Bài thực hành số 5: Xây dựng và triển khai hợp đồng thông minh cho NFT Marketplace

1. Mục tiêu

Sau khi công nghệ token ERC-20 đã được giới thiệu trong bài lab 04, dday cũng là công nghệ trở nên phổ biến trên Ethereum. Trong bài lab số 5, NFT được giới thiệu, đây cũng là công nghệ ngày càng được ưa chuộng và đang được nhiều người sử dụng. Mọi người mua bán nghệ thuật số dưới dạng NFT. Bài thực hành nhằm giúp sinh viên nắm vững các kiến thức và kỹ năng cơ bản trong việc xây dựng và triển khai NFC trên Ethereum bằng ngôn ngữ Vyper. Sau khi hoàn thành, sinh viên có thể:

- Trình bày được nguyên lý hoạt động của NFT.
- Tiêu chuẩn ERC-721.
- Tạo hợp đồng thông minh cho NFT.

2. Thời gian làm bài

Dự kiến: 3 giờ.

3. Cơ sở lý thuyết và bài tập tại lớp

A. NFT là gì?

NFT là từ viết tắt của non-fungible token (token không thể thay thế). Định nghĩa "token" ở đây giống với định nghĩa "token" trong token theo chuẩn ERC-20. Token là đại diện cho một "thứ gì đó". Ví dụ, token ERC-20 có thể đại diện cho tiền kỹ thuật số hoặc tiền ảo. Còn với NFT, điểm khác biệt nằm ở cụm từ "non-fungible" (không thể thay thế).

Vậy "fungible" (có thể thay thế) nghĩa là gì? Khi một thứ có tính fungible, nghĩa là một token này có giá trị ngang bằng với một token khác cùng loại. Ví dụ, giả sử ta tạo ra 1000 token ERC-20 tên là PHO. Điều này có nghĩa là bất kỳ một token PHO cụ thể nào cũng có giá trị như bất kỳ token PHO nào khác.

Nếu điều đó làm các em bối rối, các em có thể thay thế PHO bằng đô la Mỹ (USD). Giả sử các em có 100 USD trong ví và bạn em cũng có 100 USD trong ví của bạn em. 100 USD của em có giá trị tương đương với 100 USD của bạn. Nói cách khác, USD có tính fungible (có thể thay thế). 1 USD có cùng giá trị với 1 USD khác.

Quay lại với NFT, token trong NFT là non-fungible (không thể thay thế). 1 token NFT này không nhất thiết có giá trị tương đương với 1 token NFT khác. Mỗi token NFT là duy nhất. Giả sử các em có một bộ sưu tập NFT gồm 10 token: NFT token số 1, NFT token số 2, NFT token số 3, và cứ thế cho đến NFT token số 10. NFT token số 1 khác với NFT token số 2. Mỗi token NFT là độc nhất. Các em không thể xử lý với NFT token số 3 giống như cách xử lý với NFT token số 7.

Nếu điều này quá trừu tượng, ta có thể thay thế bộ sưu tập NFT bằng một bộ sưu tập thẻ Pokemon. Một lần nữa, giả sử bộ sưu tập thẻ Pokemon của chúng ta có 10 món. Món số 1 là thẻ có hình Pikachu (một nhân vật Pokemon được yêu thích). Món số 2 là thẻ có hình Bulbasaur (một nhân vật Pokemon được yêu thích khác). Ta không thể nói thẻ Pikachu có giá trị tương đương với thẻ Bulbasaur, đây là 2 thẻ hoàn toàn khác nhau, và tất nhiên chúng có giá trị khác nhau. Một thẻ Pikachu có thể có giá trị hơn một thẻ Bulbasaur hoặc ngược lại.

Nếu token ERC-20 tương tự như đô la Mỹ, thì token NFT tương tự như thẻ Pokemon. Nói cách khác, nếu token ERC-20 là tiền tệ ảo, thì token NFT là tác phẩm nghệ thuật số.

Tuy nhiên, nghệ thuật số trong ngữ cảnh này không giống như những bức ảnh kỹ thuật số mà một người tạo ra bằng ứng dụng như Photoshop hay Adobe Illustrator. Một bức ảnh kỹ thuật số là bức ảnh bạn phải xem trên một nền tảng kỹ thuật số như điện thoại di động, máy tính xách tay hoặc máy tính để bàn. Một thẻ Pokemon dưới dạng NFT có một thứ mà một bức ảnh kỹ thuật số thông thường không có, đó là quyền sở hữu được nhúng vào chính token. Ngoài ra, NFT cho phép bạn đưa dữ liệu vào blockchain.

Đề cập đến giá trị đề xuất về quyền sở hữu, mọi NFT đều đi kèm với thông tin về chủ sở hữu. Trong ngữ cảnh của những bức ảnh kỹ thuật số thông thường thì khác. Ví dụ, một bức ảnh kỹ thuật số về một con mèo không ngụ ý ai là chủ sở hữu. Bạn phải đặt thông tin quyền sở hữu của bức ảnh con mèo kỹ thuật số đó ở một nơi khác. Ví dụ, bạn có thể đặt nó trong cơ sở dữ liệu của một ứng dụng web lưu trữ hình ảnh, bao gồm cả bức ảnh con mèo kỹ thuật số đó. Quyền sở hữu có thể được đặt trong một bảng với một cột có dữ liệu có tên là url_picture, tham chiếu đến vị trí lưu trữ của bức ảnh con mèo kỹ thuật số, và một cột có tên là profile_id, tham chiếu đến ID của một hàng trong bảng profiles - nơi chứa thông tin của người sở hữu.

Do đó, chúng ta có thể nói rằng một token NFT đi kèm với ý nghĩa về quyền sở hữu. Điều này tương tự như trường hợp mọi token ERC-20 đều đi kèm với thông tin quyền sở hữu. Trong ERC-20, thông tin quyền sở hữu nằm trong biến trạng thái balanceOf trong hợp đồng thông minh.

