# 647.https://stackoverflow.com/questions/68639104/not-able-to-access-erc721-openzeppelin-nfts-from-separate-contract-that-inherits

**T:**Not able to access ERC721 OpenZeppelin NFTs from separate contract that inherits NFT contract

**Q:**I'm creating an NFT that inherits the openzeppelin ERC721 NFT smart contract. I have a contract, BookCreation, that inherits ERC721. This smart contract is where I mint the NFTs in the function mintBook():  
  
 function mintBook(uint256 \_bookID, uint256 \_editionID) external onlyBookAuthor(\_bookID) { \_tokenIds = \_tokenIds + 1; books[\_bookID].\_numMinted = books[\_bookID].\_numMinted + 1; books[\_bookID].editions[\_editionID].\_numMinted = books[\_bookID].editions[\_editionID].\_numMinted + 1; emit BookMinted(\_tokenIds, \_editionID, \_bookID, books[\_bookID].authorID); \_safeMint(msg.sender, \_tokenIds); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I then have another smart contract, BookStore, that will be the marketplace where you can buy and sell these NFTs.  
  
I have overwritten the ERC721 function ownerOf(uint256 tokenID) as so in my BookCreation contract.  
  
 function ownerOf(uint256 tokenID) public view virtual override returns (address) { return super.ownerOf(tokenID); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
And then I call this function in BookStore like this (I have also tried super.ownerOf(\_tokenID) and ownerOf(\_tokenID) in place of (BookCreation.ownerOf(\_tokenID)):  
  
 modifier onlyBookOwner(uint256 \_tokenID) { require(BookCreation.ownerOf(\_tokenID) == msg.sender,"This isn't your book!"); \_; }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I am running into a problem where, while I can mint a Book in the BookCreation smart contract and see this NFT reflected on the blockchain by calling ownerOf(tokenId) in BookCreation, when I try to call this function in BookStore on the same tokenID by calling BookCreation.ownerOf(tokenId), it is not able to see the NFT that was created.  
  
I am a little unsure about how to be able to read NFTs created in a separate smart contract, any guidance would be helpful!  
  
Other relevant parts of BookCreation class:  
  
import "../node\_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol";import "../node\_modules/@openzeppelin/contracts/access/Ownable.sol";contract BookCreation is ERC721, Ownable { uint256 private \_tokenIds; /\*\* \* @dev represents a submitted book \* Along with editions associated with it (initially empty) \*/ struct Book { string title; uint256 authorID; uint256 \_bookID; uint256 \_numMinted; uint256 \_numEditions; mapping (uint256 => Edition) editions; } /\*\* @dev Represents specific edition of a specific book \* (Advanced Readers Copy, Initial Publishing, 1yr Special Edition, etc) \*/ struct Edition{ uint256 \_editionID; uint256 \_bookID; uint256 \_numMinted; string editionName; } // BookId mapped to the Book it represents mapping (uint256 => Book) private books; /\*\* \* @dev Constructs ERC721 "Book" token collection with symbol "TLB" \*/ constructor() ERC721("Book", "TLB"){ }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

1 **Answer**

**A1:**If you could provide more of the code that would be helpful. Otherwise, my guess is something with the super keyword.https://ethereum.stackexchange.com/questions/12920/what-does-the-keyword-super-in-solidity-do

**C1:**Thanks! I edited the post with more code to hopefully give a better idea of what I am trying to do.

**C2:**contract BookCreation is ERC721, Ownable Solidity checks inheritance from right-to-left, most-derived-to-most-based. You could try switching the order of these to: Ownable, ERC721