

LAPORAN PRAKTIKUM PENGUJIAN PERANGKAT LUNAK

Dibuat Oleh:

Evi Fitriya 1201222005

Program Studi Rekayasa Perangkat Lunak

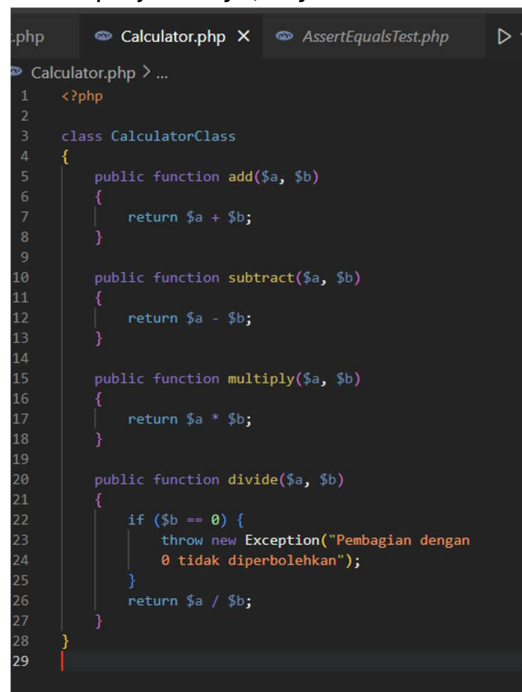
Fakultas Informatika

Telkom University

Surabaya

Pertemuan 6 (Unit Test)

- Buat Sebuah Folder di directory/file explorer
- Buka Folder
- Klik Kanan
- Pilih “Buka Terminal”
- Ketik “Code .”, Enter
- Vs Code akan terbuka dengan sendirinya
- Silahkan Install Composer dg cara:
 - o Buka terminal di vs Code
 - o Ketik “composer require --dev phpunit/phpunit”, enter
 - o Tunggu hingga proses selesai
- Silahkan buat terlebih dahulu programnya
- Untuk project saya, saya buat calculator Simple dg codingan seperti berikut:

A screenshot of a Visual Studio Code editor window. The top bar shows two tabs: 'Calculator.php' and 'AssertEqualsTest.php'. The 'Calculator.php' tab is active, displaying a PHP class named 'CalculatorClass'. The class contains four public methods: 'add(\$a, \$b)', 'subtract(\$a, \$b)', 'multiply(\$a, \$b)', and 'divide(\$a, \$b)'. The 'divide' method includes a check for division by zero, throwing an exception with the message 'Pembagian dengan 0 tidak diperbolehkan' if the denominator is zero. The code is as follows:

```
1 <?php
2
3 class CalculatorClass
4 {
5     public function add($a, $b)
6     {
7         return $a + $b;
8     }
9
10    public function subtract($a, $b)
11    {
12        return $a - $b;
13    }
14
15    public function multiply($a, $b)
16    {
17        return $a * $b;
18    }
19
20    public function divide($a, $b)
21    {
22        if ($b == 0) {
23            throw new Exception("Pembagian dengan
24                                0 tidak diperbolehkan");
25        }
26        return $a / $b;
27    }
28 }
29
```

- Selanjutnya kita prepare untuk testnya dengan membuat fungsi testnya:
 - o Buat Folder dengan nama “tests”, Open
 - o Kemudian buat file php untuk fungsi testnya
 - o Berikut contohnya:

- AssertContainsTest

```
tests > AssertContainsTest.php > ...
1  <?php
2
3  use PHPUnit\Framework\TestCase;
4
5  class AssertContainsTest extends TestCase
6  {
7      public function testAssertContains()
8      {
9          $buah = ['apel', 'jeruk', 'nanas', 'durian'];
10
11          $this->assertContains('apel', $buah,
12              'Tidak ada buah tsb di dalam array'); //benar
13
14      }
15  }
16
17
18  ?>
```

- AssertEqualsTest

```
<?php
use PHPUnit\Framework\TestCase;
class AssertEqualsTest extends TestCase
{
    public function testAssertEquals()
    {
        $this->assertEquals(20, 10 + 10,
            '10 + 10 hasilnya 20');
    }
}
?>
```

- AssertFalseTest

```
AssertFalseTest.php > PHP Intelephense > AssertFalseTest
<?php
use PHPUnit\Framework\TestCase;
class AssertFalseTest extends TestCase
{
    public function testAssertFalse()
    {
        $value = 24;
        $condition = is_string($value);
        $this->assertFalse($condition,
            'Value ini string'); // salah
    }
}
?>
```

