SOAL UTS PROJECT PRAKTIKUM REKAYASA PERANGKAT LUNAK

by Nurul Firdaus, S.Kom., M.Inf.

UNS UNIVERSITAS SEBELAS MARET

PENTING

1. Jelaskan maksud dari setiap perintah yang dijalankan berikut source code PHP dan konsep OOP PHP **(*class, object, property* dan *method)*** dan *Test Cases* yang dibuat untuk melakukan pengujian unit/*unit testing* menggunakan PHPUnit!
2. Screenshoot setiap *result* yang dihasilkan baik ada *error, failure* maupun OK/berhasil kemudian analisalah sesuai dengan hasil yang didapatkan!
3. Deskripsikan pembagian *jobdesk* setiap anggota kelompok!
4. Hint! *perintah testing menggunakan* ***vendor\bin\phpunit*** ***lokasi file***
5. Letakkan file *class* di folder *src* dan file test di folder *tests atau* sesuai setting yang telah dibuat
6. Pastikan isi **composer.json** sebagaimana berikut

```
{
    "name": "lenovo/latihan-php-unit",
    "description": "Hanya untuk latihan composer dan unit testing dengan php unit",
    "type": "project",
    "license": "GPL",
    "authors": [
        {
            "name": "Nurul Firdaus",
            "email": "nurul.firdaus@staff.uns.ac.id"
        }
    ],
    "autoload": {
        "classmap": [
            "src/"
        ]
    },
      "require": {
        "nesbot/carbon": "^2.41"
    },
    "require-dev": {
        "phpunit/phpunit": "^9.4"
    }
}
```

Otomatis me-load file *class*
yang ada di folder src
tanpa menggunakan *use nama class;* di file test

7. Jalankan pertintah ***composer dump-autoload*** untuk mengenerate files autoload. Dengan memakai autoload, programmer tidak perlu lagi meng-*include*-kan file berisi *class* satu persatu, namun cukup dengan menginstansikan *class* dan PHP secara otomatis akan mengenali nama *class* serta mencari file yang memiliki nama yang sama dengan *class* tersebut.

8. Pastikan mengumpulkan file laporan dalam format **.pdf** dan file project dalam bentuk **.rar**

SOAL

Unit Testing menggunakan PHPUnit Assertions

Menggunakan assertion methods (ex: assertEquals())

1. Test *function/method* **testTambah()** dan *function/method* **testPengurangan()** dalam class **testoperasibilangan.php** dibawah ini. Analisa hasilnya (Jika gagal/*failures,* perbaiki code PHPnya kemudian test kembali)!

```php
<?php

use PHPUnit\Framework\TestCase;

 class testoperasibilangan extends TestCase
 {
 //test penambahan...
 public function testTambah()
 {
 $this->assertEquals(10,5+6);
 }
 //test pengurangan...
 public function testPengurangan()
 {
 $this->assertEquals(-2,5-6);
 }
 }

?>
```

2. Test *function/method* **pangkatBilangan()** pada *class* **Matematika.php** dibawah ini. Jelaskan!

   File *source code class* **Matematika.php**

```php
<?php

class Matematika
{
    /**
     * @param integer $bilanganBasic
     * @param integer $bilanganExponen
     *
     * @return integer
     */
    public static function pangkatBilangan($bilanganBasic, $bilanganExponen)
    {
        $nilaiSekarang = 1;
        for ($i = 1; $i <= $bilanganExponen; $i++) {
            $nilaiSekarang = $nilaiSekarang * $bilanganBasic;
        }
        return $nilaiSekarang;
    }
}
```

File tests **MatematikaTest.php**

```php
<?php
use PHPUnit\Framework\TestCase;
class MatematikaTest extends TestCase
{
    //Menguji pangkat bilangan positif, positif
    public function testPositifSemua()
    {
        $hasilPangkat = Matematika::pangkatBilangan(2, 3);
        $this->assertEquals(8, $hasilPangkat);
    }

    // Menguji pangkat bilangan negatif, positif
    public function testNegatifPositif()
    {
        $hasilPangkat = Matematika::pangkatBilangan(-3, 2);
        $this->assertEquals(9, $hasilPangkat);
    }

    //Menguji pangkat bilangan positif, negatif
    public function testPositifNegatif()
    {
        $hasilPangkat = Matematika::pangkatBilangan(4, -2);
        $this->assertEquals(1 / 16, $hasilPangkat);
    }

    //Menguji pangkat bilangan input tidak sesuai
    public function testInputNgawur()
    {
        $hasilPangkat = Matematika::pangkatBilangan("a", 4);
        $this->assertEquals(10, $hasilPangkat);
    }
}
?>
```

