TYPO3 Flow Coding Guidelines on one page

Namespace starts with vendor name followed by package key (name) and subparts as needed

One *use* statement per line.
One *use* statement per namespace.
Order statements alphabetically.
Don't import namespaces unless you use them.

No empty line between DocComment and class, member var or method.

Use *@var* tag. Optional description goes in the first comment line followed by a blank comment line.

Prefer relative namespaces, unless Fully Qualified Namespace is more readable

Param tag: type, name, description.

Indent with tabs.

Multiline conditions:
Indent them and add a extra indent to following code. Put the boolean operators at beginning of line.

@return tag with type, even if it is "void". Only __construct() has no return tag.

@api tag defines public API

static and *abstract* keywords before the visibility modifier

```php
<?php
namespace Acme\TestPackage;

/*                                                                        *
 * This script belongs to the TYPO3 Flow package "Acme.TestPackage".      *
 *                                                                        *
 * It is free software; you can redistribute it and/or modify it under    *
 * the terms of the GNU General Public License, either version 3 of the   *
 * License, or (at your option) any later version.                        *
 *                                                                        *
 * The TYPO3 project - inspiring people to share!                         *
 *                                                                        */

use Acme\TestPackage\Service\FooGenerator;
use TYPO3\Flow\Annotations as Flow;

/**
 * Here goes the description of the class. It should explain what the main
 * purpose of this class is...
 *
 * @Flow\Scope("singleton")
 */
class UniverseAnalyzer extends BaseClass implements SomeInterface {

    /**
     * Some injected dependency
     *
     * @Flow\Inject
     * @var FooGenerator
     */
    protected $someDependency = NULL;

    /**
     * Shows if you are addicted to TYPO3 Flow
     *
     * @var boolean
     */
    static protected $addictedToFlow = TRUE;

    /**
     * Shows if you are a fan of TYPO3 Flow
     *
     * @var boolean
     */
    protected $fanOfFlow;

    /**
     * A great method which shows how to indent control structures.
     *
     * @param MyClass $object An instance of MyClass
     * @param array $someArray Some array
     * @return void
     * @throws \Exception
     */
    public function analyzeUniverse(MyClass $object, array $someArray = array()) {
        $subObjects = $object->getSubObjects();
        foreach ($subObjects as $subObject){
            /** @var $subObject MySubClass */
            $subObject->doSomethingCool();
        }
        if (isset($someArray['question'])
            && $this->answerToEverything === 42
            || count($someArray) > 3) {
                $this->fanOfTYPO3Flow = TRUE;
        } else {
            throw new \Exception('We cannot tolerate that.', 1223391710);
        }
    }

    /**
     * This is a setter for the fanOfFlow property.
     *
     * @param boolean $isFan Pass TRUE to mark a fan, FALSE for a Zend follower
     * @return mixed
     */
    public function setFanOfFlow($isFan) {
        $this->fanOfFlow = $isFan;
    }

    /**
     * As simple as it gets - a boolean getter.
     *
     * @return boolean Whether a foo was detected (TRUE) or not (FALSE)
     * @api
     */
    static public function isAddictedToFlow() {
        return self::$addictedToFlow;
    }

}
?>
```

Capture the joy of coding as you create excellent web solutions. Enjoy coding. Enjoy Flow.

Description of the class. Make it as long as needed, feel free to explain how to use it.

UpperCamelCase class name. Class names should be nouns.
In other packages, import \Acme\TestPackage\UniverseAnalyzer and refer to it as UniverseAnaler.

List *@Flow|** before other tags: *@var, @param, @return, @throws, @api, @since, @deprecated*

Description of the method. Make it as long as needed.

Method names should be verbs.

Use type hinting

Only use inline *@var* annotations when type can't be derived (like in an array of objects) to increase readability and trigger IDE auto-completion.

UNIX timestamp at time of writing the throw clause.

Write what went wrong, give helpful details and give a hint for a possible solution.

Setter methods should start with "set".

Methods returning boolean values should start with "has" or "is". Other getters should start with "get".

Opening brace on same line with opening token. One space before.

Also check out the latest documentation: http://docs.typo3.org/flow/TYPO3FlowDocumentation/TheDefinitiveGuide/PartV/CodingGuideLines/Index.html