Ứng dụng của NFT không chỉ giới hạn ở nghệ thuật số trên blockchain. Nó có thể đại diện cho những thứ khác như vé concert chẳng hạn. Một chiếc vé có thể được gắn với một chỗ ngồi cụ thể. Một chỗ ngồi ở hàng ghế đầu sẽ khác với một chỗ ngồi ở hàng ghế thứ hai, mặc dù chúng có thể có cùng giá bán. Vì vậy, theo một cách nào đó, một chiếc vé được gắn với một chỗ ngồi cụ thể là duy nhất hay có tính "non-fungible" (không thể thay thế). Nó giống như một NFT vậy.

Bây giờ các em bắt tay vào tạo ra một smart contract NFT đơn giản để minh hoạ. Sử dụng Ape freamwork để tạo một project đặt tên là Lab05_MSSV. Sau khi tạo thành công các em chụp hình cấu trúc project bỏ vào trong file báo cáo.

Trong thư mục contracts, các em tạo file có tên VerySimpleNFT.vy có nội dung như hình số 1.

```
# @version ^0.4.3

ownerOf: public(HashMap[uint256, address])
@deploy
def __init__():
    for i: uint256 in range(10):
        self.ownerOf[i] = msg.sender
@external
def transfer(tokenId: uint256, destination: address):
    if self.ownerOf[tokenId] == msg.sender:
        self.ownerOf[tokenId] = destination
```

Hình 1. Đoạn chương trình minh hoạ smart contract NFT đơn giản

Để theo dõi quyền sở hữu của một NFT, ta sử dụng biến trạng thái ownerOf. Trong token ERC-20, ta dùng biến trạng thái balanceOf để theo dõi một địa chỉ ví đang sở hữu bao nhiêu token. Còn ở đây, ta dùng biến trạng thái ownerOf để theo dõi tài khoản nào đang sở hữu NFT nào. Trong biến balanceOf, khóa (key) của biến là một địa chỉ ví và giá trị (value) của biến là một con số. Biến này trả lời cho câu hỏi một người dùng có bao nhiêu token. Hãy lưu ý rằng ta không cần phải theo dõi xem một địa chỉ đang sở hữu những token cụ thể nào, bởi vì mọi token trong hợp đồng thông minh ERC-20 đều có cùng giá trị, nghĩa là có thể thay thế cho nhau (fungible).

Tuy nhiên, trong hợp đồng thông minh NFT, ta cần phải theo dõi quyền sở hữu cụ thể. Vì vậy, khóa của biến trạng thái ownerOf là một số nguyên (integer) và giá trị của biến ownerOf là một địa chỉ ví. Biến này trả lời cho câu hỏi ai là chủ sở hữu của NFT đó. Một NFT trong hợp đồng thông minh được đại diện bởi một con số hoặc một số nguyên. NFT số 0 thuộc về địa chỉ A. NFT số 1 thuộc về địa chỉ B, v.v... NFT số 0 là khác biệt so với NFT số 1.

Để chuyển nhượng một NFT hay quyền sở hữu của một NFT, ta có thể sử dụng phương thức transfer. Trong phương thức này, ta thay đổi giá trị của biến trạng thái ownerOf ứng với một khóa (ID của NFT) cụ thể. Nếu NFT số 0 thuộc về địa chỉ A, thì địa chỉ A có thể chuyển chủ sở hữu của NFT số 0 sang một địa chỉ khác, chẳng hạn như địa chỉ B. Kết quả là, NFT số 0 sẽ thuộc về địa chỉ B. Đây chính là cốt lõi của một hợp đồng thông minh NFT.

Bây giờ, ta hãy cùng tạo file conftest.py thực hiện kiểm thử cho hợp đồng thông minh này có nội dung như hình số 2.

```python
import pytest
@pytest.fixture
def deployer(accounts):
    return accounts[0]
@pytest.fixture
def contract(deployer, project):
    return deployer.deploy(project.VerySimpleNFT)
```

Hình 2. Thực hiện kiểm thử cho smart contract NFT đơn giản.

File conftest.py là tệp hỗ trợ cho việc kiểm tra, giúp ta lấy được hợp đồng thông minh đã được triển khai và địa chỉ của người triển khai. Sau đó, các em hãy tạo file kiểm tra có tên test_VerySimpleNFT.py bên trong thư mục tests và thêm đoạn mã như hình số 3.

```python
def test_ownerOf(contract, deployer):
    for i in range(10):
        assert deployer == contract.ownerOf(i)

def test_transfer(contract, deployer, accounts):
    destination_account = accounts[1]
    tokenId = 2
    contract.transfer(tokenId, destination_account, sender=deployer)
    assert destination_account == contract.ownerOf(tokenId)
    for i in range(2):
        assert deployer == contract.ownerOf(i)
    for i in range(3, 10):
        assert deployer == contract.ownerOf(i)
```

Hình 3. Đoạn chương trình kiểm tra chủ sở hữu và chức năng chuyển tiền.

Trong hình số 3, hàm test_ownerOf kiểm tra xem chủ sở hữu của tất cả các token NFT có phải là người triển khai hợp đồng hay không. Sau đó, hàm test_transfer kiểm tra xem hàm chuyển tiền có hoạt động hay không, trước tiên ta gọi hàm transfer, sau đó kiểm tra xem chủ sở hữu của NFT có ID là 2 đã thay đổi từ địa chỉ của người triển khai sang địa chỉ đích hay không.

Sau khi thực hiện xong rồi, các em có thể dùng lệnh biên dịch chương trình: ape compile.

Chạy chương trình test bằng câu lệnh sau: py.test tests/test_VerySimpleNFT.py

Các em chụp kết quả chạy được đưa vào file báo cáo.

Nhưng làm thế nào để ta lưu trữ tác phẩm nghệ thuật kỹ thuật số (chẳng hạn như một bức ảnh) trên hợp đồng thông minh? Về cơ bản, đó là một quy ước. NFT với ID là 0 sẽ ám chỉ thẻ Pikachu. NFT với ID là 1 sẽ ám chỉ thẻ Bulbasaur. Việc các em thiết lập quy ước này như thế nào là tùy thuộc vào dự án các em triển khai. Vậy, tại sao điều này lại quan trọng?

