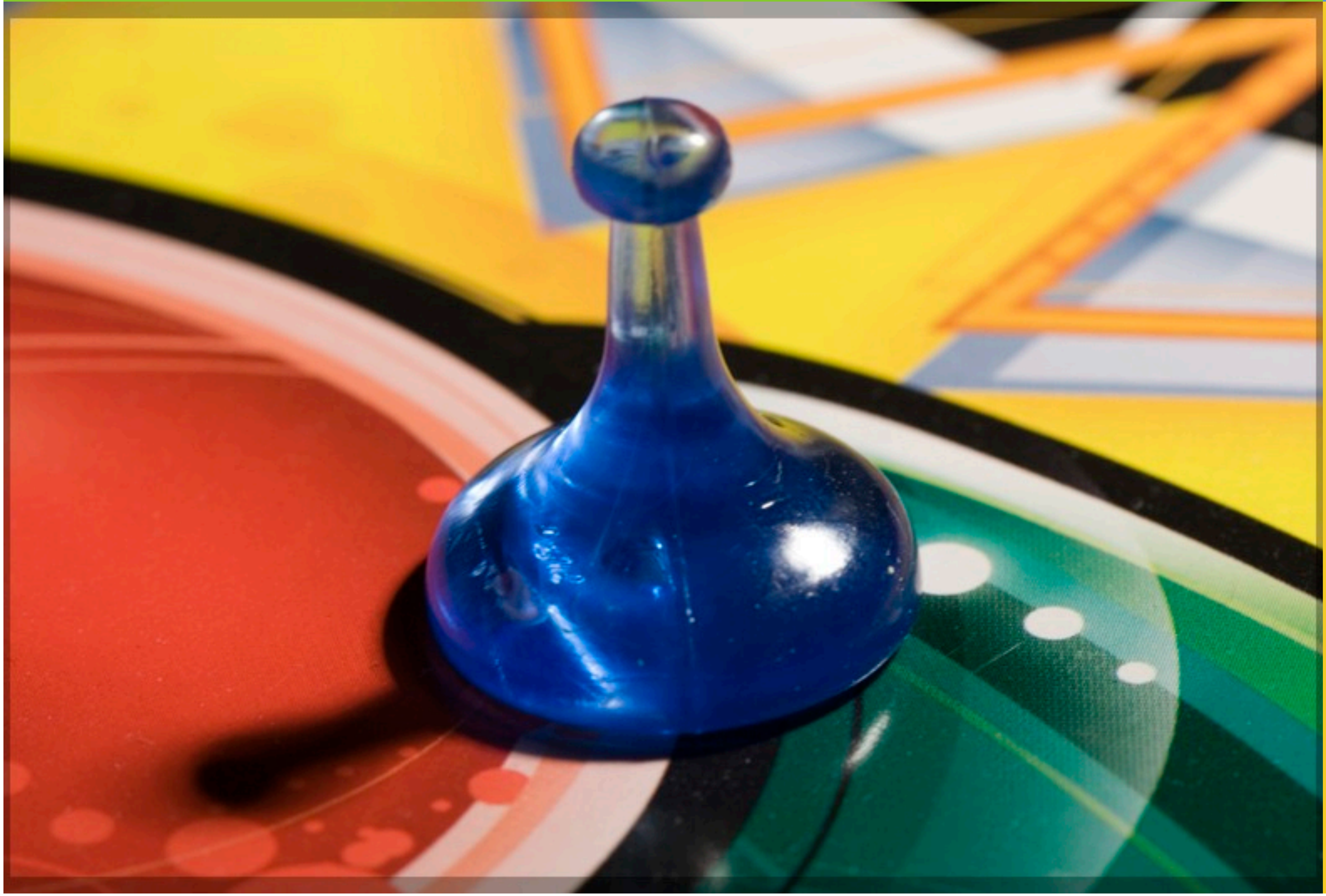
Developer Testing 201



Goals

- Learn advanced unit testing techniques
- Build a toolbox for integration testing
- Begin to be able to assess appropriate testing needs and usages

Non-Goals

- Not an introduction to testing
- Not to necessarily learn how to write the best (code) tests, but getting closer

Resources

Code Samples

https://github.com/elblinkin/DeveloperTesting201

PHPUnit Manual

http://www.phpunit.de/manual/current/en/

PHPUnit GitHub

https://github.com/sebastianbergmann/phpunit

Etsy PHPUnit Extensions

https://github.com/etsy/phpunit-extensions/wiki

Mockery

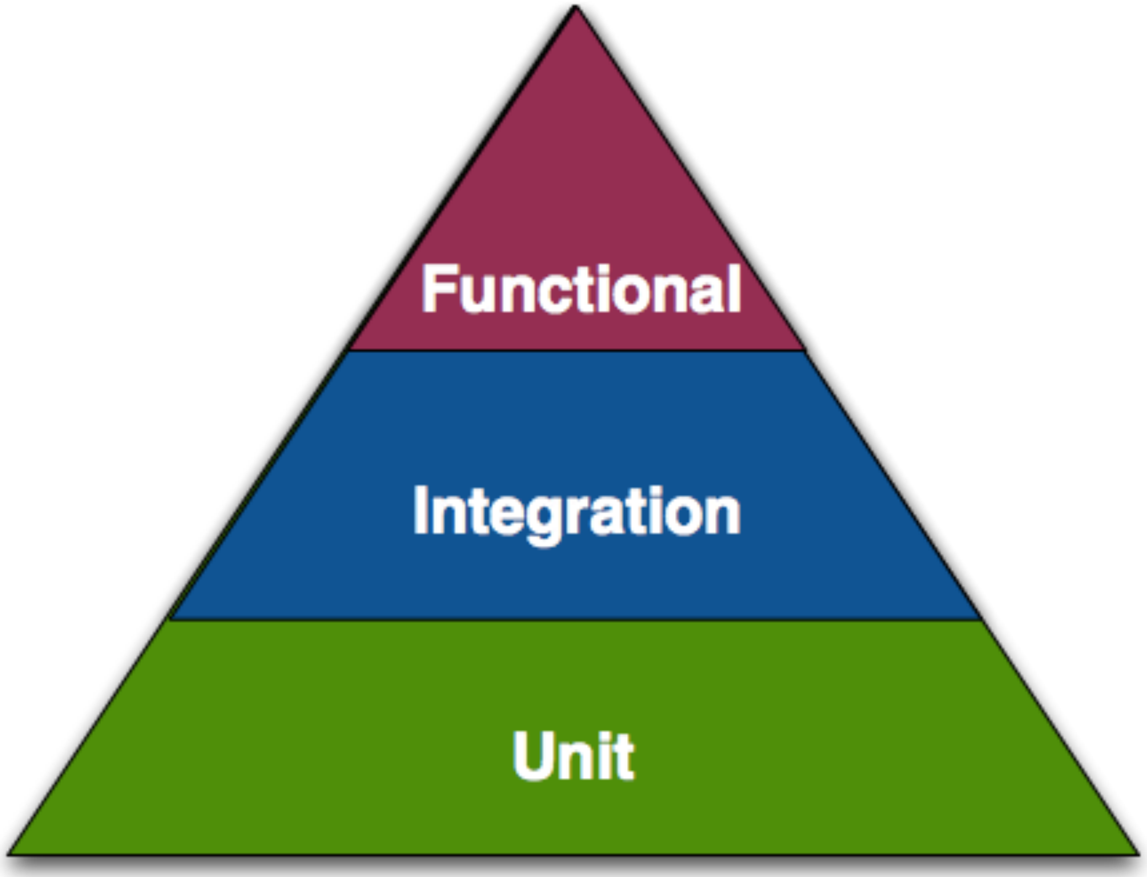
https://github.com/padraic/mockery

Topics

- Review Test Sizes
- Database Testing
- Custom Assertions
- Mock Objects
- Filesystem Testing
- Network Testing

Test Pyramid

Food Pyramid of the 90s



Test Types

Vocabulary Break!

Functional Tests

Does the overall product satisfy the the requirements?

Integration Tests

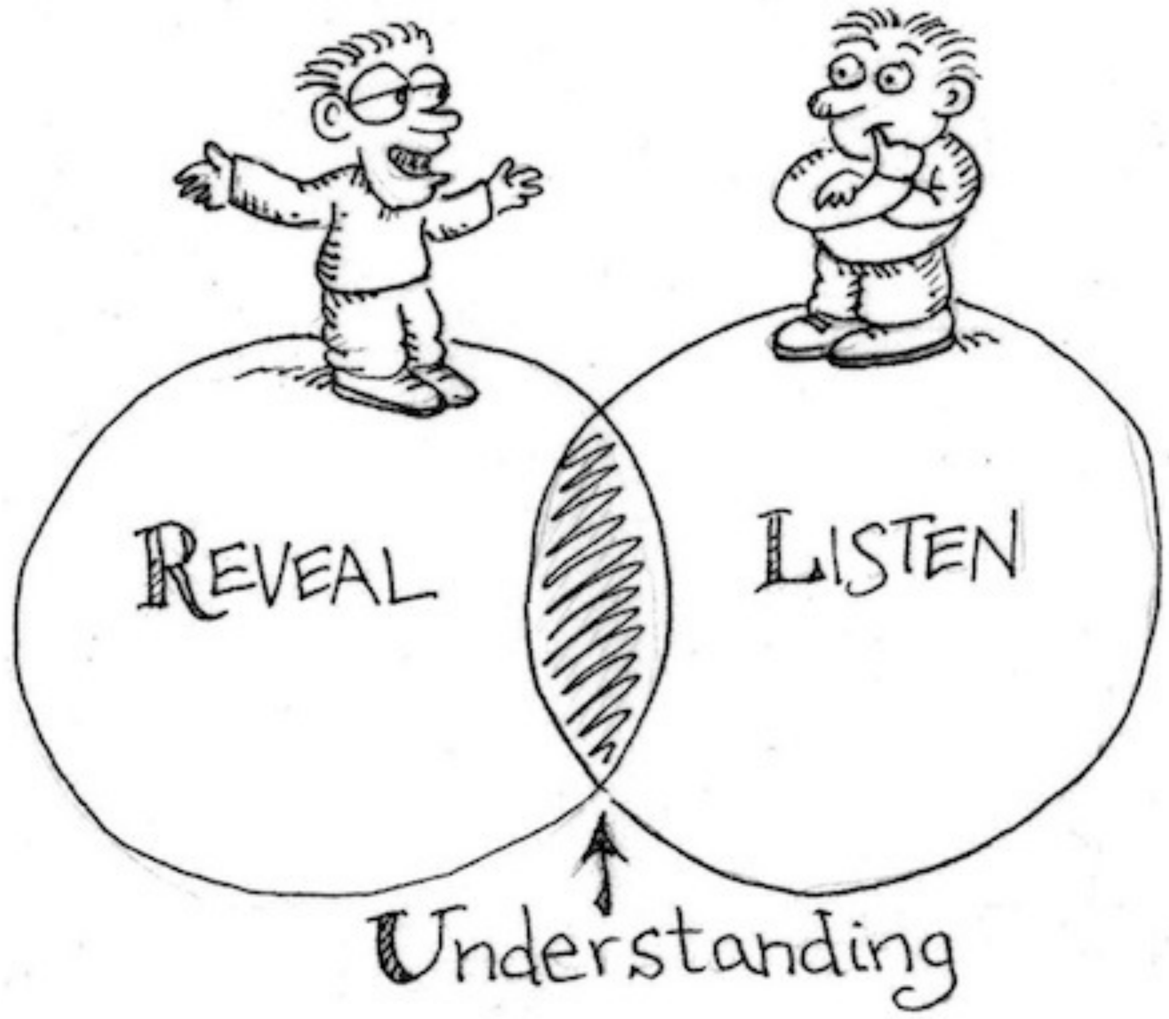
Do the pieces fit together?

Unit Tests

Is the logic correct in that function?

Functional

Understand the product.



The Tools

The Tools

- BeHat (Cucumber)

The Tools

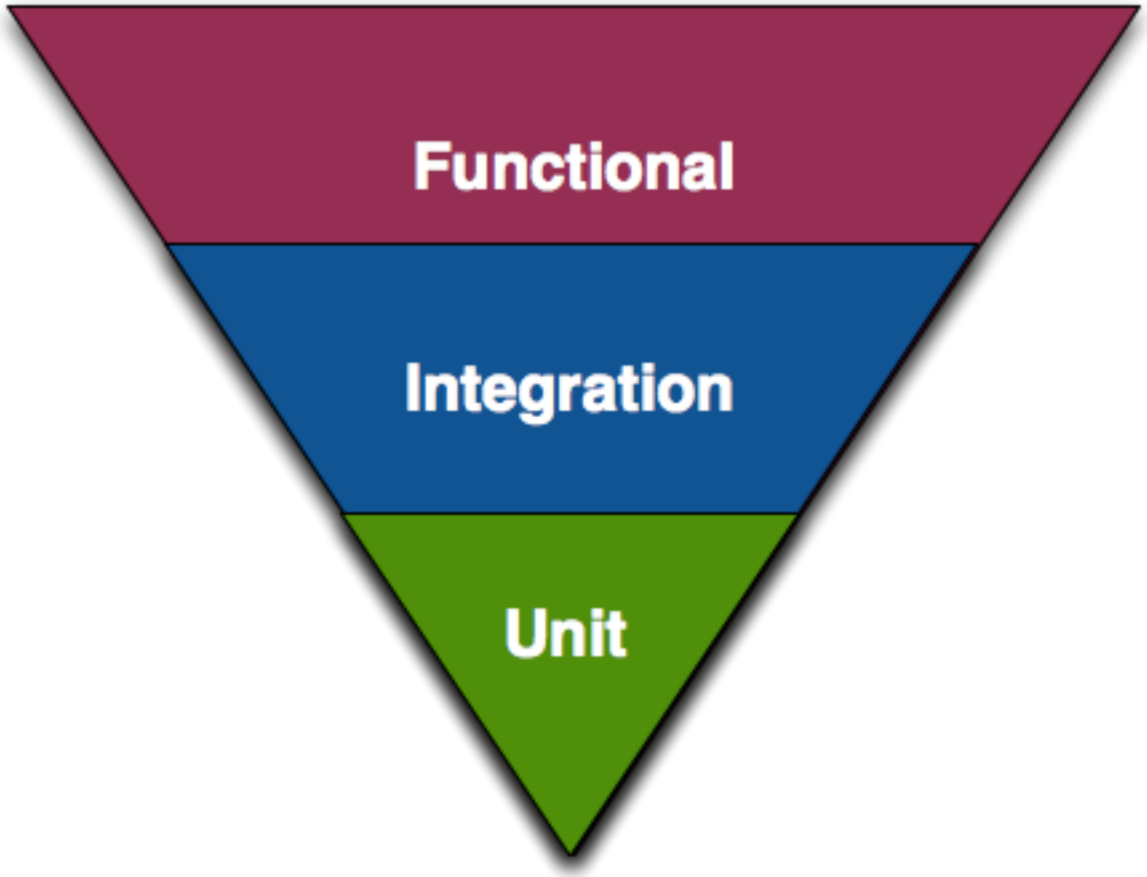
- BeHat (Cucumber)
- PHPSpec

The Tools

- BeHat (Cucumber)
- PHPSpec
- Keyboard, Mouse, and You

When Do They Work?

- During prototyping
- Focused on the product requirements
- Refactoring
- Regression of key features
- Better for smaller teams



After Greenfield

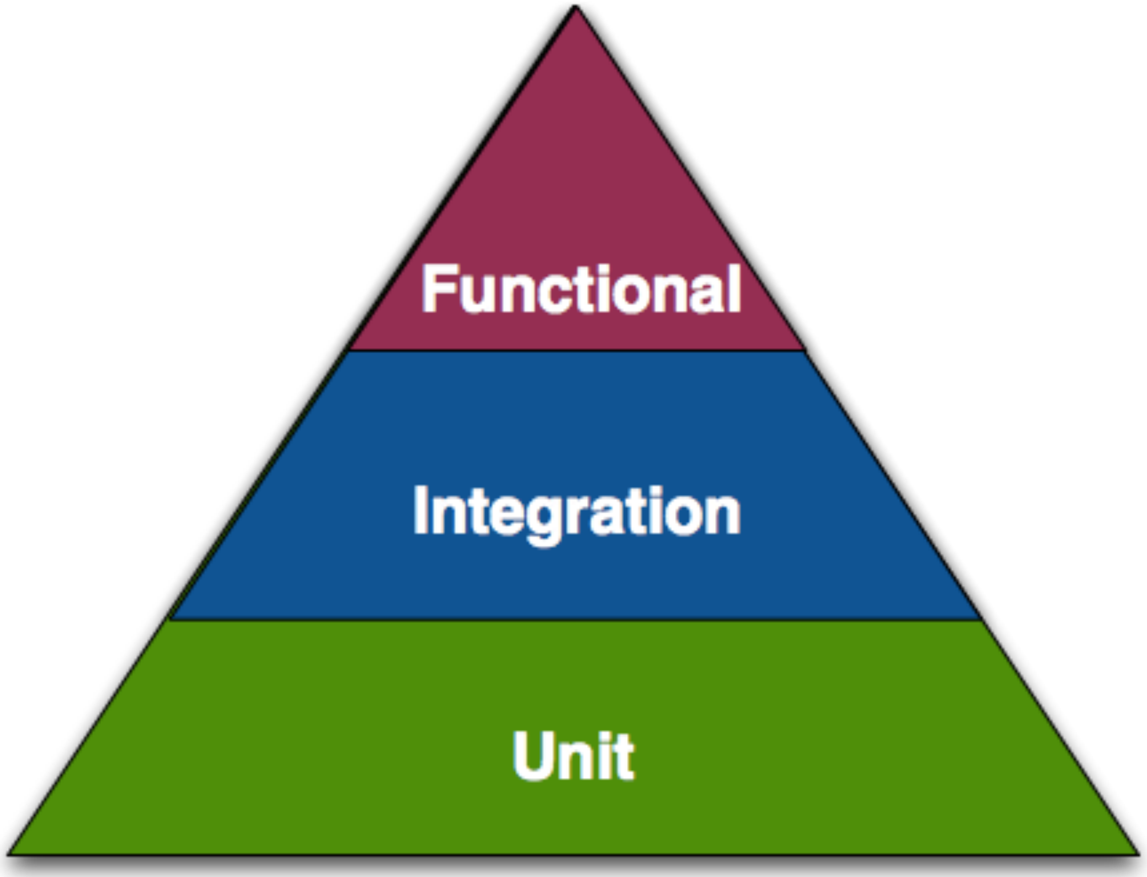
Inverted Test Pyramid

...Stop Helping?

- Focused on the implementation
- Rapidly changing functionality
- Large organizations

Test Pyramid

Food Pyramid of the 90s



Unit Tests

Simple and to the point.

Verify Correctness

- Line coverage
- Branch coverage
- Icky Bits-o-Logic

Gain Confidence

- Individual functions
- Variety of parameters
- Works for expected interactions with collaborators

Communication

- Show how to use the function
- Show expected interactions with other collaborators
- Increase discoverability of possible reuse

Tests should always be treated like every other consumer of the subject under test.

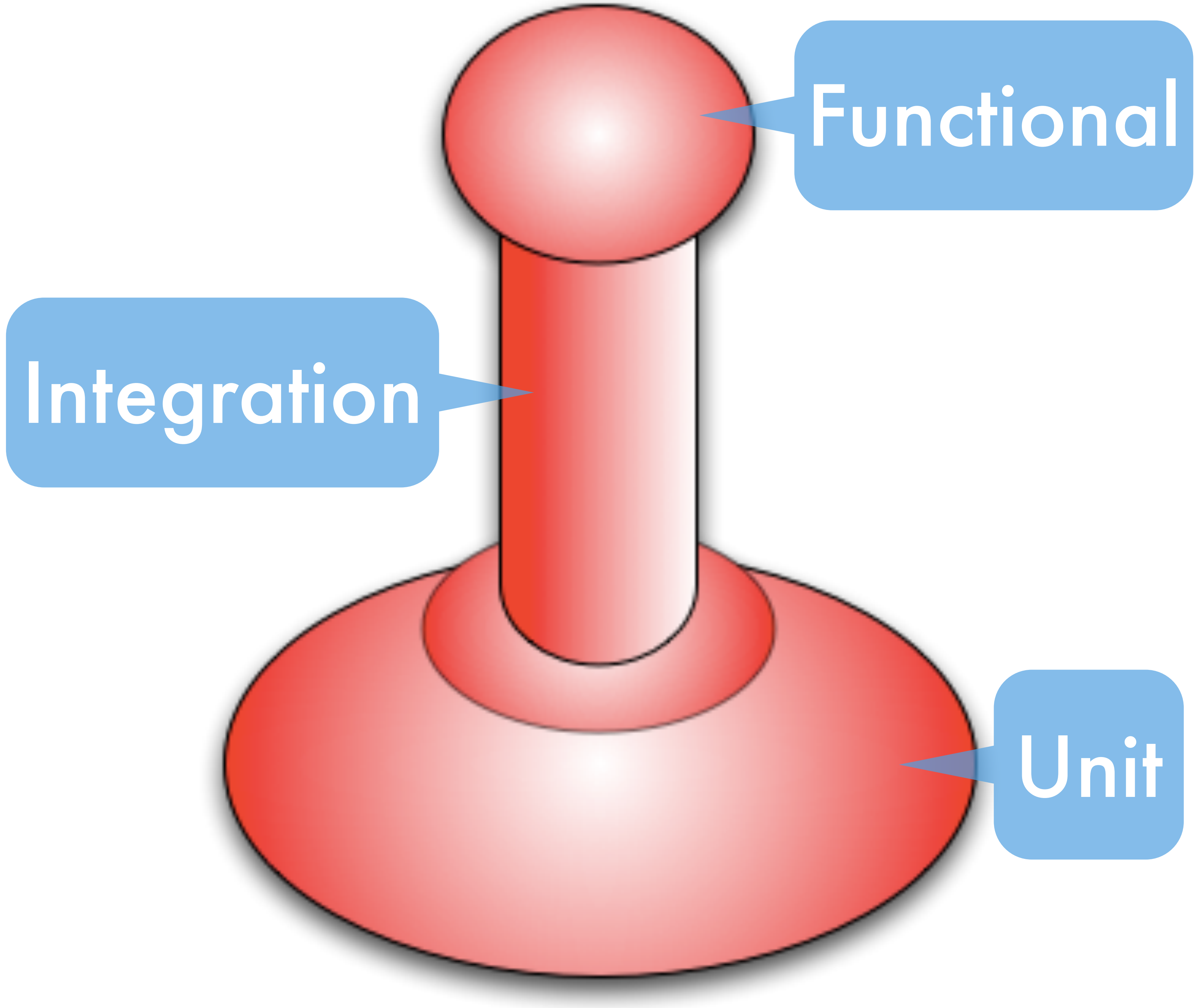
Integration

For everything in-between.

