Web Adv

PHP: PHPUnit (deel I)

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

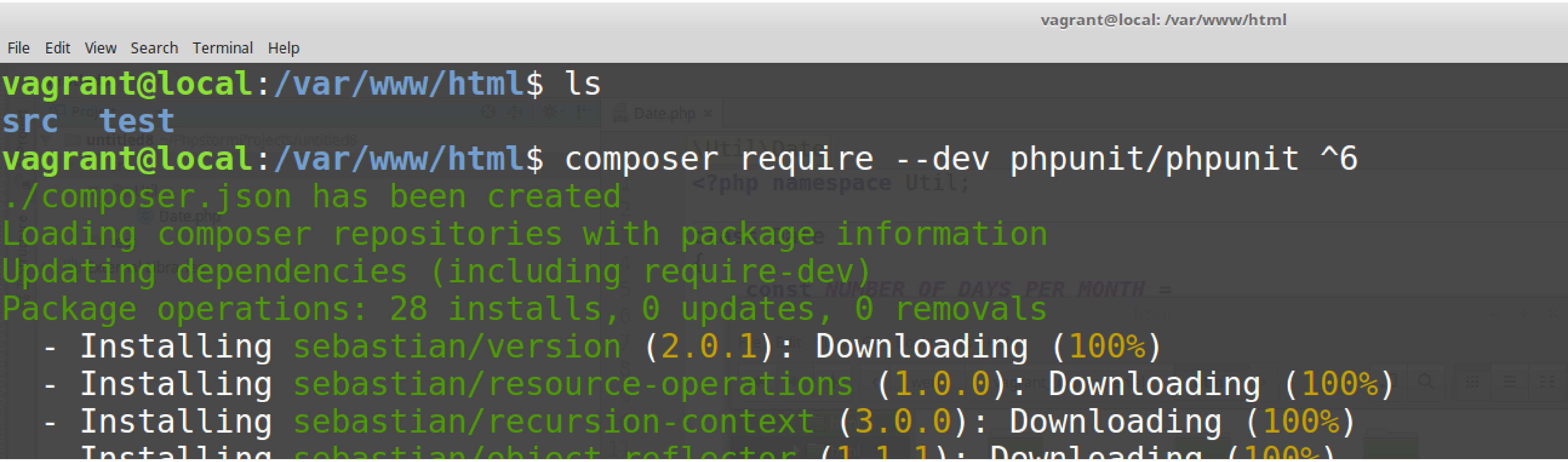
PHPUnit

Unit-test:

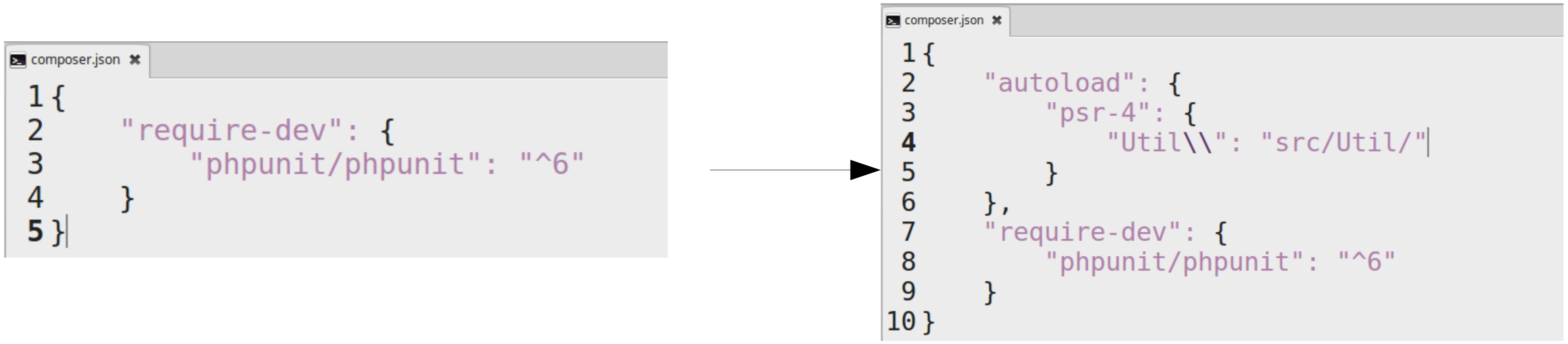
- testen van één klasse onafhankelijk van 'buitenwereld'
- groot project, veel programmeurs
- gaat mijn wijziging de rest vd code verstoren?

    doe de wijziging run alle unit tests

PHPUnit

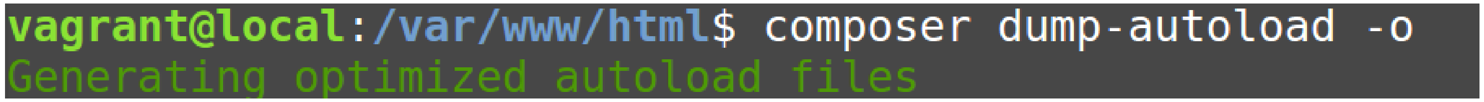

vagrant@local: /var/www/html
File Edit View Search Terminal Help

```
vagrant@local:/var/www/html$ ls
src  test
vagrant@local:/var/www/html$ composer require --dev phpunit/phpunit ^6
./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 28 installs, 0 updates, 0 removals
  - Installing sebastian/version (2.0.1): Downloading (100%)
  - Installing sebastian/resource-operations (1.0.0): Downloading (100%)
  - Installing sebastian/recursion-context (3.0.0): Downloading (100%)
  - Installing sebastian/object-reflector (1.1.1): Downloading (100%)
```