- AssertSameTest

```
1  <?php
2
3  use PHPUnit\Framework\TestCase;
4
5  class AssertSameTest extends TestCase
6  {
7      public function testAssertSame()
8      {
9          $this->assertSame(5, 5,
10             "Harusnya sama!");
11      }
12  }
13
14
15  ?>
```

- AssertTrueTest

```
AssertTrueTest.php > PHP Intelephense > AssertTrueT
1  <?php
2
3  use PHPUnit\Framework\TestCase;
4
5  class AssertTrueTest extends TestCase
6  {
7      public function testAssertTrue()
8      {
9          $this->assertTrue(5 > 2);
10
11          $this->assertTrue(
12              is_array([1, 2, 3]),
13              'Ini bukan array');
14      }
15  }
16
17
18  ?>
```

■ CalculatorTest

```
CalculatorTest.php > PHP Intelephense > CalculatorTest > testAdd
<?php

use PHPUnit\Framework\TestCase;
require_once 'Calculator.php';

class CalculatorTest extends TestCase
{
    protected $calculator;

    protected function setUp(): void
    {
        $this->calculator = new CalculatorClass();
    }

    //pengecekan menggunakan assert equals karena calculator perlu validasi hasil perhitungan operasi

    public function testAdd()
    {
        $this->assertEquals(8,
            $this->calculator->add(2, 6), 'Penjumlahan 2 dan 6 seharusnya Memberikan nilai 8');
        $this->assertEquals(2, $this->calculator->add(1, 1), 'Penjumlahan 1 dan 1 seharusnya Memberikan nilai 2');
    }

    public function testSubtract()
    {
        $this->assertEquals(-2, $this->calculator->subtract(-1, 1), 'Pengurangan -1 oleh 1 seharusnya Memberikan nilai -2');
        $this->assertEquals(1, $this->calculator->subtract(-2, -3), 'Pengurangan -2 oleh -3 seharusnya Memberikan nilai 1');
    }

    public function testMultiply()
    {
        $this->assertEquals(4, $this->calculator->multiply(2, 2), 'Perkalian 2 dan 2 seharusnya Memberikan nilai 4');
        $this->assertEquals(-6, $this->calculator->multiply(-2, 3), 'Perkalian -2 dan 3 seharusnya Memberikan nilai -6');
    }

    public function testDivide()
    {
        $this->assertEquals(2, $this->calculator->divide(8, 4), 'Pembagian 8 oleh 4 seharusnya Memberikan nilai 2');
        $this->assertEquals(2, $this->calculator->divide(2, 1), 'Pembagian 2 oleh 1 seharusnya Memberikan nilai 2');
    }

    public function testDivideByZero()
    {
        $this->expectException(Exception::class);
        $this->expectExceptionMessage('Pembagian dengan nol tidak diperbolehkan');
        $this->calculator->divide(1, 0);
    }
}
```

■ User

```
User.php > PHP Intelephense > User > activate
<?php
class User
{
    public string $name;
    public string $email;
    public bool $active;

    public function __construct($name, $email, $active = false)
    {
        $this->name = $name;
        $this->email = $email;
        $this->active = $active;
    }

    public function isValidEmail(): bool
    {
        return filter_var($this->email, FILTER_VALIDATE_EMAIL) !== false;
    }

    public function activate()
    {
        $this->active = true;
    }
}
```

- Lalu kita buat apa yg akan dites:

```

erTest.php > PHP Intelephense > UserTest > testActivateUser
<?php

require_once "User.php";

use PHPUnit\Framework\TestCase;

class UserTest extends TestCase
{
    public function testUserProperties()
    {
        $user = new User("John Doe", "john@example.com");

        // assertTrue: Memeriksa apakah bernilai true
        $this->assertTrue($user->isValidEmail(), "Email tidak valid");

        // assertFalse: Memeriksa apakah bernilai false
        $this->assertFalse($user->active, "User harusnya tidak aktif");

        // assertEquals: Memeriksa apakah nilai sama
        $this->assertEquals("John Doe", $user->name, "Nama tidak sama");
        $this->assertEquals("john@example.com", $user->email, "Email tidak sama");

        // assertSame: Memeriksa apakah referensi sama
        $this->assertSame("John Doe", $user->name, "Nama tidak sama");
    }
}