You're Paranoid

- Experimenting with third-party code or service
- You do not trust that your collaborators work as specified

Sorry! Piece of Testing



Test Classification

External Dependencies

- Caches, ie. Memcache
- Databases, ie. MySQL, Postgres, etc.
- 3rd Party Services
- Filesystem

More Sources of Flake

- Sleep
- Date / Time
- Random Number Generators
- Multiple-Processes

@group

- `@group cache`

  for test that use memcache

- `@group dbunit`

  for test that use DBUnit and databases

- `@group network`

  for tests that talk to external services

- `@group flaky`

  for tests that fail without a code change

Unit Tests are DEFAULT

There is **NO** `@group` for unit tests.

Test Sizes

- New in PHPUnit 3.6
- `@small` run in less than one second
- `@medium` run in less than 10 seconds
- `@large` run in less than 60 seconds
- Note: All times are configurable

Database Testing

DBUnit: Basic Steps

1. Set Up Fixture
   - `getDataSet()`
   - `getConnection()`
2. Exercise System Under Test
3. Verify Outcome
4. Tear Down

Supported Databases

- MySQL
- PostgreSQL
- Oracle
- SQLite
- IBM DB2 or Microsoft SQL Server
  - via Zend Framework or Doctrine 2

DataSet Options

- XML
- Flat XML
- MySQL XML
- YAML
- CSV
- Query (SQL)
- Database (DB)
- Array*

- Replacement
- Filter
- Composite

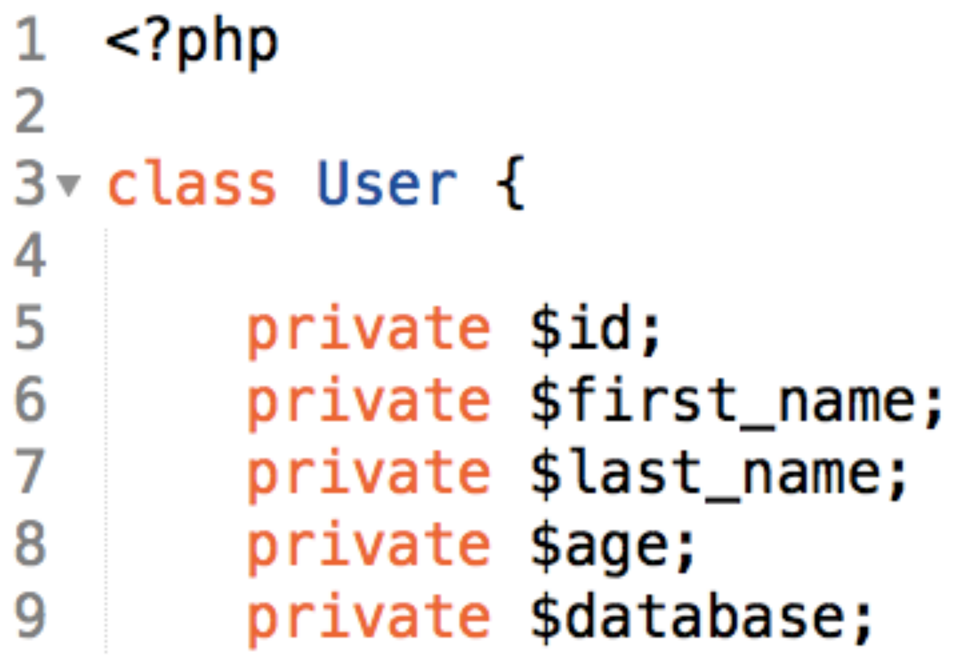
* from Etsy Extensions

Example Deep Dive

- `01 - Database`
  - `01 - Simple` (Read-Only DBUnit Test)
  - `02 - TestCase` (Extract Custom)
  - `03 - Modify` (Modifying Existing)
  - `04 - Create` (Adding Data)
  - `05 - DRYing` (Reducing Integration Points)

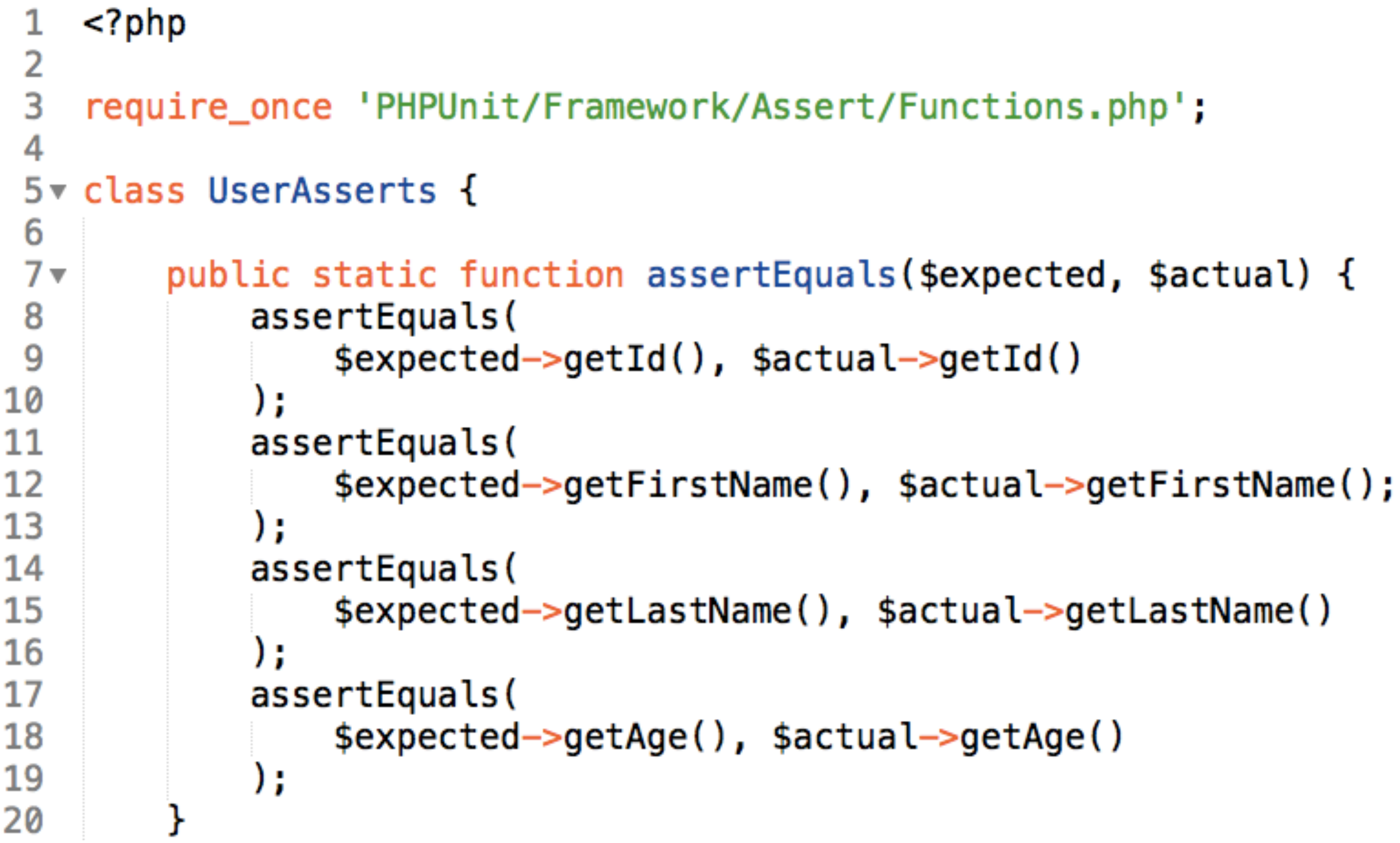
Custom Assertions

Custom Assert



```php
<?php

class User {

    private $id;
    private $first_name;
    private $last_name;
    private $age;
    private $database;
```

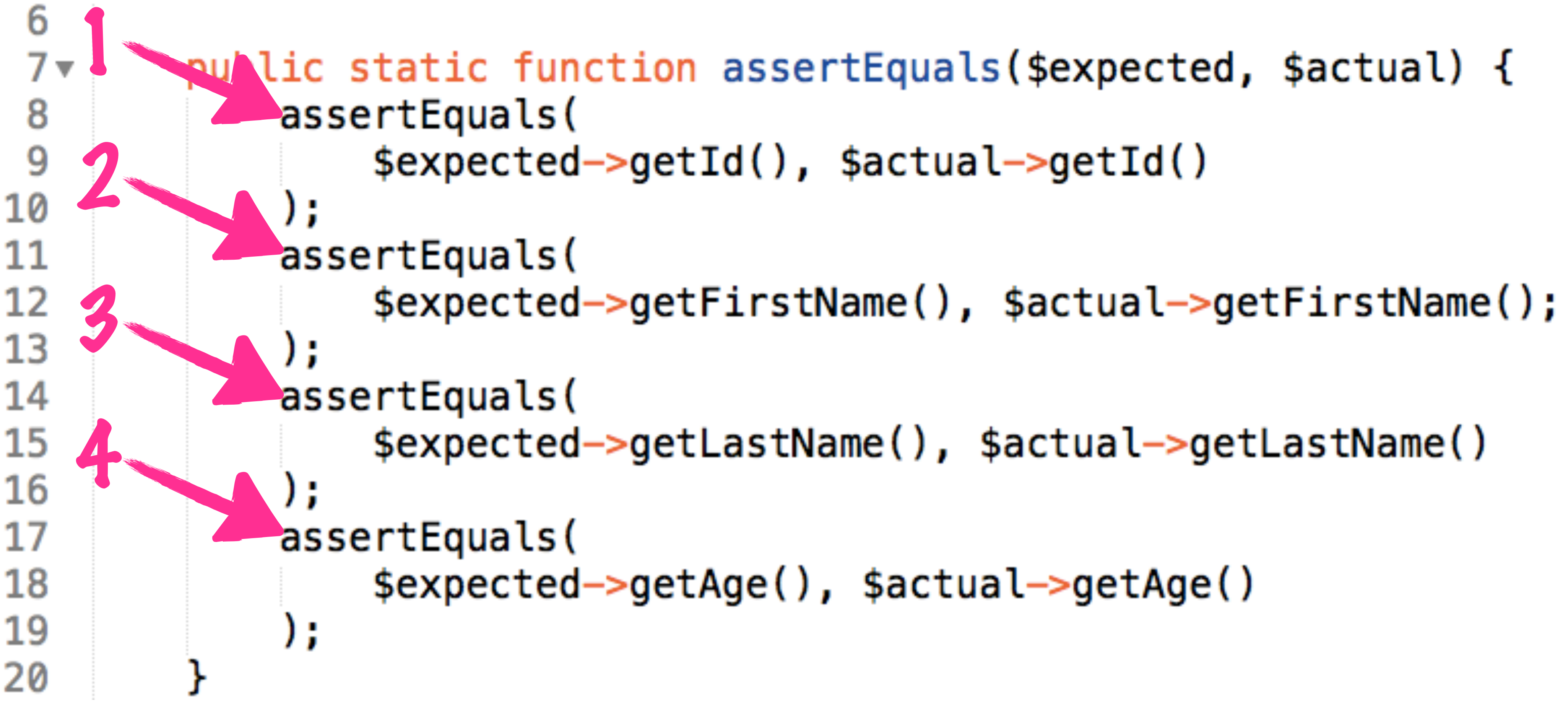
Custom Assert: Lv. 1



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual) {
        assertEquals(
            $expected->getId(), $actual->getId()
        );
        assertEquals(
            $expected->getFirstName(), $actual->getFirstName();
        );
        assertEquals(
            $expected->getLastName(), $actual->getLastName()
        );
        assertEquals(
            $expected->getAge(), $actual->getAge()
        );
    }
```

Custom Assert: Lv. 1

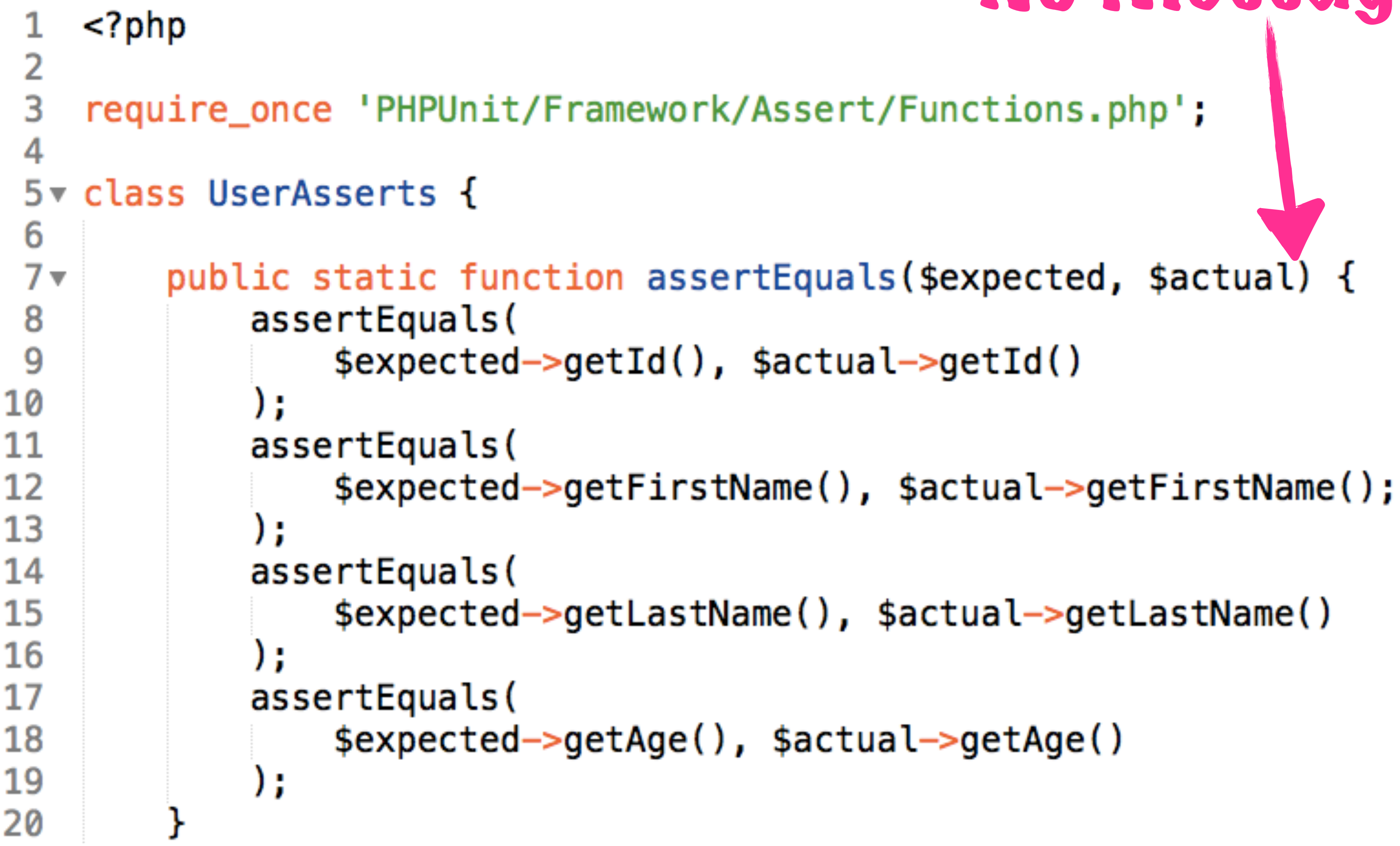
Multiple Asserts!!



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual) {
        assertEquals(
            $expected->getId(), $actual->getId()
        );
        assertEquals(
            $expected->getFirstName(), $actual->getFirstName();
        );
        assertEquals(
            $expected->getLastName(), $actual->getLastName()
        );
        assertEquals(
            $expected->getAge(), $actual->getAge()
        );
    }
```

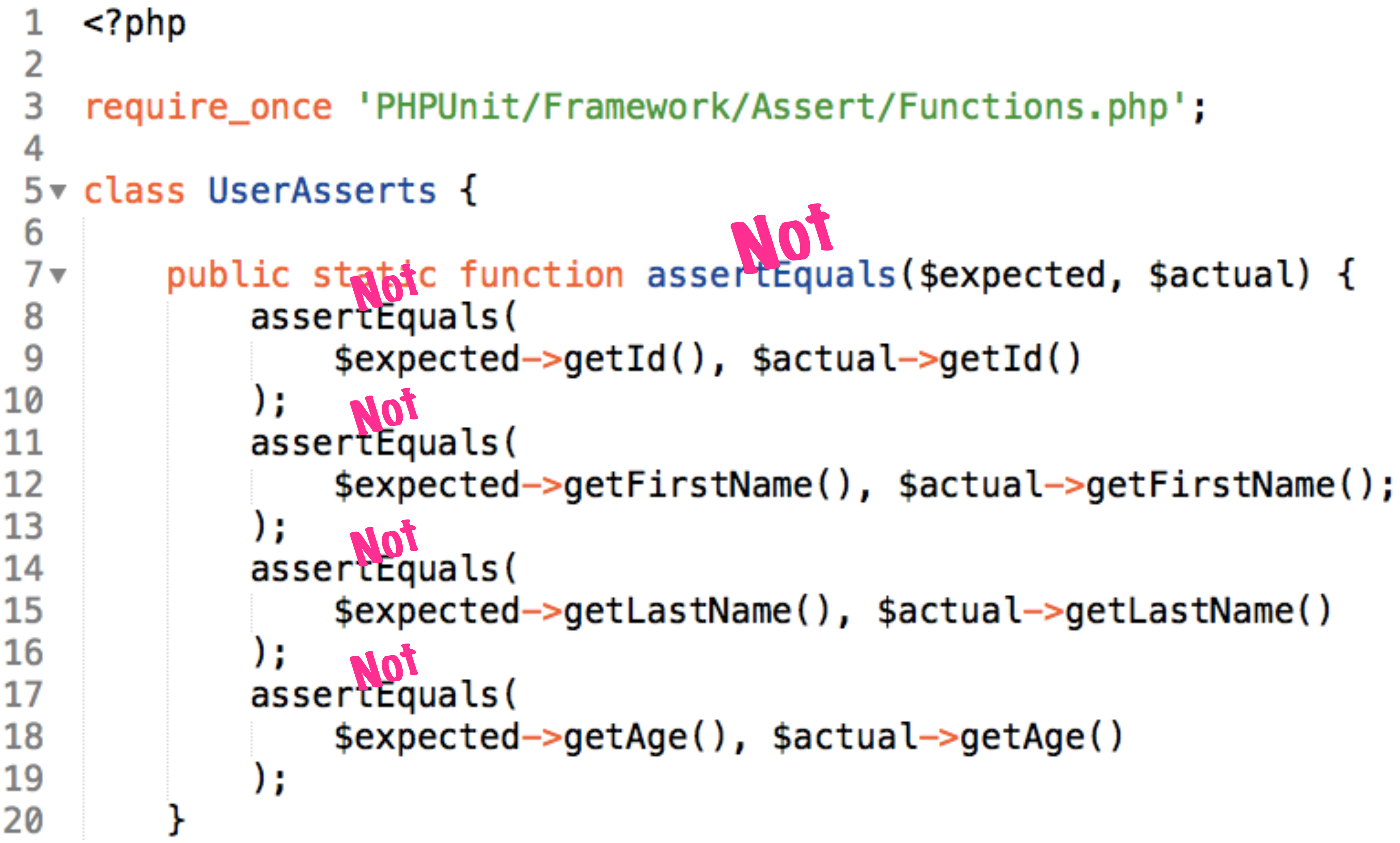
Custom Assert: Lv. 1

No Message!!



