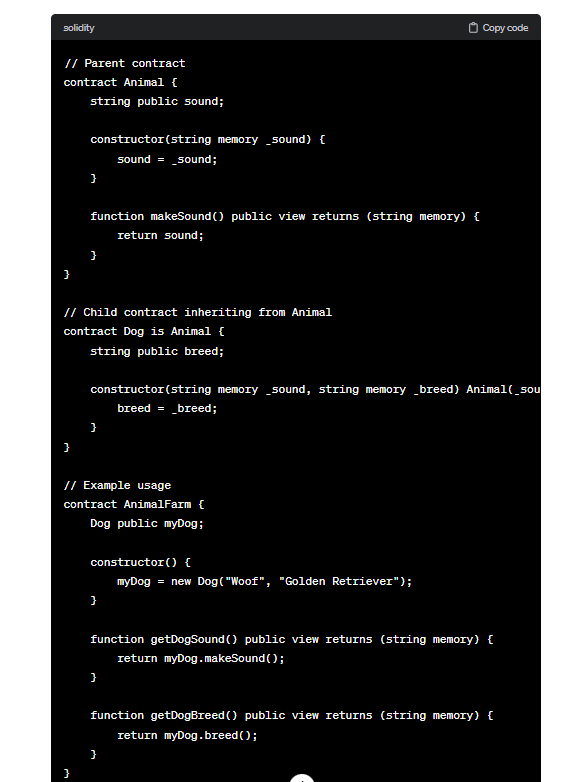
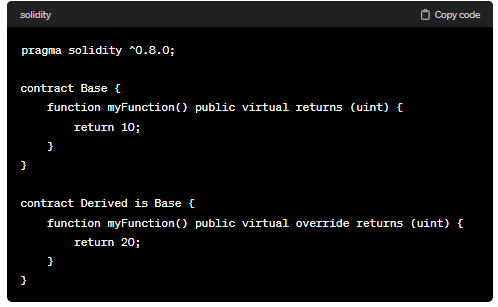
**Inheritance:**

Inheritance in Solidity is a way to structure and organize smart contract code by allowing one contract to inherit properties and functionalities from another contract. This concept promotes code reusability and makes it easier to manage and maintain your code.



Inheritance can be used to create more complex contract structures, allowing you to build on existing functionality while keeping your code organized and modular.

**virtual**: This keyword is used to indicate that a function in a base contract is meant to be overridden by derived contracts. It allows a function to be redefined in the derived contracts.



**override**: This keyword is used to explicitly state that a function in a derived contract is intended to override a function in the base contract. It ensures that the overridden function signature matches the one in the base contract.

**Code:**

//SPDX-License-Identifier: GPL-3.0

pragma solidity 0.8.0;

contract A {

    function fun1() public pure returns(string memory){

        return "i'm in contract A";

    }

    function fun2() public pure returns(string memory){

        return "i'm in contract A";

    }

    function fun3() public pure virtual returns(string memory){

        return "i'm in contract A";

    }

    function fun4() public pure virtual returns(string memory){

        return "i'm in contract A";

    }

}

contract B is A{

    function fun3() public pure override returns(string memory){

        return "i'm in contract B";

    }

    function fun4() public pure virtual override returns(string memory){

        return "i'm in contract B";

    }

}

contract C is B{

    function fun4() public pure override returns(string memory){

        return "i'm in contract C";

    }

}

**Output:**