Trong trường hợp của token ERC-20, ta đánh giá cao giá trị của tính minh bạch về các quy tắc trong hợp đồng thông minh ERC-20 và khả năng chống kiểm duyệt của token ERC-20. Khi ai đó tạo ra một hợp đồng thông minh mà trong đó họ không thể tạo thêm token, điều đó có nghĩa là họ không thể tùy ý thay đổi luật đưa ra ban đầu. Khi người

khởi tạo contract thay đổi luật thì tất cả người tham gia trong smart contract có thể thấy điều đó và quyết định xem họ có muốn tham gia sử dụng hợp đồng thông minh đó hay không. Trong một hợp đồng thông minh cụ thể, nếu một người gửi chuyển một số token cho người dùng khác và quy tắc của hợp đồng thông minh đã thực thi điều đó, thì người gửi không thể hoàn tác giao dịch, bất kể họ có mong muốn đều đó đến đâu. Người triển khai hợp đồng thông minh có thể tạo ra một hợp đồng mà ngay cả chính họ cũng không thể kiểm duyệt các giao dịch đang diễn ra bên trong hợp đồng đó.

Trong trường hợp của NFT, điều này cũng tương tự. Ta có thể triển khai một hợp đồng thông minh mà bất kỳ người dùng nào, kể cả chính người khởi tạo, cũng không thể hủy bỏ việc chuyển nhượng một NFT. Ta có thể tạo ra một số lượng token NFT giới hạn trong một hợp đồng thông minh. Ví dụ: ta có thể tạo một bộ sưu tập 10 token đại diện cho thẻ Pokemon trong một hợp đồng thông minh và đặt ra quy tắc không thể thêm token sau khi hợp đồng thông minh được triển khai. Vì vậy, hãy tưởng tượng, ta mua token NFT số 1 từ người triển khai hợp đồng thông minh NFT hoặc có thể nhận được token NFT số 1 như một món quà từ họ. Rồi một ngày nào đó, NFT của ta có thể trở nên vô cùng giá trị. Khi điều này xảy ra, người triển khai có thể cảm thấy hối hận về quyết định của mình. Họ muốn lấy lại nó nhưng không thể.

Điều này có nghĩa là NFT thuộc về bạn, và không ai có thể lấy nó khỏi bạn. Theo cách này, công nghệ NFT mang lại quyền sở hữu kỹ thuật số đối với các vật phẩm hiếm. Quyền sở hữu có thể được chứng minh ngay lập tức trên blockchain. Do đó, có thể nói rằng nếu công nghệ token ERC-20 mang lại tiền tệ cho blockchain, thì công nghệ NFT mang lại nghệ thuật số cho blockchain. Một điều thú vị nữa về NFT là ta có thể xem được lịch sử sở hữu của một NFT cụ thể.

Mọi người có thể thấy liệu bạn có được NFT từ đợt bán hàng sơ cấp (bạn mua NFT trực tiếp từ chủ sở hữu hợp đồng thông minh) hay từ đợt bán hàng thứ cấp (bạn mua từ một người đã mua nó từ người khác). Mọi người có thể xác minh rằng bạn đã sở hữu NFT ngay từ đầu và chưa từng bán nó đi, qua đó chứng minh được sự gắn bó của bạn với vật phẩm đó.

NFT có thể được sử dụng trong nhiều lĩnh vực khác nhau, ở các ngữ cảnh khác nhau từ lĩnh vực bất động sản, bán lẻ, truyền hình, nghệ thuật, giáo dực, …

Có một điều thú vị là ta có thể thực hiện chuyển nhượng hoặc bán NFT của mình cho người khác. Điều này mở ra một ứng dụng khác. Giả sử bạn mua một NFT để đăng ký một dịch vụ phim trực tuyến như Netflix, Disney+, FPT play, Galaxy, …. Nếu bạn sở hữu NFT, bạn có thể xem phim từ dịch vụ đó. Nếu không, bạn không thể xem. Nhưng hãy nhớ rằng bạn có thể chuyển nhượng hoặc bán NFT đó cho người khác. Điều đó có nghĩa là bạn có thể chuyển nhượng hoặc bán gói đăng ký dịch vụ phát trực tuyến phim của mình. NFT cũng có thể đại diện cho các tài sản được sử dụng trong trò chơi điện tử, chẳng hạn như nhân vật, đất đai ảo, vũ khí hoặc quần áo. Thông thường, người chơi có thể mua các vật phẩm nâng cấp trong trò chơi điện tử thông qua các giao dịch mua hàng trong ứng dụng. Tuy nhiên, việc này cũng có thể được thực hiện bằng NFT. Bên cạnh

việc đại diện cho các vật phẩm ảo, NFT có thể đại diện cho các tài sản trong thế giới thực, chẳng hạn như bất động sản. Việc sở hữu một NFT có thể cấp cho bạn quyền sử dụng nhà cửa hoặc các bất động sản. NFT cũng có thể được sử dụng trong giáo dục. Ví dụ, các chứng chỉ, bằng cấp của bạn có thể được đại diện bởi NFT. Để chứng minh bạn đã theo học tại một trường đại học nào đó, bạn có thể cung cấp một NFT do chính trường đại học đó cấp cho bạn như một chứng chỉ minh chứng bạn đã hoàn thành khoá học ở đó.

