**Coding Standards and Guidelines**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** | **Standards and Guidelines** | **Status of Code 1** | **Status of Code 2** |
| 1 | Code MUST use 4 spaces for indenting, not tabs. | YES | NO |
| 2 | There MUST NOT be a hard limit on line length; the soft limit MUST be 120 characters; lines SHOULD be 80 characters or less. | YES | YES |
| 3 | Opening braces for classes MUST go on the next line, and closing braces MUST go on the next line after the body. | YES | MISSING |
| 4 | Opening braces for methods MUST go on the next line, and closing braces MUST go on the next line after the body. | NO | NO |
| 5 | Control structure keywords MUST have one space after them; method and function calls MUST NOT. | MISSING | NO |
| 6 | Opening braces for control structures MUST go on the same line, and closing braces MUST go on the next line after the body. | MISSING | NO |
| 7 | Opening parentheses for control structures MUST NOT have a space after them. | MISSING | NO |
| 8 | Closing parentheses for control structures MUST NOT have a space before. | MISSING | YES |
| 9 | All PHP files MUST end with a single blank line. | YES | NO |
| 10 | There MUST NOT be trailing whitespace at the end of non-blank lines. | NO | NO |
| 11 | Blank lines MAY be added to improve readability and to indicate related blocks of code. | YES | YES |
| 12 | There MUST NOT be more than one statement per line. | YES | NO |
| 13 | PHP keywords MUST be in lower case. | YES | MISSING |
| 14 | The PHP constants true, false, and null MUST be in lower case. | YES | MISSING |
| 15 | The ‘extends’ and ‘implements’ keywords MUST be declared on the same line as the class name. | YES | MISSING |
| 16 | Lists of implements MAY be split across multiple lines, where each subsequent line is indented once. When doing so, the first item in the list MUST be on the next line, and there MUST be only one interface per line. | MISSING | MISSING |
| 17 | Visibility MUST be declared on all methods. | NO | YES |
| 18 | Method names SHOULD NOT be prefixed with a single underscore to indicate protected or private visibility. | YES | YES |
| 19 | The opening brace MUST go on its own line, and the closing brace MUST go on the next line following the body. There MUST NOT be a space after the opening parenthesis, and there MUST NOT be a space before the closing parenthesis. | NO | NO |
| 20 | In the argument list, there MUST NOT be a space before each comma, and there MUST be one space after each comma. | MISSING | YES |
| 21 | Method arguments with default values MUST go at the end of the argument list. | MISSING | MISSING |
| 22 | Argument lists MAY be split across multiple lines, where each subsequent line is indented once. When doing so, the first item in the list MUST be on the next line, and there MUST be only one argument per line. | MISSING | MISSING |
| 23 | When the argument list is split across multiple lines, the closing parenthesis and opening brace MUST be placed together on their own line with one space between them. | MISSING | MISSING |
| 24 | When present, the abstract and final declarations MUST precede the visibility declaration. | MISSING | MISSING |
| 25 | When present, the static declaration MUST come after the visibility declaration. | MISSING | MISSING |
| 26 | There MUST be one space between the closing parenthesis and the opening brace | YES | NO |
| 27 | The structure body MUST be indented once | NO | YES |
| 28 | The placement of parentheses, spaces, and braces; and that else and elseif should be on the same line as the closing brace from the earlier body. | YES | YES |
| 29 | The keyword elseif SHOULD be used instead of else if so that all control keywords look like single words. | MISSING | MISSING |
| 30 | The case statement MUST be indented once from switch, and the break keyword (or other terminating keyword) MUST be indented at the same level as the case body. | MISSING | MISSING |
| 31 | A while statement looks like the following. Note the placement of parentheses, spaces, and braces.  **<?php**  **while** ($expr) {  *// structure body*  } | MISSING | MISSING |
| 32 | A do-while statement looks like the following. Note the placement of parentheses, spaces, and braces.  **<?php**  **do** {  *// structure body;*  } **while** ($expr); | MISSING | MISSING |
| 33 | A for statement looks like the following. Note the placement of parentheses, spaces, and braces.  **<?php**  **for** ($i **=** 0; $i **<** 10; $i**++**) {  *// for body*  } | MISSING | MISSING |

<?php   
class multipleInheritance   
{   
    function callClass($class\_to\_call)   
    {   
        return new $class\_to\_call();   
    }   
}   
  
class A   
{   
    function insideA()   
    {   
        echo "I'm inside A!<br />";   
    }   
}   
  
class B   
{   
  
    function insideB ()   
    {   
        echo "I'm inside B!<br />";   
    }   
 }   
  
class C extends multipleInheritance   
{   
    function insideC() {    
        $a = parent::callClass('A');   
        $a->insideA();   
        $b = parent::callClass('B');   
        $b->insideB();}   
 }   
  
$c = new C();   
$c->insideC();   
?>

**Program 2**

class Demo {

private $data = array();

public function \_\_set($variable, $value){

echo 'Setting ' . $variable . ' to ' . $value; $this->data[$variable] = $value; }

public function \_\_get($variable){ if(isset($this->data[$variable])){

return $this->data[$variable]; }else{ die('Unknown variable.'); }

}

}

$d = new Demo; // Set a non-existent variable $d->test = 'Test Variable';

// Get what we just stored echo $d->test;

// Get a non-existant variable echo $d->testFail;

}