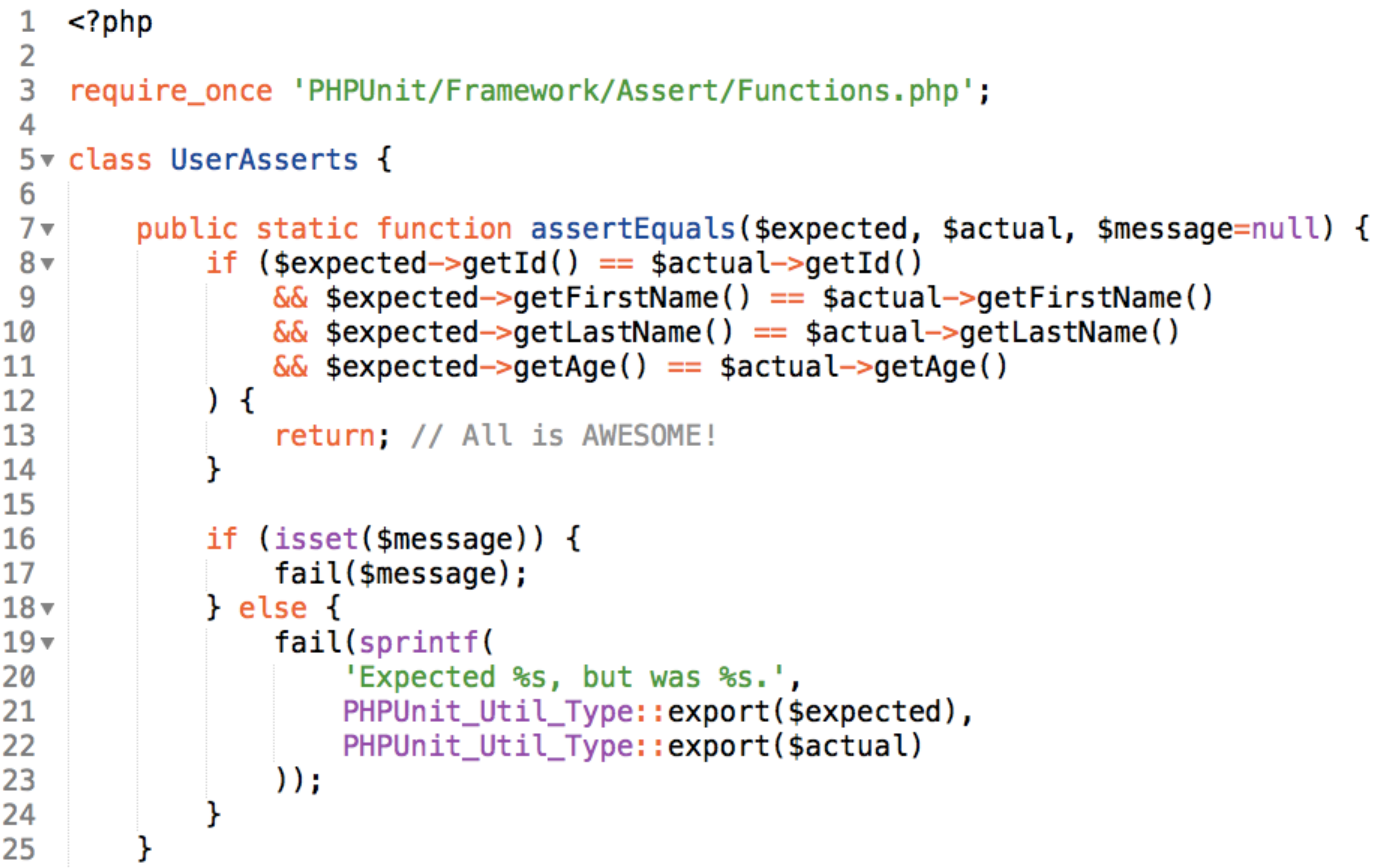
```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual) {
        assertEquals(
            $expected->getId(), $actual->getId()
        );
        assertEquals(
            $expected->getFirstName(), $actual->getFirstName();
        );
        assertEquals(
            $expected->getLastName(), $actual->getLastName()
        );
        assertEquals(
            $expected->getAge(), $actual->getAge()
        );
    }
```

Custom Assert: Lv. 1



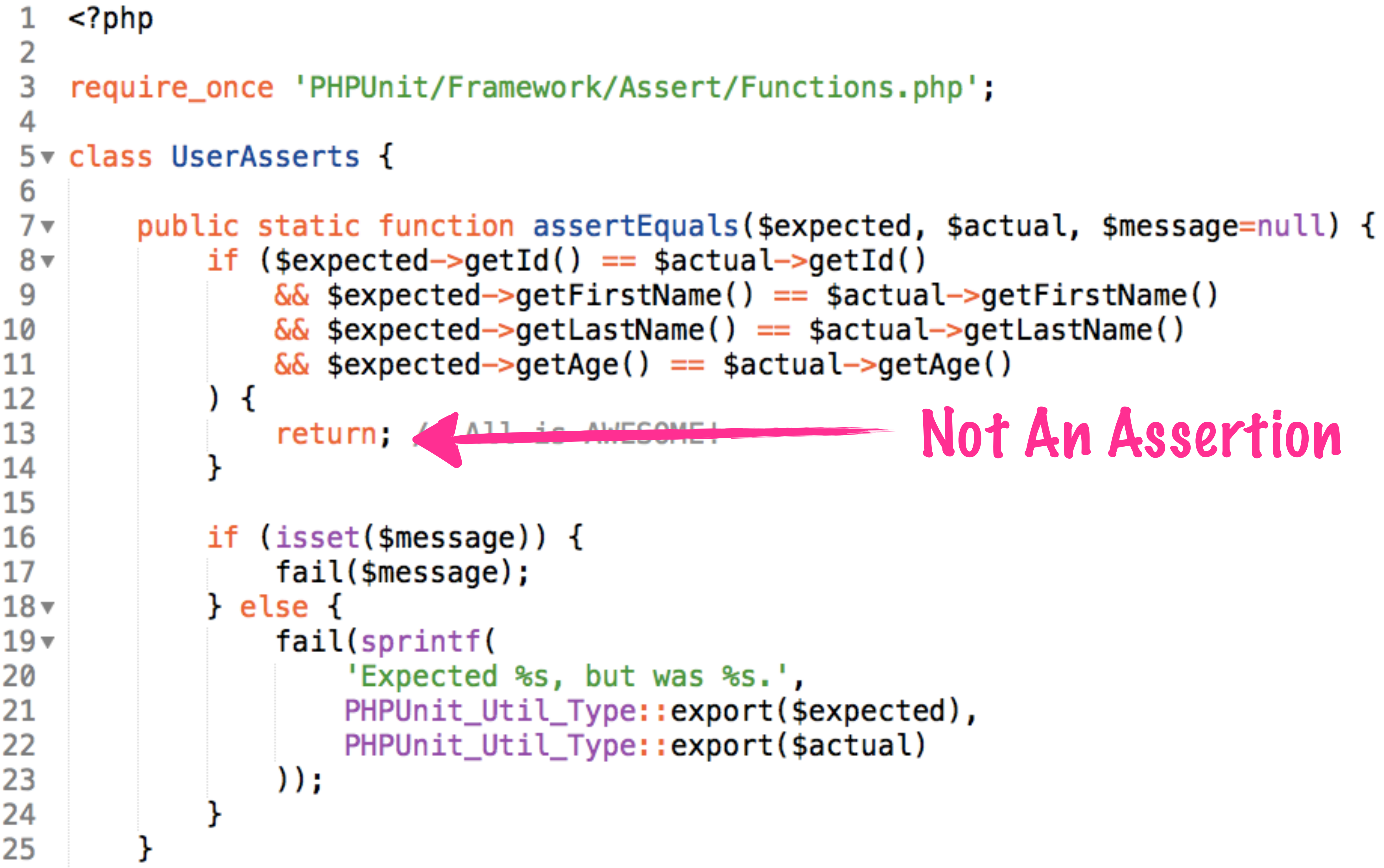
```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual) {
        assertEquals(
            $expected->getId(), $actual->getId()
        );
        assertEquals(
            $expected->getFirstName(), $actual->getFirstName();
        );
        assertEquals(
            $expected->getLastName(), $actual->getLastName()
        );
        assertEquals(
            $expected->getAge(), $actual->getAge()
        );
    }
```

Custom Assert: Lv. 2



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        if ($expected->getId() == $actual->getId()
            && $expected->getFirstName() == $actual->getFirstName()
            && $expected->getLastName() == $actual->getLastName()
            && $expected->getAge() == $actual->getAge()
        ) {
            return; // All is AWESOME!
        }

        if (isset($message)) {
            fail($message);
        } else {
            fail(sprintf(
                'Expected %s, but was %s.',
                PHPUnit_Util_Type::export($expected),
                PHPUnit_Util_Type::export($actual)
            ));
        }
    }
```

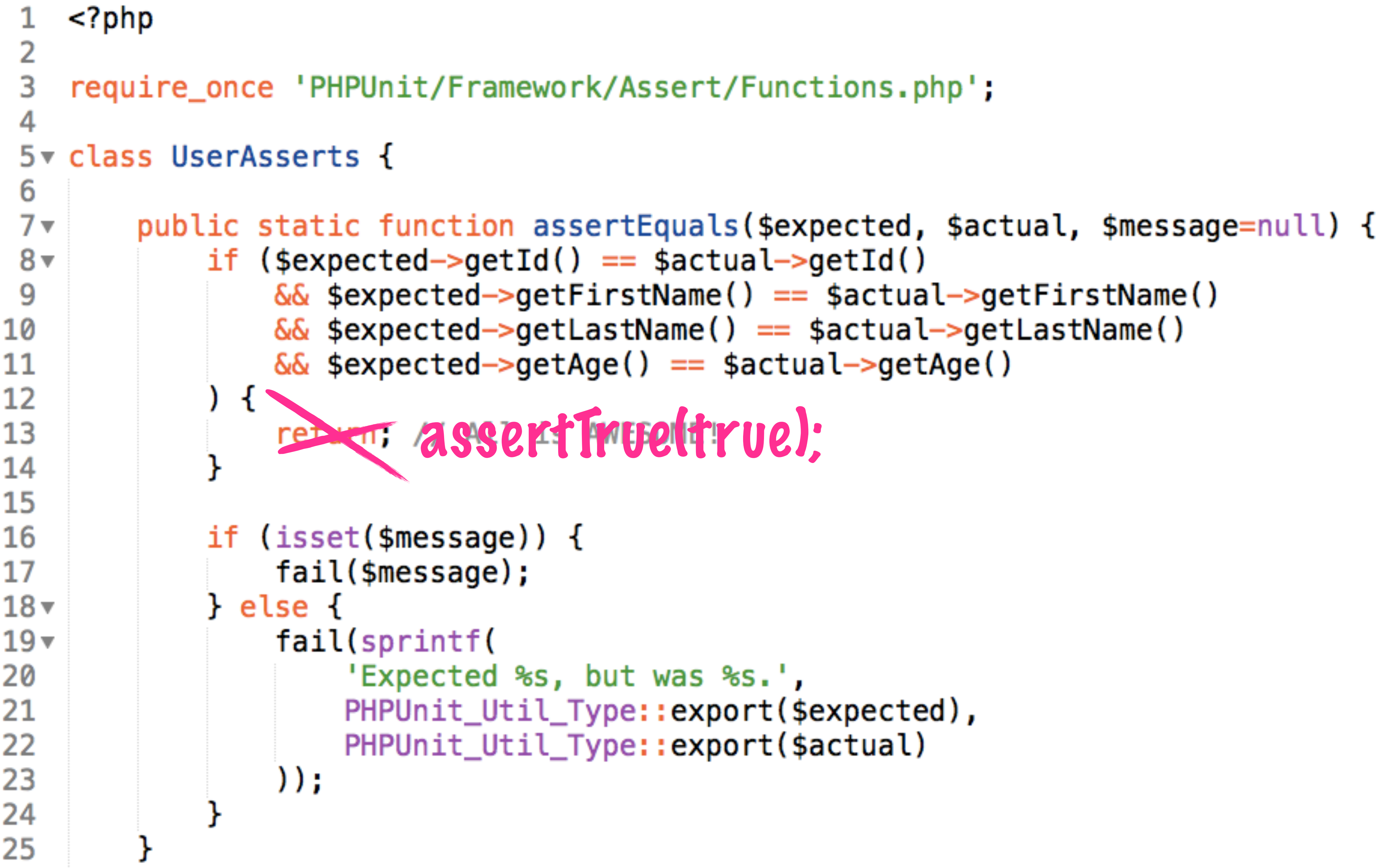
Custom Assert: Lv. 2



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        if ($expected->getId() == $actual->getId()
            && $expected->getFirstName() == $actual->getFirstName()
            && $expected->getLastName() == $actual->getLastName()
            && $expected->getAge() == $actual->getAge()
        ) {
            return; // All is AWESOME!
        }

        if (isset($message)) {
            fail($message);
        } else {
            fail(sprintf(
                'Expected %s, but was %s.',
                PHPUnit_Util_Type::export($expected),
                PHPUnit_Util_Type::export($actual)
            ));
        }
    }
```

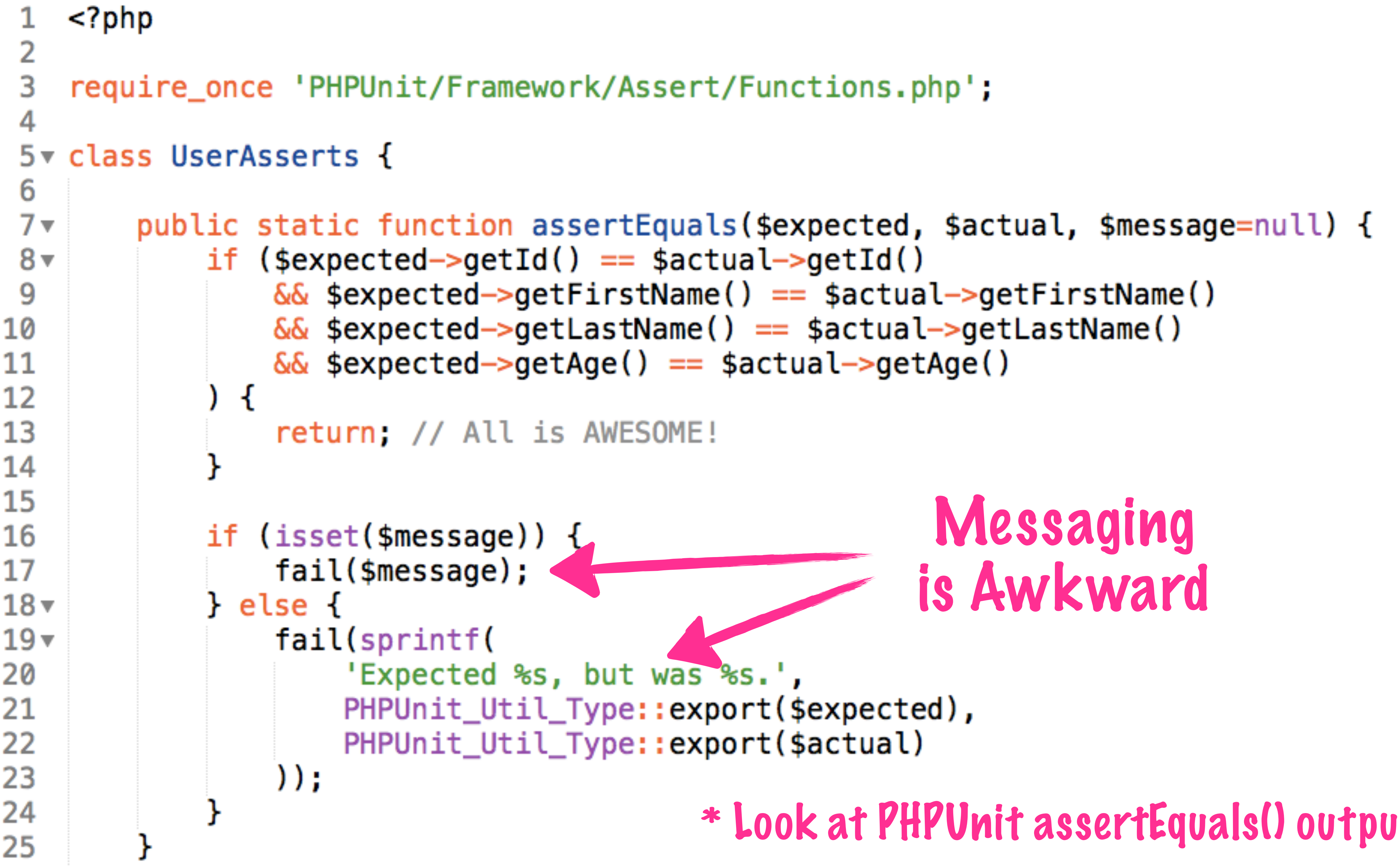
Not An Assertion

Custom Assert: Lv. 2



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        if ($expected->getId() == $actual->getId()
            && $expected->getFirstName() == $actual->getFirstName()
            && $expected->getLastName() == $actual->getLastName()
            && $expected->getAge() == $actual->getAge()
        ) {
            ~~return;~~ // ALL IS AWESOME! assertTrue(true);
        }

        if (isset($message)) {
            fail($message);
        } else {
            fail(sprintf(
                'Expected %s, but was %s.',
                PHPUnit_Util_Type::export($expected),
                PHPUnit_Util_Type::export($actual)
            ));
        }
    }
```

Custom Assert: Lv. 2

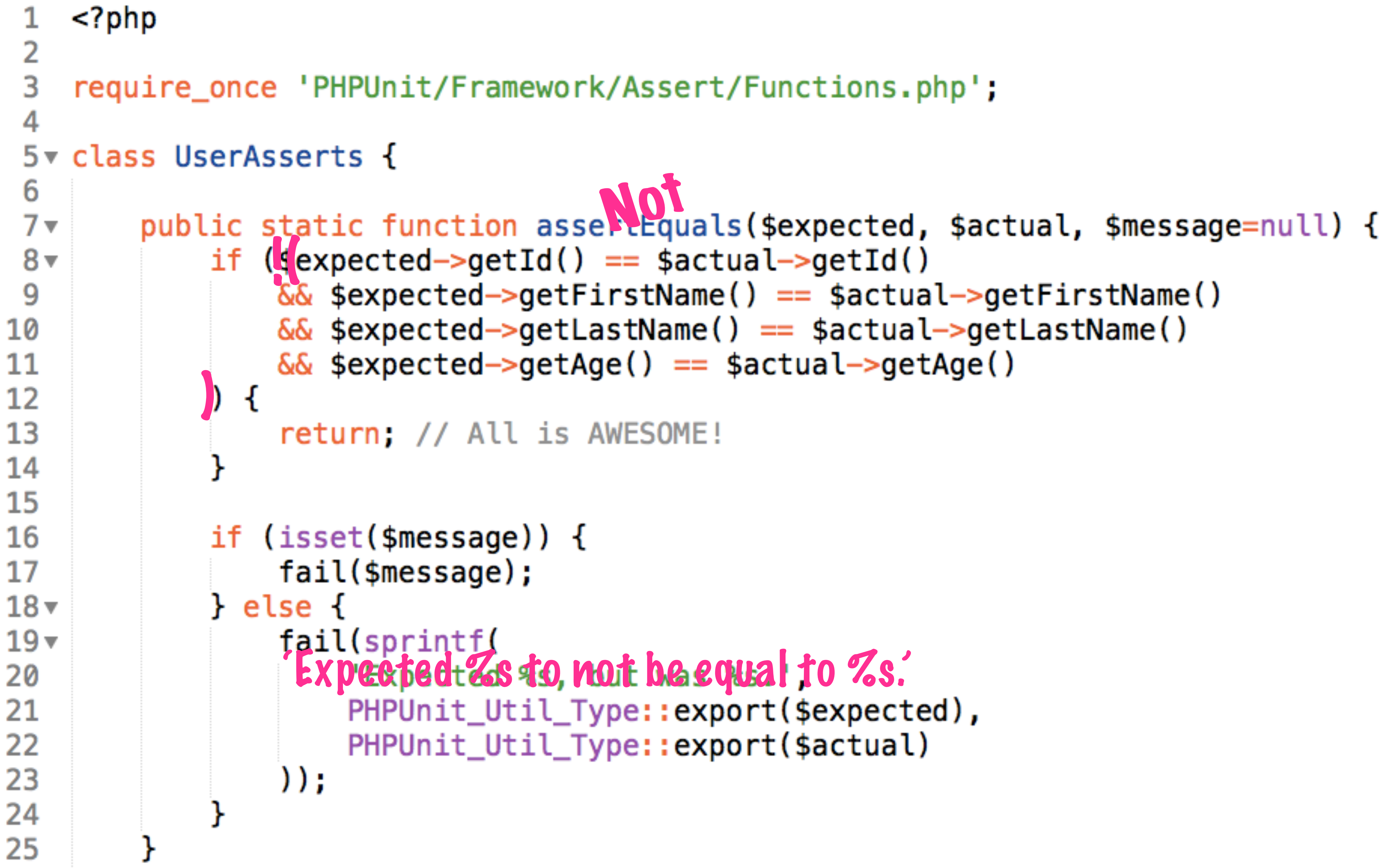


```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        if ($expected->getId() == $actual->getId()
            && $expected->getFirstName() == $actual->getFirstName()
            && $expected->getLastName() == $actual->getLastName()
            && $expected->getAge() == $actual->getAge()
        ) {
            return; // All is AWESOME!
        }