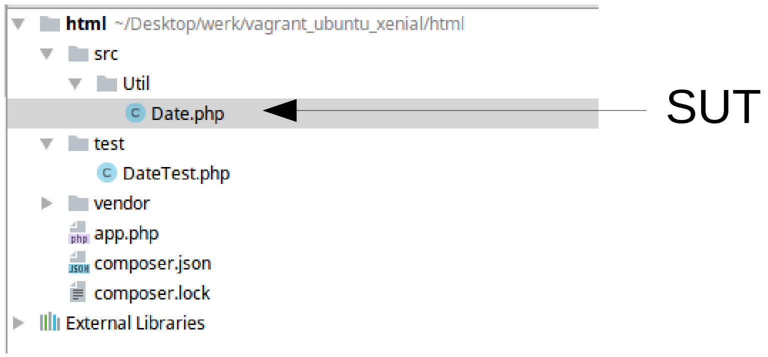
```json
{
    "require-dev": {
        "phpunit/phpunit": "^6"
    }
}
```

```json
{
    "autoload": {
        "psr-4": {
            "Util\\": "src/Util/"
        }
    },
    "require-dev": {
        "phpunit/phpunit": "^6"
    }
}
```

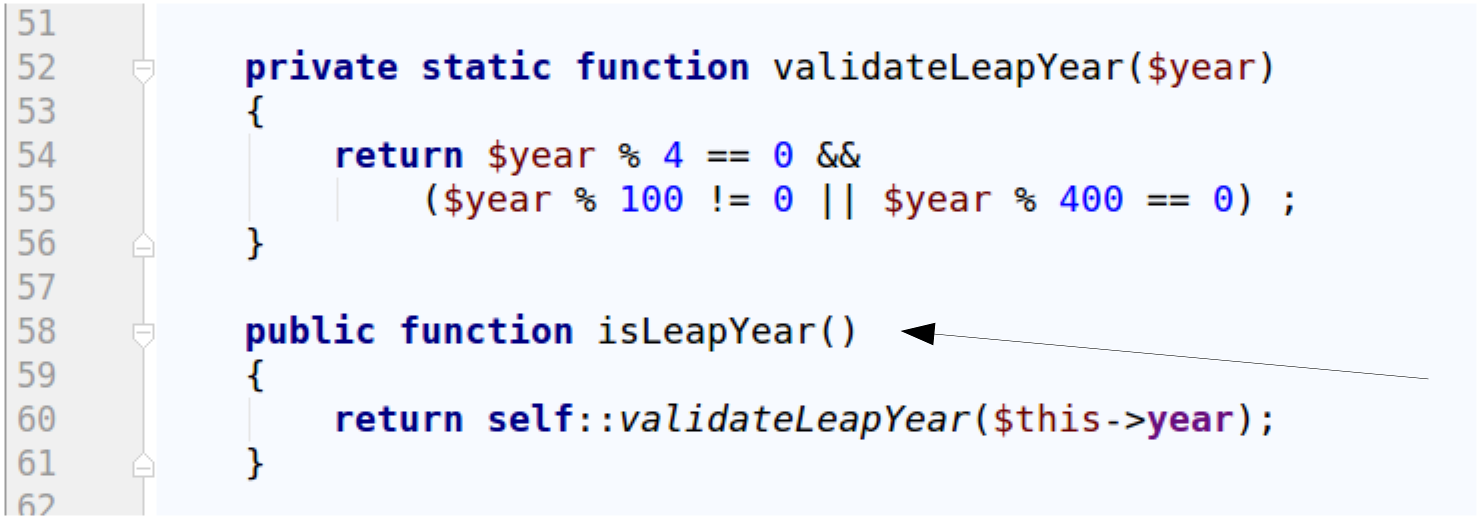


```
vagrant@local:/var/www/html$ composer dump-autoload -o
Generating optimized autoload files
```

