phpunit.xml | PracticeTest.php | routes.php | statistic.php | runner.php | composer.json

```php
<?php namespace Statistic;

class Statistic
{
    private $data;
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;
    private $validated;

    public function __construct($Data)
    {
        if($this->validate($Data))
        {
            $this->validated = $this->validate($Data);
            $this->data = $this->radixSort($Data);
            $this->isOdd = (count($Data)%2 == 0) ? false : true;
        }
    }
```

SOFTWARE TESTING

OLEH: RIJALUL FIKRI

Requirement

Sebelum bisa merancang test case kita terlebih dahulu membutuhkan requirement program, agar test case tidak melenceng jauh dari hasil yang diharapkan.

Adapun requirement dari aplikasi adalah sebagai berikut:

- Aplikasi menerima inputan data berupa array
- Anggota array tersebut harus berupa bilangan integer
- Anggota array minimal terdapat 1 data
- Aplikasi menghasilkan **median** dan **rata-rata** dari data yang diberikan
- Inputan data berupa array akan otomatis di sortir oleh sistem (menggunakan radix sort)
- Rumus rata-rata:

$$\text{rata-rata} = \frac{x_1+x_2+x_3+\ldots+x_n}{n}$$

- Rumus Median:

$$\text{Median}_{\text{(jumlah bilangan ganjil)}} = \frac{\text{jumlah array}}{2}$$

dari rumus diatas akan didapatkan index array dari mediannya. Kemudian nilai dari array dengan index tersebut dikembalikan

$$\text{Index Median}_{\text{(jumlah bilangan genap)}} = \frac{\text{jumlah array}}{2}$$

$$\text{Median} = \frac{\text{array[index]} + \text{array[index+1]}}{2}$$

Akan ada beberapa metoda / fungsi yang diperlukan untuk membuat aplikasi tersebut diantaranya:

- constructor (berfungsi sebagai setter, initial sorting, validasi, dsb)
- radix_sort (berfungsi untuk menyortir array yang diberikan secara ascending)
- validate (berfungsi untuk memvalidasi data yang diberikan ke objek)
- getData (kembalikan nilai datanya yang telah disortir)
- getMedian (berfungsi untuk mengambil Median)
- getAverage (berfungsi untuk mengambil Rata-Rata)

RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

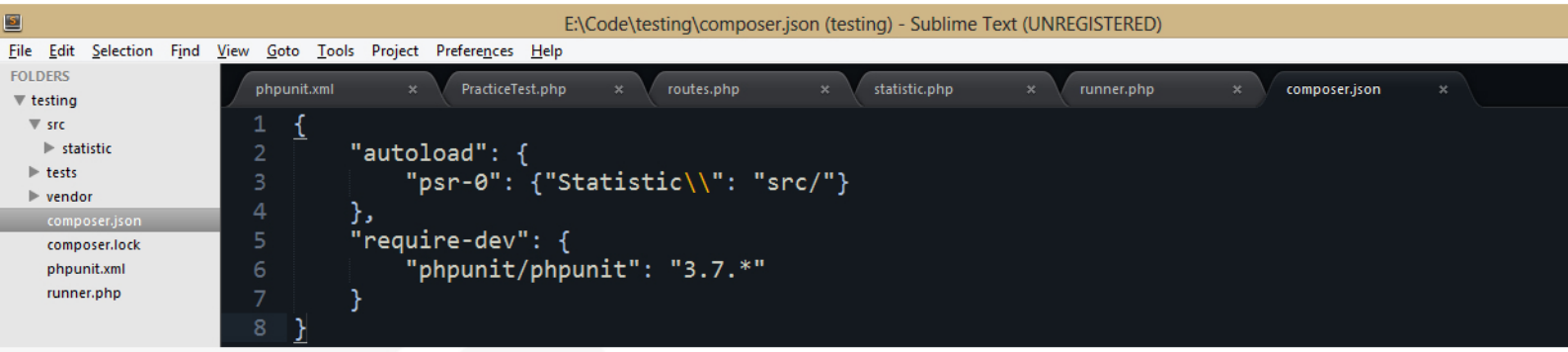
Prerequisite

Bahasa pemrograman yang akan penulis pakai untuk melakukan testing adalah PHP, oleh karena itu penulis membutuhkan dependency tambahan untuk membantu penulis dalam melakukan testing yaitu dengan menggunakan **PHPUnit** serta **Composer** untuk memanage dependency nya.

