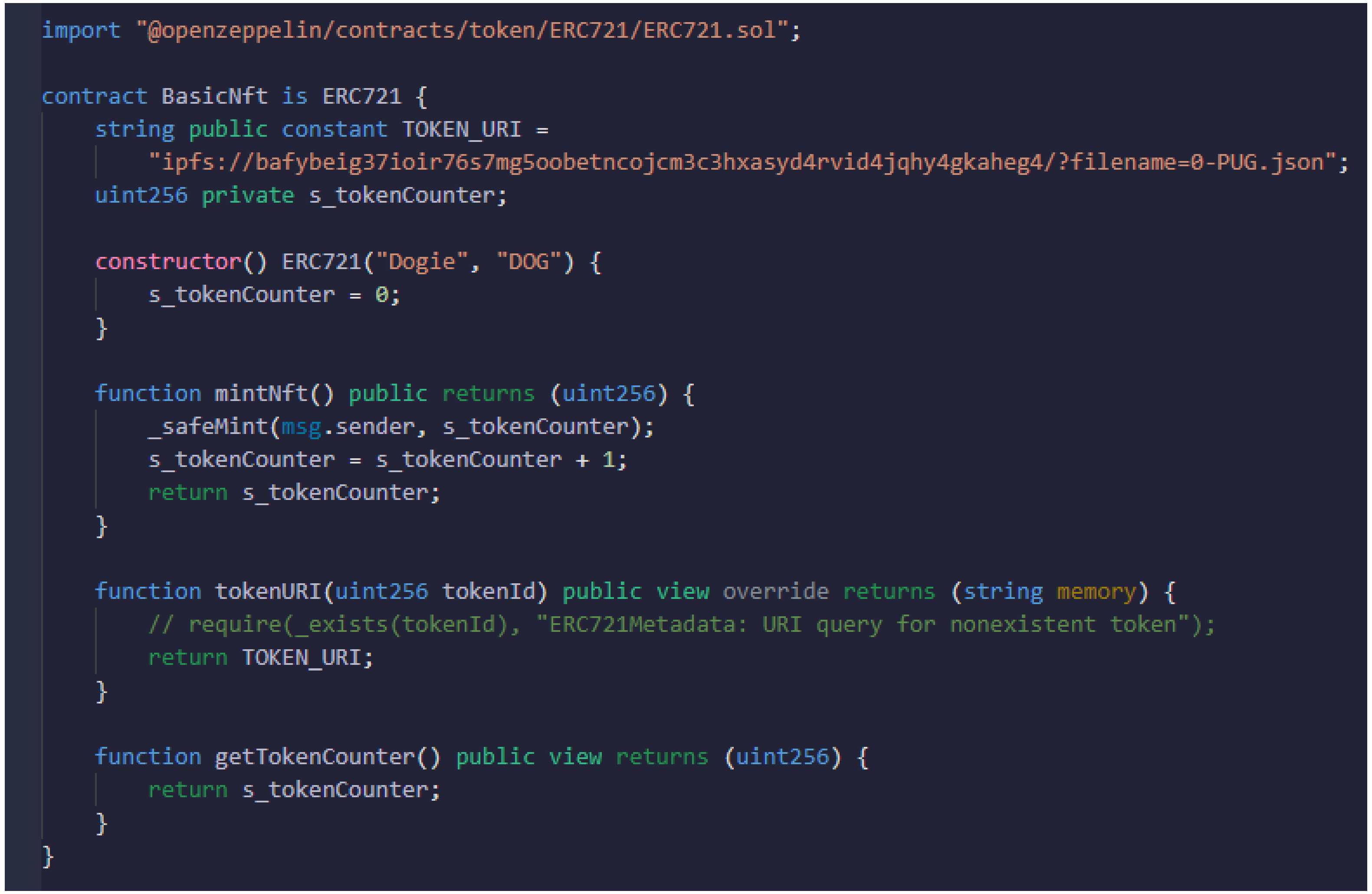
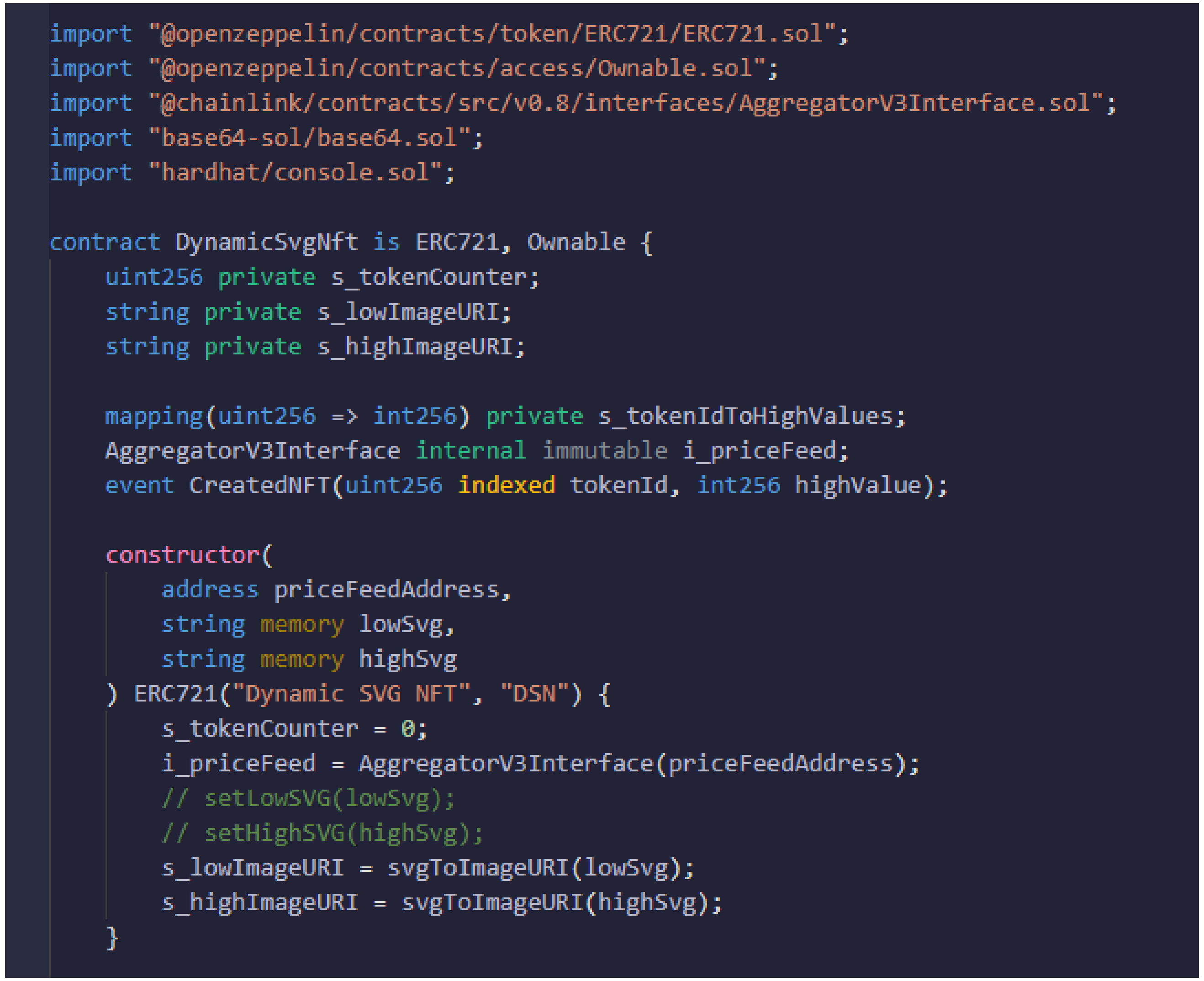
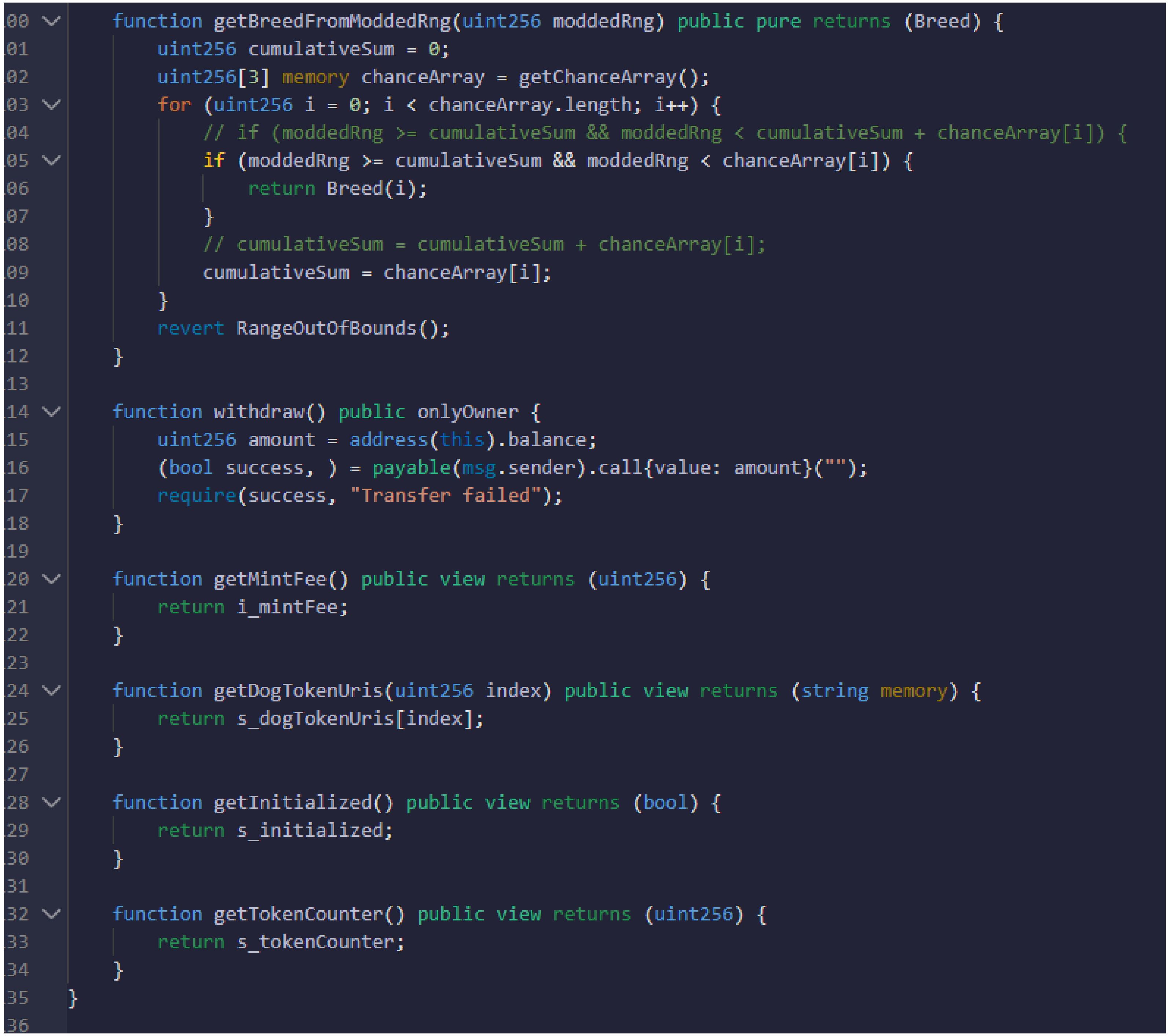
Hardhat NFTs



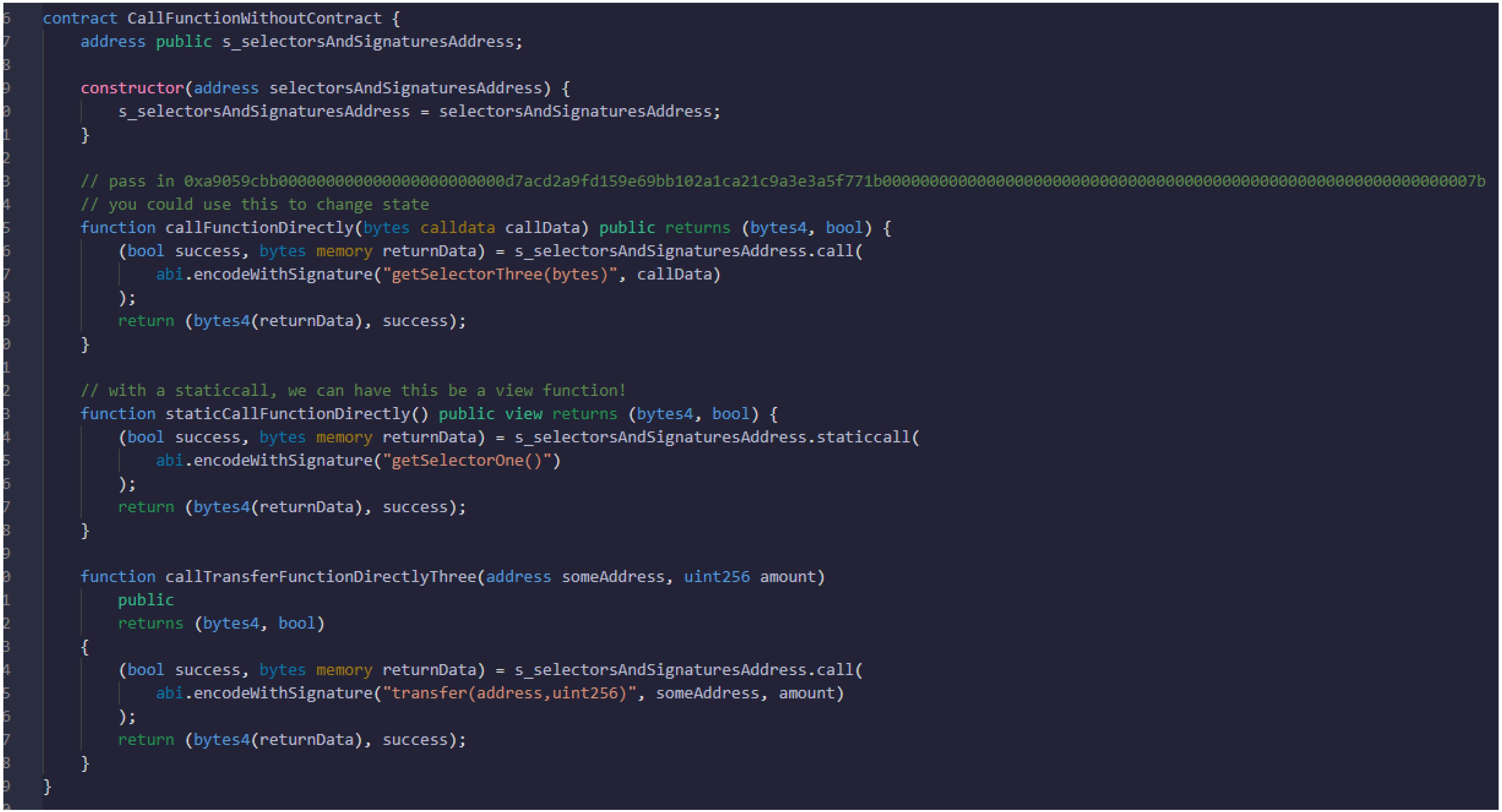
```solidity
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract BasicNft is ERC721 {
    string public constant TOKEN_URI =
        "ipfs://bafybeig37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filename=0-PUG.json";
    uint256 private s_tokenCounter;

    constructor() ERC721("Dogie", "DOG") {
        s_tokenCounter = 0;
    }

    function mintNft() public returns (uint256) {
        _safeMint(msg.sender, s_tokenCounter);
        s_tokenCounter = s_tokenCounter + 1;
        return s_tokenCounter;
    }

    function tokenURI(uint256 tokenId) public view override returns (string memory) {
        // require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