PHPUnit

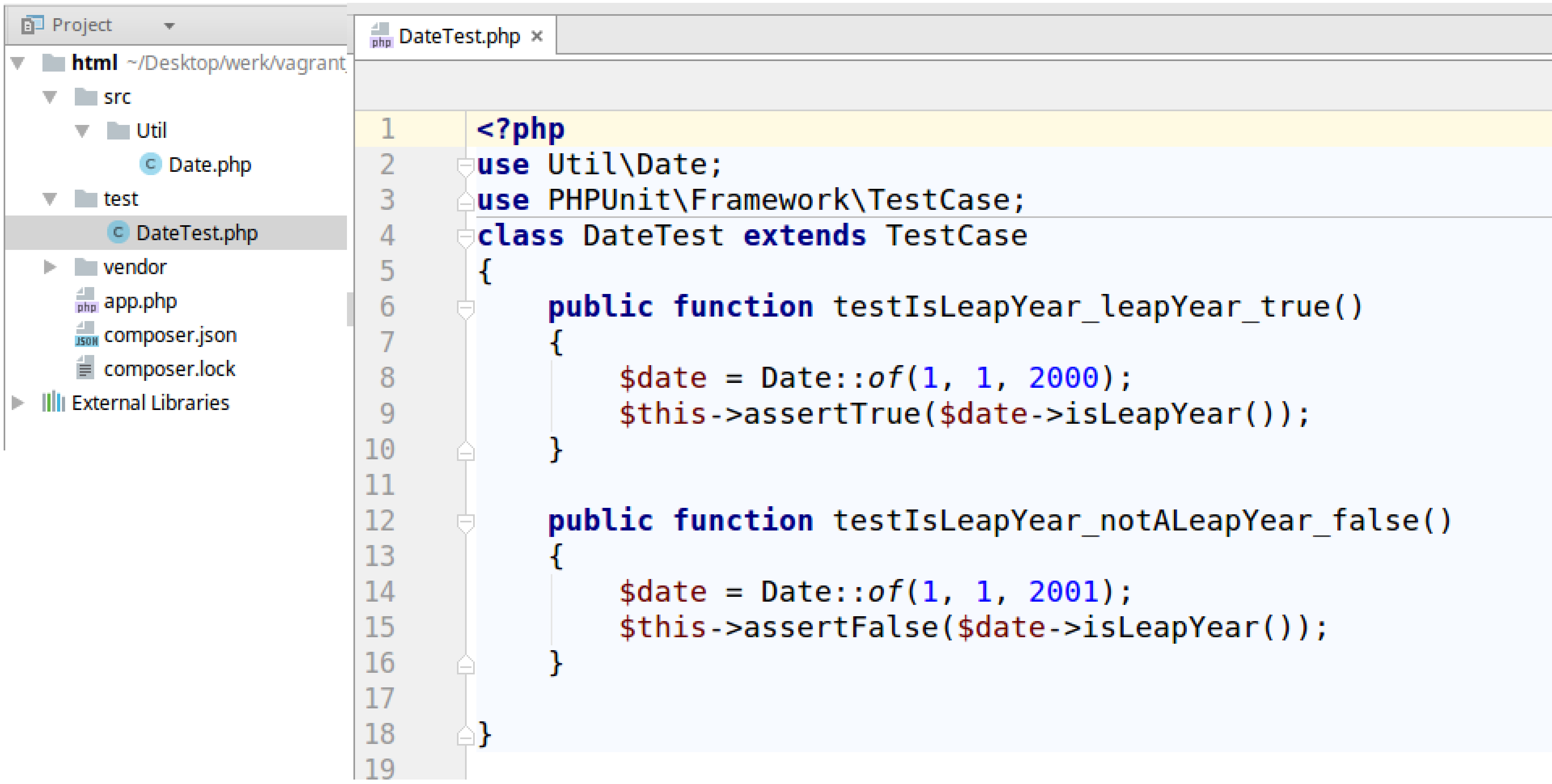


```
html ~/Desktop/werk/vagrant_ubuntu_xenial/html
  src
    Util
      Date.php
  test
    DateTest.php
  vendor
  app.php
  composer.json
  composer.lock
External Libraries
```

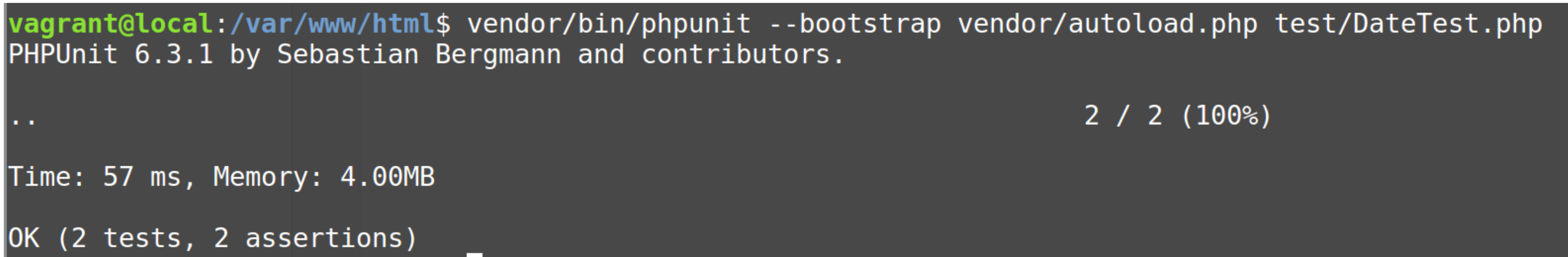


```php

    private static function validateLeapYear($year)
    {
        return $year % 4 == 0 &&
            ($year % 100 != 0 || $year % 400 == 0) ;
    }

    public function isLeapYear()
    {
        return self::validateLeapYear($this->year);
    }