Dari keempat method yang ditest, kenapa hasilnya ada 1 *error* dan 1 *failure* ? Analisa kenapa **$hasilPangkat** pada *function/method* **testInputNgawur()** hasilnya error dan pada *function/method* **testPositifNegatif()** hasilnya gagal ?

```
D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit>vendor\bin\phpunit tests\MatematikaTest.php
PHPUnit 9.4.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.4.10
Configuration: D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\phpunit.xml

..FE                                                                4 / 4 (100%)

Time: 00:00.089, Memory: 6.00 MB

There was 1 error:

1) MatematikaTest::testInputNgawur
A non-numeric value encountered

D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\src\Matematika.php:16
D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\tests\MatematikaTest.php:50

--

There was 1 failure:

1) MatematikaTest::testPositifNegatif
Failed asserting that 1 matches expected 0.0625.

D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\tests\MatematikaTest.php:40

ERRORS!
Tests: 4, Assertions: 3, Errors: 1, Failures: 1.

D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit>
```

3. Buat class **file.php** dan file test **ClassTest.php** sebagaimana berikut:

   File *source code class* **file.php**

```php
<?php

$url = 'http://mediabisnisonline.com';

function create_link($url){

}

create_link($url);
```

   File tests **ClassTest.php**

```php
<?php
use PHPUnit\Framework\TestCase;

class TestClass extends TestCase
{
        public function testFile()
        {
                ob_start();
                include 'src\file.php';
                $content = ob_get_contents();
                ob_end_clean();
                $this->assertEquals('<a href="http://mediabisnisonline.com">Klik link ini</a>', $content);
        }
}
?>
```

assertEquals() *berfungsi untuk menyatakan apakah argumen pertama sama dengan argumen kedua. Fungsi ini digunakan untuk membandingkan dan nilainya antara kedua argumen* ***harus sama****. Jika tidak, maka hasil testnya gagal/failures serta dijelaskan penyebab gagalnya sebagaimana gambar dibawah ini.*

```
D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit>vendor\bin\phpunit tests\TestClass.php
PHPUnit 9.4.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.4.10
Configuration: D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\phpunit.xml

F                                                                   1 / 1 (100%)

Time: 00:00.050, Memory: 6.00 MB

There was 1 failure:

1) TestClass::testFile
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'<a href="http://mediabisnisonline.com">Klik link ini</a>'
+''

D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\tests\TestClass.php:12

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

Jelaskan mana argument pertama dan argument kedua! Bandingkan kenapa berbeda! Kemudian perbaiki code **file.php** sebagaimana berikut dan test kembali. Analisa hasilnya!

TestClass.php | file.php

```php
<?php

$url = 'http://mediabisnisonline.com';

function create_link($url){
  echo '<a href="'.$url.'">Klik link ini</a>';
}

create_link($url);

/*
<?php

$url = 'http://mediabisnisonline.com';

function create_link($url){

}

create_link($url);
*/
```

4. Buat class **Employee.php** dan file test-nya

Employee.php | EmployeeTest.php

```php
<?php

class Employee
{
    private $id;
    private $name;
    private $basicSalary;

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

    public function getId()
    {
        return $this->id;
    }

    public function getName()
    {
        return $this->name;
    }

    public function getBasicSalary()
    {
        return $this->basicSalary;
    }
}
```

File tests **EmployeeTest.php**

Employee.php | EmployeeTest.php

```php
<?php

use PHPUnit\Framework\TestCase;

class EmployeeTest extends TestCase
{
    /** @test */
    public function shouldCreateObject()
    {
        $id = 1;
        $name = 'John Smith';
        $basicSalary = 1000000;

        $obj = new Employee(
            $id,
            $name,
            $basicSalary
        );

        $this->assertEquals($id, $obj->getId());
        $this->assertEquals($name, $obj->getName());
        $this->assertEquals($basicSalary, $obj->getBasicSalary());
    }
}
```

Jelaskan hasilnya kenapa ada **3 assertions** yang berhasil dalam **sebuah test**! Sebutkan!

Coba ganti isi method **$name = 'John Smith';** dengan **$name = John Smith (tanpa '');**

Analisa hasilnya!

Unit Testing menggunakan PHPUnit TestDox

*The **--testdox** option produces a nicer output, with checkboxes ([✓]), where an unchecked box ([✗]) means the test failed.*

Symbol	Color	Meaning
✔	green	test passed
✗	red	assertion failed
✗	yellow	PHPUnit error or warning
Ø	yellow	incomplete test
☢	yellow	risky test
→	yellow	skipped test

5. Buatlah class **Average.php** dan file test-nya **AverageTest.php**, kemudian test menggunakan PHPUnit TestDox. Jelaskan hasilnya!

   File class **Average.php**

```php
<?php
class Average
{
    /**
     * Calculate the mean average
     * @param array $numbers Array of numbers
     * @return float Mean average
     */
    public function mean(array $numbers)
    {
        return array_sum($numbers) / count($numbers);
    }

