# 170.https://stackoverflow.com/questions/72582656/invalid-contract-specified-in-override-list

**T:**Invalid contract specified in override list?

**Q:**I am having an issue trying to deploy a contract i've just finished working on...  
  
Here's the code :`  
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.4;import '@openzeppelin/contracts/access/Ownable.sol';import '@openzeppelin/contracts/utils/Strings.sol';import 'erc721a/contracts/IERC721A.sol';import 'erc721a/contracts/extensions/erc721aQueryable.sol';contract unNamed is ERC721AQueryable, Ownable { using Strings for uint256; uint256 public constant TOTAL\_MAX\_SUPPLY = 10000; uint256 public totalFreeMints = 1000; uint256 public teamAmount = 777; uint256 public maxFreeMintPerWallet = 2; uint256 public maxPublicMintPerWallet = 10; uint256 public publicTokenPrice = .0069 ether; string \_contractURI; bool public saleStarted = false; uint256 public freeMintCount; mapping(address => uint256) public freeMintClaimed; string private \_baseTokenURI; constructor() ERC721A('unNamed', 'unn') {} modifier callerIsUser() { require(tx.origin == msg.sender, 'unNamed: The caller is another contract'); \_; } modifier underMaxSupply(uint256 \_quantity) { require( \_totalMinted() + \_quantity <= TOTAL\_MAX\_SUPPLY - teamAmount, 'unNamed: Purchase exceeds max supply' ); \_; } function mint(uint256 \_quantity) external payable callerIsUser underMaxSupply(\_quantity) { require(balanceOf(msg.sender) < maxPublicMintPerWallet, "unNamed: Mint limit exceeded"); require(saleStarted, 'unNamed: Sale has not begun '); if (\_totalMinted() < (TOTAL\_MAX\_SUPPLY - teamAmount)) { if (freeMintCount >= totalFreeMints) { require(msg.value >= \_quantity \* publicTokenPrice, 'unNamed: Send more ETH!'); \_mint(msg.sender, \_quantity); } else if (freeMintClaimed[msg.sender] < maxFreeMintPerWallet) { uint256 \_mintableFreeQuantity = maxFreeMintPerWallet - freeMintClaimed[msg.sender]; if (\_quantity <= \_mintableFreeQuantity) { freeMintCount += \_quantity; freeMintClaimed[msg.sender] += \_quantity; } else { freeMintCount += \_mintableFreeQuantity; freeMintClaimed[msg.sender] += \_mintableFreeQuantity; require( msg.value >= (\_quantity - \_mintableFreeQuantity) \* publicTokenPrice, 'unNamed: Find more ETH to send' ); } \_mint(msg.sender, \_quantity); } else { require(msg.value >= (\_quantity \* publicTokenPrice), 'unNamed: Find more ETH to send'); \_mint(msg.sender, \_quantity); } } } function \_baseURI() internal view virtual override returns (string memory) { return \_baseTokenURI; } function tokenURI(uint256 tokenId) public view virtual override(ERC721A, IERC721A) returns (string memory) { if (!\_exists(tokenId)) revert URIQueryForNonexistentToken(); string memory baseURI = \_baseURI(); return bytes(baseURI).length != 0 ? string(abi.encodePacked(baseURI, tokenId.toString())) : ''; } function numberMinted(address owner) public view returns (uint256) { return \_numberMinted(owner); } function \_startTokenId() internal view virtual override returns (uint256) { return 1; } function ownerMint(uint256 \_numberToMint) external onlyOwner underMaxSupply(\_numberToMint) { \_mint(msg.sender, \_numberToMint); } function ownerMintToAddress(address \_recipient, uint256 \_numberToMint) external onlyOwner underMaxSupply(\_numberToMint) { \_mint(\_recipient, \_numberToMint); } function setFreeMintCount(uint256 \_count) external onlyOwner { totalFreeMints = \_count; } function setTeamAmount(uint256 \_count) external onlyOwner { teamAmount = \_count; } function setMaxFreeMintPerWallet(uint256 \_count) external onlyOwner { maxFreeMintPerWallet = \_count; } function setMaxPublicMintPerWallet(uint256 \_count) external onlyOwner { maxPublicMintPerWallet = \_count; } function setBaseURI(string calldata baseURI) external onlyOwner { \_baseTokenURI = baseURI; } // Storefront metadata // https://docs.opensea.io/docs/contract-level-metadata function contractURI() public view returns (string memory) { return \_contractURI; } function setContractURI(string memory \_URI) external onlyOwner { \_contractURI = \_URI; } function withdrawFunds() external onlyOwner { (bool success, ) = msg.sender.call{ value: address(this).balance }(''); require(success, 'unNamed: Transfer failed.'); } function withdrawFundsToAddress(address \_address, uint256 amount) external onlyOwner { (bool success, ) = \_address.call{ value: amount }(''); require(success, 'unNamed: Transfer failed.'); } function flipSaleStarted() external onlyOwner { saleStarted = !saleStarted; }}`  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
And Here's the error :  
  
TypeError: Invalid contract specified in override list: "IERC721A".--> contracts/unNamed.sol:75:58: | 75 | function tokenURI(uint256 tokenId) public view virtual override(ERC721A, IERC721A) returns(string memory) { |^^^^^^^^^^^^^^^^^^^^^^^^^^^ Note: This contract: -->erc721a/contracts/IERC721A.sol:10:1: | 10 | interface IERC721A {| ^ (Relevant source part starts here and spans across multiplelines).  
  
Error HH600: Compilation failed  
  
I don't see the error though, it says invalid contract specified but I have imported it as well as made sure it's in the erc721A folder...  
  
Any ideas? Thanks in advance

**C1:**any help would be awesome :D i am stuuck!

1 **Answer**

**A1:**The use of the override keyword is no longer usable with interfaces. Remove IERC721A from override(ERC721A, IERC721A)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
https://github.com/ethereum-optimism/optimism/pull/1491  
  
  
  
Or just remove your upper-version indicator from your solidity version ( ^ ), that way you'll specifically use Solc 0.8.4, and i think that should do the trick as well, I hope :)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]