        return TOKEN_URI;
    }

    function getTokenCounter() public view returns (uint256) {
        return s_tokenCounter;
    }
}
```



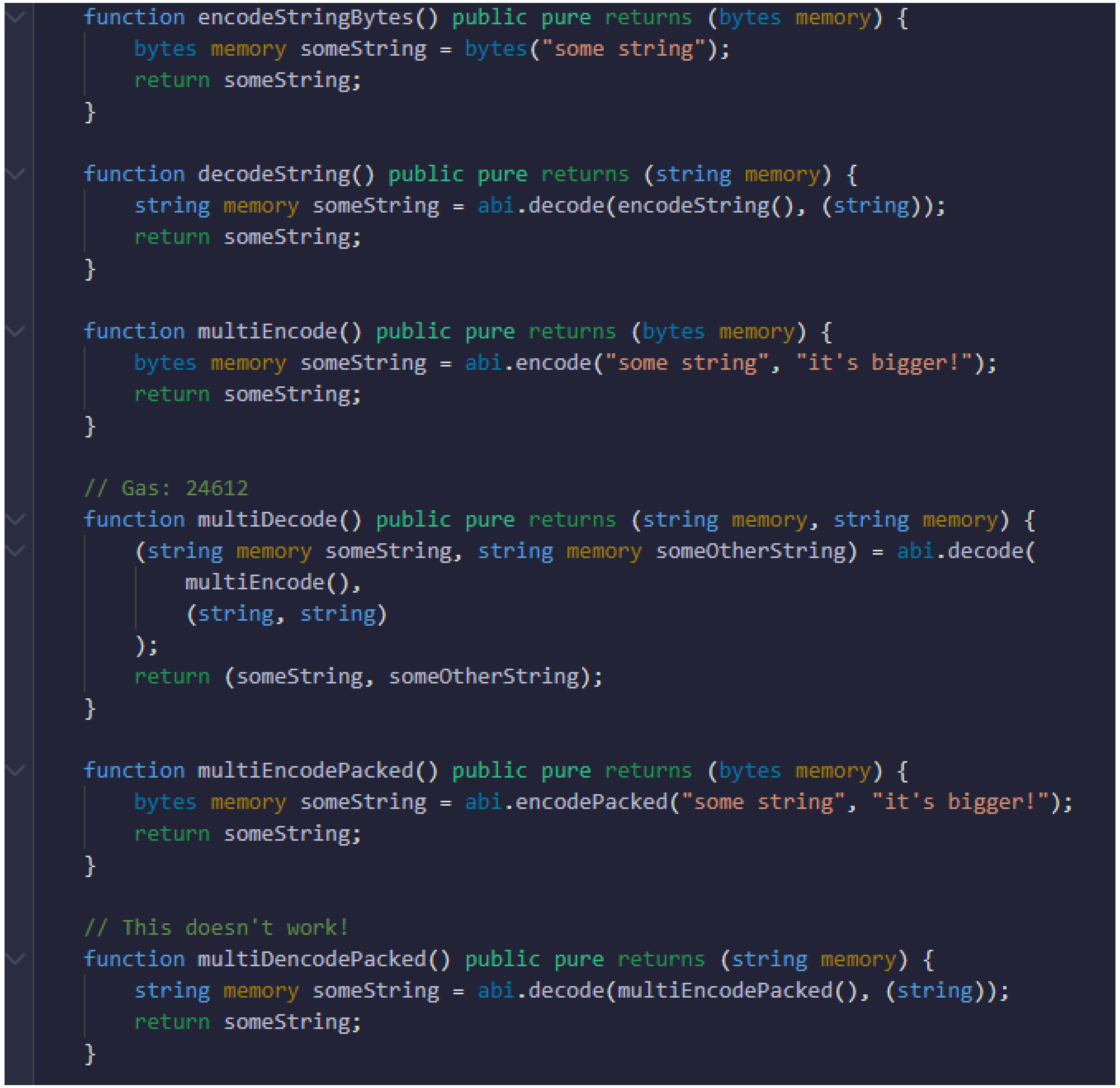
```solidity
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
import "base64-sol/base64.sol";
import "hardhat/console.sol";

contract DynamicSvgNft is ERC721, Ownable {
    uint256 private s_tokenCounter;
    string private s_lowImageURI;
    string private s_highImageURI;

    mapping(uint256 => int256) private s_tokenIdToHighValues;
    AggregatorV3Interface internal immutable i_priceFeed;
    event CreatedNFT(uint256 indexed tokenId, int256 highValue);

    constructor(
        address priceFeedAddress,
        string memory lowSvg,
        string memory highSvg
    ) ERC721("Dynamic SVG NFT", "DSN") {
        s_tokenCounter = 0;
        i_priceFeed = AggregatorV3Interface(priceFeedAddress);
