Interface

You can interact with other contracts by declaring an `Interface`.

Interface

- cannot have any functions implemented
- can inherit from other interfaces
- all declared functions must be external
- cannot declare a constructor
- cannot declare state variables

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Counter {
    uint public count;

    function increment() external {
        count += 1;
    }
}

interface ICounter {
    function count() external view returns (uint);

    function increment() external;
}

contract MyContract {
    function incrementCounter(address _counter) external {
        ICounter(_counter).increment();
    }

    function getCount(address _counter) external view returns (uint) {
        return ICounter(_counter).count();
    }
}

// Uniswap example
interface UniswapV2Factory {
    function getPair(address tokenA, address tokenB)
        external
        view
        returns (address pair);
}

interface UniswapV2Pair {
    function getReserves()
        external
        view
        returns (
            uint112 reserve0,
            uint112 reserve1,
            uint32 blockTimestampLast
        );
}

contract UniswapExample {
    address private factory = 0x5C69bEe701ef814a2B6a3EDD4B1652CB9cc5aA6f;
    address private dai = 0x6B175474E89094C44Da98b954EedeAC495271d0F;
    address private weth = 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;

    function getTokenReserves() external view returns (uint, uint) {
        address pair = UniswapV2Factory(factory).getPair(dai, weth);
        (uint reserve0, uint reserve1, ) = UniswapV2Pair(pair).getReserves();
        return (reserve0, reserve1);
    }
}
```