        if (isset($message)) {
            fail($message);
        } else {
            fail(sprintf(
                'Expected %s, but was %s.',
                PHPUnit_Util_Type::export($expected),
                PHPUnit_Util_Type::export($actual)
            ));
        }
    }
```

Messaging is Awkward

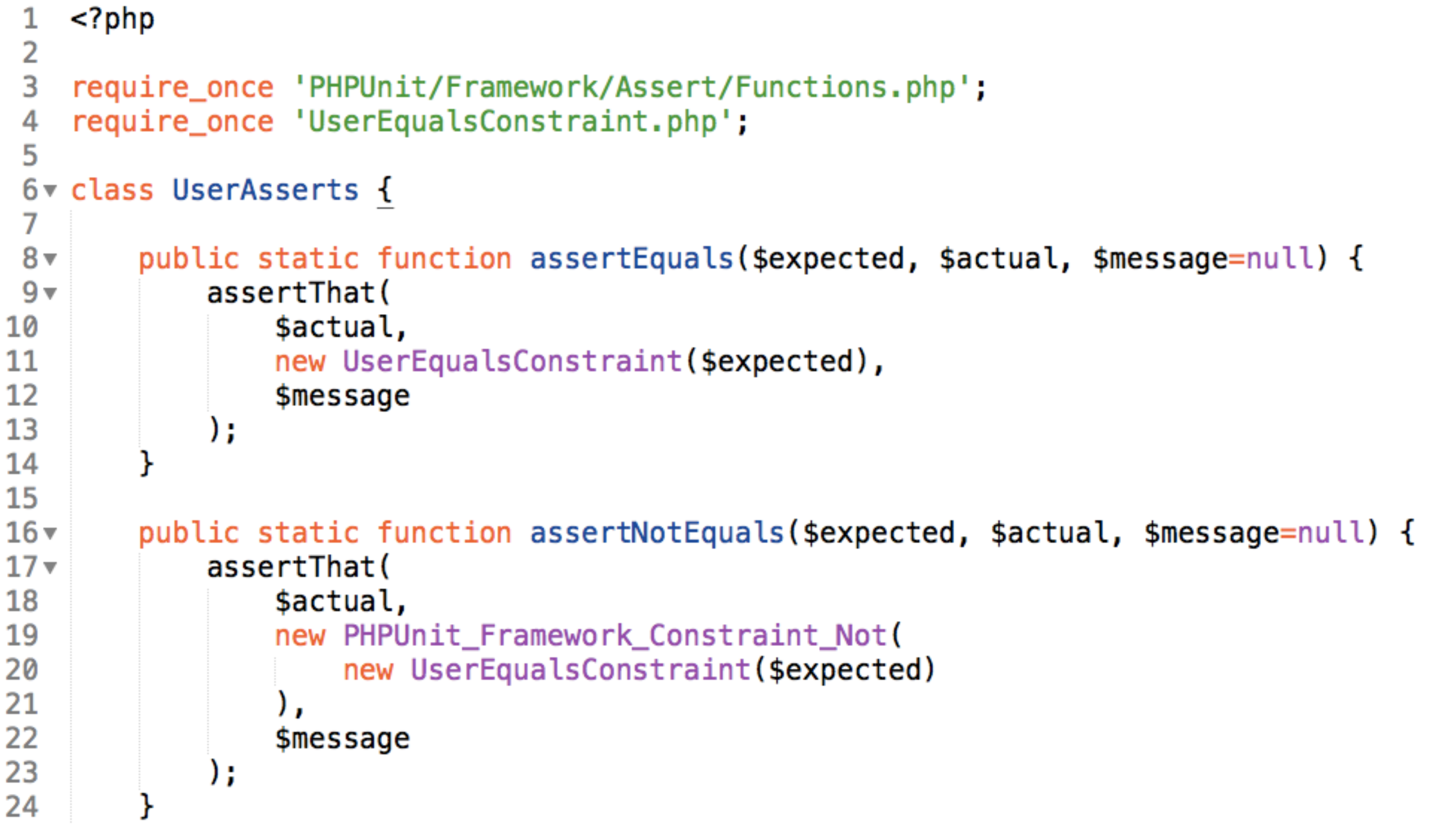
* Look at PHPUnit assertEquals() output.

Custom Assert: Lv. 2



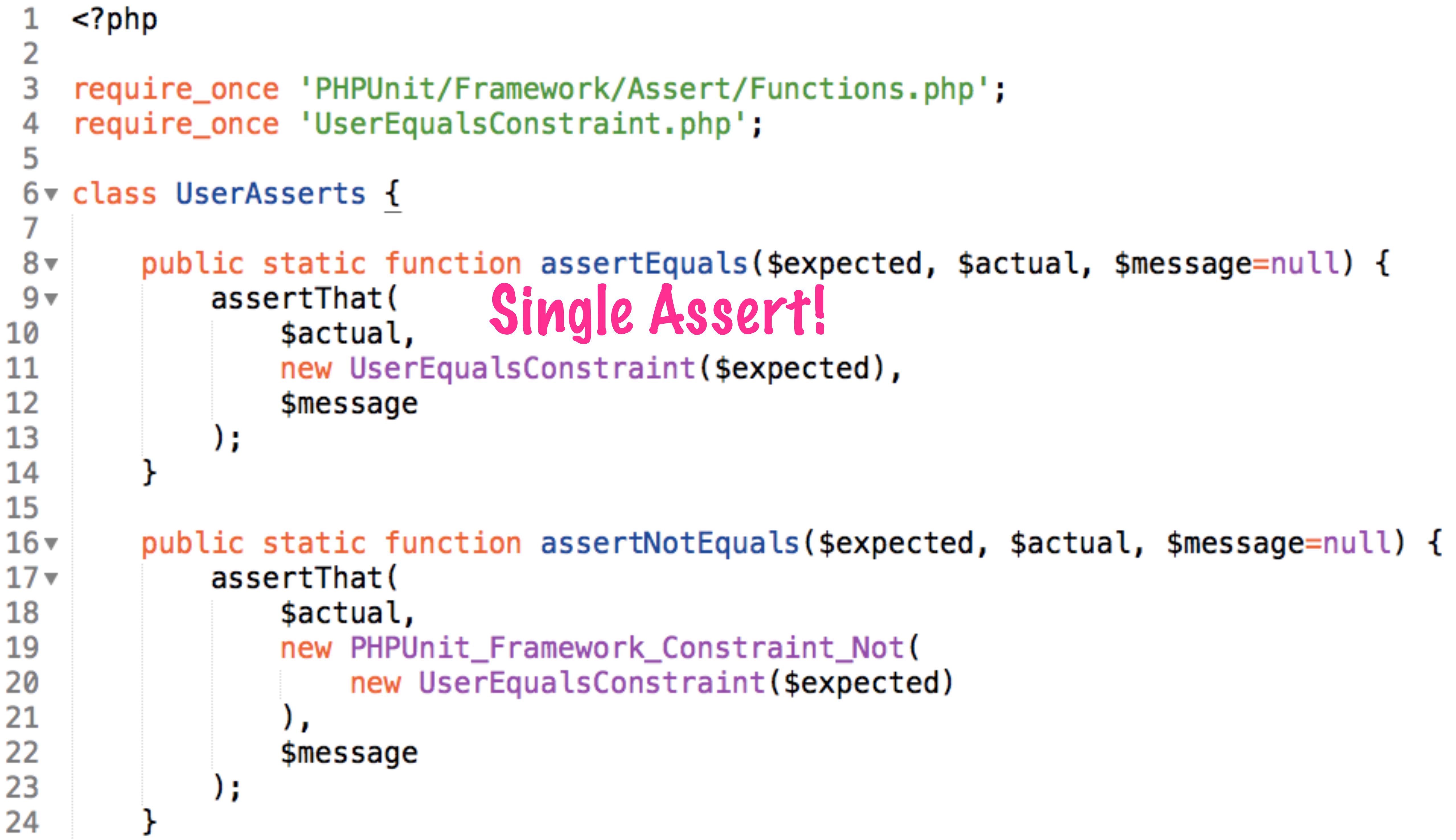
```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        if ($expected->getId() == $actual->getId()
            && $expected->getFirstName() == $actual->getFirstName()
            && $expected->getLastName() == $actual->getLastName()
            && $expected->getAge() == $actual->getAge()
        ) {
            return; // All is AWESOME!
        }

        if (isset($message)) {
            fail($message);
        } else {
            fail(sprintf(
                'Expected %s, but was %s!',
                PHPUnit_Util_Type::export($expected),
                PHPUnit_Util_Type::export($actual)
            ));
        }
    }
```

Custom Assert: Lv. 3



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';
require_once 'UserEqualsConstraint.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new UserEqualsConstraint($expected),
            $message
        );
    }

    public static function assertNotEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new PHPUnit_Framework_Constraint_Not(
                new UserEqualsConstraint($expected)
            ),
            $message
        );
    }
```

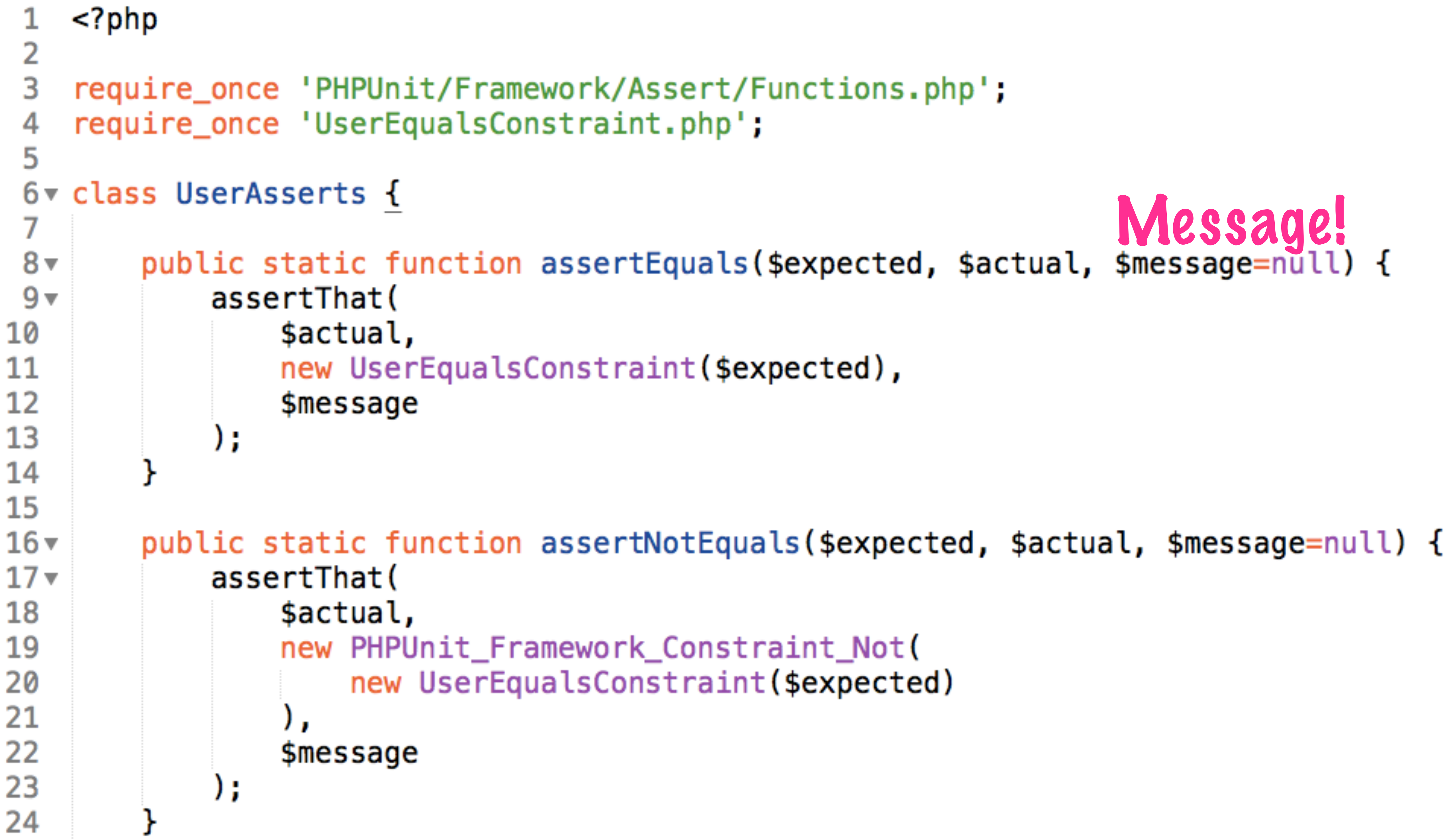
Custom Assert: Lv. 3



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';
require_once 'UserEqualsConstraint.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new UserEqualsConstraint($expected),
            $message
        );
    }

    public static function assertNotEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new PHPUnit_Framework_Constraint_Not(
                new UserEqualsConstraint($expected)
            ),
            $message
        );
    }
```

Single Assert!

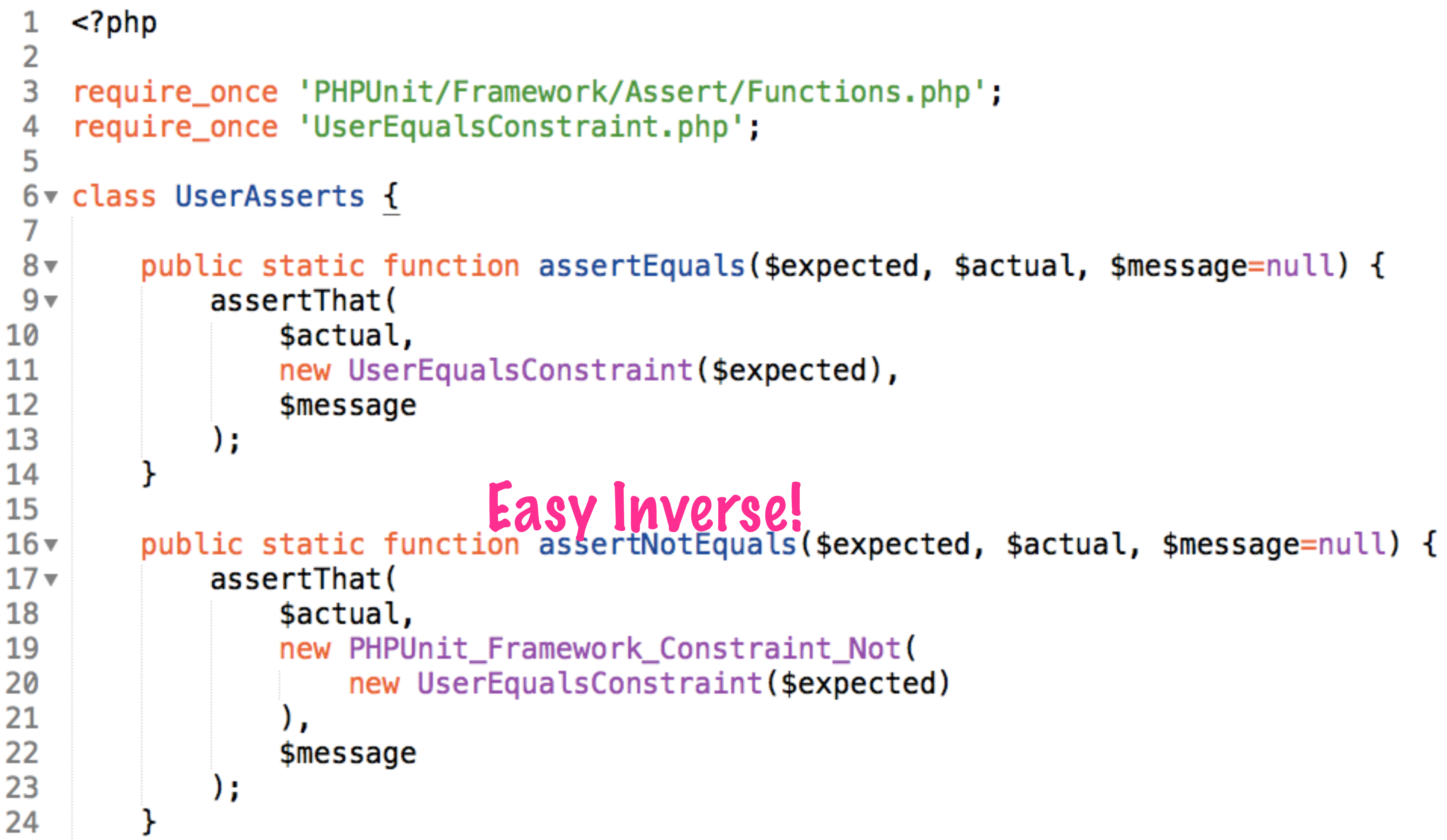
Custom Assert: Lv. 3



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';
require_once 'UserEqualsConstraint.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new UserEqualsConstraint($expected),
            $message
        );
    }

    public static function assertNotEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new PHPUnit_Framework_Constraint_Not(
                new UserEqualsConstraint($expected)
            ),
            $message
        );
    }
```

Message!

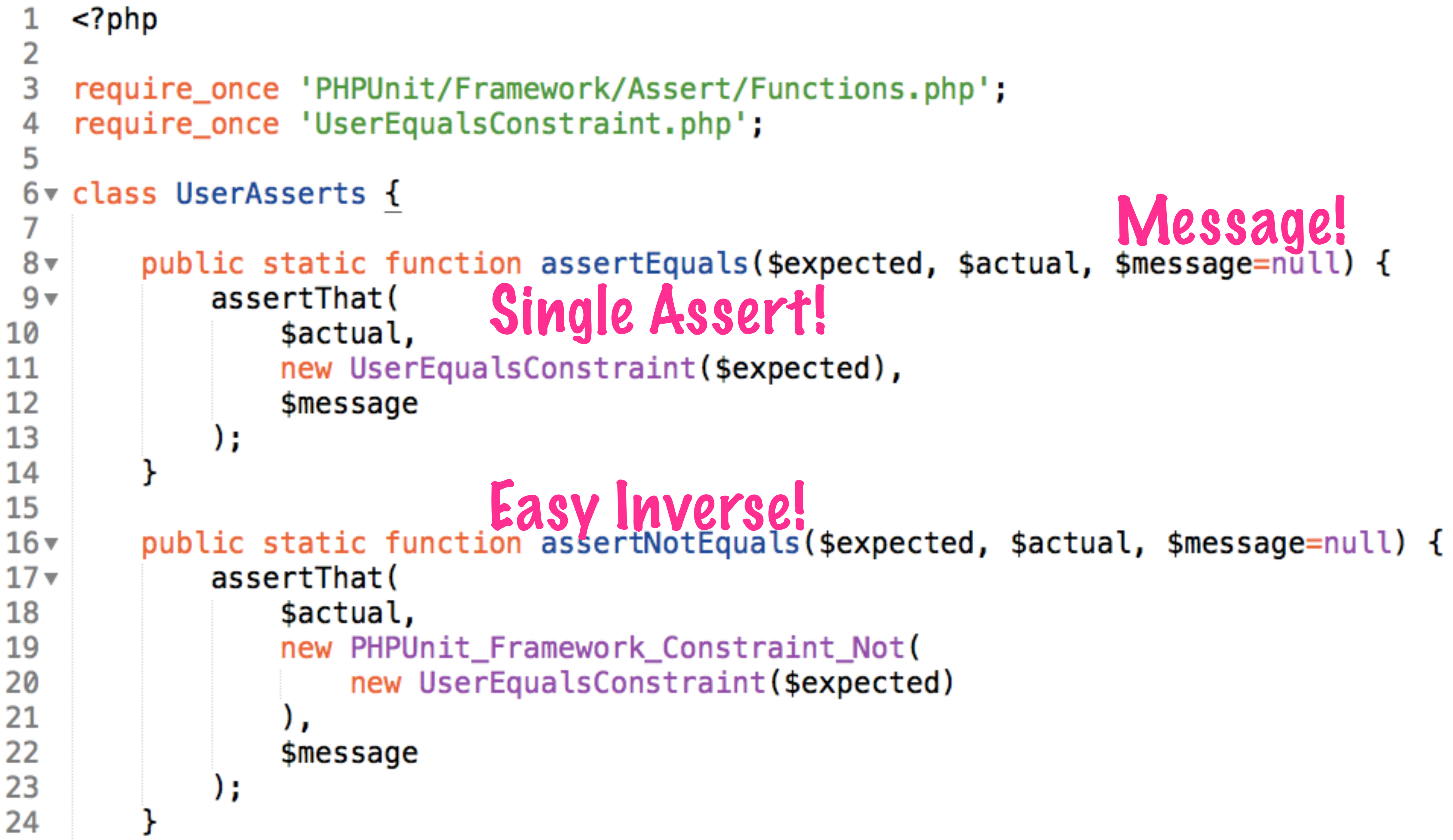
Custom Assert: Lv. 3



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';
require_once 'UserEqualsConstraint.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new UserEqualsConstraint($expected),
            $message
        );
    }

    public static function assertNotEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new PHPUnit_Framework_Constraint_Not(
                new UserEqualsConstraint($expected)
            ),
            $message
        );
    }
```

Easy Inverse!

Custom Assert: Lv. 3



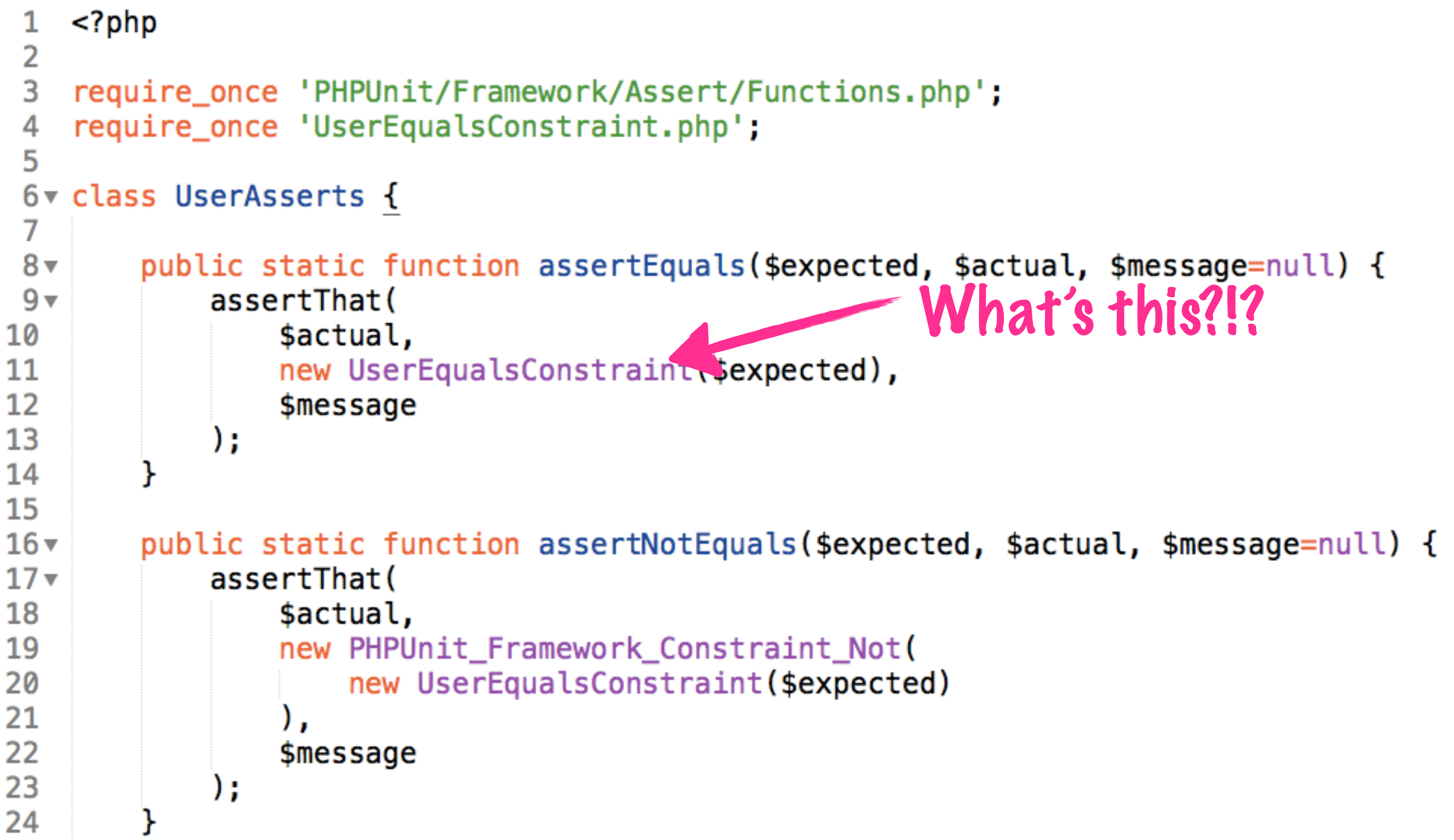
```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';
require_once 'UserEqualsConstraint.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new UserEqualsConstraint($expected),
            $message
        );
    }

    public static function assertNotEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new PHPUnit_Framework_Constraint_Not(
                new UserEqualsConstraint($expected)
            ),
            $message
        );
    }
```

Message!

Single Assert!

Easy Inverse!

Custom Assert: Lv. 3



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';
require_once 'UserEqualsConstraint.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new UserEqualsConstraint($expected),
            $message
        );
    }

    public static function assertNotEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new PHPUnit_Framework_Constraint_Not(
                new UserEqualsConstraint($expected)
            ),
            $message
        );
    }
```

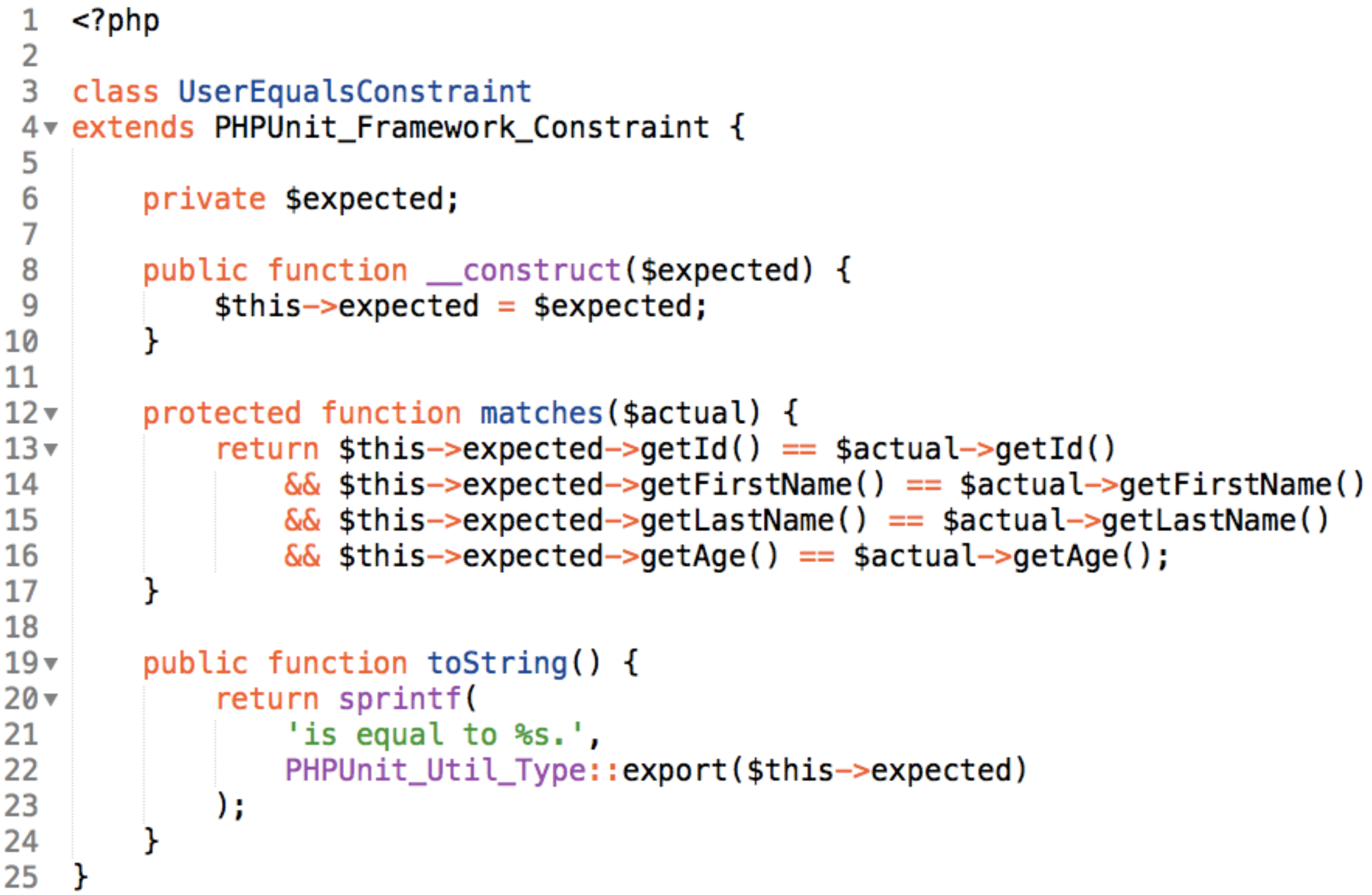
What's this?!?