```

PHPUnit



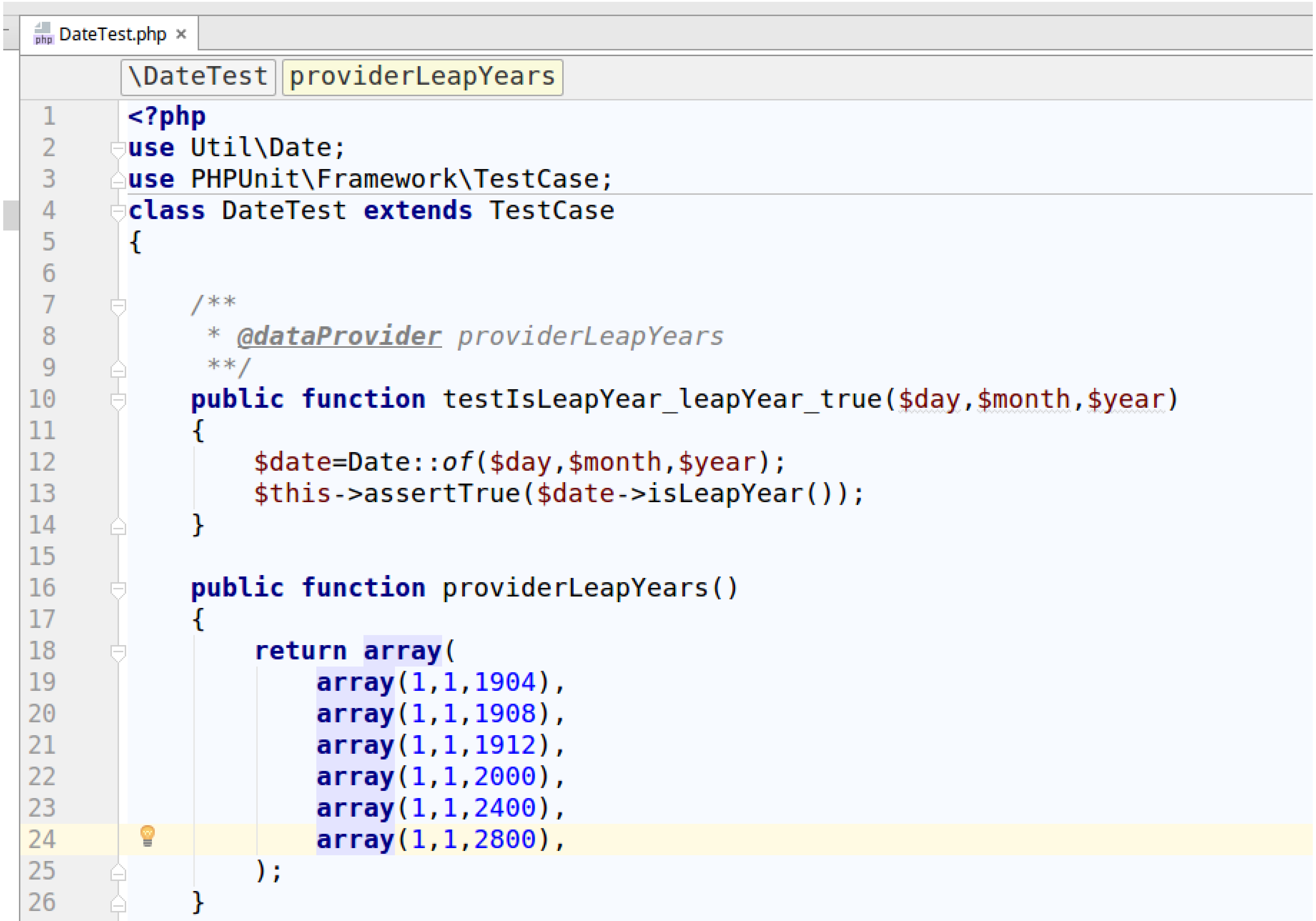
```php
<?php
use Util\Date;
use PHPUnit\Framework\TestCase;
class DateTest extends TestCase
{
    public function testIsLeapYear_leapYear_true()
    {
        $date = Date::of(1, 1, 2000);
        $this->assertTrue($date->isLeapYear());
    }

    public function testIsLeapYear_notALeapYear_false()
    {
        $date = Date::of(1, 1, 2001);
        $this->assertFalse($date->isLeapYear());
    }

}
```



```
vagrant@local:/var/www/html$ vendor/bin/phpunit --bootstrap vendor/autoload.php test/DateTest.php
PHPUnit 6.3.1 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 57 ms, Memory: 4.00MB

OK (2 tests, 2 assertions)
```

assertEquals(expected, actual)	kijk of de bekomen waarde (actual) gelijk is aan een verwachte waarde (expected)
assertTrue(actual)	kijk of de bekomen waarde True is
assertFalse(actual)	kijk of de bekomen waarde True is
assertLessThan(expected, actual)	kijk of de bekomen waarde kleiner is dan de verwachte waarde
assertNull(expected)	kijk of de bekomen waarde gelijk is aan null
assertRegExp(pattern, actual)	kijk of de bekomen waarde voldoet aan een patroon
assertInstanceOf(expected, object)	kijk object een instantie is van een klasse

