Flow Coding Guidelines (PSR-2) on one page

Prefer strict types

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

Indent with spaces

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

Param annotation: type, name, description if they provide additional information over type hints

@return annotation with type, only if it provides additional information over the type hint

@api tag defines public API

Opening brace on the next line

```php
<?php
declare(strict_types=1);
namespace Acme\TestPackage;

/*
 * This file is part of the Acme.TestPackage package.
 *
 * (c) Whoever wrote this
 *
 * This package is Open Source Software. For the full copyright and license
 * information, please view the LICENSE file which was distributed with this
 * source code.
 */

use Acme\TestPackage\Service\FooGenerator;
use Neos\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;

    /**
     * @var bool
     */
    static protected $addictedToFlow = true;

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

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

    /**
     * This is a setter for the fanOfFlow property.
     *
     * @param boolean $isFan true to mark a fan, false for a Zend follower
     * @return void since this is a setter, duh
     */
    public function setFanOfFlow(bool $isFan): void
    {
        $this->fanOfFlow = $isFan;
    }

    /**
     * As simple as it gets – a boolean getter.
     *
     * @api
     */
    public static function isAddictedToFlow(): bool
    {
        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 UniverseAnalyer

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“.

Full PSR-2 Coding standard https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md
Also check out the latest documentation: http://flowframework.readthedocs.org/en/stable/TheDefinitiveGuide/PartV/CodingGuideLines/index.html