Constraints

Constraints

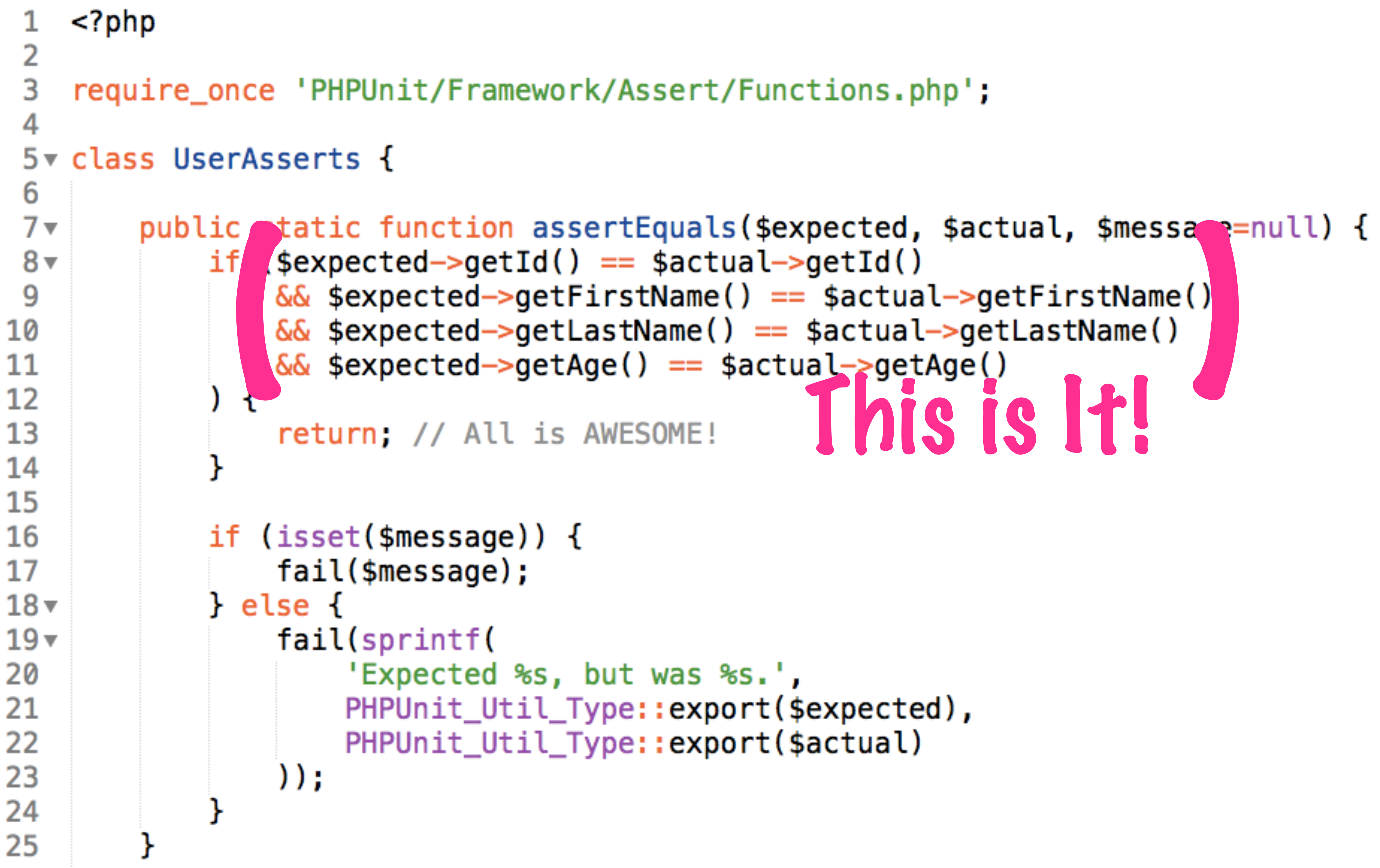
- Based on `Hamcrest` matchers
- Can be used in `assertThat()` for Custom Asserts
- Can also be used in `with()` for Mock Objects

Constraints



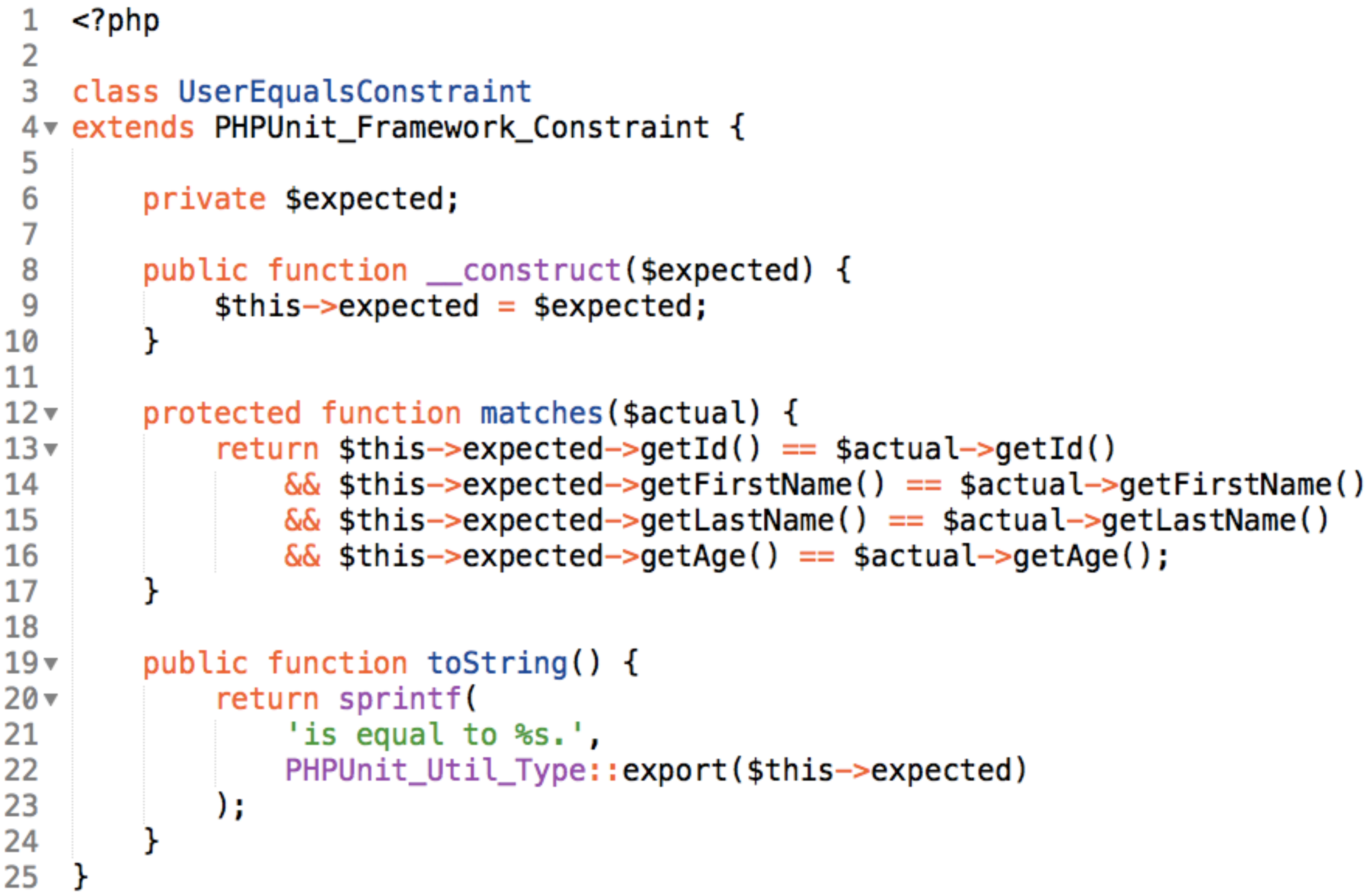
```php
<?php

class UserEqualsConstraint
extends PHPUnit_Framework_Constraint {

    private $expected;

    public function __construct($expected) {
        $this->expected = $expected;
    }

    protected function matches($actual) {
        return $this->expected->getId() == $actual->getId()
            && $this->expected->getFirstName() == $actual->getFirstName()
            && $this->expected->getLastName() == $actual->getLastName()
            && $this->expected->getAge() == $actual->getAge();
    }

    public function toString() {
        return sprintf(
            'is equal to %s.',
            PHPUnit_Util_Type::export($this->expected)
        );
    }
}
```

Custom Assert: Lv. 2



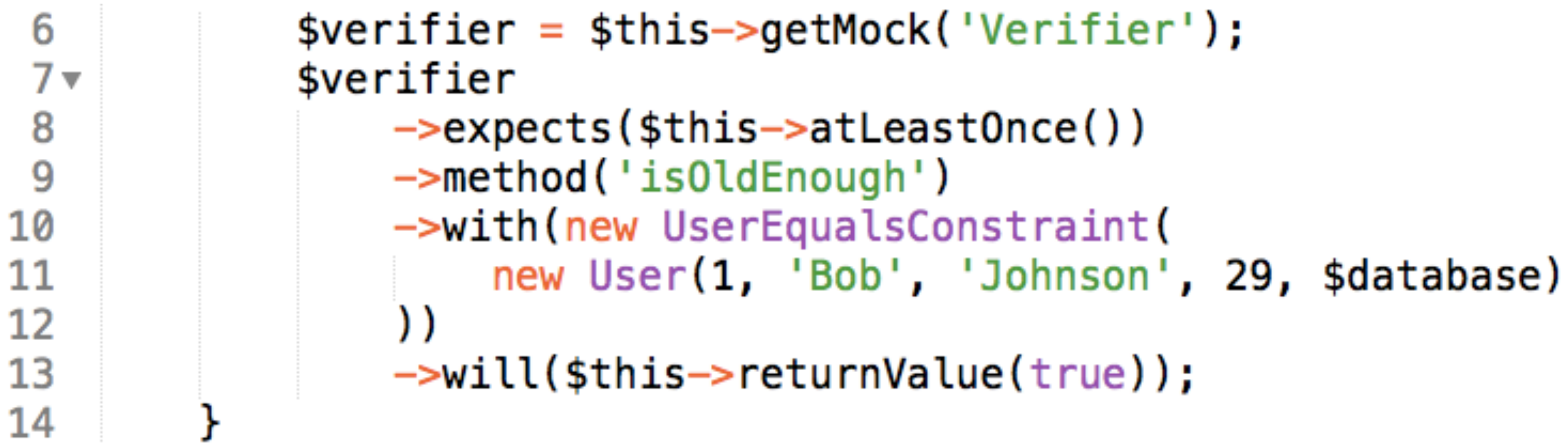
```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        if ($expected->getId() == $actual->getId()
            && $expected->getFirstName() == $actual->getFirstName()
            && $expected->getLastName() == $actual->getLastName()
            && $expected->getAge() == $actual->getAge()
        ) {
            return; // All is AWESOME!
        }

        if (isset($message)) {
            fail($message);
        } else {
            fail(sprintf(
                'Expected %s, but was %s.',
                PHPUnit_Util_Type::export($expected),
                PHPUnit_Util_Type::export($actual)
            ));
        }
    }
```

Constraints



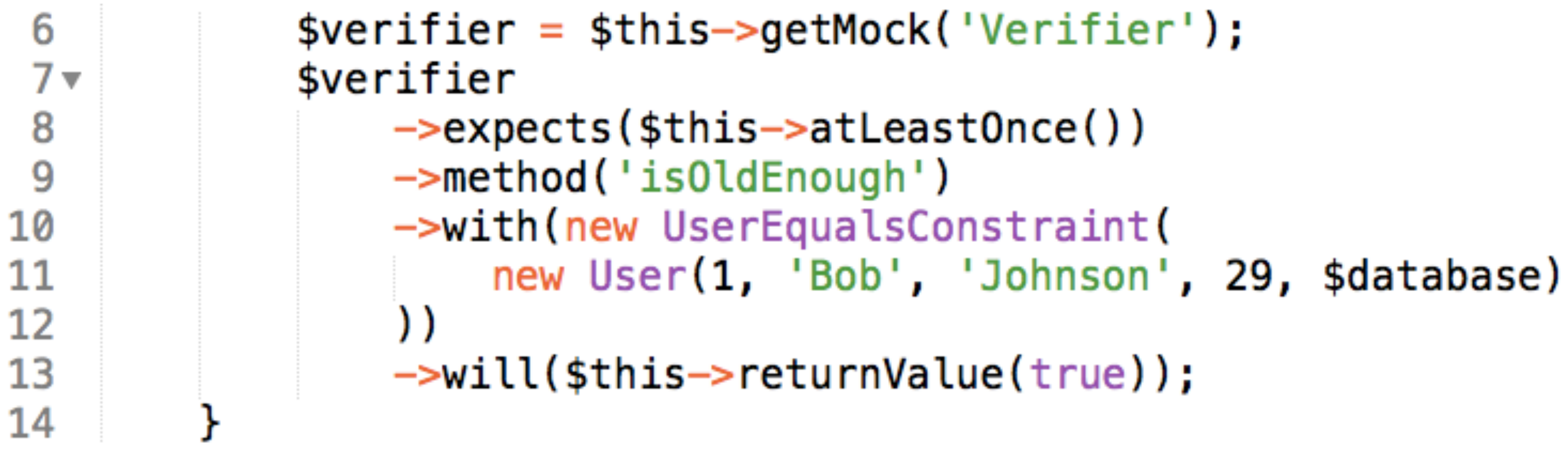
```php
<?php

class UserEqualsConstraint
extends PHPUnit_Framework_Constraint {

    private $expected;

    public function __construct($expected) {
        $this->expected = $expected;
    }

    protected function matches($actual) {
        return $this->expected->getId() == $actual->getId()
            && $this->expected->getFirstName() == $actual->getFirstName()
            && $this->expected->getLastName() == $actual->getLastName()
            && $this->expected->getAge() == $actual->getAge();
    }

    public function toString() {
        return sprintf(
            'is equal to %s.',
            PHPUnit_Util_Type::export($this->expected)
        );
    }
}
```

Constraints `with()` Mocks



```php
        $verifier = $this->getMock('Verifier');
        $verifier
            ->expects($this->atLeastOnce())
            ->method('isOldEnough')
            ->with(new UserEqualsConstraint(
                new User(1, 'Bob', 'Johnson', 29, $database)
            ))
            ->will($this->returnValue(true));
    }
```

Constraints `with()` Mocks

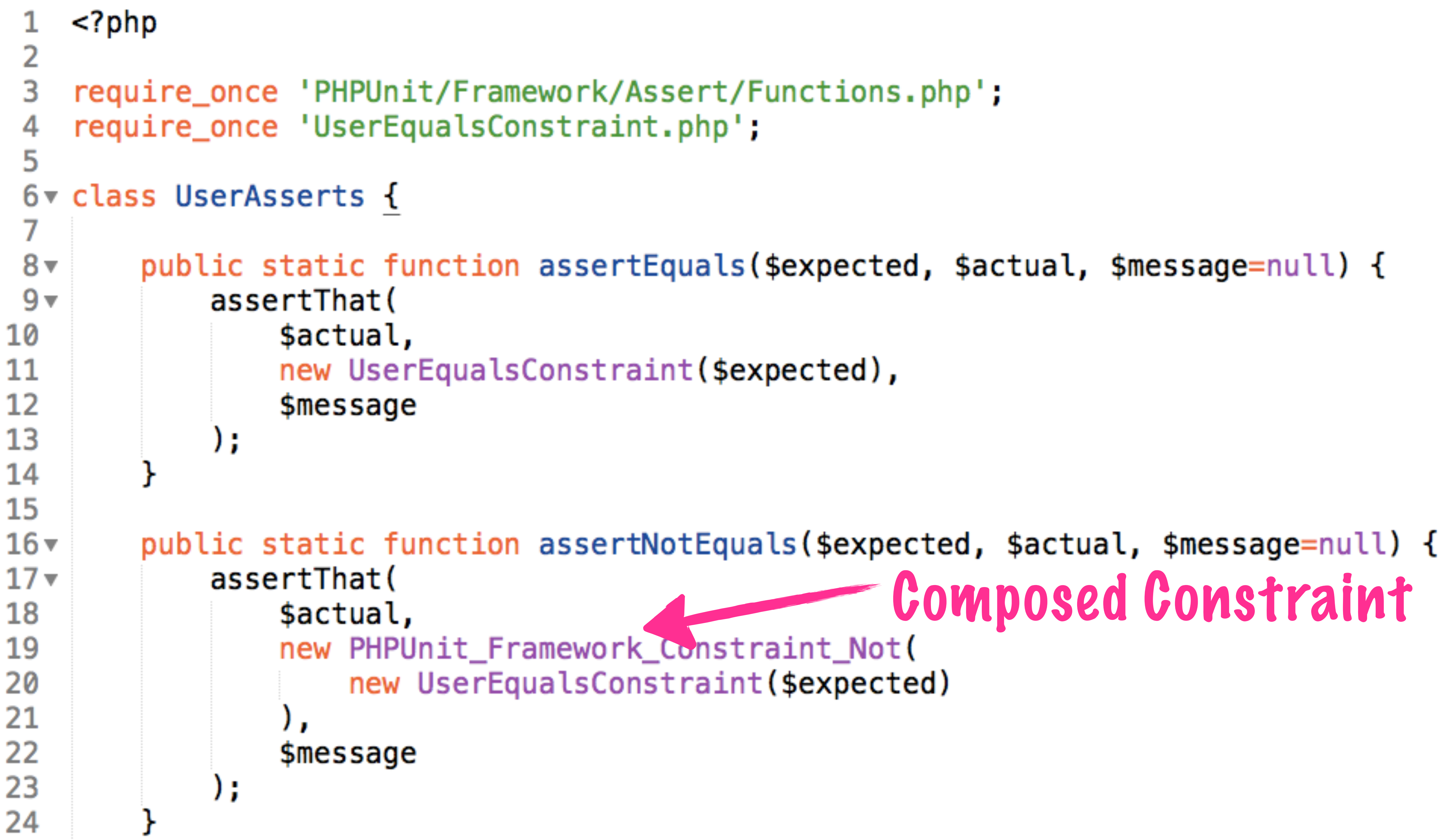


```
    $verifier = $this->getMock('Verifier');
    $verifier
        ->expects($this->atLeastOnce())
        ->method('isOldEnough')
        ->with(new UserEqualsConstraint(
            new User(1, 'Bob', 'Johnson', 29, $database)
        ))
        ->will($this->returnValue(true));
}
```

with()

- equalTo()
- **anything()**
- isTrue()
- isFalse()
- isNull()
- contains()
- containsOnly()
- arrayHasKey()
- isEmpty()
- greaterThan()
- greaterThanOrEqual()
- lessThan()
- lessThanOrEqual()
- identicalTo()
- isInstanceOf()
- isType()
- matchesRegularExpression
- matches()
- stringStartWith()
- stringEndWith()
- stringContains()
- **logicalAnd()**
- **logicalOr()**
- **logicalNot()**
- **logicalXor()**
- more...

