Shadowing State Variables

Description

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract `A` with a variable `x` could inherit contract `B` that also has a state variable `x` defined. This would result in two separate versions of `x`, one of them being accessed from contract `A` and the other one from contract `B`. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

Remediation

Review storage variable layouts for your contract systems carefully and remove any ambiguities. Always check for compiler warnings as they can flag the issue within a single contract.

Example:

Code:

```
1 pragma solidity 0.4.24;
2
3 contract ShadowingInFunctions {
4     uint n = 2;
5     uint x = 3;
6
7     function test1() constant returns (uint n) {
8         return n; // Will return 0
9     }
10
11    function test2() constant returns (uint n) {
12        n = 1;
13        return n; // Will return 1
14    }
15
16    function test3() constant returns (uint x) {
17        uint n = 4;
18        return n+x; // Will return 4
19    }
20 }
```

Explanation:

In the test1() function, the n state variable is declared as a return local variable with a default value of 0. However, it did not inherit the value set at line 4.

The n state variable is defined and initialized with value in the test2() function. Therefore, it behaves as a local state variable and returns the value that was just initialised.

The n and x state variables are defined and initialized inside the test3() function. Therefore, it did not inherit the state variable from the contract and performed calculations using the value of the local variable.