    /**
     * Calculate the median average
     * @param array $numbers Array of numbers
     * @return float Median average
     */
    public function median(array $numbers)
    {
        sort($numbers);
        $size = count($numbers);
        if ($size % 2) {
            return $numbers[$size / 2];
        } else {
            return $this->mean(
                array_slice($numbers, ($size / 2) - 1, 2)
            );
        }
    }
}
```

   File tests **AverageTest.php**

```php
<?php

use PHPUnit\Framework\TestCase;

class AverageTest extends TestCase
{
    protected $Average;

    public function setUp() : void
    {
        $this->Average = new Average();
    }

    public function testCalculationOfMean()
    {
        $numbers = [3, 7, 6, 1, 5];
        $this->assertEquals(4.4, $this->Average->mean($numbers));
    }

    public function testCalculationOfMedian()
    {
        $numbers = [3, 7, 6, 1, 5];
        $this->assertEquals(5, $this->Average->median($numbers));
    }
}
```

```
D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit>vendor\bin\phpunit --testdox tests\AverageTest.php
PHPUnit 9.4.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.4.10
Configuration: D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\phpunit.xml

Average
 ✓ Calculation of mean  13 ms
 ✓ Calculation of median  1 ms

Time: 00:00.090, Memory: 6.00 MB

OK (2 tests, 2 assertions)
```

Analisa hasil testing berikut ini apabila hasil mean diubah dari **4.4** menjadi **4!**

```
D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit>vendor\bin\phpunit --testdox tests\AverageTest.php
PHPUnit 9.4.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.4.10
Configuration: D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\phpunit.xml

Average
 ✗ Calculation of mean  33 ms
   ┐
   ├ Failed asserting that 4.4 matches expected 4.
   │
   │ D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\tests\AverageTest.php:17
   ┴

 ✓ Calculation of median  2 ms

Time: 00:00.238, Memory: 6.00 MB

Summary of non-successful tests:

Average
 ✗ Calculation of mean  33 ms
   ┐
   ├ Failed asserting that 4.4 matches expected 4.
   │
   │ D:\#Semester Ganjil 2020\Bahan Ajar\Praktikum RPL\latihan-php-unit\tests\AverageTest.php:17
   ┴

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
```

Analisa hasil testing apabila median **$numbers** diubah menjadi **[3, 7, 6, 1, 9],** kemudian ubah isi **$numbers** sesuai dengan hasil testing tanpa merubah hasil median **[5]** !

Writing Tests for PHPUnit (Exceptions and Errors)

PHPUnit has a nice way of testing exceptions, using the **exceptException()** method to test whether an exception is thrown by the code under test.. PHPUnit converts errors to exceptions, errors behave the same way in unit tests as exceptions do. Any code that follows an error being triggered will not executed while it is being tested.

6. Jelaskan hasil pengujian dan analisa pengujian *error condition* dengan PHPUnit pada file **MyTest.php** dibawah ini:

MyTest.php

```php
<?php
use PHPUnit\Framework\TestCase;
class MyTest extends TestCase
{
    private $errors;
    protected function setUp() : void {
        $this->errors = array();
        set_error_handler(array($this, "errorHandler"));
    }
    public function errorHandler($errno, $errstr, $errfile, $errline, $errcontext) {
        $this->errors[] = compact("errno", "errstr", "errfile",
            "errline", "errcontext");
    }
    public function assertError($errstr, $errno) {
        foreach ($this->errors as $error) {
            if ($error["errstr"] === $errstr
                && $error["errno"] === $errno) {
                return;
            }
        }
        $this->fail("Error with level " . $errno .
            " and message '" . $errstr . "' not found in ",
            var_export($this->errors, TRUE));
    }
    public function testDoStuff() {
        // execute code that triggers a warning
        $this->assertError("Message for the expected error",
            E_USER_WARNING);
    }
}
```

7. Jelaskan hasil pengujian dan analisa *testing exceptions* pada file **ExceptionTest.php** dibawah ini:

ExceptionTest.php

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ExceptionTest extends TestCase
{
    public function testException(): void
    {
        $this->expectException(InvalidArgumentException::class);
    }
}
```

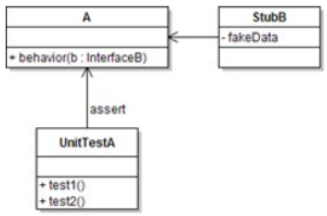
Test Doubles (Mocks and Stubs)

Ada beberapa definisi objek, yang tidak nyata. Istilah umum adalah *test doubles*/tes ganda.