dataProvider



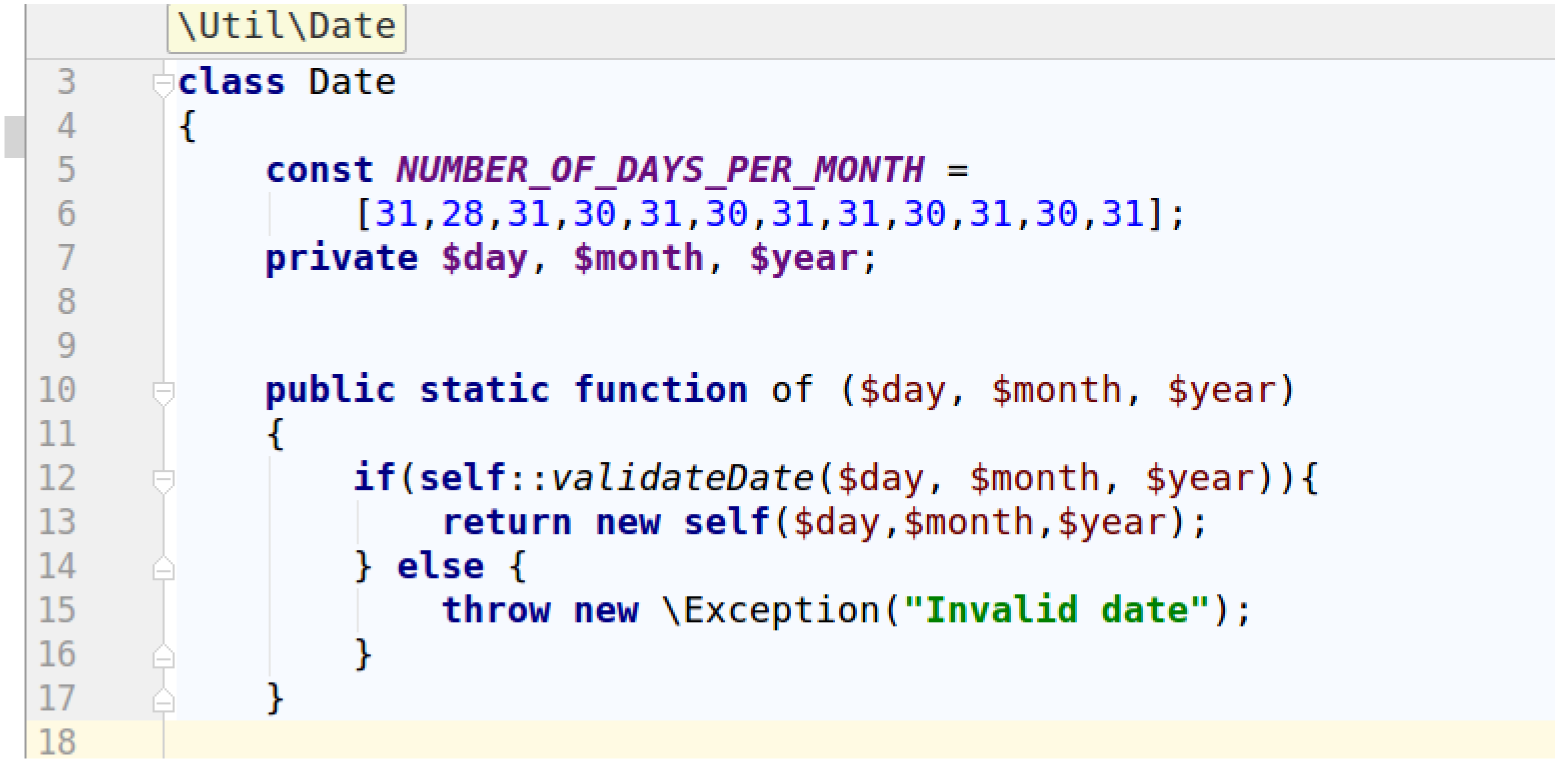
DateTest.php ×

\DateTest providerLeapYears

```php
<?php
use Util\Date;
use PHPUnit\Framework\TestCase;
class DateTest extends TestCase
{

    /**
     * @dataProvider providerLeapYears
     **/
    public function testIsLeapYear_leapYear_true($day,$month,$year)
    {
        $date=Date::of($day,$month,$year);
        $this->assertTrue($date->isLeapYear());
    }

    public function providerLeapYears()
    {
        return array(
            array(1,1,1904),
            array(1,1,1908),
            array(1,1,1912),
            array(1,1,2000),
            array(1,1,2400),
            array(1,1,2800),
        );
    }
```