PHPUnit

PHPUnit adalah sebuah framework unit testing yang di buat oleh ***Sebastian Bergmann***. PHPUnit menggunakan pernyataan untuk memverifikasi bahwa kode yang diuji (biasa disebut SUT, atau System Under Test) berperilaku seperti yang diharapkan.

Diasumsikan, composer sudah terinstall secara global pada komputer, maka kita tinggal menambahkan sebuah file baru bernama composer.json di folder utama kemudian tambahkan baris berikut didalam file tersebut.



Composer akan membaca secara otomatis file-file yang terdapat dalam folder src dalam namespace 'Statistic' (penulis menggunakan standar psr-0) serta aplikasi ini akan membutuhkan phpunit versi 3.7.*

Kemudian kita butuh untuk mengupdate dependency nya dengan mengetikkan:

```
composer install --dev
```

Untuk menjalankan phpunit cukup dengan menjalankan excecutable phpunit kemudian namafile yang akan ditest, dalam hal ini php unit terletak didalam folder **vendor/bin/phpunit** kemudian file yang akan di test didalam folder **tests/PracticeTest.php**

RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

Implementasi Testing

Test case yang dirancang disesuaikan dengan requirement seperti yang telah dibahas pada halaman pertama. Test case yang dibuat antara lain:

Pastikan bahwa data yang diberikan bertipe array

Test ini bertujuan untuk memastikan bahwa data yang diberikan pada saat pembuatan instan kelas benar-benar bertipe array. PHPUnit memiliki metoda **assertInternalType** untuk mencocokkan tipe data.

E:\Code\testing\tests\PracticeTest.php (testing) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
- testing
  - src
    - statistic
  - tests
  - vendor
  - composer.json
  - composer.lock
  - phpunit.xml
  - runner.php

phpunit.xml | PracticeTest.php | routes.php | statistic.php | runner.php

```php
<?php
class PracticeTest extends PHPUnit_Framework_TestCase
{

    public function testReturnTrueIfParameterOfDataTypeArray()
    {
        //data yang diberikan
        $data = [1,3,7,15,2,5,10];

        //eksekusi scriptnya
        //buat instance baru dari objek statistik dengan data
        //yang diberikan tadi
        $test = new \Statistic\Statistic($data);

        //nyatakan bahwa data yang diberikan bertipe array
        $this->assertInternalType('array', $test->getData()); // true
    }
```

Pastikan program akan mereturn NULL jika data yang diberikan bukan bertipe Array

Pada Test Case ini misal data yang diberikan bukan bertipe array maka seharusnya 3 metoda utama dalam class tersebut akan mengembalikan nilai NULL, hal ini kita nyatakan dengan menggunakan metoda **assertEquals** dari PHPUnit yang membutuhkan parameter yaitu **assertEqual($expected, $actual)**.

runner.php

```php
    public function testReturnNullIfDataTypeNotArray()
    {
        //data yang diberikan
        $data = 15;

        //eksekusi scriptnya
        //buat instance baru dari objek statistik
        $test = new \Statistic\Statistic($data);

        //nyatakan bahwa aplikasi
        //akan mengembalian nilai NULL
        //jika data yang diberikan tidak bertipe array
        $this->assertEquals(NULL, $test->getData()); // true
        $this->assertEquals(NULL, $test->getMedian()); // true
        $this->assertEquals(NULL, $test->getAverage()); // true
    }
```


RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

```
    private $data = array();
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;

    public function __construct($data)
```

Pastikan bahwa panjang data yang diberikan minimal satu

Sesuai dengan requirement pada halaman pertama bahwa aplikasi akan membutuhkan minimal satu data, jika data tidak diberikan maka akan menyebabkan division by zero pada operasi median dan rata-rata. Berikut test case untuk mengatasi hal tersebut.

phpunit.xml
runner.php

```
36      public function testReturnNullIfDataLengthLessThenOne()
37      {
38          $data = 0;
39
40          //eksekusi scriptnya
41          //buat instance baru dari objek statistik
42          $test = new \Statistic\Statistic($data);
43
44          //nyatakan bahwa setiap metode atau fungsi
45          //akan menghasilkan NULL jika panjang data
46          //kurang dari satu
47          $this->assertEquals(NULL, $test->getData()); // true
48          $this->assertEquals(NULL, $test->getMedian()); // true
49          $this->assertEquals(NULL, $test->getAverage()); // true
50      }
```

Dari Test case diatas dapat kita lihat bahwa untuk membuat test tersebut lolos maka kode kita nantinya harus mengembalikan nilai NULL jika panjang datanya kurang dari satu. Jika aplikasi kita tidak mengakomodir hal tersebut maka test tersebut akan gagal.