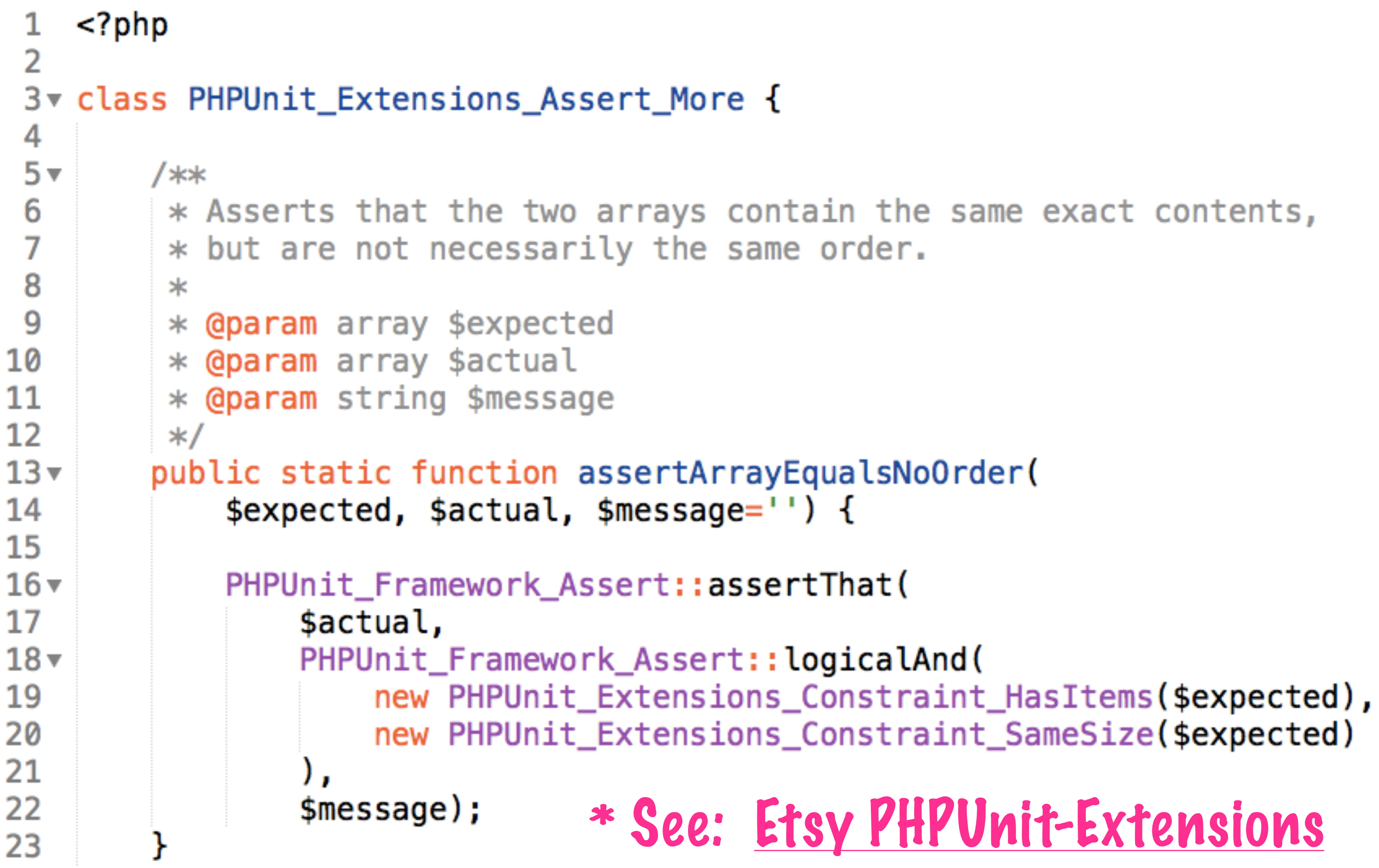
Custom Assert: Lv. 3



```php
<?php

require_once 'PHPUnit/Framework/Assert/Functions.php';
require_once 'UserEqualsConstraint.php';

class UserAsserts {

    public static function assertEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new UserEqualsConstraint($expected),
            $message
        );
    }

    public static function assertNotEquals($expected, $actual, $message=null) {
        assertThat(
            $actual,
            new PHPUnit_Framework_Constraint_Not(
                new UserEqualsConstraint($expected)
            ),
            $message
        );
    }
```

Composed Constraint

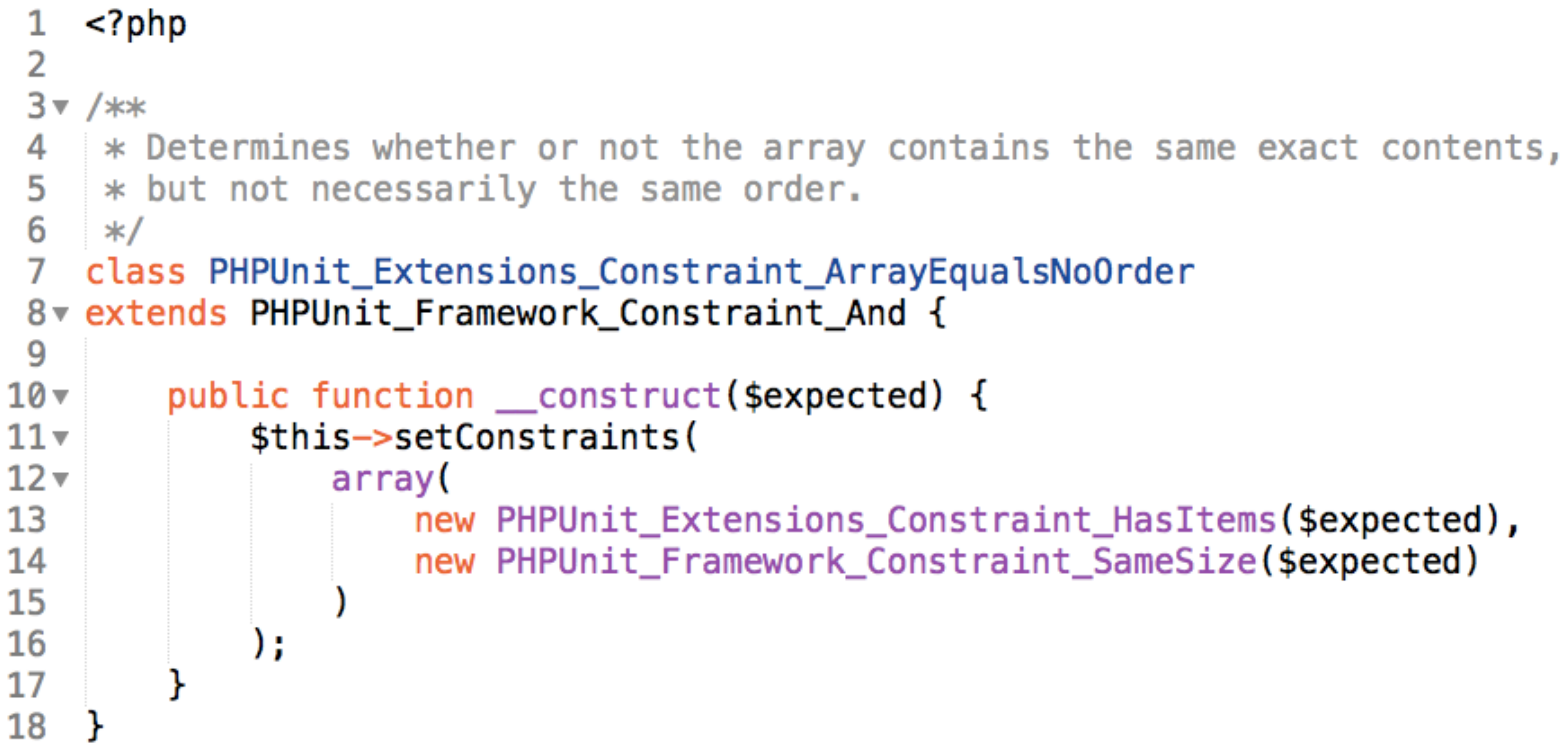
Composed Constraint



```php
<?php

class PHPUnit_Extensions_Assert_More {

    /**
     * Asserts that the two arrays contain the same exact contents,
     * but are not necessarily the same order.
     *
     * @param array $expected
     * @param array $actual
     * @param string $message
     */
    public static function assertArrayEqualsNoOrder(
        $expected, $actual, $message='') {

        PHPUnit_Framework_Assert::assertThat(
            $actual,
            PHPUnit_Framework_Assert::logicalAnd(
                new PHPUnit_Extensions_Constraint_HasItems($expected),
                new PHPUnit_Extensions_Constraint_SameSize($expected)
            ),
            $message);
    }
```

* See: Etsy PHPUnit-Extensions

Composed Constraint



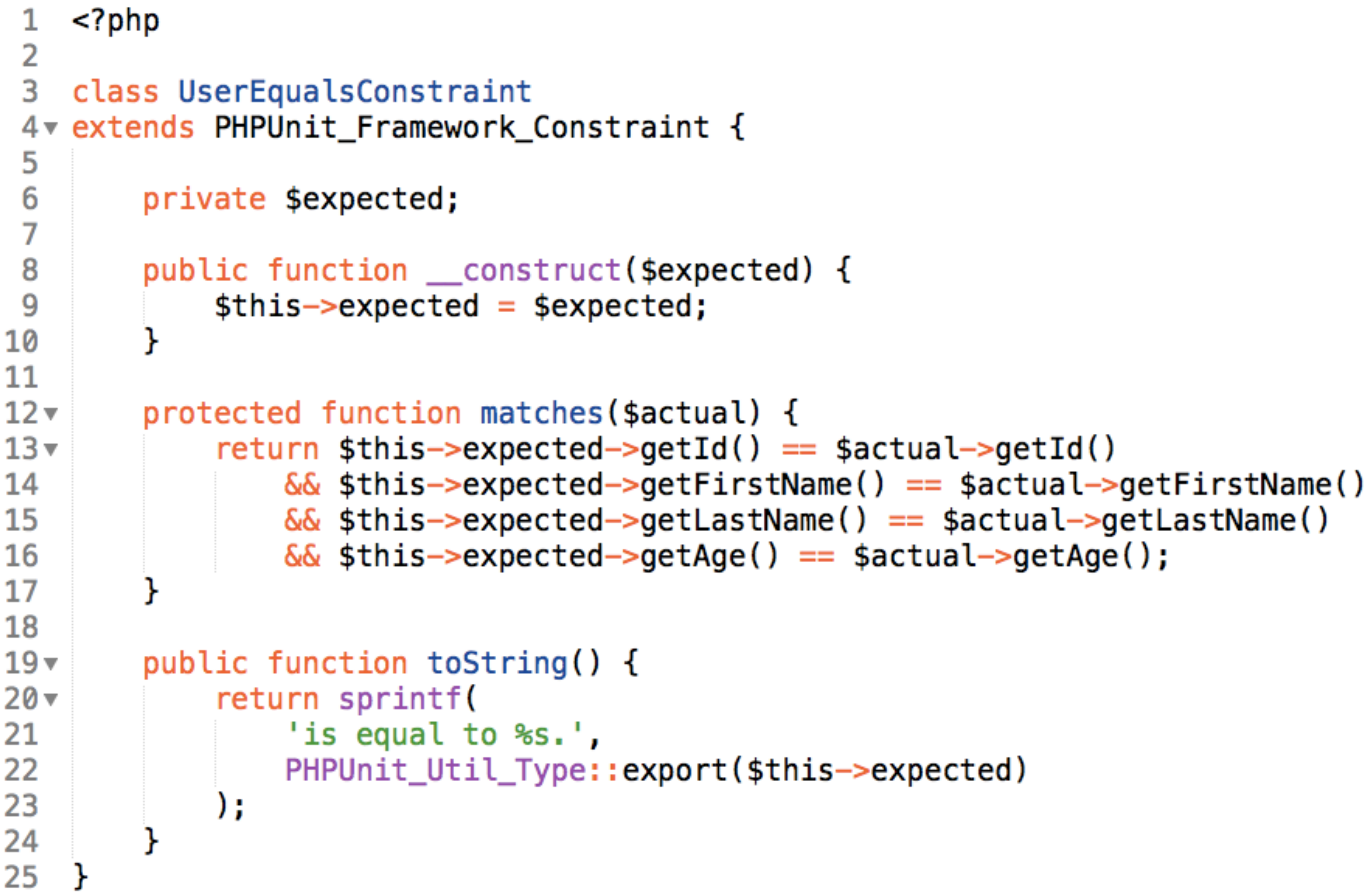
```php
<?php

/**
 * Determines whether or not the array contains the same exact contents,
 * but not necessarily the same order.
 */
class PHPUnit_Extensions_Constraint_ArrayEqualsNoOrder
extends PHPUnit_Framework_Constraint_And {

    public function __construct($expected) {
        $this->setConstraints(
            array(
                new PHPUnit_Extensions_Constraint_HasItems($expected),
                new PHPUnit_Framework_Constraint_SameSize($expected)
            )
        );
    }
}
```

* See: Etsy PHPUnit-Extensions

Constraints v. Matcher(?)

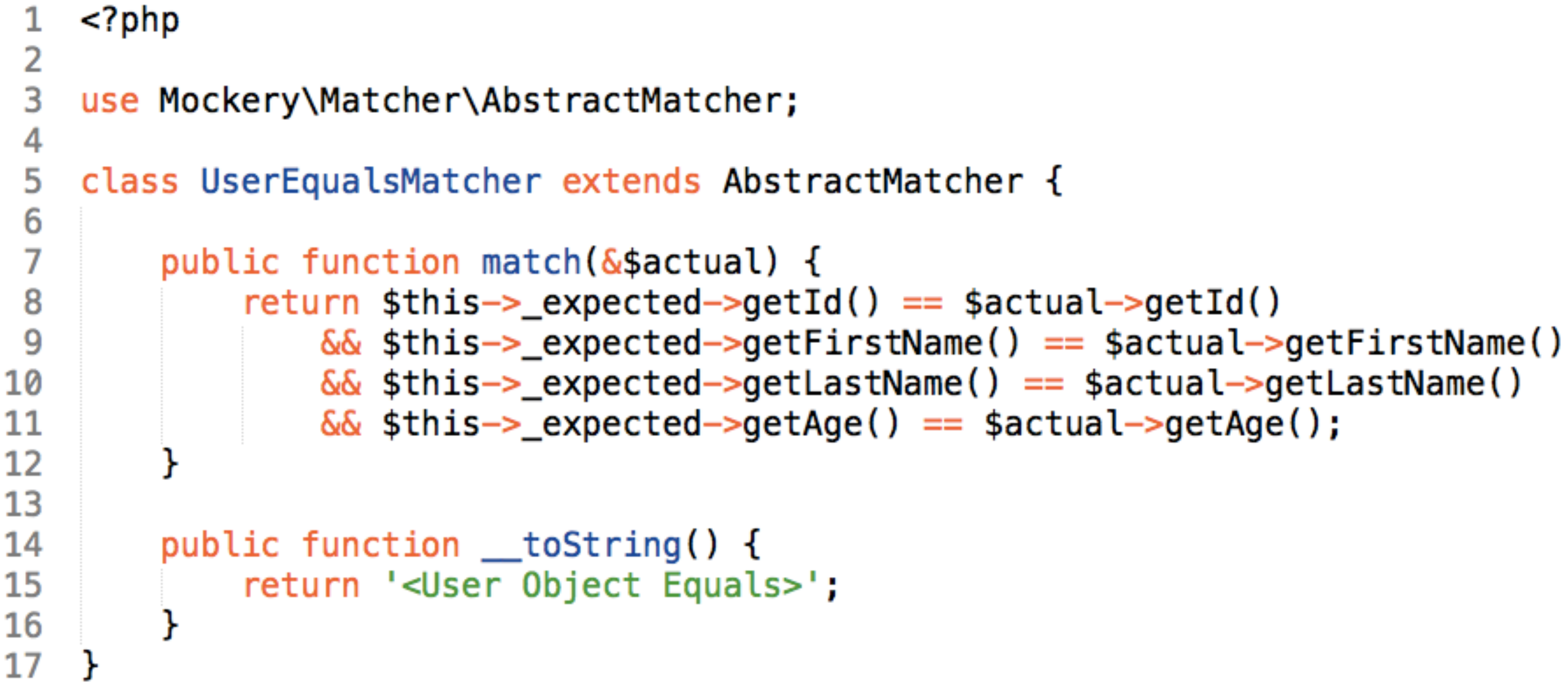
- PHPUnit Constraint
  - `evaluate()`
- Hamcrest Matcher
  - `matches()`
- Mockery Matcher
  - `match()`

Constraints



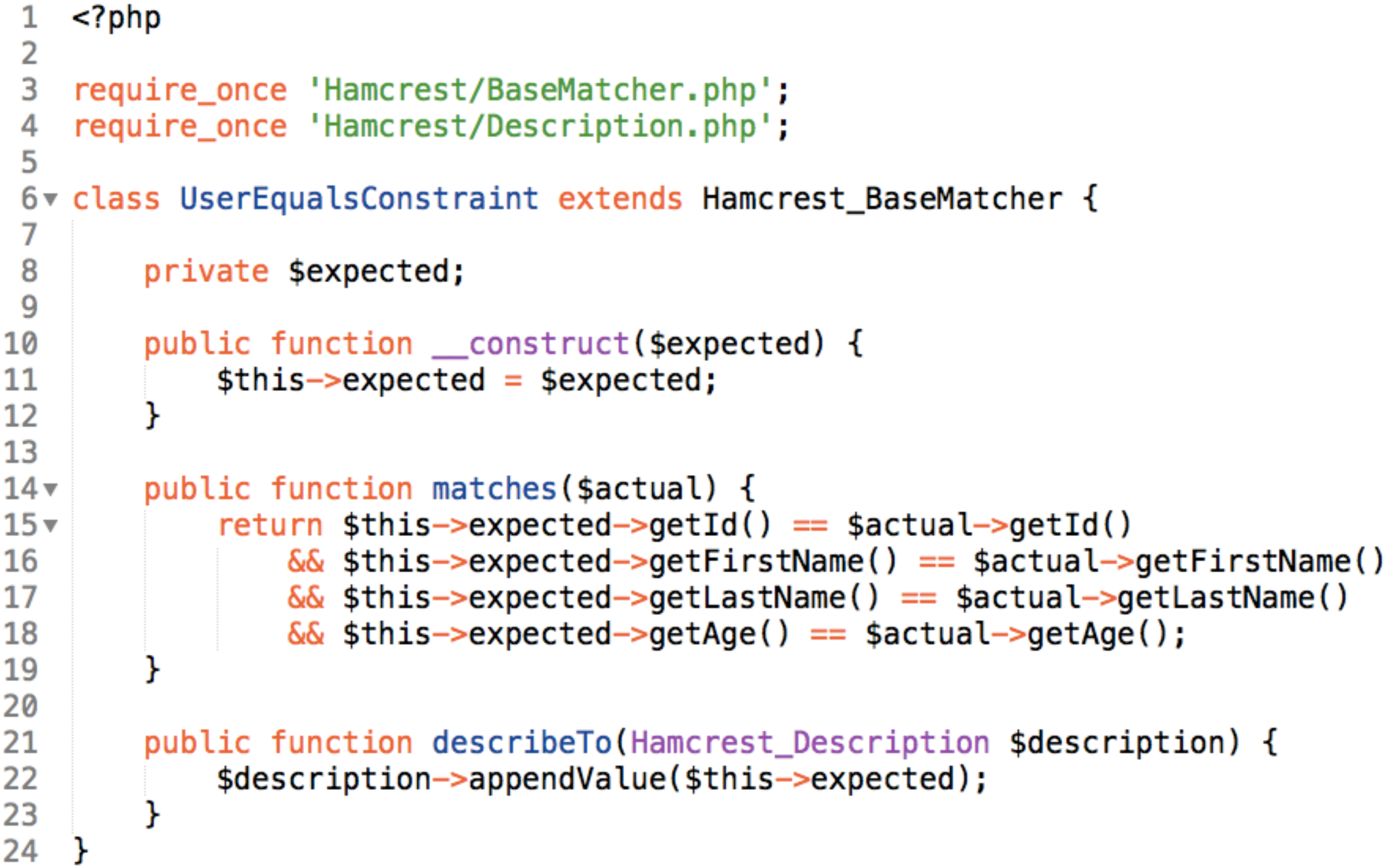
```php
<?php

class UserEqualsConstraint
extends PHPUnit_Framework_Constraint {

    private $expected;

    public function __construct($expected) {
        $this->expected = $expected;
    }

    protected function matches($actual) {
        return $this->expected->getId() == $actual->getId()
            && $this->expected->getFirstName() == $actual->getFirstName()
            && $this->expected->getLastName() == $actual->getLastName()
            && $this->expected->getAge() == $actual->getAge();
    }

    public function toString() {
        return sprintf(
            'is equal to %s.',
            PHPUnit_Util_Type::export($this->expected)
        );
    }
}
```

Mockery Matcher



```php
<?php

use Mockery\Matcher\AbstractMatcher;

class UserEqualsMatcher extends AbstractMatcher {

    public function match(&$actual) {
        return $this->_expected->getId() == $actual->getId()
            && $this->_expected->getFirstName() == $actual->getFirstName()
            && $this->_expected->getLastName() == $actual->getLastName()
            && $this->_expected->getAge() == $actual->getAge();
    }

    public function __toString() {
        return '<User Object Equals>';
    }
}
```

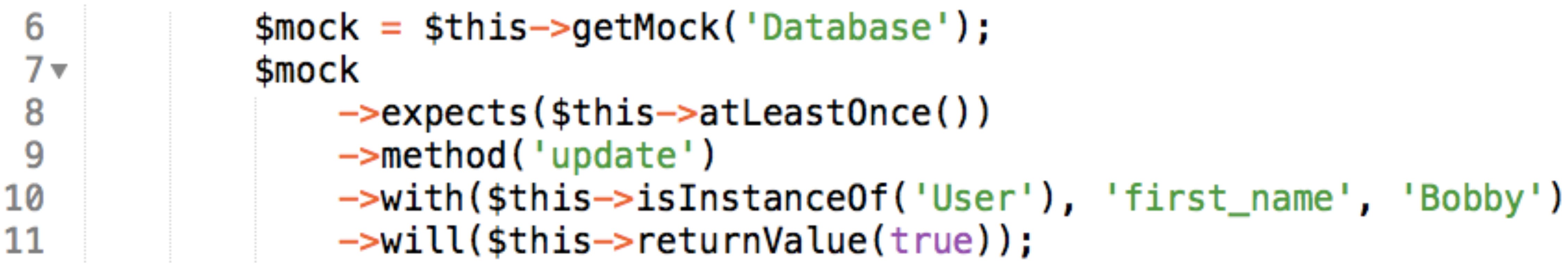
Hamcrest Matcher



```php
<?php

require_once 'Hamcrest/BaseMatcher.php';
require_once 'Hamcrest/Description.php';

class UserEqualsConstraint extends Hamcrest_BaseMatcher {

    private $expected;

    public function __construct($expected) {
        $this->expected = $expected;
    }

    public function matches($actual) {
        return $this->expected->getId() == $actual->getId()
            && $this->expected->getFirstName() == $actual->getFirstName()
            && $this->expected->getLastName() == $actual->getLastName()
            && $this->expected->getAge() == $actual->getAge();
    }

    public function describeTo(Hamcrest_Description $description) {
        $description->appendValue($this->expected);
    }
}
```