B. Chuẩn ERC-721

Giống như token trên Ethereum có chuẩn ERC-20, NFT trên Ethereum có chuẩn ERC-721. Để một hợp đồng thông minh tuân thủ ERC-721, nó cần đáp ứng cả tiêu chuẩn ERC-721 và tiêu chuẩn ERC-165. Tiêu chuẩn này rất hữu ích nếu bạn muốn hợp đồng thông minh NFT của mình có khả năng tương tác với các hợp đồng thông minh khác, ví dụ: nếu bạn muốn người khác có thể giao dịch token NFT từ hợp đồng thông minh NFT của bạn.

```
event Transfer:
    _from: indexed(address)
    _to: indexed(address)
    _tokenId: indexed(uint256)

event Approval:
    _owner: indexed(address)
    _approved: indexed(address)
    _tokenId: indexed(uint256)

event ApprovalForAll:
    _owner: indexed(address)
    _operator: indexed(address)
    _approved: bool
```

a. Các sự kiện

```
@view
@external
def balanceOf(_owner: address) -> uint256:
    return 0
@view
@external
def ownerOf(_tokenId: uint256) -> address:
    return empty(address)
@external
def safeTransferFrom(_from: address, _to: address, _tokenId: uint256, _data: Bytes[1024]) -> bool:
    return False
@external
def safeTransferFromWithoutData(_from: address, _to: address, _tokenId: uint256) -> bool:
    return False
@external
def transferFrom(_from: address, _to: address, _tokenId: uint256) -> bool:
    return False
@external
def approve(_approved: address, _tokenId: uint256) -> bool:
    return False
@external
def setApprovalForAll(_operator: address, _approved: bool) -> bool:
    return False
@view
@external
def getApproved(_tokenId: uint256) -> address:
    return empty(address)
@view
@external
def isApprovedForAll(_owner: address, _operator: address) -> bool:
    return False
```

b. Các hàm trong smart contract theo chuẩn ERC-721

Hình 4. Interface cho xây dựng smart contract theo chuẩn ERC-721.

Để đáp ứng tiêu chuẩn ERC-721, bạn cần triển khai các sự kiện (events) và các hàm như hình 4. Ý nghĩa của các hàm và sự kiện:

- Sự kiện đầu tiên cần triển khai là sự kiện Transfer. Sự kiện này được kích hoạt khi một người dùng chuyển một NFT cho một người dùng khác.
- Sự kiện thứ hai cần triển khai là sự kiện Approval. Sự kiện này được kích hoạt khi một chủ sở hữu cho phép một người dùng khác chuyển một NFT thay mặt cho chủ sở hữu. Sự chấp thuận này chỉ dành cho một NFT duy nhất.
- Sự kiện cuối cùng cần triển khai là sự kiện ApprovalForAll. Sự kiện này được kích hoạt khi một chủ sở hữu cho phép một người dùng khác chuyển TẤT CẢ

token NFT thay cho chủ sở hữu. Sự chấp thuận này áp dụng cho toàn bộ token NFT mà chủ sở hữu đang nắm giữ.

- Hàm đầu tiên cần hiện thực là hàm balanceOf. Mục đích của hàm này là cho biết một người dùng có bao nhiêu token NFT trong hợp đồng thông minh này.
- Hàm thứ hai cần hiện thực là hàm ownerOf. Hàm này dùng để cho biết ai là chủ sở hữu của một NFT cụ thể.
- Hàm thứ ba và thứ tư cần hiện thực là các hàm safeTransferFrom. Chúng có chung tên hàm nhưng các đối số (arguments) thì khác nhau. Hàm safeTransferFrom đầu tiên có đối số data với kiểu dữ liệu bytes. Hàm này chuyển một NFT từ người gửi đến nơi nhận. Nhưng đối số data trong hàm này dùng để làm gì? Nếu các em chuyển đến một địa chỉ bình thường (ví dụ: ví cá nhân), đối số data sẽ không được sử dụng. Đối số data chỉ được sử dụng khi đích đến là một hợp đồng thông minh. Khi đích đến là hợp đồng thông minh, hàm safeTransferFrom sẽ gọi hàm onERC721Received trên hợp đồng thông minh đó và truyền đối số data vào.
- Hàm thứ năm cần hiện thực là hàm transferFrom. Mục đích của hàm này giống với mục đích của hàm safeTransferFrom. Điểm khác biệt duy nhất là hàm transferFrom không quan tâm liệu đích đến có phải là một hợp đồng thông minh hay không.
- Hàm thứ sáu cần hiện thực là phương thức approve. Đây là hàm để phê duyệt một NFT cụ thể, cho phép một người dùng có thể chuyển nó thay mặt cho người dùng khác.
- Hàm thứ bảy cần hiện thực là hàm setApprovalForAll. Thay vì chỉ phê duyệt một NFT, hàm này sẽ phê duyệt TẤT CẢ token NFT, cho phép một người dùng có thể chuyển toàn bộ chúng thay mặt cho người dùng khác.
- Hàm thứ tám cần hiện thực là hàm getApproved. Đây là hàm để kiểm tra xem một người dùng có thể chuyển một NFT cụ thể thay mặt cho người dùng khác hay không.
- Hàm thứ chín cần hiện thực là hàm isApprovedForAll. Đây là hàm để kiểm tra xem một người dùng có thể chuyển TẤT CẢ token NFT thay mặt cho người dùng khác hay không.

Để đáp ứng chuẩn ERC-165, các em chỉ cần hiện thực hàm supportsInterface như hình 5. Mục đích của hàm supportsInterface là để xác định xem một hợp đồng thông minh có hỗ trợ một giao diện (interface) cụ thể hay không. Ví dụ, nếu các em muốn biết một hợp đồng thông minh có hỗ trợ giao diện ERC-721 hay không, hay nói cách khác, liệu hợp đồng thông minh đó có phải là một hợp đồng NFT hay không.

```
@view
@external
def supportsInterface(interfaceID: bytes4) -> bool:
    return False
```

Hình 5. Hàm cần hiện thực theo chuẩn ERC-165.