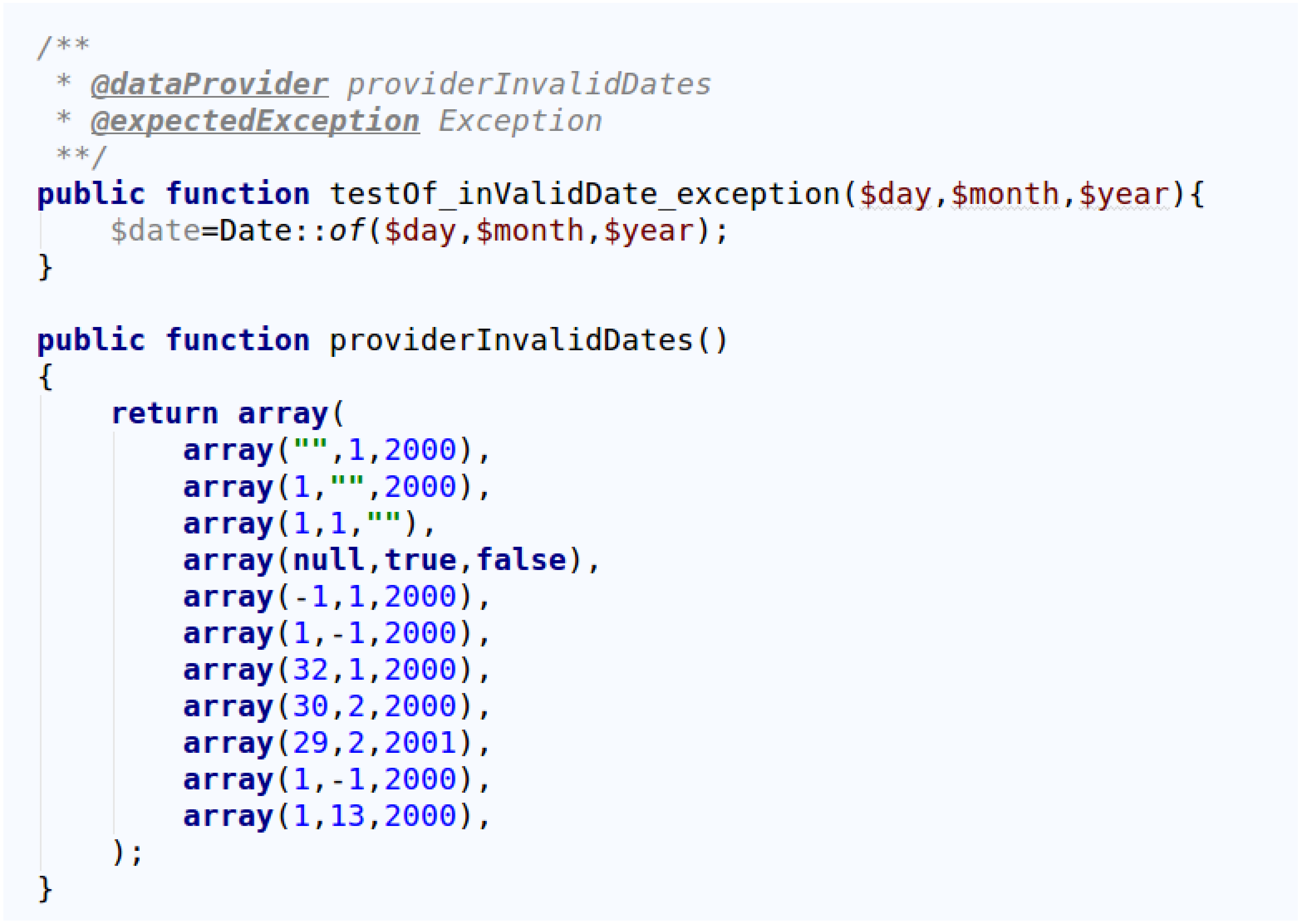
Exceptions

SUT



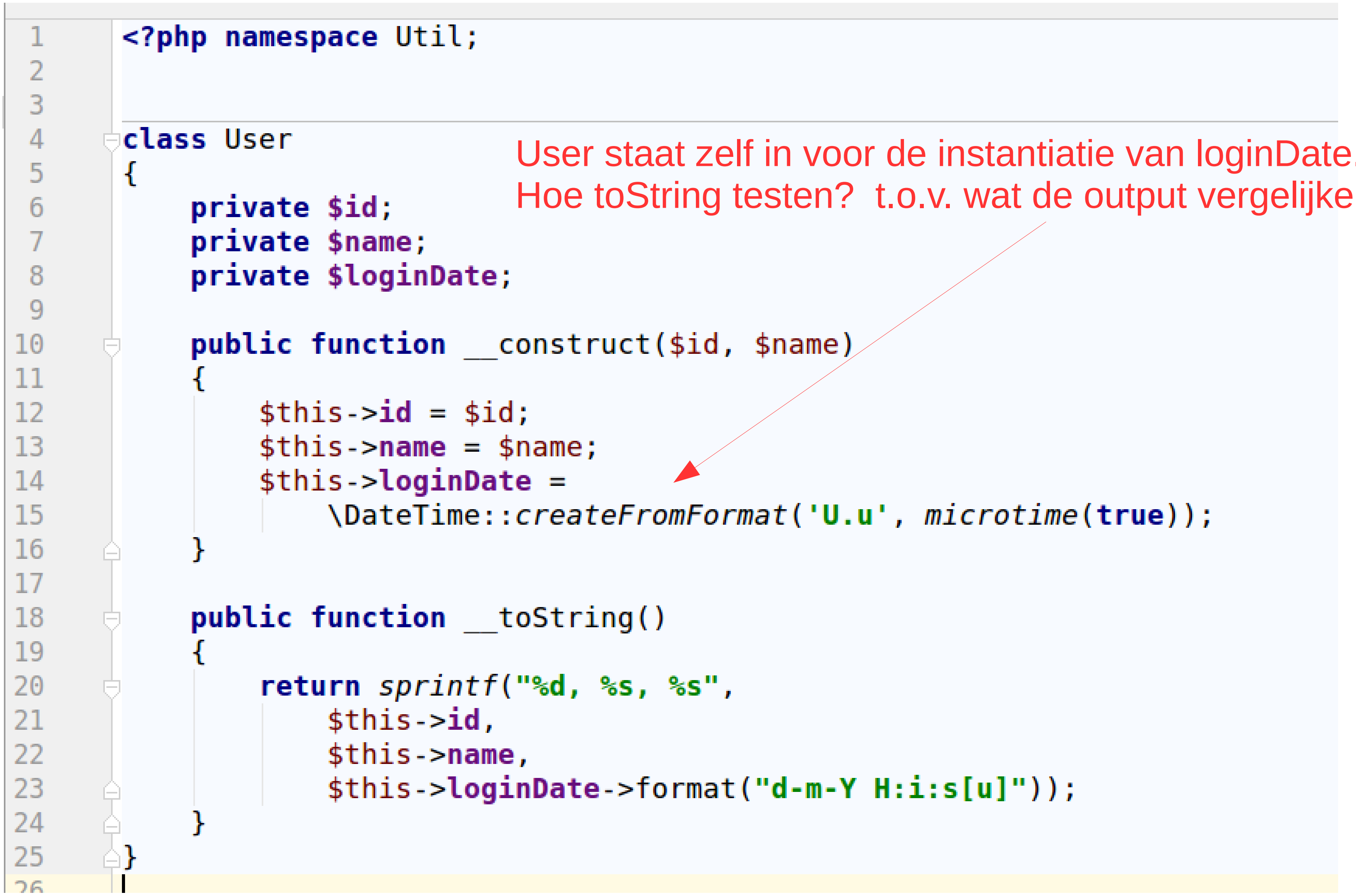
```
\Util\Date
class Date
{
    const NUMBER_OF_DAYS_PER_MONTH =
        [31,28,31,30,31,30,31,31,30,31,30,31];
    private $day, $month, $year;


    public static function of ($day, $month, $year)
    {
        if(self::validateDate($day, $month, $year)){
            return new self($day,$month,$year);
        } else {
            throw new \Exception("Invalid date");
        }
    }