Mock Objects

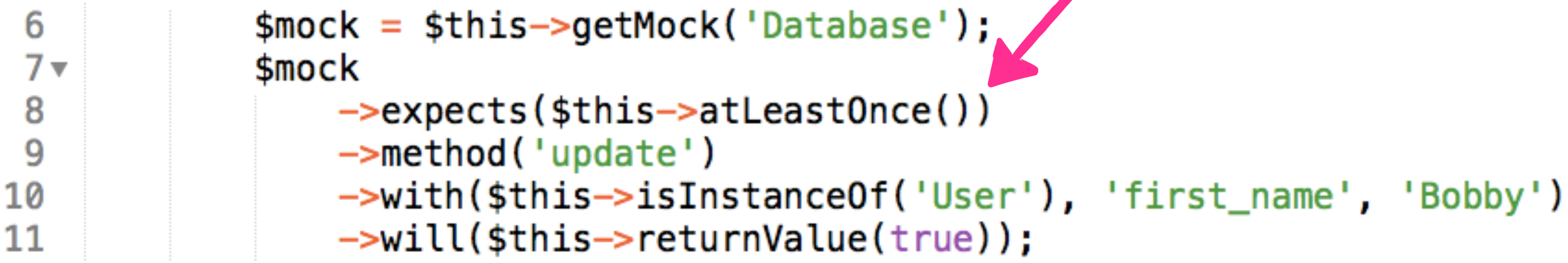
Mock Object



```
        $mock = $this->getMock('Database');
        $mock
            ->expects($this->atLeastOnce())
            ->method('update')
            ->with($this->isInstanceOf('User'), 'first_name', 'Bobby')
            ->will($this->returnValue(true));
```

Mock Object

Have To Specify Occurrences Every Time



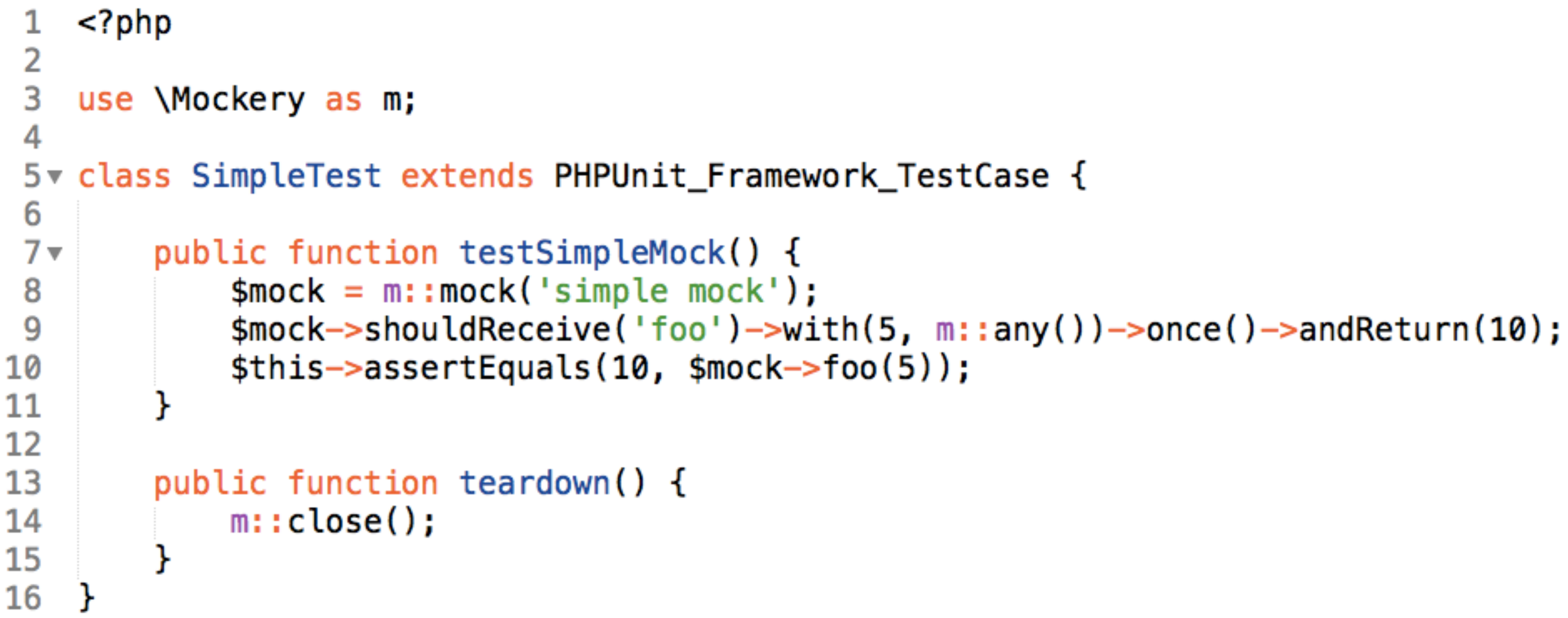
```
$mock = $this->getMock('Database');
$mock
    ->expects($this->atLeastOnce())
    ->method('update')
    ->with($this->isInstanceOf('User'), 'first_name', 'Bobby')
    ->will($this->returnValue(true));
```

Can Have Only ONE Expectation Per Method

Workarounds

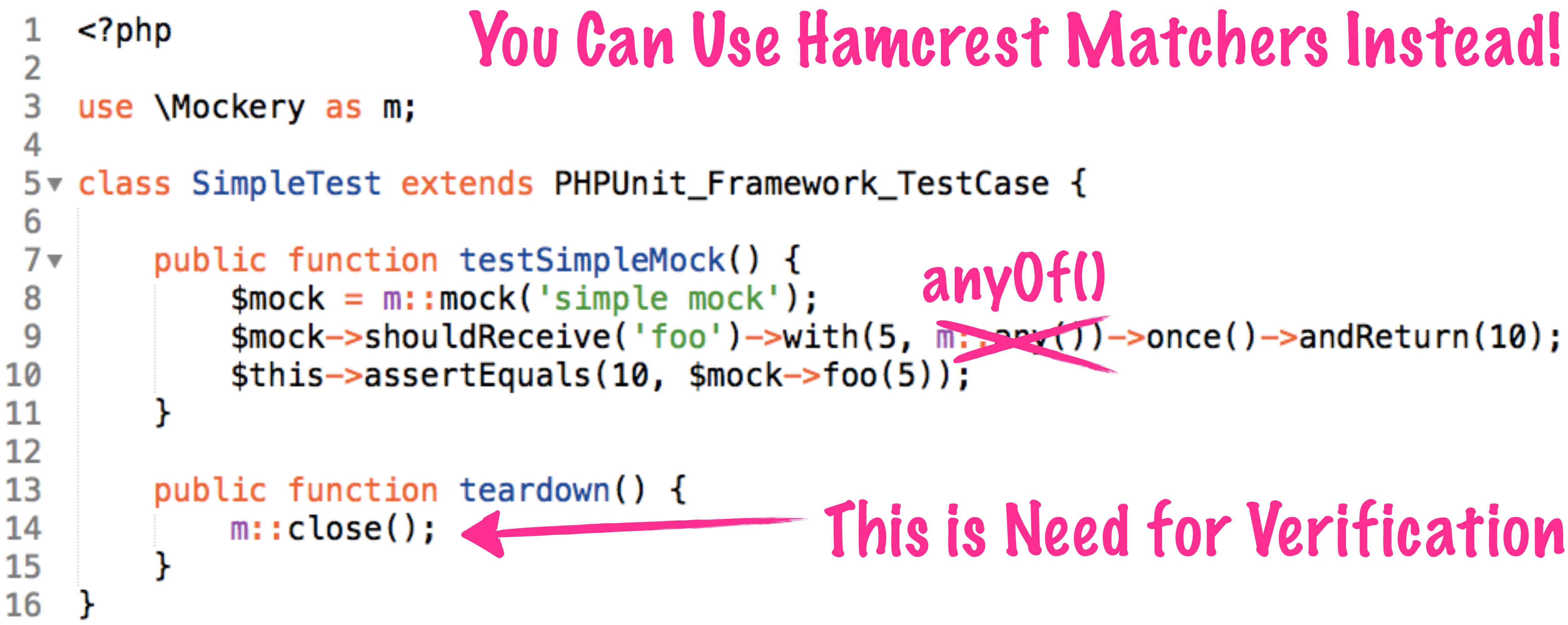
- There are **hacks** to allow multiple expectations per method
- See: https://github.com/etsy/phpunit-extensions/wiki/Mock-object

Mockery



```php
<?php

use \Mockery as m;

class SimpleTest extends PHPUnit_Framework_TestCase {

    public function testSimpleMock() {
        $mock = m::mock('simple mock');
        $mock->shouldReceive('foo')->with(5, m::any())->once()->andReturn(10);
        $this->assertEquals(10, $mock->foo(5));
    }

    public function teardown() {
        m::close();
    }
}
```

Mockery



```php
<?php

use \Mockery as m;

class SimpleTest extends PHPUnit_Framework_TestCase {

    public function testSimpleMock() {
        $mock = m::mock('simple mock');
        $mock->shouldReceive('foo')->with(5, m::~~any()~~)->once()->andReturn(10);
        $this->assertEquals(10, $mock->foo(5));
    }

    public function teardown() {
        m::close();
    }
}
```

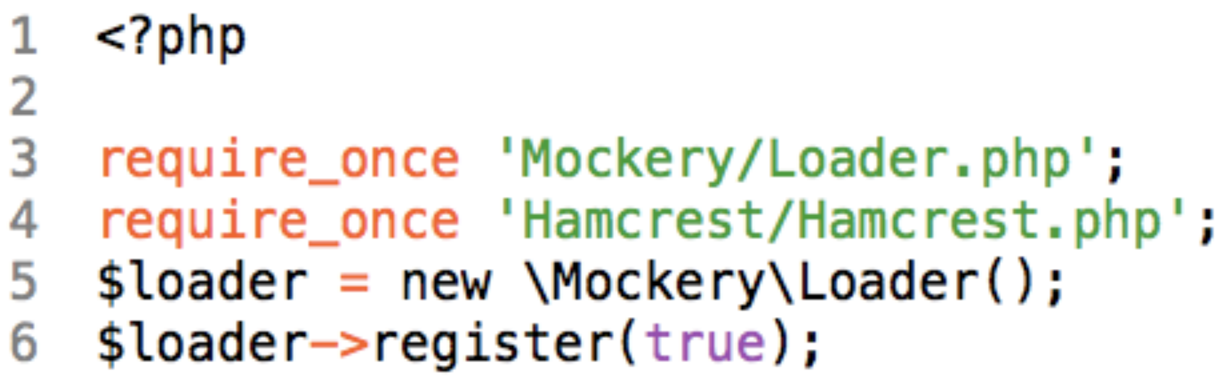
You Can Use Hamcrest Matchers Instead!

anyOf()

This is Need for Verification

Bootstrap for Mockery

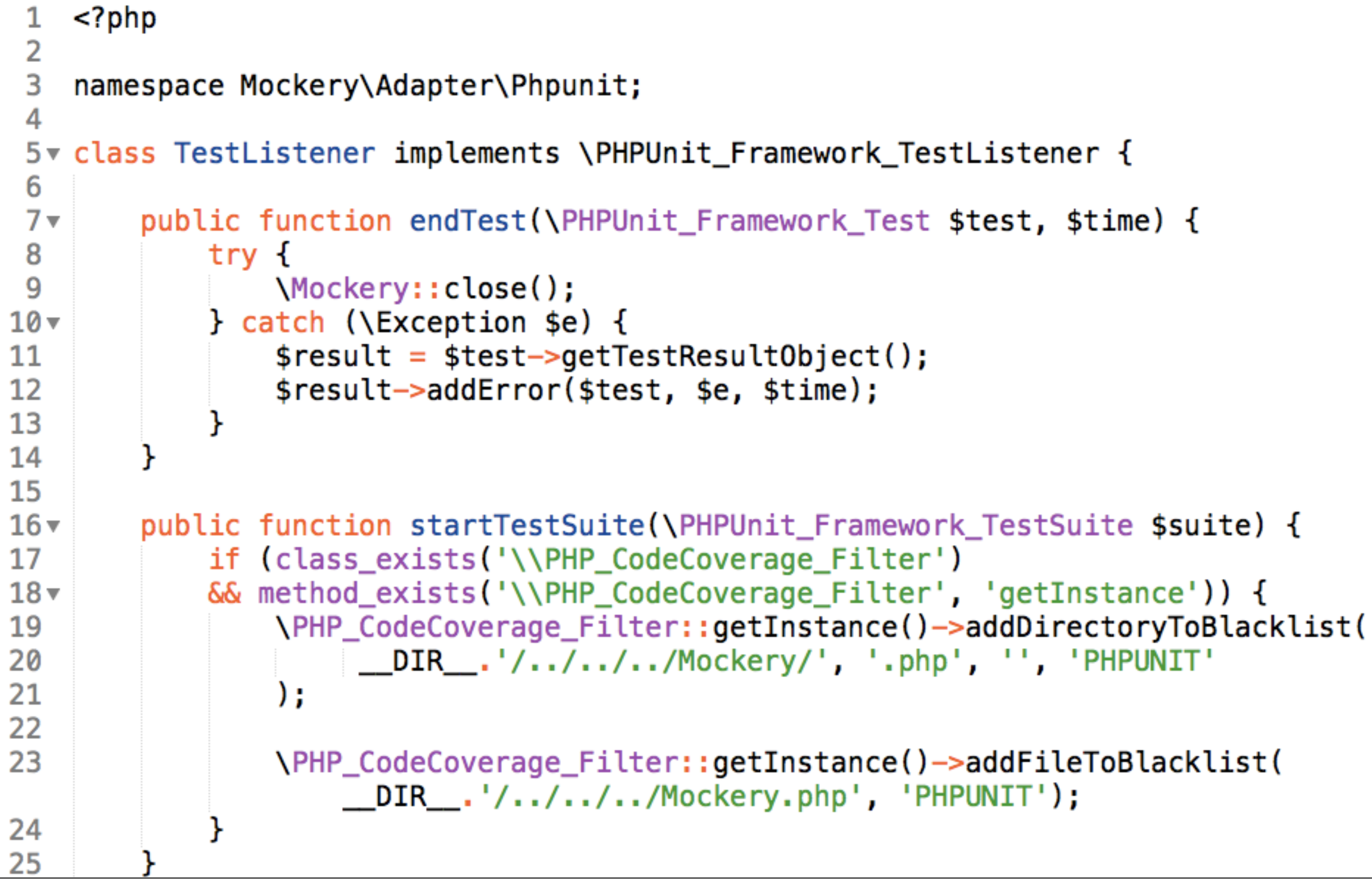
bootstrap.php



```php
<?php

require_once 'Mockery/Loader.php';
require_once 'Hamcrest/Hamcrest.php';
$loader = new \Mockery\Loader();
$loader->register(true);
```

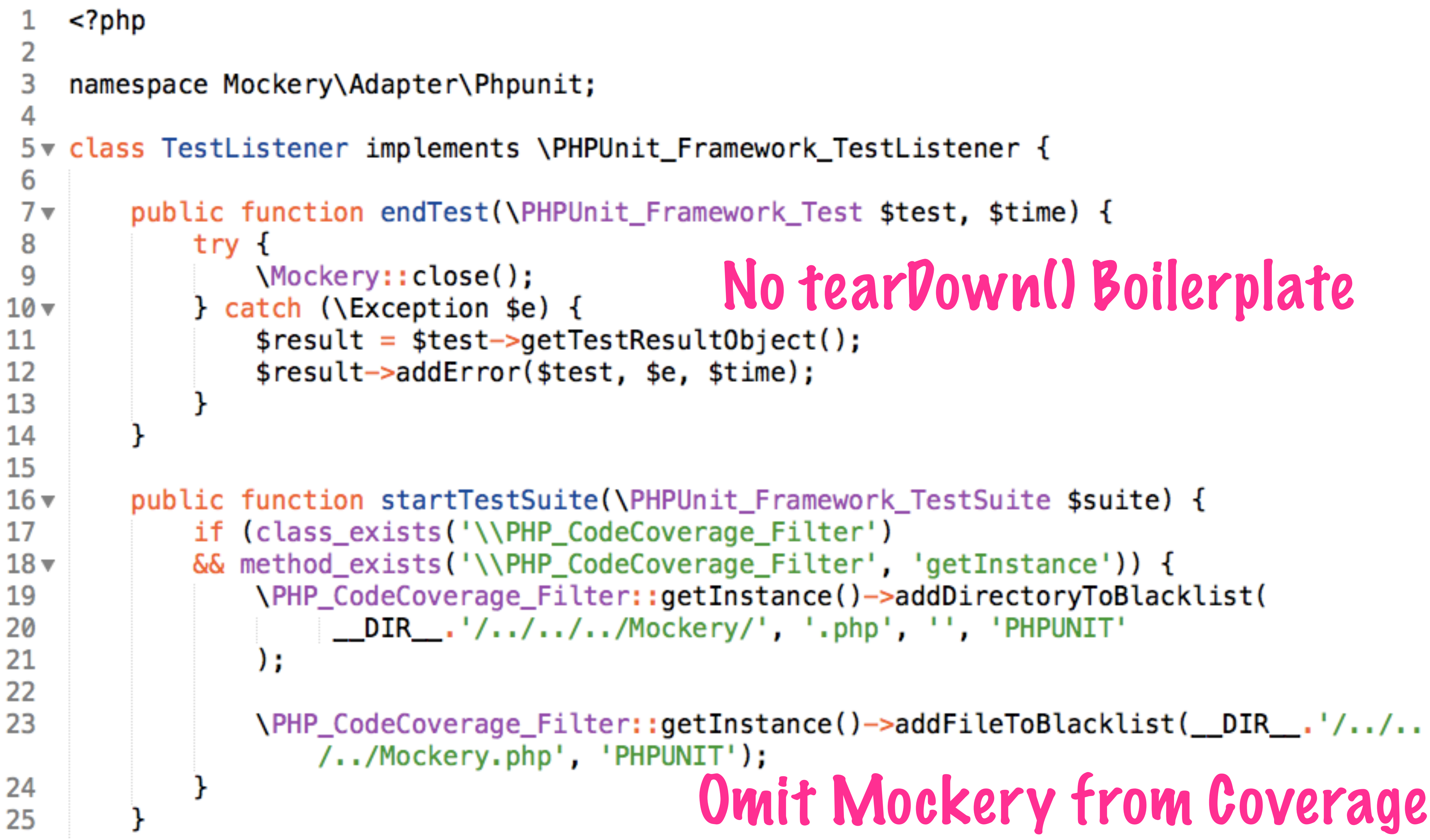
phpunit --boostrap=bootstrap.php
or set in your phpunit.xml

Mockery PHPUnit Listener



```php
<?php

namespace Mockery\Adapter\Phpunit;

class TestListener implements \PHPUnit_Framework_TestListener {

    public function endTest(\PHPUnit_Framework_Test $test, $time) {
        try {
            \Mockery::close();
        } catch (\Exception $e) {
            $result = $test->getTestResultObject();
            $result->addError($test, $e, $time);
        }
    }

    public function startTestSuite(\PHPUnit_Framework_TestSuite $suite) {
        if (class_exists('\\PHP_CodeCoverage_Filter')
        && method_exists('\\PHP_CodeCoverage_Filter', 'getInstance')) {
            \PHP_CodeCoverage_Filter::getInstance()->addDirectoryToBlacklist(
                 __DIR__.'/../../../Mockery/', '.php', '', 'PHPUNIT'
            );

            \PHP_CodeCoverage_Filter::getInstance()->addFileToBlacklist(
                __DIR__.'/../../../Mockery.php', 'PHPUNIT');
        }
    }
```

Mockery PHPUnit Listener

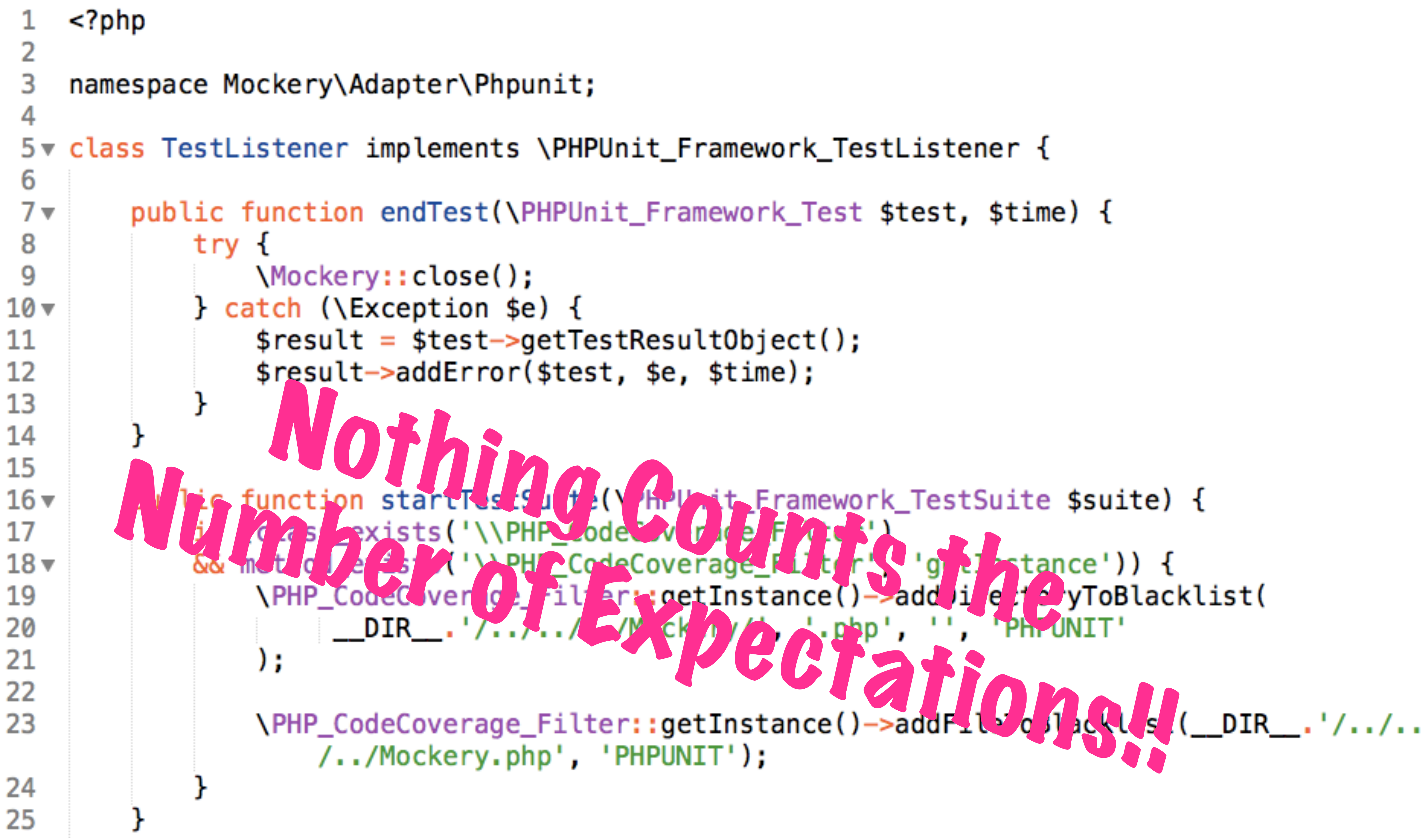


```php
<?php

namespace Mockery\Adapter\Phpunit;

class TestListener implements \PHPUnit_Framework_TestListener {

    public function endTest(\PHPUnit_Framework_Test $test, $time) {
        try {
            \Mockery::close();
        } catch (\Exception $e) {
            $result = $test->getTestResultObject();
            $result->addError($test, $e, $time);
        }
    }

    public function startTestSuite(\PHPUnit_Framework_TestSuite $suite) {
        if (class_exists('\\PHP_CodeCoverage_Filter')
        && method_exists('\\PHP_CodeCoverage_Filter', 'getInstance')) {
            \PHP_CodeCoverage_Filter::getInstance()->addDirectoryToBlacklist(
                 __DIR__.'/../../../Mockery/', '.php', '', 'PHPUNIT'
            );

            \PHP_CodeCoverage_Filter::getInstance()->addFileToBlacklist(__DIR__.'/../../../Mockery.php', 'PHPUNIT');
        }
    }
```

