Flow Coding Guidelines (PSR-2) on one page

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

/*
 * This ·le 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 ·le 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.
     *
     * @param MyClass $object An instance of MyClass
     * @param array $someArray Some array
     * @return void
     * @throws \Exception
     */
    public function analyzeUniverse(MyClass $object, array $someArray = [])
    {
        $subObjects = $object->getSubObjects();
        foreach ($subObjects as $subObject) {
            /** @var $subObject MySubClass */
            $subObject->doSomethingCool();
        }
        if (isset($someArray['question'])
            && $this->answerToEverything === 42
            || count($someArray) > 3 {
                $this->fanOfFlow = true;
        } else {
            throw new \Exception('We cannot tolerate that.', 1223391710);
        }
    }

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

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

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

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

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

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

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

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

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

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

Prefer relative namespaces, unless Fully Qualified Namespace is more readable

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

Param tag: type, name, description.

Method names should be verbs.

Indent with spaces.

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.

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

UNIX timestamp at time of writing the throw clause.

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

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

Setter methods should start with "set".

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

@api tag defines public API

Opening brace on the next line

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