```

Exceptions



```
/**
 * @dataProvider providerInvalidDates
 * @expectedException Exception
 **/
public function testOf_inValidDate_exception($day,$month,$year){
    $date=Date::of($day,$month,$year);
}

public function providerInvalidDates()
{
    return array(
        array("",1,2000),
        array(1,"",2000),
        array(1,1,""),
        array(null,true,false),
        array(-1,1,2000),
        array(1,-1,2000),
        array(32,1,2000),
        array(30,2,2000),
        array(29,2,2001),
        array(1,-1,2000),
        array(1,13,2000),
    );
}
```

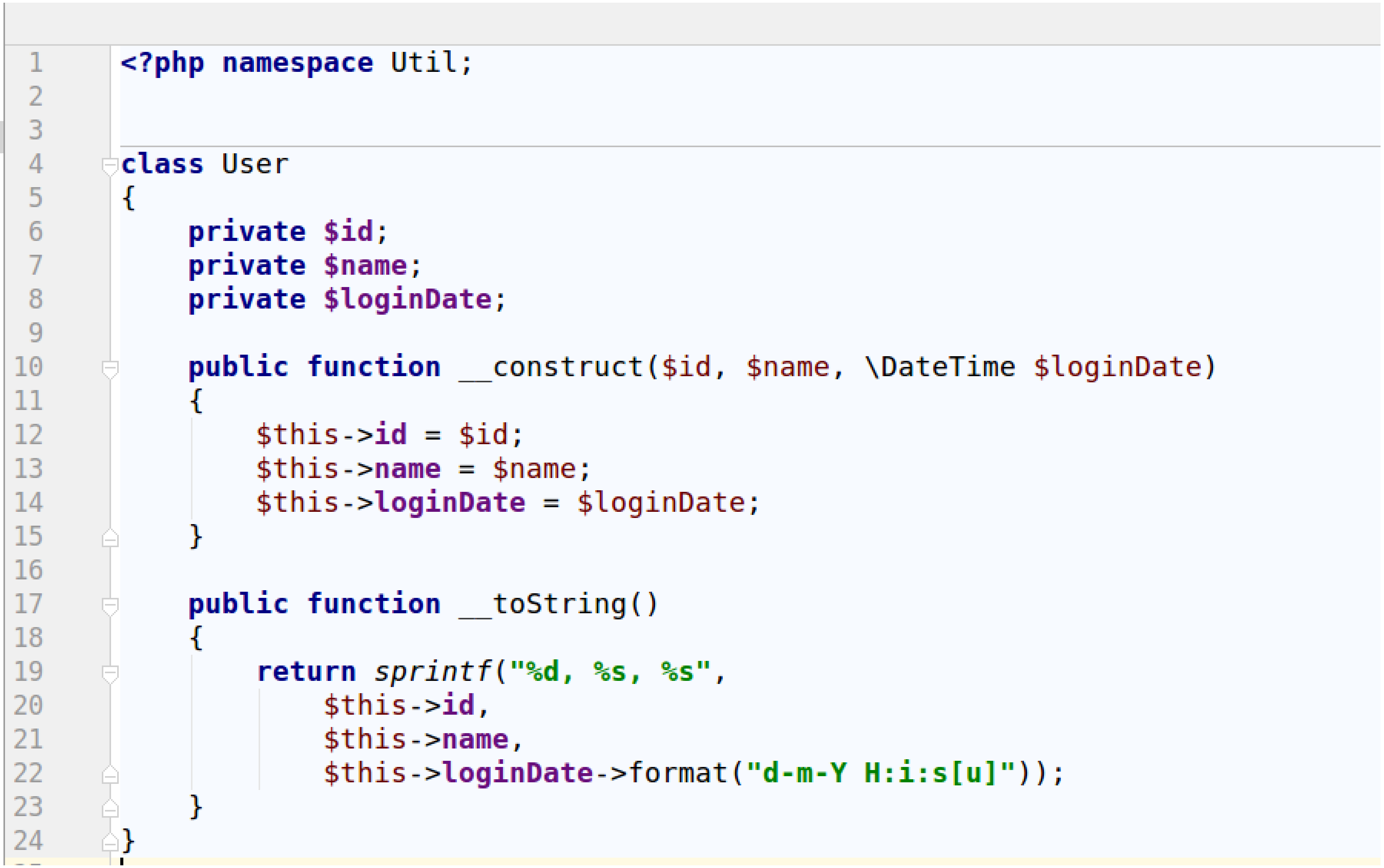
Ontestbaar



```
<?php namespace Util;


class User
{
    private $id;
    private $name;
    private $loginDate;

    public function __construct($id, $name)
    {
        $this->id = $id;
        $this->name = $name;
        $this->loginDate =
            \DateTime::createFromFormat('U.u', microtime(true));
    }

    public function __toString()
    {
        return sprintf("%d, %s, %s",
            $this->id,
            $this->name,
            $this->loginDate->format("d-m-Y H:i:s[u]"));
    }
}
```