```

```

public function testEmailValidation()
{
    /// Format email: namaEmail @ domain

    $validUser = new User("Alice", "alice@example.com");
    $invalidUser = new User("Bob", "invalid-email");

    // assertTrue: Email valid harus return true
    $this->assertTrue($validUser->isValidEmail()); // true --> SUCCESS

    // assertFalse: Email invalid harus return false
    $this->assertFalse($invalidUser->isValidEmail()); // false --> SUCCESS
}

public function testSameReference()
{
    $user1 = new User("User1", "user1@example.com");
    $user2 = $user1;

    $user3 = new User("Charlie", "charlie@example.com");

    // assertSame: Pastikan referensi sama
    $this->assertSame($user1, $user2); // sama --> SUCCESS
}

```

```

public function testActivateUser()
{
    $user = new User("Charlie", "charlie@example.com");

    // assertFalse sebelum diaktifkan
    $this->assertFalse($user->active); // false --> SUCCESS

    // Aktivasi user
    $user->activate();

    // assertTrue setelah diaktifkan
    $this->assertTrue($user->active); // true --> SUCCESS
}

public function testUserExistsInArray()
{
    $user1 = new User("Alice", "alice@example.com");
    $user2 = new User("Bob", "bob@example.com");
    $user3 = new User("Charlie", "charlie@example.com");
    $user4 = new User("Joko", "joko@mail.com");

    $users = [$user1, $user2, $user3];

    // assertContains: Memastikan bahwa $user2 ada dalam array $users
    $this->assertContains($user2, $users);
}

```

```
public function testStringInArray()
{
    $fruits = ["Apple", "Banana", "Orange"];

    // assertContains: Memastikan bahwa "Banana" ada di dalam array
    $this->assertContains("Banana", $fruits);
}

public function testObjectInstance()
{
    $user = new User("David", "david@example.com");

    // assertInstanceOf: Pastikan objek bertipe User
    $this->assertInstanceOf(User::class, $user);
}

public function testArrayHasKey()
{
    $userData = ["name" => "Eve", "email" => "eve@example.com"];

    // assertArrayHasKey: Pastikan array punya key "email"
    $this->assertArrayHasKey("email", $userData);
}

public function testCountUsers()
{
    $users = [
        new User("User1", "user1@example.com"),
        new User("User2", "user2@example.com")
    ];

    // assertCount: Pastikan ada 2 user
    $this->assertCount(2, $users);
}
```

```

public function testStringContains()
{
    $message = "Welcome to PHPUnit testing";

    // assertStringContainsString: Pastikan teks mengandung kata "PHPUnit"
    $this->assertStringContainsString("PHPUnit", $message);
}

public function testNotSameReference()
{
    $user1 = new User("User1", "user1@example.com");
    $user2 = new User("User1", "user1@example.com");

    $user3 = $user1;

    // assertNotSame: Pastikan referensi berbeda
    $this->assertNotSame($user1, $user2); // beda --> SUCCESS

    $this->assertNotSame($user1, $user3); // sama --> FAILED
}

```

- Lalu Run dengan buka terminal ketik “./vendor/bin/phpunit --bootstrap Calculator.php tests”
- Enter
- Tunggu hingga muncul seperti berikut:

```

PS C:\APRIBADI\EVI FITRIYA\ITTELKOM\SEMESTER 6\PENGUJIAN - NIE\TUGAS\UNIT TEST> ./vendor/bin/ph
punit --bootstrap Calculator.php tests
PHPUnit 10.5.45 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.25

.....F                                     10 / 10 (100%)

Time: 00:01.583, Memory: 8.00 MB

There was 1 failure:

1) CalculatorTest::testDivideByZero
Failed asserting that exception message 'Pembagian dengan
    0 tidak diperbolehkan' contains 'Pembagian dengan nol tidak diperbolehkan'.

FAILURES!
Tests: 10, Assertions: 16, Failures: 1.
PS C:\APRIBADI\EVI FITRIYA\ITTELKOM\SEMESTER 6\PENGUJIAN - NIE\TUGAS\UNIT TEST>

```

artinya ada 10 test yg dilakukan, dengan 16 Assertion, semuanya berhasil kecuali ada 1 yang gagal, total waktu test 1 detik 583 mili detik.

- Selesai
- Link Github: <https://github.com/evftrya/Pengujian-Perangkat-Lunak/tree/main/TUGAS/UNIT%20TEST>