No tearDown() Boilerplate

Omit Mockery from Coverage

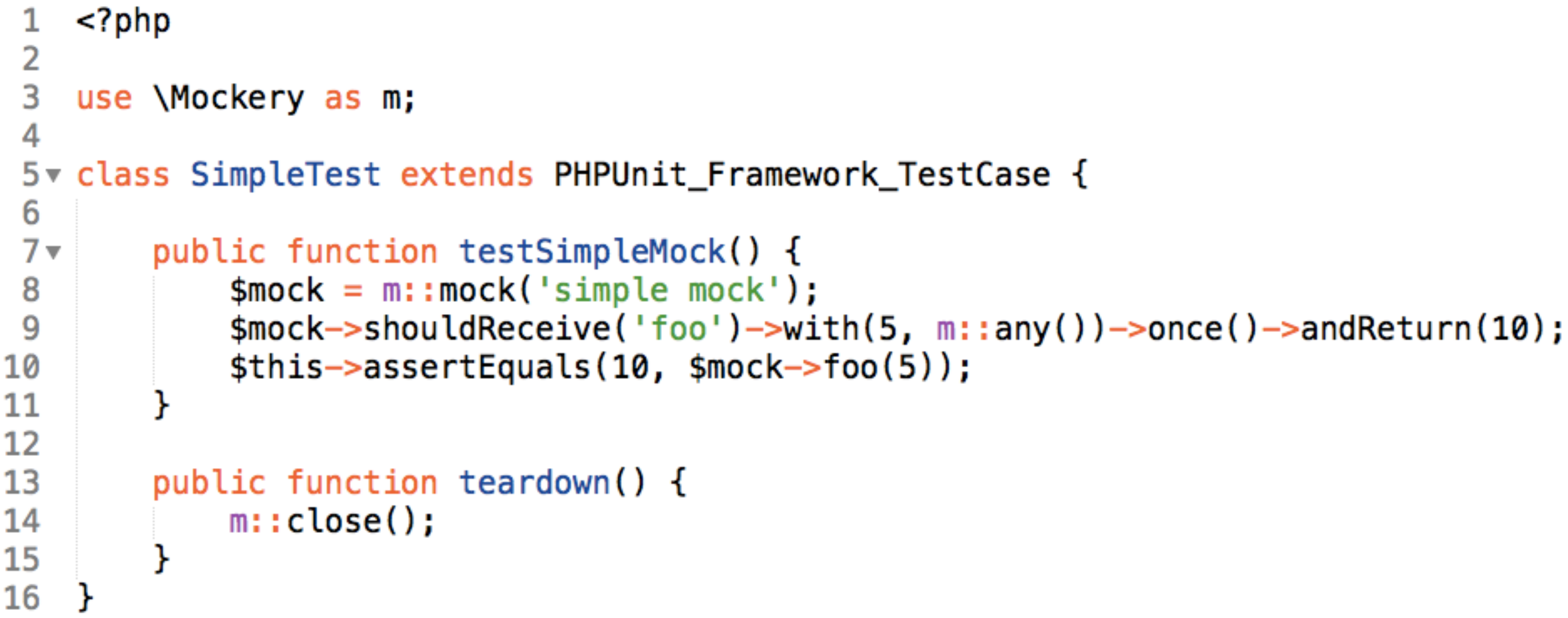
Incompatible with `--strict`



```php
<?php

namespace Mockery\Adapter\Phpunit;

class TestListener implements \PHPUnit_Framework_TestListener {

    public function endTest(\PHPUnit_Framework_Test $test, $time) {
        try {
            \Mockery::close();
        } catch (\Exception $e) {
            $result = $test->getTestResultObject();
            $result->addError($test, $e, $time);
        }
    }

    public function startTestSuite(\PHPUnit_Framework_TestSuite $suite) {
        if (class_exists('\\PHP_CodeCoverage_Filter')
        && method_exists('\\PHP_CodeCoverage_Filter', 'getInstance')) {
            \PHP_CodeCoverage_Filter::getInstance()->addDirectoryToBlacklist(
                 __DIR__.'/../../../Mockery/', '.php', '', 'PHPUNIT'
            );

            \PHP_CodeCoverage_Filter::getInstance()->addFileToBlacklist(__DIR__.'/../../../Mockery.php', 'PHPUNIT');
        }
    }
```

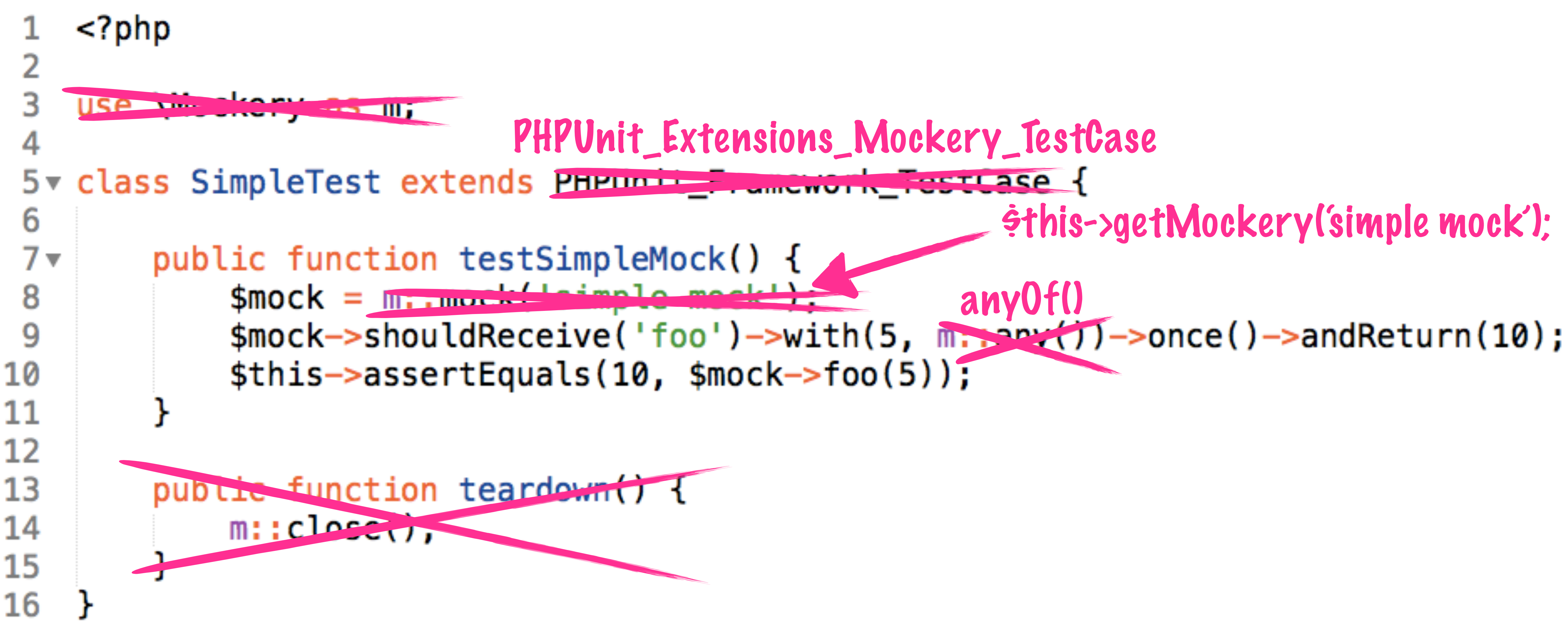
Nothing Counts the Number of Expectations!!

Mockery



```php
<?php

use \Mockery as m;

class SimpleTest extends PHPUnit_Framework_TestCase {

    public function testSimpleMock() {
        $mock = m::mock('simple mock');
        $mock->shouldReceive('foo')->with(5, m::any())->once()->andReturn(10);
        $this->assertEquals(10, $mock->foo(5));
    }

    public function teardown() {
        m::close();
    }
}
```

Mockery TestCase



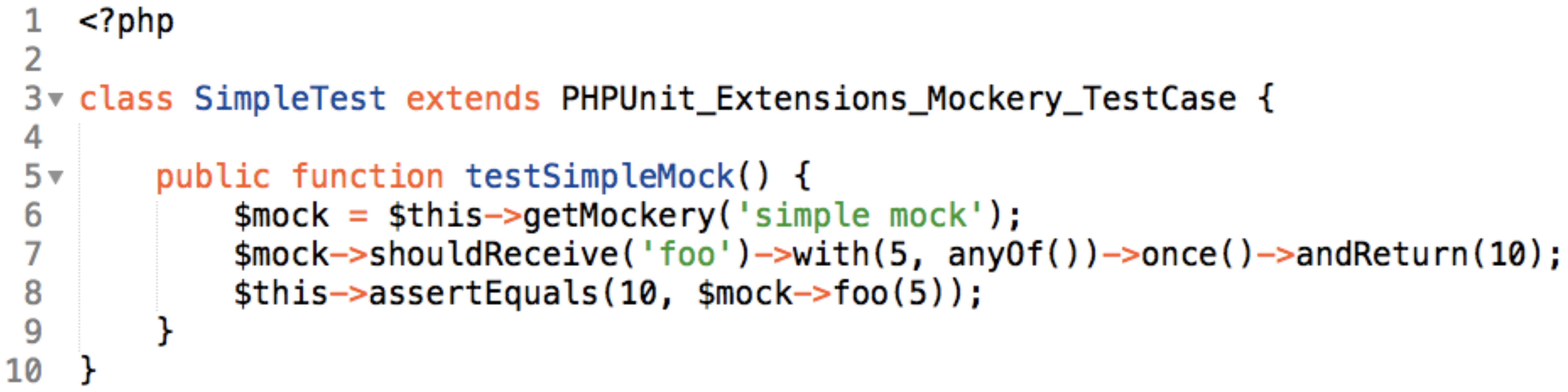
```php
<?php

~~use \Mockery as m;~~

class SimpleTest extends ~~PHPUnit_Framework_TestCase~~ {

    public function testSimpleMock() {
        $mock = ~~m::mock('simple mock')~~;
        $mock->shouldReceive('foo')->with(5, ~~m::any()~~)->once()->andReturn(10);
        $this->assertEquals(10, $mock->foo(5));
    }

    ~~public function teardown() {~~
        ~~m::close();~~
    ~~}~~
}
```

PHPUnit_Extensions_Mockery_TestCase

$this->getMockery('simple mock');

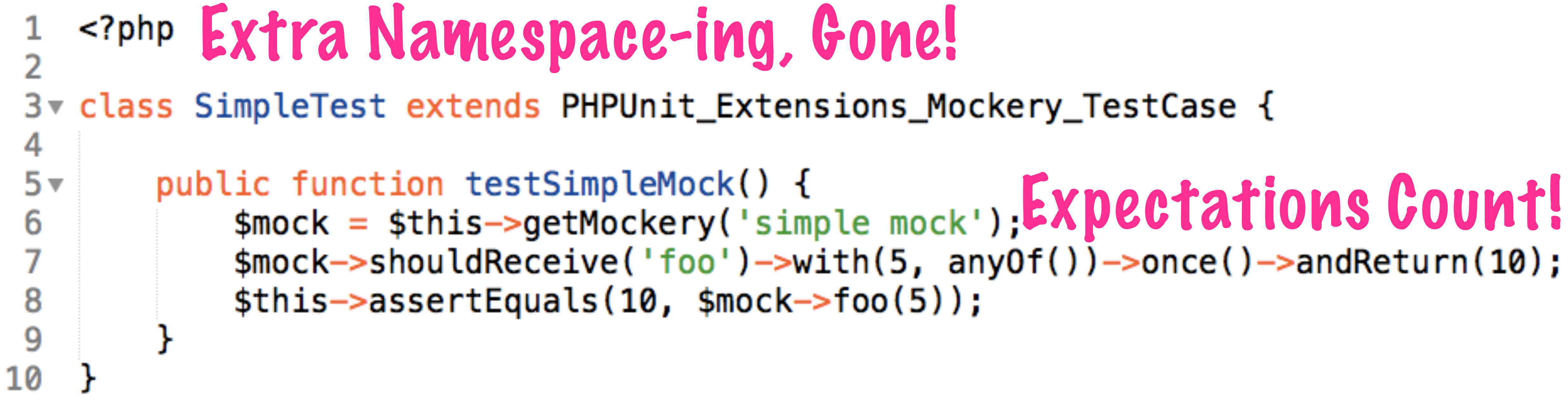
anyOf()

Mockery TestCase



```php
<?php

class SimpleTest extends PHPUnit_Extensions_Mockery_TestCase {

    public function testSimpleMock() {
        $mock = $this->getMockery('simple mock');
        $mock->shouldReceive('foo')->with(5, anyOf())->once()->andReturn(10);
        $this->assertEquals(10, $mock->foo(5));
    }
}
```

Mockery TestCase



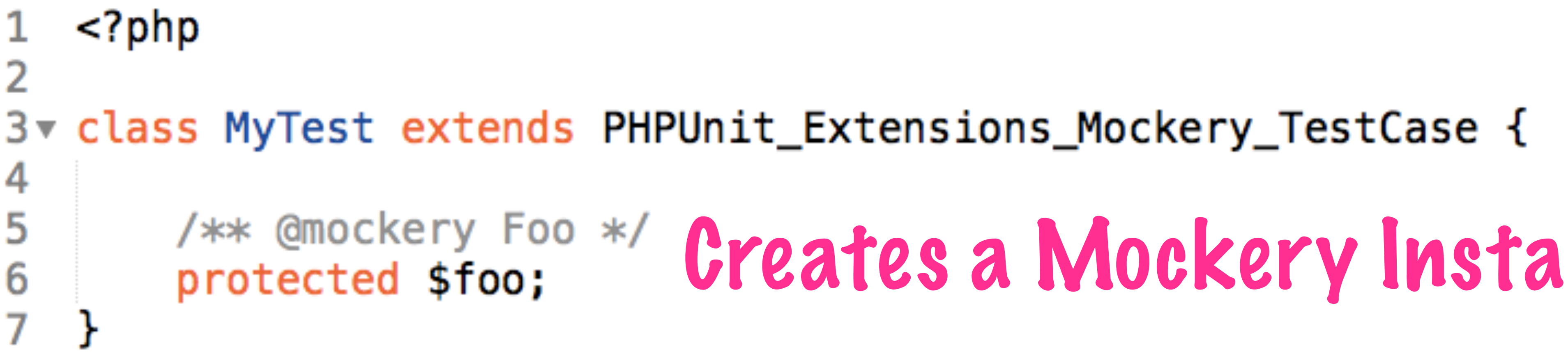
```php
<?php

class SimpleTest extends PHPUnit_Extensions_Mockery_TestCase {

    public function testSimpleMock() {
        $mock = $this->getMockery('simple mock');
        $mock->shouldReceive('foo')->with(5, anyOf())->once()->andReturn(10);
        $this->assertEquals(10, $mock->foo(5));
    }
}
```

Extra Namespace-ing, Gone!

Expectations Count!

No bootstrap.php or TestListener necessary!!

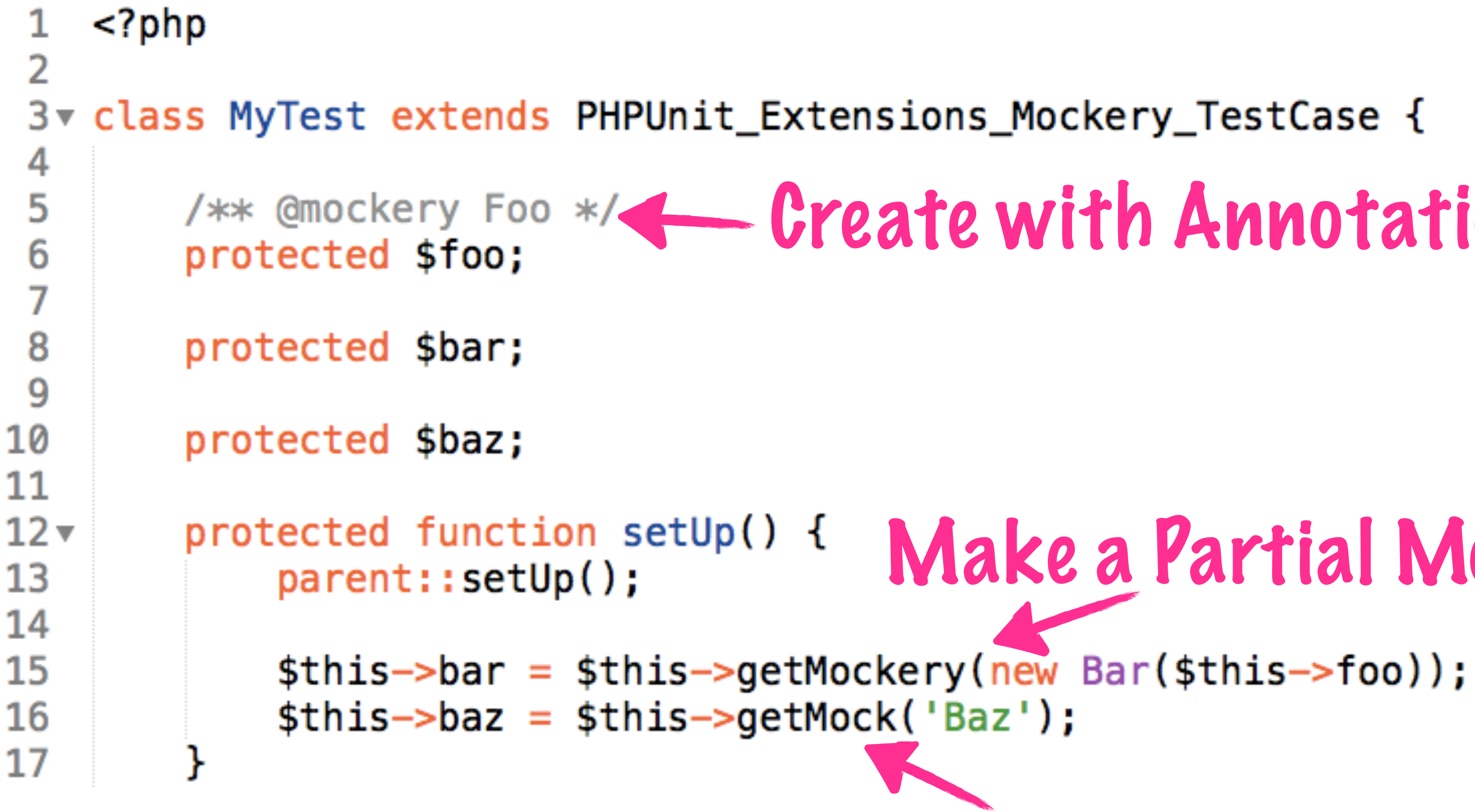
@mockery



```php
<?php

class MyTest extends PHPUnit_Extensions_Mockery_TestCase {

    /** @mockery Foo */
    protected $foo;
}
```

Creates a Mockery Instance

Mix and Match



```php
<?php

class MyTest extends PHPUnit_Extensions_Mockery_TestCase {

    /** @mockery Foo */
    protected $foo;

    protected $bar;

    protected $baz;

    protected function setUp() {
        parent::setUp();

        $this->bar = $this->getMockery(new Bar($this->foo));
        $this->baz = $this->getMock('Baz');
    }
```

Create with Annotation

Make a Partial Mock

Use PHPUnit Mock Objects too!

Filesystem Testing

vfsStream

- Virtual FileSystem
- In-Memory
- Utilizes `stream_wrapper_register`
    - `vfs://path/to/my/file.txt`

Works With...

- `file_exists`
- `copy`
- `mkdir`
- `unlink`
- `is_file`
- `is_dir`
- `...`

Pure Filenames Only

- `realpath()`
- `symlink()`
- `SplFileInfo::getRealPath()`

Not Supported in 5.3

- `touch()`
- `chmod()`
- `chown()`
- `chgrp()`

Other Limitations

- `ext/zip`
  - does not support user-land stream wrappers
- `is_executable()`
  - always returns `false`

Not Implemented

- `stream_set_blocking()`
- `stream_set_timeout()`
- `stream_set_write_buffer()`

Examples

- 05 - Filesystem
  - vfsStream/examples
  - Taggle

Network Tests

Web Services

- SOAP/WSDL
    - `$this->getMockFromWsdl();`
- Thin Wrap Your Web Service Client

Closing Thoughts

- Minimize Integration Points
- Minimize Integration Tests
- Wrap External (Hostile) Dependencies
- Try to Avoid Test Fixture Files
- Test Your Custom Test Tools
- Consider Alert Systems, not CI

Resources

Code Samples

https://github.com/elblinkin/DeveloperTesting201

PHPUnit Manual

http://www.phpunit.de/manual/current/en/

PHPUnit GitHub

https://github.com/sebastianbergmann/phpunit

Etsy PHPUnit Extensions

https://github.com/etsy/phpunit-extensions/wiki

Mockery

https://github.com/padraic/mockery