$this->cart->add(7, 14);

$this->integer($this->cart->getAmount())->isEqualTo(17);

}

public function testEmptyStorage()

{

$cart = new TestedCart();

$this->exception(function () use ($cart) { $cart->getItems();

})->hasMessage('Storage must be set');

}

}

5. Add a separated test for checking the SessionStorage class:

<?php

namespace book\cart\tests\units\storage;

use book\cart\storage\SessionStorage as TestedStorage; use book\cart\tests\TestCase;

class SessionStorage extends TestCase {

/\*\*

\* @var TestedStorage \*/

private $storage;

public function beforeTestMethod($method)

{

parent::beforeTestMethod($method);

$this->storage = new TestedStorage(['key' => 'test']);

} public function testEmpty()

{

$this

->given($storage = $this->storage) ->then

->array($storage->load()) ->isEqualTo([]);

}

public function testStore()

{

$this

->given($storage = $this->storage) ->and($storage->save($items = [1 => 5, 6 => 12])) ->then

->array($this->storage->load())

->isEqualTo($items)

;

}

}

6. Now we will get the following structure:

book

1 cart

— src

storage

I— SessionStorage.php I— StorageInterface.php Cart.php