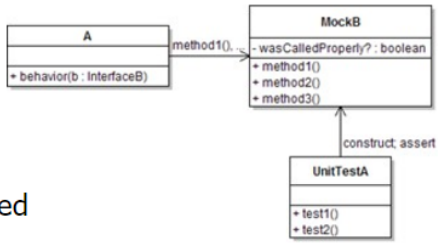
Istilah ini meliputi: stub dan mock.

Stubs vs. mocks

- A **stub** gives out data that goes to the object/class under test.
- The unit test directly asserts against class under test, to make sure it gives the right result when fed this data.



- A **mock** waits to be called by the class under test (A).
  - Maybe it has several methods it expects that A should call.
- It makes sure that it was contacted in exactly the right way.
  - If A interacts with B the way it should, the test passes.



21

8. Pengujian unit dengan *mocking.* Jelaskan apa yang dimaksud dengan *mocking* pada unit test kemudian analisa hasil pengujian source code dibawah ini.

   class **Logger.php**

```php
<?php

class Logger {
    public function log($text) {

        echo $text;
    }
}
```

   class **Application.php**

```php
<?php
class Application {
  public function run(Logger $logger) {
    // some code that starts up the application

    // send out a log that the application has started
   $logger->log('Application has started');

  }
}
?>
```

File test **ApplicationTest.php**

```php
<?php
use PHPUnit\Framework\TestCase;
class ApplicationTest extends TestCase {
  public function testThatRunLogsApplicationStart() {
    // create the observer
    $mock = $this->createMock(Logger::class);
    $mock->expects($this->once())
        ->method('log')
        ->with('Application has started');
    // run the application with the observer which ensures the log method was called
    $app = new Application();
    $app->run($mock);
  }
}
?>
```

9. Pengujian unit dengan *stubbing.* Jelaskan apa yang dimaksud dengan *stubbing* pada unit test kemudian analisa hasil pengujian source code dibawah ini.
   class **Answer.php**

```php
<?php declare(strict_types=1);
class Answer
{
    // prompt the user and check if the answer is yes or no, anything else, return null
    public function getYesNoAnswer($prompt) {

        $answer = $this->readUserInput($prompt);

        $answer = strtolower($answer);
        if (($answer === "yes") || ($answer === "no")) {
            return $answer;
        } else {
            return null;
        }

    }

    // Simply prompt the user and return the answer
    public function readUserInput($prompt) {
        return readline($prompt);
    }
}
?>
```

File tests **AnswerTest.php**

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;
class AnswerTest extends TestCase
{

    public function test_yes_no_answer() : void{

        $stub = $this->getMockBuilder(Answer::class)
                ->setMethods(["readUserInput"])
                ->getMock();

        $stub->method('readUserInput')
            ->will($this->onConsecutiveCalls("yes","junk"));

        // stub will return "yes"
        $answer = $stub->getYesNoAnswer("Student? (yes/no)");
        $this->assertSame("yes",$answer);

        // stub will return "junk"
        $answer = $stub->getYesNoAnswer("Student? (yes/no)");
        $this->assertNull($answer);
    }
}
?>
```

10. Jelaskan apa yang dimaksud dengan Fixtures pada unit test kemudian jelaskan hasil dari pengujian source code berikut

Card.php

Card.php CardTest.php

```php
<?php
class Card{
    private $number;
    private $suit;
    public function __construct($number, $suit)
    {
    $this->number = $number;
    $this->suit = $suit;
    }
    public function getNumber()
    {
    return $this->number;
    }
    public function getSuit()
    {
    return $this->suit;
    }
    public function isInMatchingSet(Card $card)
    {
    return ($this->getNumber() == $card->getNumber());
    }
}
```

CardTest.php

Card.php CardTest.php

```php
<?php
use PHPUnit\Framework\TestCase;
class CardTest extends TestCase {
    private $card;
    public function setUp() : void
    {
        $this->card = new Card('4', 'spades');
    }
    public function testGetNumber()
    {
        $actualNumber = $this->card->getNumber();
        $this->assertEquals(4, $actualNumber, 'Number should be <4>');
    }
    public function testGetSuit()
    {
        $actualSuit = $this->card->getSuit();
        $this->assertEquals('spades', $actualSuit, 'Suit should be <spades>');
    }
    public function testIsInMatchingSet()
    {
        $matchingCard = new Card('4', 'hearts');
        $this->assertTrue($this->card->isInMatchingSet($matchingCard),       '<4 of Spades> should match <4 of Hearts>');
    }
    public function testIsNotInMatchingSet()
    {
        $matchingCard = new Card('5', 'hearts');
        $this->assertFalse($this->card->isInMatchingSet($matchingCard),       '<4 of Spades> should not match <5 of Hearts>');
    }
}
```

Referensi:

Lively, M., 2013. *Instant Hands-on Testing with PHPUnit How-to*. Packt Publishing Ltd.