Cũng giống như trong trường hợp của chuẩn ERC-20, chuẩn ERC-721 cũng có các hàm tuỳ chọn hiện thực như hình số 6.

```
@view
@external
def name() -> String[64]:
    return ""
@view
@external
def symbol() -> String[32]:
    return ""
@view
@external
def tokenURI(_tokenId: uint256) -> String[128]:
    return ""
```

Hình 6. Các hàm tuỳ chọn hiện thực theo chuẩn ERC-721

C. Tạo smart contract NFT

Bây giờ các em sẽ bắt đầu tạo smart contract NFT. Các em tạo 1 file tên HelloNFT.vy

Bước 1: Thực hiện các khai báo thư viện như hình 7.

```
# @version ^0.4.3
from ethereum.ercs import IERC165
from ethereum.ercs import IERC721

implements: IERC721
implements: IERC165
```

Hình 7. Khai báo thư viện

Bước 2: Khai báo các sự kiện theo chuẩn ERC-721 như hình 8. Sự kiện Transfer được sử dụng khi ai đó chuyển một NFT cho người khác. Sự kiện Approval được sử dụng khi chủ sở hữu của một NFT ủy quyền cho một người khác, để người đó có thể chuyển NFT thay mặt cho chủ sở hữu. Sự kiện ApprovalForAll được sử dụng khi chủ sở hữu của các token NFT ủy quyền cho một người khác, để người đó có thể chuyển TẤT CẢ các token NFT thay mặt cho chủ sở hữu.

```
event Transfer:
    _from: indexed(address)
    _to: indexed(address)
    _tokenId: indexed(uint256)

event Approval:
    _owner: indexed(address)
    _approved: indexed(address)
    _tokenId: indexed(uint256)

event ApprovalForAll:
    _owner: indexed(address)
    _operator: indexed(address)
    _approved: bool
```

Hình 8. Các sự kiện theo chuẩn ERC-721.

Bước 3: Viết 1 interface ERC721Receiver như hình 9, interface này cho phép thực thi phương thức onERC721Received của một hợp đồng thông minh nếu ta chuyển một NFT đến một hợp đồng thông minh.

```
interface ERC721Receiver:
    def onERC721Received(
            _operator: address,
            _from: address,
            _tokenId: uint256,
            _data: Bytes[1024]
        ) -> bytes4: nonpayable
```

Hình 9. Interface ERC721Receiver

Bước 4: Thêm tên và ký hiệu giúp người dùng phân biệt NFT của chúng ta với các NFT khác như hình 10.

```
name: public(String[32])

symbol: public(String[32])
```

Hình 10. Tên và ký hiệu cho NFT

Bước 5: Định nghĩa các biến HashMap, có nhiệm vụ theo dõi thông tin về NFT, chủ sở hữu và các ủy quyền. Biến trạng thái idToOwner liên kết mã định danh token NFT (token ID) với địa chỉ của chủ sở hữu. Biến trạng thái idToApprovals liên kết mã định danh token NFT với địa chỉ tài khoản được phép chuyển NFT này thay mặt cho chủ sở hữu. Biến trạng thái ownerToNFTokenCount đếm số lượng token NFT mà một chủ sở hữu đang sở hữu. Biến trạng thái ownerToOperators liên kết chủ sở hữu với các địa chỉ tài khoản được ủy quyền chuyển tất cả token NFT của chủ sở hữu đó thay mặt họ. Biến trạng thái minter là địa chỉ tài khoản có quyền đúc (mint) hoặc tạo ra các token NFT. Biến trạng thái baseURL là địa chỉ URL, tạo thành một phần của đường dẫn đến tài sản NFT. Tất cả các công việc này được thể hiện trong hình 11.

```
idToOwner: HashMap[uint256, address]

idToApprovals: HashMap[uint256, address]

ownerToNFTokenCount: HashMap[address, uint256]

ownerToOperators: HashMap[address, HashMap[address, bool]]

minter: address

baseURL: String[53]
```

Hình 11. Các biến trạng thái của smart contract NFT.

Bước 6: Tiếp theo khai báo các giá trị hằng, để thông báo smart contract hỗ trợ chuẩn ERC-165 và ERC-721 như hình 12. Các em tìm hiểu và giải thích các giá trị hằng trong interface này và viết vào báo cáo.

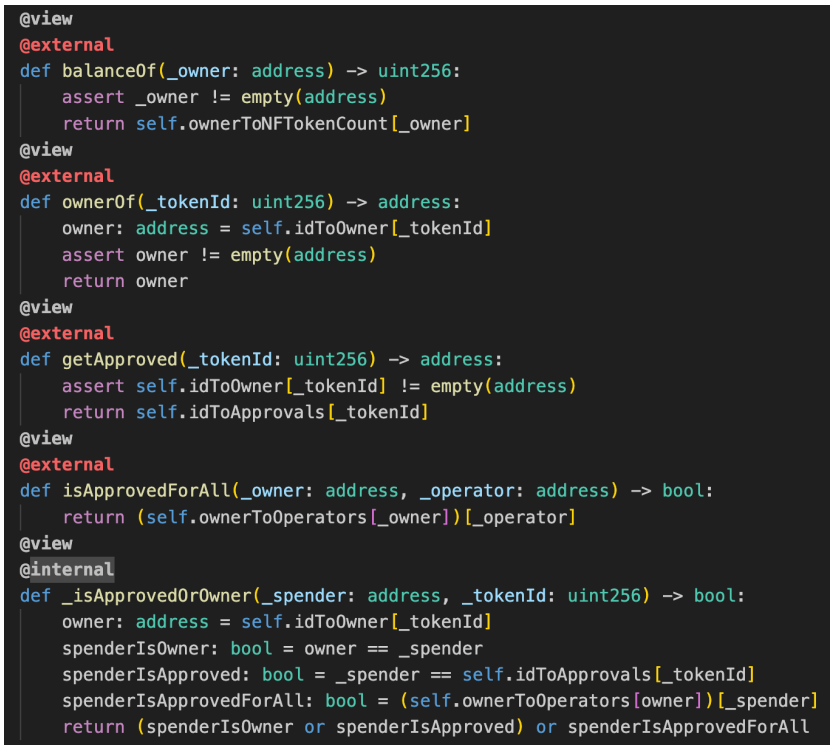
```
SUPPORTED_INTERFACES: constant(bytes4[2]) = [
    # ERC165 interface ID of ERC165
    0x01ffc9a7,
    # ERC165 interface ID of ERC721
    0x80ac58cd,
]
```

