# 275.https://stackoverflow.com/questions/71880843/i-wrote-a-solidity-that-pays-erc-20-tokens-for-erc-721-nft-transaction-but-it-d

**T:**I wrote a solidity that pays ERC-20 tokens for ERC-721 NFT transaction, but it doesn't work

**Q:**I wrote a solidity that pays ERC-20 tokens for ERC-721 NFT transactions, but it doesn't work. On purchase, you trigger the purchaseToken method. But I get an unknown error. I'm testing it on ropsten. Can you find the problem with my solidity file? I can't solve it and I'm on the verge of dying. Please advice from seniors. please.  
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.0;import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";import "@openzeppelin/contracts/utils/Counters.sol";import "./saleNFT.sol";contract MintNFT is ERC721Enumerable { using Counters for Counters.Counter; Counters.Counter private \_tokenIds; address owner; // constructor constructor() ERC721("NAME", "SYMBOL") { owner = msg.sender; } SaleNFT public saleNFT; // struct struct tokenData { uint256 tokenId; string tokenURI; uint256 tokenPrice; } // modifier modifier onlyOwner { require(msg.sender == owner, "Caller not owner"); \_; } // mapping mapping(uint256 => string) private \_tokenURIs; // variable uint256 balanceLength; function mintNftToken(string memory \_tokenURI) public onlyOwner returns (uint256) { \_tokenIds.increment(); uint256 nftTokenId = \_tokenIds.current(); \_mint(msg.sender, nftTokenId); \_setTokenURI(nftTokenId, \_tokenURI); return nftTokenId; } function \_setTokenURI(uint256 \_tokenId, string memory \_tokenURI) internal { require(\_exists(\_tokenId), "ERC721Metadata: URI set of nonexistent token"); \_tokenURIs[\_tokenId] = \_tokenURI; } }// SPDX-License-Identifier: MITpragma solidity ^0.8.0;import "@openzeppelin/contracts/token/ERC20/ERC20.sol";import "./mintNFT.sol";contract SaleNFT { MintNFT public mintNftTokenAddress; IERC20 public currencyTokenAddress; uint256 feePercent; address payable feeAddress; address owner; // constructor constructor(address \_mintNftTokenAddress, IERC20 \_currencyTokenAddress, address payable \_feeAddress, uint256 \_feePercent) { mintNftTokenAddress = MintNFT(\_mintNftTokenAddress); currencyTokenAddress = \_currencyTokenAddress; feeAddress = \_feeAddress; feePercent = \_feePercent; owner = msg.sender; } // struct struct Trade { uint256 item; uint256 price; } // modifier modifier onlyOwner { require(msg.sender == owner, "Caller not owner"); \_; } // mapping mapping(uint256 => Trade) public trades; // array uint256[] public onSaleTokenArray; function changeFee(uint256 \_feePercent , address payable \_feeAddress) onlyOwner external returns (bool) { feePercent = \_feePercent; feeAddress = \_feeAddress; return true; } function \_pay(address from, address to, uint256 amount) internal returns (bool) { return IERC20(currencyTokenAddress).transferFrom(from, to, amount); } function setForSaleNft(uint256 \_tokenId, uint256 \_price) public { address tokenOwner = mintNftTokenAddress.ownerOf(\_tokenId); require(tokenOwner == msg.sender, "Caller is not token owner."); require(\_price > 0, "Price is zero or lower."); require(trades[\_tokenId].price == 0, "This token is already on sale."); require(mintNftTokenAddress.isApprovedForAll(tokenOwner, address(this)), "token owner did not approve token."); trades[\_tokenId] = Trade({ item: \_tokenId, price: \_price }); onSaleTokenArray.push(\_tokenId); } function purchaseToken(uint256 \_tokenId) external { uint256 price = trades[\_tokenId].price; address tokenOwner = mintNftTokenAddress.ownerOf(\_tokenId); uint256 balance = currencyTokenAddress.balanceOf(msg.sender); uint256 premium = price \* feePercent / 10000; require(price > 0, "token not sale"); require(tokenOwner != msg.sender, "Caller is token owner."); require((balance + premium) >= price, "Lack of balance"); require(balance >= (premium + price), "Caller sent lower than price."); require(\_pay(msg.sender, tokenOwner, price), "token transfer error for owner"); mintNftTokenAddress.safeTransferFrom(tokenOwner, msg.sender, \_tokenId); require(\_pay(msg.sender, feeAddress, premium), "token transfer error for feeaddress"); trades[\_tokenId].price = 0; for(uint256 i = 0; i < onSaleTokenArray.length; i++) { if (trades[onSaleTokenArray[i]].price == 0) { if ((onSaleTokenArray.length - 1) != i) { onSaleTokenArray[i] = onSaleTokenArray[onSaleTokenArray.length - 1]; } onSaleTokenArray.pop(); } } } }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**Can you add in your questions mintNFT smart contract code?

**C2:**yes i added the code.

**C3:**Looks like you are importing a saleNFT.sol file. Can you share the content of saleNFT.sol? to understand your problem, we need the whole files.

0 **Answer**