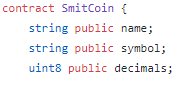
**Creating a Token**

**Setting Solidity Version:**

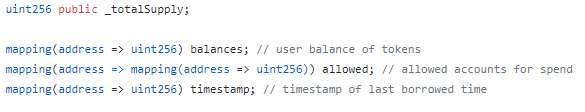
****

* Sets version of solidity between versions 0.6.0 and 0.8.0

**Start with the variables for token Info:**

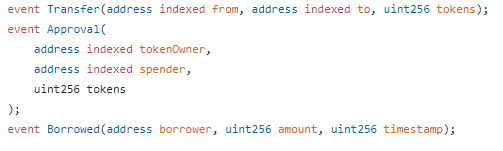
****

* Creates variables for the name of the token, the symbol, and the decimal values the token will calculate
  + Decimals should be set to 18 since that is the standard for ether
  + The symbol should correspond to the name
* Make sure to use the correct type for whatever variable you want to create. In this example it is <string> for all the words/symbol and uin8 for the numbers



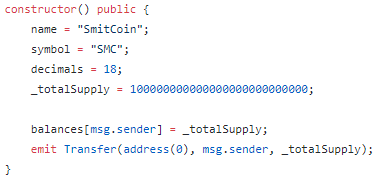
* Creates variables for the total supply of the token, the user balance, the allowed spending, and the timestamp of when the transaction occurred
  + These will be called several times throughout the contract
* The <mapping> command attaches the address of the user to the total balance, allowed accounts , and the timestamp (These values will be called on in later functions)

**Creating Events for Transfer, Approval, and Borrowing**



* Parameters:
  + <address indexed from>: The address sending the tokens
  + <address indexed to>: The address receiving the tokens
  + <uint256 tokens/amount>: The number of tokens being sent
  + <address borrower>: The address of the borrower
  + <uint256 timestamp>: The time the tokens were sent (so the transaction can be verified)
* Sets up the events with all some of the basic info needed to call them later on in the contract
* Transfer will be for moving tokens between accounts, Approval will authenticate the transaction, and Borrowed will be for receiving tokens from a faucet.
* These events will be called again later on in the contract

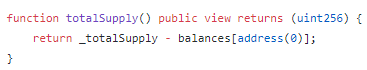
**The Constructor**

****

* The constructor sets up some of the basic values of the token
* It establishes the name, symbol, and decimals from the beginning
* It also sets the total supply of tokens and matches it to the balance of the account that created the token

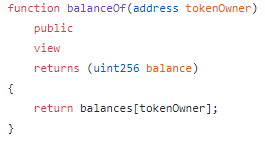
**Creating Functions**

**Total Supply**

****

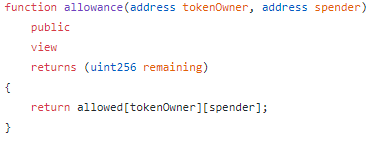
* Sets totalSupply() as the balance of account 0 (the account that created the token)

**Balance Of**



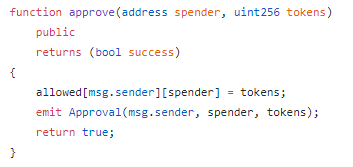
* Returns the balance of tokens in the current account that is being used

**Allowance**



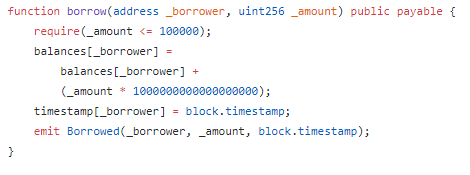
* Gives an allowance to another address in order to retrieve tokens from it.
* Returns the remaining number of tokens the spender can spend on behalf of the owner

**Approve**

****

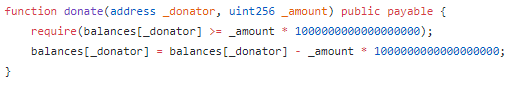
* Approves the transfer being made, returns the number of tokens and spender

**Borrow**



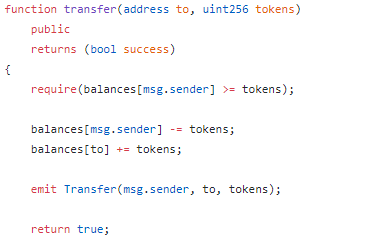
* Allows someone to borrow a certain number of tokens from faucet (in this case <= 100k)

**Donate**



* Allows someone to donate money back to the treasury, if they have a sufficient balance

**Transfer**

****

* Allows for transfers between addresses, assuming each has a sufficient balance

**Transfer From**

****

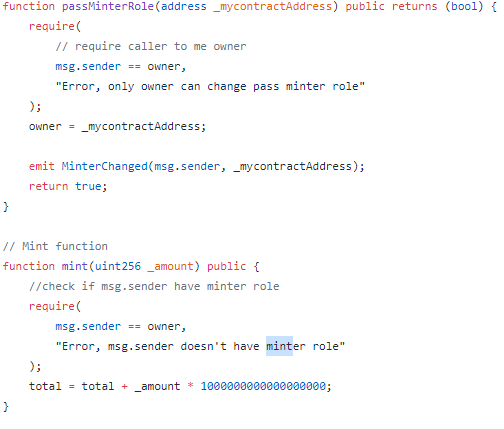
* Similar to “transfer” except it allows other contracts to handle the transfer

**Mint**

****

****

* Add these to the events at the beginning of the contract

****

* Sets up two events in the beginning which correspond to the address of the minter and the actual mint event
* Once the minter is established and approved (through passMinterRole) the Mint function will be called and more tokens will be creates
* This will add to the totalSupply established earlier.