        // setLowSVG(lowSvg);
        // setHighSVG(highSvg);
        s_lowImageURI = svgToImageURI(lowSvg);
        s_highImageURI = svgToImageURI(highSvg);
    }
```

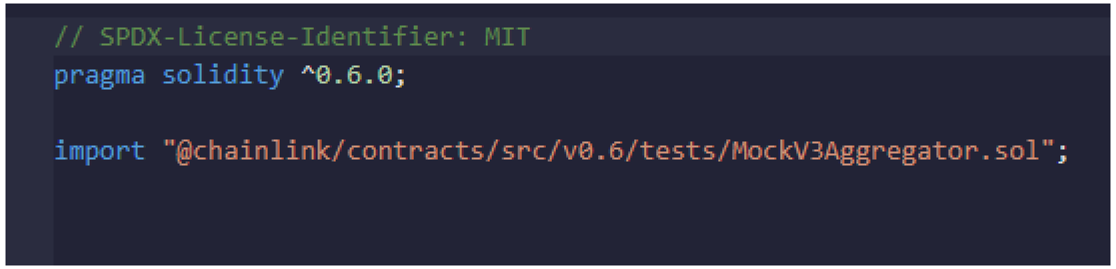


```
    function getBreedFromModdedRng(uint256 moddedRng) public pure returns (Breed) {
        uint256 cumulativeSum = 0;
        uint256[3] memory chanceArray = getChanceArray();
        for (uint256 i = 0; i < chanceArray.length; i++) {
            // if (moddedRng >= cumulativeSum && moddedRng < cumulativeSum + chanceArray[i]) {
            if (moddedRng >= cumulativeSum && moddedRng < chanceArray[i]) {
                return Breed(i);
            }
            // cumulativeSum = cumulativeSum + chanceArray[i];
            cumulativeSum = chanceArray[i];
        }
        revert RangeOutOfBounds();
    }

    function withdraw() public onlyOwner {
