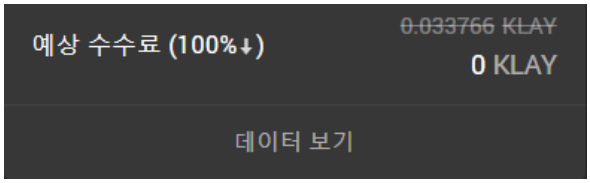
2022/05/27

수수료 테스트

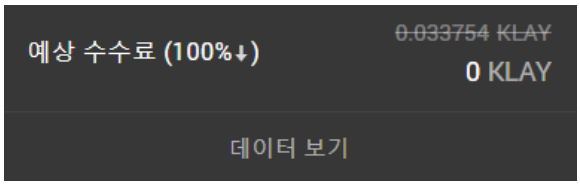
저장하는 변수가 작아지면 수수료가 더 작아질것으로 생각하고 토큰 id, price변수 크기를 조절하여 테스트 하였다.

```
contract SoundgramAlbumSales {
  KIP17Full public nftAddress;

  mapping(uint64 => uint128) public tokenPrice;

  constructor(address _tokenAddress) public {
      nftAddress = KIP17Full(_tokenAddress); //YTT 주소를 넘김. 컨트랙에 있는 함수 사용 가능.
  }

  function setForSale(uint64 _tokenId, uint128 _price) public {
      address tokenOwner = nftAddress.ownerOf(_tokenId);    //public, external 함수만 이렇게 호출 가능
      require(tokenOwner == msg.sender, "caller is not token owner");
      require(_price > 0, "price is zero or lower");
      require(nftAddress.isApprovedForAll(tokenOwner,address(this)), "token owner did not approve TokenSales contract");

      tokenPrice[_tokenId] = _price; //특정 토큰의 가격 저장.
  }

  function purchaseToken(uint64 _tokenId) public payable {
      uint256 price = tokenPrice[_tokenId];
      address tokenSeller = nftAddress.ownerOf(_tokenId);

      require(msg.value == price, "caller sent klay not same price"); //가격 이상의 비용을 지불해야 함.
      require(msg.sender != tokenSeller, "caller is token seller");    //본인은 구매 불가
      address payable payableTokenSeller = address(uint160(tokenSeller));

      payableTokenSeller.transfer(msg.value);

      nftAddress.safeTransferFrom(tokenSeller, msg.sender, _tokenId); //tokenSeller->구매자 토큰 전송
      tokenPrice[_tokenId] = 0;
  }

  function removeTokenOnSale(uint64 _tokenId) public {
      address tokenSeller = nftAddress.ownerOf(_tokenId);
      require(msg.sender == tokenSeller, "caller is not tokenSeller");
      tokenPrice[_tokenId] = 0;
  }
}
```



