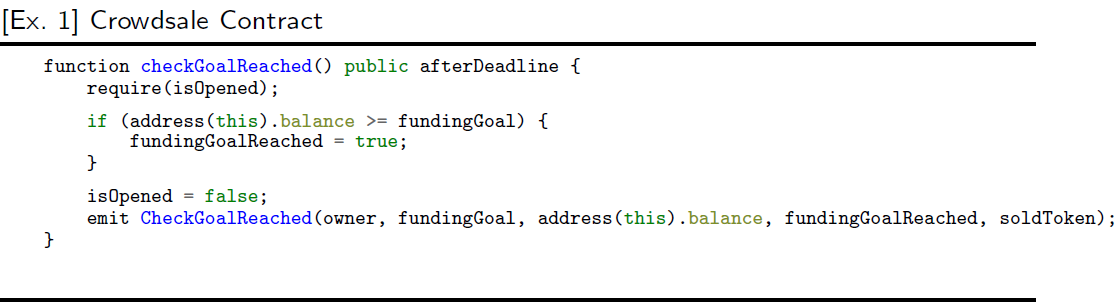
Reserve 함수는 토큰 예약하는 것



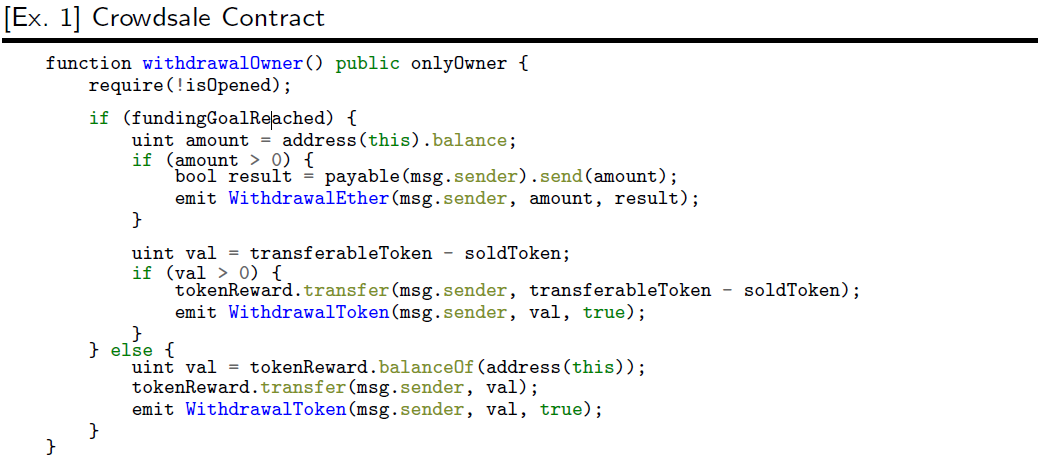
Start 함수

Emit CrowdsaleStart(fundingGoal, deadline, transferableToken, owner); 확인

getRemainingTimeEthToken중 view 키워드는 가스를 소모하지 않고 조회를 위해 사용되는 키워드



if문 emit에서 address(this) <- this 한번 보기 함수 호출자가 아닌 계약의 인스턴스 주소!



여기서는 require(!isOpened) <- ! 확인, 그리고 msg.sender 확인