Hình 12. Khai báo interface thông báo hỗ trợ chuẩn ERC-721 & ERC-165

Bước 7: ta hiện thực hàm khởi tạo cho các biến minter, biến baseURL, biến name và biến symbol như hình 13. Tên token NFT trong bài thực hành là "Hello NFT". Tài sản NFT lưu trữ bên trong đường dẫn: https://ethereum.org/developers/tutorials/(đây là địa chỉ máy chủ cục bộ). Đúng vậy, tài sản NFT không cần phải lưu trữ trên IPFS. Chúng có thể lưu trữ trên các máy chủ web thông thường. Đồng thời hợp đồng thông minh NFT trong bài thực hành chỉ hỗ trợ các giá trị được xác định trong SUPPORTED_INTERFACES, đó là interface theo chuẩn ERC-721 và interface theo chuẩn ERC-165. Người dùng có thể gọi phương thức supportsInterface để kiểm tra xem hợp đồng thông minh có tuân thủ chuẩn ERC-721 hay không.

```
@deploy
def __init__():
    self.minter = msg.sender
    self.baseURL = "https://ethereum.org/developers/tutorials/"
    self.name = "Hello NFT"
    self.symbol = "HEL"
@view
@external
def supportsInterface(interface_id: bytes4) -> bool:
    return interface_id in SUPPORTED_INTERFACES
```

Hình 13. Hàm khởi tạo và hàm supportsInterface



```
@view
@external
def balanceOf(_owner: address) -> uint256:
    assert _owner != empty(address)
    return self.ownerToNFTokenCount[_owner]
@view
@external
def ownerOf(_tokenId: uint256) -> address:
    owner: address = self.idToOwner[_tokenId]
    assert owner != empty(address)
    return owner
@view
@external
def getApproved(_tokenId: uint256) -> address:
    assert self.idToOwner[_tokenId] != empty(address)
    return self.idToApprovals[_tokenId]
@view
@external
def isApprovedForAll(_owner: address, _operator: address) -> bool:
    return (self.ownerToOperators[_owner])[_operator]
@view
@internal
def _isApprovedOrOwner(_spender: address, _tokenId: uint256) -> bool:
    owner: address = self.idToOwner[_tokenId]
    spenderIsOwner: bool = owner == _spender
    spenderIsApproved: bool = _spender == self.idToApprovals[_tokenId]
    spenderIsApprovedForAll: bool = (self.ownerToOperators[owner])[_spender]
    return (spenderIsOwner or spenderIsApproved) or spenderIsApprovedForAll
```

Hình 14. Các hàm lấy thông tin token NFT từ chủ sở hữu.

Bước 8: Mục đích của hàm balanceOf là kiểm tra xem một chủ sở hữu có bao nhiêu token NFT. Hàm này lấy thông tin từ biến trạng thái ownerToNFTokenCount. Trước hết cần đảm bảo địa chỉ không phải là địa chỉ rỗng vì địa chỉ rỗng không thể sở hữu token NFT. Khi thực hiện chuyển một NFT đến địa chỉ rỗng, điều đó có nghĩa là đang hủy NFT đó. Chức năng của hàm ownerOf là tìm chủ sở hữu từ một ID token NFT. Ta có thể lấy thông tin này từ biến trạng thái idToOwner. Ở đây, ta cũng cần đảm bảo địa chỉ không phải là địa chỉ rỗng. Mục tiêu của hàm getApproved là lấy địa chỉ được ủy quyền của một NFT. Địa chỉ được ủy quyền này có thể chuyển NFT thay mặt cho chủ sở hữu. Ta lấy nó từ biến idToApprovals. Trước tiên hãy đảm bảo rằng ID token NFT có chủ sở hữu. Chức năng của hàm isApprovedForAll là kiểm tra xem một địa chỉ có được ủy quyền chuyển tất cả token NFT từ một chủ sở hữu thay mặt cho chủ sở hữu hay không. Ta lấy thông tin này từ biến ownerToOperators. Các hàm này được thực hiện chi tiết trong hình 14.

```
@internal
def _addTokenTo(_to: address, _tokenId: uint256):
    assert self.idToOwner[_tokenId] == empty(address)
    self.idToOwner[_tokenId] = _to
    self.ownerToNFTokenCount[_to] += 1

@internal
def _removeTokenFrom(_from: address, _tokenId: uint256):
    assert self.idToOwner[_tokenId] == _from
    self.idToOwner[_tokenId] = empty(address)
    self.ownerToNFTokenCount[_from] -= 1

@internal
def _clearApproval(_owner: address, _tokenId: uint256):
    assert self.idToOwner[_tokenId] == _owner
    if self.idToApprovals[_tokenId] != empty(address):
        self.idToApprovals[_tokenId] = empty(address)
```

Hình 15. Hiện thực các hàm thêm và bớt Token từ 1 tài khoản.

Bước 9: hiện thực các hàm thay đổi, biến động sở hữu số lượng token NFT của chủ sở hữu như hình 15. Hàm _addTokenTo thêm tokem NFT cho chủ sở hữu dựa vào biến idToOwner. Sau đó, thực hiện tăng số lượng NFT của chủ sở hữu trong biến ownerToNFTokenCount. Hàm _removeTokenFrom thực hiện đặt giá trị địa chỉ rỗng cho chủ sở hữu của ID token NFT trong biến idToOwner. Sau đó, thực hiện giảm số lượng token NFT của chủ sở hữu trong biến ownerToNFTokenCount. Đối với hàm _clearApproval, để thực hiện xoá các token của chủ sở hữu, ta thực hiện gán giá trị địa chỉ rỗng cho ID token NFT trong biến idToApprovals.

