# 13.https://stackoverflow.com/questions/71899314/how-to-fix-undeclared-identifier-even-though-its-present-in-erc721enumerable

**T:**How to fix undeclared identifier even though its present in ERC721Enumerable

**Q:**I am trying to prepare a solidity smart contract and everything appears fine except a single undeclared error on remix compiler. The error code is as follows:  
  
from solidity:DeclarationError: Undeclared identifier. --> contracts/WTest.sol:66:22: |66 | uint256 supply = totalSupply(); | ^^^^^^^^^^^  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
For reference, my code is as follows:  
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.2;import "@openzeppelin/contracts/token/ERC721/ERC721.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";import "@openzeppelin/contracts/access/Ownable.sol";import "@openzeppelin/contracts/utils/math/SafeMath.sol";contract WTest is ERC721, ERC721Burnable, Ownable { using Strings for uint256; using SafeMath for uint256; uint256 public mintPrice; address public blackHoleAddress; ERC721 public crateContract; string public baseURI; string public baseExtension = ".json"; mapping(uint256 => bool) private \_crateProcessList; bool public paused = false; bool public revealed = false; uint256 public maxSupply = 5000; uint256 public maxPrivateSupply = 580; uint256 public maxMintAmount = 20; string public notRevealedUri; event OperationResult(bool result, uint256 itemId); constructor() ERC721("WTest", "WTST") {} function \_baseURI() internal view virtual override returns (string memory) { return baseURI; } function setBASEURI(string memory newuri) public onlyOwner { baseURI = newuri; } function setMintPrice(uint256 \_mintPrice) public onlyOwner returns(bool success) { mintPrice = \_mintPrice; return true; } function getMintPrice() public view returns (uint256) { return mintPrice; } function setBlackHoleAddress(address \_blackHoleAddress) public onlyOwner returns(bool success) { blackHoleAddress = \_blackHoleAddress; return true; } function setcrateContractAddress(ERC721 \_crateContractAddress) public onlyOwner returns (bool success) { crateContract = \_crateContractAddress; return true; }function mint(uint256 \_mintAmount) public payable { uint256 supply = totalSupply(); require(!paused); require(\_mintAmount > 0); require(\_mintAmount <= maxMintAmount); require(supply + \_mintAmount <= maxSupply); if (msg.sender != owner()) { require(msg.value >= mintPrice \* \_mintAmount); } for (uint256 i = 1; i <= \_mintAmount; i++) { \_safeMint(msg.sender, supply + i); } }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
As you can see I have referenced ERC721Enumerable.sol  
  
Any help in understanding where I am going wrong would be greatly appreciated.

1 **Answer**

**A1:**You must extend ERC721Enumerable, and only then you can use totalSupply() function.Your smart contract should be similtar to this:  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.2;import "@openzeppelin/contracts/token/ERC721/ERC721.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";import "@openzeppelin/contracts/access/Ownable.sol";import "@openzeppelin/contracts/utils/math/SafeMath.sol";contract WTest is ERC721, ERC721Enumerable, ERC721Burnable, Ownable { using Strings for uint256; using SafeMath for uint256; uint256 public mintPrice; address public blackHoleAddress; ERC721 public crateContract; string public baseURI; string public baseExtension = ".json"; mapping(uint256 => bool) private \_crateProcessList; bool public paused = false; bool public revealed = false; uint256 public maxSupply = 5000; uint256 public maxPrivateSupply = 580; uint256 public maxMintAmount = 20; string public notRevealedUri; event OperationResult(bool result, uint256 itemId); constructor() ERC721("WTest", "WTST") {} function \_baseURI() internal view virtual override returns (string memory) { return baseURI; } function setBASEURI(string memory newuri) public onlyOwner { baseURI = newuri; } function setMintPrice(uint256 \_mintPrice) public onlyOwner returns(bool success) { mintPrice = \_mintPrice; return true; } function getMintPrice() public view returns (uint256) { return mintPrice; } function setBlackHoleAddress(address \_blackHoleAddress) public onlyOwner returns(bool success) { blackHoleAddress = \_blackHoleAddress; return true; } function setcrateContractAddress(ERC721 \_crateContractAddress) public onlyOwner returns (bool success) { crateContract = \_crateContractAddress; return true; } function mint(uint256 \_mintAmount) public payable { uint256 supply = totalSupply(); require(!paused); require(\_mintAmount > 0); require(\_mintAmount <= maxMintAmount); require(supply + \_mintAmount <= maxSupply); if (msg.sender != owner()) { require(msg.value >= mintPrice \* \_mintAmount); } for (uint256 i = 1; i <= \_mintAmount; i++) { \_safeMint(msg.sender, supply + i); } } // NOTE: Override ERC721Enumerable functions function \_beforeTokenTransfer(address from, address to, uint256 tokenId) internal override(ERC721, ERC721Enumerable) { super.\_beforeTokenTransfer(from, to, tokenId); } function supportsInterface(bytes4 interfaceId) public view override(ERC721, ERC721Enumerable) returns (bool) { return super.supportsInterface(interfaceId); }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]