        uint256 amount = address(this).balance;
        (bool success, ) = payable(msg.sender).call{value: amount}("");
        require(success, "Transfer failed");
    }

    function getMintFee() public view returns (uint256) {
        return i_mintFee;
    }

    function getDogTokenUris(uint256 index) public view returns (string memory) {
        return s_dogTokenUris[index];
    }

    function getInitialized() public view returns (bool) {
        return s_initialized;
    }

    function getTokenCounter() public view returns (uint256) {
        return s_tokenCounter;
    }
}
```

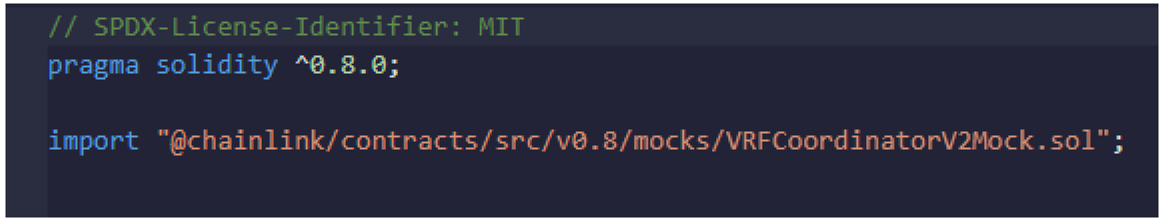


```solidity
contract CallFunctionWithoutContract {
    address public s_selectorsAndSignaturesAddress;

    constructor(address selectorsAndSignaturesAddress) {
        s_selectorsAndSignaturesAddress = selectorsAndSignaturesAddress;
