Solidity-Inheritance

Inheritance is a way to extend functionality of a contract. Solidity supports both single as well as multiple inheritance. Following are the key highlighsts.

A derived contract can access all non-private members including internal methods and state variables. But using this is not allowed.

Function overriding is allowed provided function signature remains same. In case of difference of output parameters, compilation will fail.

We can call a super contract's function using super keyword or using super contract name.

In case of multiple inheritance, function call using super gives preference to most derived contract.

Example

```
pragma solidity ^0.5.0;

contract C {
  //private state variable
  uint private data;

  //public state variable
  uint public info;

  //constructor
  constructor() public {
    info = 10;
  }
  //private function
```

```
   function increment(uint a) private pure returns(uint) { return a + 1; }

   //public function
   function updateData(uint a) public { data = a; }
   function getData() public view returns(uint) { return data; }
   function compute(uint a, uint b) internal pure returns (uint) { return a +
b; }}//Derived Contract
contract E is C {
   uint private result;
   C private c;
   constructor() public {
      c = new C();
   }
   function getComputedResult() public {
      result = compute(3, 5);
   }
   function getResult() public view returns(uint) { return result; }
   function getData() public view returns(uint) { return c.info(); }}
```

Run the above program using steps provided in Solidity First Application chapter. Run various method of Contracts. For E.getComputedResult() followed by E.getResult() shows −

Output

```
0: uint256: 8
```