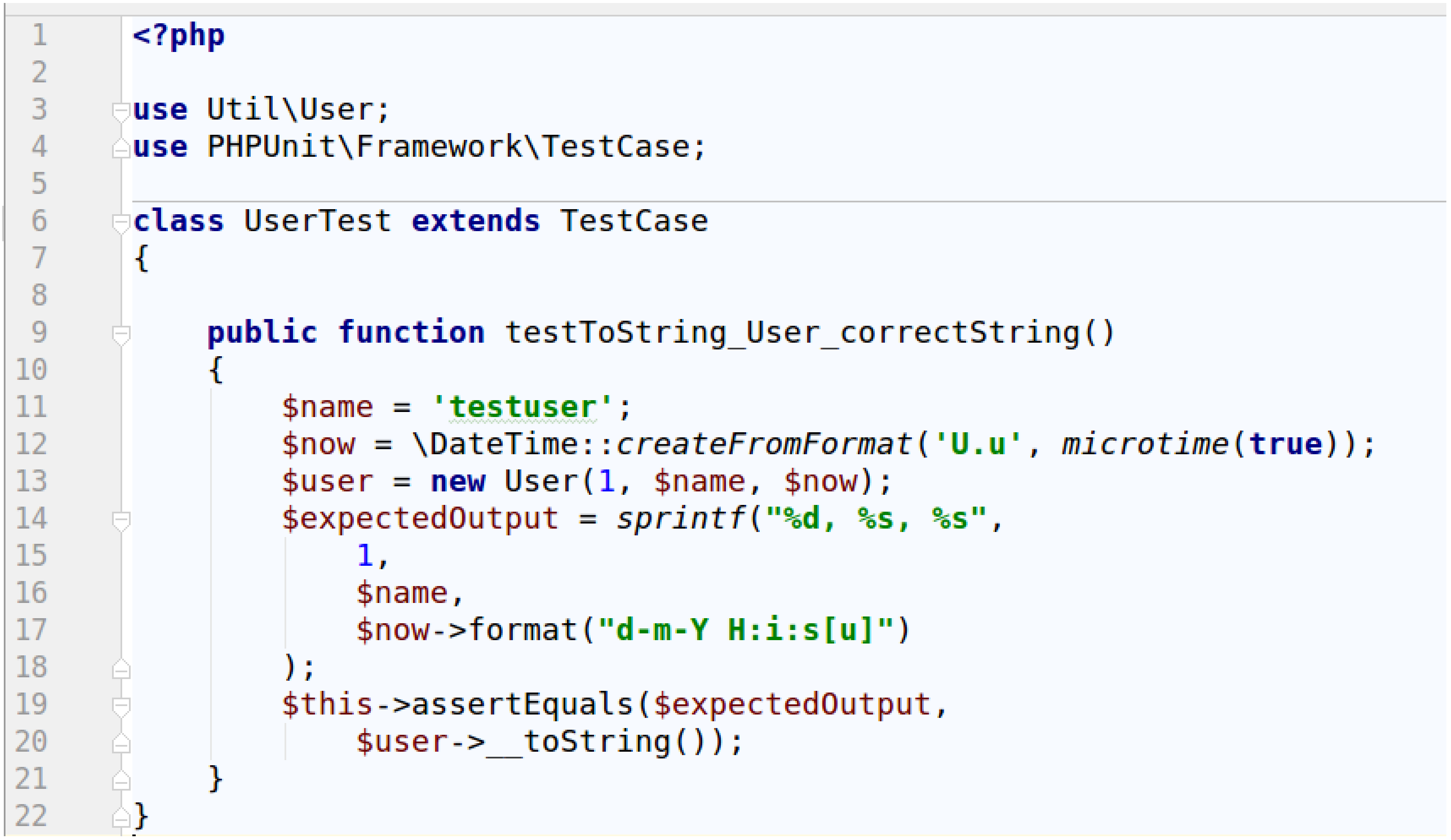
User staat zelf in voor de instantiatie van loginDate.
Hoe toString testen? t.o.v. wat de output vergelijken?

Testbaar



```php
<?php namespace Util;


class User
{
    private $id;
    private $name;
    private $loginDate;

    public function __construct($id, $name, \DateTime $loginDate)
    {
        $this->id = $id;
        $this->name = $name;
        $this->loginDate = $loginDate;
    }

    public function __toString()
    {
        return sprintf("%d, %s, %s",
            $this->id,
            $this->name,
            $this->loginDate->format("d-m-Y H:i:s[u]"));
    }
}
```

Testbaar



```php
<?php

use Util\User;
use PHPUnit\Framework\TestCase;

class UserTest extends TestCase
{

    public function testToString_User_correctString()
    {
        $name = 'testuser';
        $now = \DateTime::createFromFormat('U.u', microtime(true));
        $user = new User(1, $name, $now);
        $expectedOutput = sprintf("%d, %s, %s",
            1,
            $name,
            $now->format("d-m-Y H:i:s[u]")
        );
        $this->assertEquals($expectedOutput,
            $user->__toString());
    }
}
```

Codevoorbeeld

zie BB