    }

    // pass in 0xa9059cbb000000000000000000000000d7acd2a9fd159e69bb102a1ca21c9a3e3a5f771b000000000000000000000000000000000000000000000000000000000000007b
    // you could use this to change state
    function callFunctionDirectly(bytes calldata callData) public returns (bytes4, bool) {
        (bool success, bytes memory returnData) = s_selectorsAndSignaturesAddress.call(
            abi.encodeWithSignature("getSelectorThree(bytes)", callData)
        );
        return (bytes4(returnData), success);
    }

    // with a staticcall, we can have this be a view function!
    function staticCallFunctionDirectly() public view returns (bytes4, bool) {
        (bool success, bytes memory returnData) = s_selectorsAndSignaturesAddress.staticcall(
            abi.encodeWithSignature("getSelectorOne()")
        );
        return (bytes4(returnData), success);
    }

    function callTransferFunctionDirectlyThree(address someAddress, uint256 amount)
        public
        returns (bytes4, bool)
    {
        (bool success, bytes memory returnData) = s_selectorsAndSignaturesAddress.call(
            abi.encodeWithSignature("transfer(address,uint256)", someAddress, amount)
        );
        return (bytes4(returnData), success);
    }
}
```


```
function encodeStringBytes() public pure returns (bytes memory) {
    bytes memory someString = bytes("some string");
    return someString;
}

function decodeString() public pure returns (string memory) {
    string memory someString = abi.decode(encodeString(), (string));
    return someString;
}

function multiEncode() public pure returns (bytes memory) {
    bytes memory someString = abi.encode("some string", "it's bigger!");
    return someString;
}

// Gas: 24612
function multiDecode() public pure returns (string memory, string memory) {
    (string memory someString, string memory someOtherString) = abi.decode(
        multiEncode(),
        (string, string)
    );
    return (someString, someOtherString);
}

function multiEncodePacked() public pure returns (bytes memory) {
    bytes memory someString = abi.encodePacked("some string", "it's bigger!");
    return someString;
}

// This doesn't work!
function multiDencodePacked() public pure returns (string memory) {
    string memory someString = abi.decode(multiEncodePacked(), (string));
    return someString;
}
```



```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

import "@chainlink/contracts/src/v0.6/tests/MockV3Aggregator.sol";
```



```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@chainlink/contracts/src/v0.8/mocks/VRFCoordinatorV2Mock.sol";
```