Bước 10: Trong hình 16 là đoạn chương trình chuyển token từ người sở hữu này sang người sở hữu khác, để đảm bảo người gửi có thể chuyển tokent NFT, đầu tiên thực hiện gọi hàm _isApprovedOrOwner để kiểm tra. Sau đó, xóa địa chỉ đang sở hữu các token bằng hàm _clearApproval. Tiếp theo, xóa token NFT bên người gửi bằng hàm _removeTokenFrom và thêm token NFT bên người nhận bằng hàm _addTokenTo. Cuối cùng, ghi sự kiện xảy ra bằng hàm Transfer. Trong trường hợp muốn chuyển token NFT đến địa chỉ của người dùng thông thường, không phải hợp đồng thông minh, ta thực hiện gọi hàm transferFrom. Sau đó, tạo một hàm khác có thể chuyển token NFT đến hợp đồng thông minh. Bên trong hàm, thực hiện gọi hàm _transferFrom như bình thường. Sau đó, tiến hành kiểm tra xem địa chỉ đích có phải là hợp đồng thông minh hay không. Nếu có, gọi phương thức onERC721Received trên hợp đồng thông minh đó và đảm bảo hợp đồng thông minh có phương thức này.

```
@internal
def _transferFrom(_from: address, _to: address, _tokenId: uint256, _sender: address):
    assert self._isApprovedOrOwner(_sender, _tokenId)
    assert _to != empty(address)
    self._clearApproval(_from, _tokenId)
    self._removeTokenFrom(_from, _tokenId)
    self._addTokenTo(_to, _tokenId)
    log Transfer(_from, _to, _tokenId)

@external
@payable
def transferFrom(_from: address, _to: address, _tokenId: uint256):
    self._transferFrom(_from, _to, _tokenId, msg.sender)

@external
@payable
def safeTransferFrom(
        _from: address,
        _to: address,
        _tokenId: uint256,
        _data: Bytes[1024]=b""
    ):
    self._transferFrom(_from, _to, _tokenId, msg.sender)
    if _to.is_contract:
        returnValue: bytes4 = extcall ERC721Receiver(_to).onERC721Received(msg.sender, _from, _tokenId, _data)
        assert returnValue == method_id("onERC721Received(address,address,uint256,bytes)", output_type=bytes4)
```

Hình 16. Thực hiện chuyển token NFT từ nơi gửi đến nơi nhận.

Bước 11: thực hiện tạo một hàm external để phê duyệt một NFT, cho phép một người dùng có thể chuyển nó thay mặt cho chủ sở hữu như trong hình 17. Bên trong hàm này, ta cần đảm bảo chủ sở hữu của NFT có địa chỉ hợp lệ. Sau đó, cần đảm bảo rằng địa chỉ người gửi là operator hoặc đã được chấp thuận. Sau đó, thực hiện thay đổi giá trị của biến idToApprovals cho ID token NFT. Cuối cùng, ghi sự kiện trong Approval. Bên canh đó để thực hiện phê duyệt một địa chỉ có thể chuyển tất cả NFT thay mặt cho chủ sở hữu, ta xây dựng hàm setApprovalForAll. Bên trong hàm, ta thực hiện thay đổi giá trị của biến ownerToOperators cho chủ sở hữu và operator. Nếu giá trị là true, thì operator có thể chuyển tất cả token NFT thay mặt cho chủ sở hữu. Để hủy bỏ quyền này, ta gọi hàm này với giá trị false. Cuối cùng, thực hiện ghi sự kiện trong ApprovalForAll.

```
@external
@payable
def approve(_approved: address, _tokenId: uint256):
    owner: address = self.idToOwner[_tokenId]
    assert owner != empty(address)
    assert _approved != owner
    senderIsOwner: bool = self.idToOwner[_tokenId] == msg.sender
    senderIsApprovedForAll: bool = (self.ownerToOperators[owner])[msg.sender]
    assert (senderIsOwner or senderIsApprovedForAll)
    self.idToApprovals[_tokenId] = _approved
    log Approval(owner, _approved, _tokenId)

@external
def setApprovalForAll(_operator: address, _approved: bool):
    assert _operator != msg.sender
    self.ownerToOperators[msg.sender][_operator] = _approved
    log ApprovalForAll(msg.sender, _operator, _approved)
```

Hình 17. Hàm chấp thuận NFT.

Bước 12: tạo một hàm để mint (đúc) một NFT. Để tạo NFT, ta đảm bảo người gửi là minter. Sau đó, ta phải đảm bảo tiếp theo nữa là địa chỉ đích sẽ trở thành chủ sở hữu của NFT không phải là địa chỉ rỗng. Sau đó, ta gọi hàm _addTokenTo. Cuối cùng, ta thực hiện ghi sự kiện trong Transfer. Đồng thời, cũng phải tạo một hàm để hủy (burn) một NFT. Bên trong hàm huỷ này, ta đảm bảo người gửi đã được phê duyệt hoặc là chủ sở hữu của NFT. Sau đó, thực hiện xóa các phê duyệt bằng cách gọi hàm _clearApproval. Tiếp theo, thực hiện xóa NFT bằng cách gọi hàm _removeTokenFrom. Cuối cùng, thực hiện ghi lại sự kiện dựa trên hàm Transfer. Nói cách khác, việc hủy một NFT có nghĩa là chuyển NFT đến một địa chỉ rỗng như hình 18.

```
@external
def mint(_to: address, _tokenId: uint256) -> bool:
    assert msg.sender == self.minter
    assert _to != empty(address)
    self._addTokenTo(_to, _tokenId)
    log Transfer(empty(address), _to, _tokenId)
    return True

@external
def burn(_tokenId: uint256):
    assert self._isApprovedOrOwner(msg.sender, _tokenId)
    owner: address = self.idToOwner[_tokenId]
    assert owner != empty(address)
    self._clearApproval(owner, _tokenId)
    self._removeTokenFrom(owner, _tokenId)
    log Transfer(owner, empty(address), _tokenId)

@view
@external
def tokenURI(tokenId: uint256) -> String[132]:
    return concat(self.baseURL, uint2str(tokenId))
```

Hình 18. Hiện thực hàm mint & burn.

Lưu ý: có một giá trị của biến baseURL, biến này là một địa chỉ url https://forms.gle/5v8Qtn2FTEykBYuR6/. Vì vậy, đối với token NFT có ID là 1, chẳng hạn, hàm tokenURI sẽ trả về https://forms.gle/5v8Qtn2FTEykBYuR6/1. Đây là URL của tài sản NFT cho token NFT có ID là 1. Tất nhiên, khi tạo như thế này, ta có trách nhiệm lưu trữ và bảo vệ tài sản NFT tại URL đó.