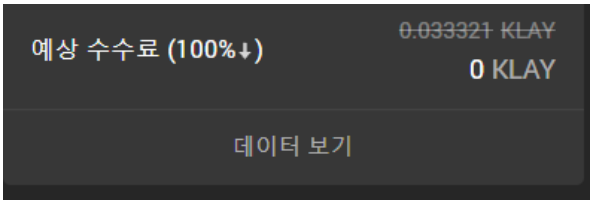
0xb8d8...5bb4e1	91398138	1 min ago	0x4e85...83d364	0xb376...926d9c	0x2271e036	Fee...Smart Contract Exec...	0.000000	0.018133

```
contract SoundgramAlbumSales {
  KIP17Full public nftAddress;

  mapping(uint256 => uint128) public tokenPrice;

  constructor(address _tokenAddress) public {
      nftAddress = KIP17Full(_tokenAddress); //YTT 주소를 넘김. 컨트랙에 있는 함수 사용 가능.
  }

  function setForSale(uint256 _tokenId, uint128 _price) public {
      address tokenOwner = nftAddress.ownerOf(_tokenId);    //public, external 함수만 이렇게 호출 가능
      require(tokenOwner == msg.sender, "caller is not token owner");
      require(_price > 0, "price is zero or lower");
      require(nftAddress.isApprovedForAll(tokenOwner,address(this)), "token owner did not approve TokenSales contract");

      tokenPrice[_tokenId] = _price; //특정 토큰의 가격 저장.
  }

  function purchaseToken(uint256 _tokenId) public payable {
      uint256 price = tokenPrice[_tokenId];
      address tokenSeller = nftAddress.ownerOf(_tokenId);

      require(msg.value == price, "caller sent klay not same price"); //가격 이상의 비용을 지불해야 함.
      require(msg.sender != tokenSeller, "caller is token seller");    //본인은 구매 불가
      address payable payableTokenSeller = address(uint160(tokenSeller));

      payableTokenSeller.transfer(msg.value);

      nftAddress.safeTransferFrom(tokenSeller, msg.sender, _tokenId); //tokenSeller->구매자 토큰 전송
      tokenPrice[_tokenId] = 0;
  }

  function removeTokenOnSale(uint256 _tokenId) public {
      address tokenSeller = nftAddress.ownerOf(_tokenId);
      require(msg.sender == tokenSeller, "caller is not tokenSeller");
      tokenPrice[_tokenId] = 0;
  }
}
```




```
contract SoundgramAlbumSales {
  KIP17Full public nftAddress;

  mapping(uint256 => uint128) public tokenPrice;

  constructor(address _tokenAddress) public {
      nftAddress = KIP17Full(_tokenAddress); //YTT 주소를 넘김. 컨트랙에 있는 함수 사용 가능.
  }

  function setForSale(uint256 _tokenId, uint256 _price) public {
      address tokenOwner = nftAddress.ownerOf(_tokenId);    //public, external 함수만 이렇게 호출 가능
      require(tokenOwner == msg.sender, "caller is not token owner");
      require(_price > 0, "price is zero or lower");
      require(nftAddress.isApprovedForAll(tokenOwner,address(this)), "token owner did not approve TokenSales contract");
```

```
        tokenPrice[_tokenId] = _price; //특정 토큰의 가격 저장.
    }

    function purchaseToken(uint256 _tokenId) public payable {
        uint256 price = tokenPrice[_tokenId];
        address tokenSeller = nftAddress.ownerOf(_tokenId);

        require(msg.value == price, "caller sent klay not same price"); //가격 이상의 비용을 지불해야 함.
        require(msg.sender != tokenSeller, "caller is token seller");    //본인은 구매 불가
        address payable payableTokenSeller = address(uint160(tokenSeller));

        payableTokenSeller.transfer(msg.value);

        nftAddress.safeTransferFrom(tokenSeller, msg.sender, _tokenId); //tokenSeller->구매자 토큰 전송
        tokenPrice[_tokenId] = 0;
    }

    function removeTokenOnSale(uint256 _tokenId) public {
        address tokenSeller = nftAddress.ownerOf(_tokenId);
        require(msg.sender == tokenSeller, "caller is not tokenSeller");
        tokenPrice[_tokenId] = 0;
    }
}
```



이더리움 기반의 스마트 컨트랙트는 변수 크기와 관계없이 실제 저장공간은 256비트이기 때문에 uint8이라도 256비트만큼을 사용하여 저장하고, 오히려 이과정에서 부족한 부분만큼 0으로 채우는것을 이더리움 가상머신이 수행하므로 실행비용이 더 나가게 된다.

작은 변수를 통합하여 수수료를 줄이고 싶다면 구조체를 사용하거나, 인코더 디코더 함수를 만들어 변수를 보낼때 통합하여 보내면 되나, 이러한 인코딩 디코딩도 가상머신의 실행 과정에서 실행 수수료가 추가로 부여되며 코드의 가독성이 떨어지므로, 줄여지는 변수량과 실행비용을 잘 고려하여 결정해야한다.

https://ko.docs.klaytn.foundation/klaytn/design/transaction-fees

코드 다듬기