Solidity Fundamentals Quizzes

Pragma Statements

Quiz 1

Instructions: Examine the incomplete code snippet below and select the correct option to complete it properly.

```
_____ solidity ^0.8.0;

contract HelloWorld {
    string public greeting = "Hello, World!";
}
```

Choose the correct option to fill in the blank:

- A) `use`
- B) `import`
- C) `pragma`
- D) `version`

Answer: C) `pragma`

Explanation: The `pragma` directive is used in Solidity to specify the compiler version. The statement `pragma solidity ^0.8.0;` tells the compiler that this code should be compiled with a Solidity version that is at least 0.8.0 but less than 0.9.0.

Quiz 2

Instructions: Identify what's wrong with the following Solidity code snippet.

```
pragma solidity 0.8.0;
pragma solidity 0.7.0;

contract MultipleVersions {
    uint256 public value = 100;
}
```

Choose the correct answer:

- A) You can't declare multiple pragma statements in a file
- B) The contract name should match the file name
- C) You can't specify two different compiler versions in the same file
- D) The contract is missing a constructor

Answer: C) You can't specify two different compiler versions in the same file

Explanation: In Solidity, you cannot specify two different compiler versions in the same file. Each Solidity file should have a single pragma directive that specifies the compiler version to use for that file.

Contracts

Quiz 1

Instructions: Complete the following Solidity code snippet by selecting the correct option to create a proper contract.

```
pragma solidity ^0.8.0;

_____ Counter {
    uint256 count;

    function increment() public {
        count += 1;
    }
}
```

Choose the correct option to fill in the blank:

- A) `interface`
- B) `library`
- C) `contract`
- D) `class`

Answer: C) `contract`

Explanation: In Solidity, the `contract` keyword is used to define a contract, which is similar to a class in object-oriented programming languages. Contracts are the fundamental building blocks in Ethereum that contain code (functions) and data (state variables).

Quiz 2

Instructions: Identify what feature is being demonstrated in the following Solidity contract.

```
pragma solidity ^0.8.0;

contract Parent {
    function getValue() public pure virtual returns (uint) {
        return 10;
    }
}

contract Child _____ Parent {
    function getValue() public pure override returns (uint) {
```

```
        return 20;
    }
}
```

Choose the correct option to fill in the blank:

- A) `extends`
- B) `is`
- C) `implements`
- D) `inherits`

Answer: B) `is`

Explanation: In Solidity, inheritance is indicated using the `is` keyword. This allows a contract to inherit and extend functionality from another contract. The `virtual` keyword in the parent contract allows the function to be overridden, while the `override` keyword in the child contract indicates that it's overriding the parent's function.

State Variables

Quiz 1

Instructions: Examine the following Solidity contract and identify which state variable is correctly declared with its visibility specifier.

```
pragma solidity ^0.8.0;

contract StateVariables {
    uint256 value1;
    uint256 internal value2;
    uint256 _____ value3;
    uint256 private value4;
}
```

Choose the correct option to fill in the blank:

- A) `external`
- B) `hidden`
- C) `public`
- D) `protected`

Answer: C) `public`

Explanation: In Solidity, state variables can have three visibility specifiers: `public`, `internal`, and `private`. The `public` visibility automatically generates a getter function that allows other contracts to read the variable. The `external` visibility is not applicable to state variables and is only used for functions.

Quiz 2

Instructions: What will be the value of `defaultValue` in the following contract?

```
pragma solidity ^0.8.0;

contract DefaultValues {
    _____ defaultValue;

    function getDefaultValue() public view returns (uint) {
        return defaultValue;
    }
}
```

Choose the correct option to fill in the blank and determine the value:

- A) `uint public` - Value will be 1
- B) `uint private` - Value will be 0
- C) `int public` - Value will be -1
- D) `uint public` - Value will be 0

Answer: D) `uint public` - Value will be 0

Explanation: In Solidity, state variables are automatically initialized to a default value if not explicitly assigned one. For `uint` (unsigned integer) types, the default value is 0. The visibility (`public`) doesn't affect this default initialization behavior.

Feel free to use these quizzes in your Solidity course for beginners. They cover the fundamental concepts of pragma statements, contracts, and state variables, which are essential building blocks for Solidity development.