Bước 13: thực hiện kiểm thử smart contract vừa được tạo ra theo chuẩn ERC-721, file conftest.py trong thư mục test thêm 2 hàm như hình 20. Sau đó các em tạo file test_HelloNFT.py, ta thực hiện kiểm thử các hàm: init, balanceOf, ownerOf, transfer như hình 19. Các em tự thực hiện kiểm thử các hàm: approve, setApprovalForAll sau đó đổi tên file thành conftest_MSSV.py để nộp. Cuối cùng các em sử dụng câu lệnh để thực hiện chạy file kiểm thử này: py.test tests/test_HelloNFT.py

```python
from ape.exceptions import ContractLogicError
import pytest

def test_init(erc721_contract, deployer):
    assert "Hello NFT" == erc721_contract.name()
    assert "HEL" == erc721_contract.symbol()

def test_balanceOf(erc721_contract, deployer, mint_nfts):
    assert 5 == erc721_contract.balanceOf(deployer)

def test_ownerOf(erc721_contract, deployer, mint_nfts):
    for i in range(5):
        assert deployer == erc721_contract.ownerOf(i)

def test_transfer(erc721_contract, deployer, mint_nfts, accounts):
    user = accounts[1]
    nftId = 2
    assert 5 == erc721_contract.balanceOf(deployer)
    assert 0 == erc721_contract.balanceOf(user)
    assert deployer == erc721_contract.ownerOf(nftId)
    erc721_contract.transferFrom(deployer, user, nftId, sender=deployer)
    assert 4 == erc721_contract.balanceOf(deployer)
    assert 1 == erc721_contract.balanceOf(user)
    assert user == erc721_contract.ownerOf(nftId)
```

Hình 19. Thực hiện kiểm thử hàm init, balanceOf, ownerOf, transfer.

```python
import pytest

@pytest.fixture
def deployer(accounts):
    return accounts[0]
@pytest.fixture
def contract(deployer, project):
    return deployer.deploy(project.VerySimpleNFT)

@pytest.fixture
def erc721_contract(deployer, project):
    return deployer.deploy(project.HelloNFT)

@pytest.fixture
def mint_nfts(erc721_contract, deployer, num_nfts=5):
    for i in range(num_nfts):
        erc721_contract.mint(deployer, i, sender=deployer)
    return num_nfts
```

Hình 20. Thêm hàm erc721_contract để thực hiện test HelloNFT.

4. Bài tập về nhà

Một công ty game muốn xây dựng một hệ thống "Bộ Sưu Tập Nhân Vật Số" (Digital Character Collection – DCC) dành cho sinh viên, với mục tiêu tạo ra một bộ NFT tượng trưng cho những nhân vật (avatars) mà người chơi có thể sở hữu, giao dịch, hoặc sử dụng để tham gia các hoạt động trong hệ thống game trực tuyến trên các event của công ty. Mỗi NFT đại diện cho một nhân vật duy nhất, ví dụ:

- "Cyber Warrior"

- “Data Wizard”
- “AI Explorer”
- “Blockchain Guardian”

Các nhân vật có mô tả riêng, chỉ số đặc biệt, và hình ảnh minh họa được lưu trữ trên một máy chủ web của công ty dưới dạng file PNG.

Công ty muốn xây dựng smart contract NFT để quản lý các nhân vật này. Contract phải đáp ứng chuẩn ERC-721 để có thể mở rộng trong tương lai, thực hiện mua bán.

Công ty yêu cầu:

- Contract phải hỗ trợ đúc (mint) nhân vật mới, chuyển nhượng, hủy (burn), và cung cấp metadata cho mỗi NFT.
- Mỗi NFT phải có thông tin nhân vật rõ ràng, không chỉ đơn thuần là token ID.
- Hệ thống trong tương lai sẽ sử dụng metadata để hiển thị avatar trong hồ sơ người chơi.

A. Nhiệm vụ bài tập

Hệ thống đảm bảo:

- Mỗi NFT là duy nhất, gắn với hình ảnh trên máy chủ trường.
- Mỗi nhân vật có thuộc tính tùy chỉnh: tên và mô tả.
- Metadata phải được trả về đúng chuẩn JSON để các ứng dụng khác của trường có thể đọc.
- Có đầy đủ chức năng chuyển nhượng và ủy quyền theo chuẩn ERC-721.

B. Mô tả chi tiết yêu cầu

i. Mỗi nhân vật phải chứa thông tin riêng

Khi tạo nhân vật (mint), sinh viên phải cung cấp:

- name: tên nhân vật (ví dụ "Cyber Warrior")
- description: mô tả nhân vật (“Một chiến binh số có khả năng phá mã CRY-128”)
- token_id: định danh nhân vật

Các thông tin này phải lưu trong HashMap.

ii. Hình ảnh nhân vật và tokenURI

Hệ thống sử dụng URL để lưu trữ hình ảnh (các em có thể sử dụng 1 server free cho phép lưu hình ảnh)

Hàm tokenURI() của bạn phải trả về JSON metadata, ví dụ:

```
{
  "name": "Cyber Warrior",
  "description": "Một chiến binh số thiện chiến.",
```

"image": "https://school.edu.vn/nft-assets/1.png"
}

iii. Chức năng mint

Chỉ minter (người triển khai hợp đồng) được quyền tạo nhân vật mới. Mint NFT phải:

- tạo owner
- lưu name
- lưu description
- kích hoạt event Minted

iv. Chức năng burn

Cho phép chủ sở hữu hủy NFT nhân vật (ví dụ: khi muốn xóa avatar cũ). Khi hủy, hệ thống phải:

- đưa token vào địa chỉ 0x0
- xóa name & description
- phát event Transfer (chuẩn ERC-721)

v. Chức năng approve & chuyển nhượng

Cho phép:

- chủ nhân ủy quyền
- người được ủy quyền chuyển NFT
- chuyển NFT qua transferFrom

vi. Kiểm thử

Mô phỏng các trường hợp:

- tạo nhân vật mới
- kiểm tra metadata
- chuyển nhượng nhân vật
- ủy quyền và chuyển nhượng
- hủy nhân vật

C. Yêu cầu nộp:

- Smart contract MyCollectibleNFT.vy
- Test file đầy đủ: test_MyCollectibleNFT.py
- Ảnh chụp kết quả chạy thử nghiệm
- Giải thích đoạn code viết trong smart contract và test file.