Pastikan bahwa program hanya menerima angka sebagai anggota dalam array

Pada test case ini kita mensimulasikan jika ada anggota array yang bukan angka maka nilai variabel **validated** yang di set pada saat konstruktor dipanggil menghasilkan nilai **FALSE** dan sebaliknya jika seluruhnya angka maka akan menghasilkan **TRUE**.

Berikut test case untuk mensimulasikan hal tersebut.

```
52      public function testReturnTrueIfArrayContainNumber()
53      {
54          //berikan data
55          $data = [1,3,5,7,10];
56          $data2 = [1,a,15,20];
57
58          //inisiasi objek
59          $test = new \Statistic\Statistic($data);
60          $test2 = new \Statistic\Statistic($data2);
61
62          //asersi pertama ini harusnya menghasilkan true
63          //karena data yang diberikan semuanya angka
64          $this->assertEquals(TRUE, $test->validated); // true
65          //asersi kedua ini harusnya menghasilkan false
66          //karena data yang diberikan memiliki huruf
67          $this->assertEquals(FALSE, $test2->validated); // true
68
69      }
```


RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

```
{
    private $data = array();
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;

    public function __construct($data)
```

Pastikan bahwa data yang diberikan dapat tersortir secara ascending

Pada requirement data yang diberikan harus di sortir dengan urutan ascending, penulis menggunakan algoritma Radix Sort untuk melakukan hal tersebut. Berikut test case nya:

composer.json
composer.lock
phpunit.xml
runner.php

```
71      public function testRadixSortingInAscendingOrder()
72      {
73          //data yang diberikan
74          $data = array(1,3,7,15,2,4);
75          //actual
76          $test = new \Statistic\Statistic($data);
77          $actual = $test->radixSort($data);
78          //expected
79          $expected = array(1,2,3,4,7,15);
80
81
82          $this->assertEquals($expected,$actual); //true
83      }
```

Pastikan bahwa metoda untuk mengambil rata-rata berjalan dengan semestinya

Berikut test case untuk membuktikan bahwa program memberikan nilai yang seharusnya untuk rata-rata.

vendor
composer.json
composer.lock
phpunit.xml
runner.php

```
85      public function testGettingCorrectAverage()
86      {
87          //data yang diberikan
88          $data = array(8,8,8,8);
89          $data2 = array(8,7,9,10);
90
91          //hasil yang diharapkan
92          $expected = 8; //dah jelas harusnya rata2 nya 8
93          $expected2 = 8.5; // berdasar rumus rata2 (34/2 = 8.5)
94
95          //assert
96          $test = new \Statistic\Statistic($data);
97          $test2 = new \Statistic\Statistic($data2);
98          $this->assertEquals($expected,$test->getAverage());
99          $this->assertEquals($expected2,$test2->getAverage());
100     }
```


RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

```
    private $data = array();
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;

    public function __construct($data)
```

Test Case untuk Median dengan panjang data ganjil

Untuk mencari median, jika datanya berjumlah ganjil maka hasil panjang data dibagi dengan dua kemudian ditambah dengan satu akan menghasilkan angka index dari nilai median tersebut. Berikut test case nya.

```
102     public function testGettingCorrectMedianForOddNumber()
103     {
104         //data yang diberikan
105         $data = array(1,3,5,7,10);
106
107         //hasil yang diharapkan
108         $expected = 5; //nilai tengah yang harusnya keluar 5
109
110         //assert
111         $test = new \Statistic\Statistic($data);
112         $this->assertEquals($expected,$test->getMedian());
113     }
```

Test Case untuk Median dengan panjang data genap

Sesuai dengan rumus median di halaman pertama kalau datanya genap maka setelah dibagi dua, akan ditambahkan elemen sebelum dan sesudah nilai tengah tersebut kemudian dibagi dengan dua. Berikut test case nya.

vendor
composer.json
composer.lock
phpunit.xml
runner.php

```
115     public function testGettingCorrectMedianForEvenNumber()
116     {
117         //data yang diberikan
118         $data = array(1,3,5,7,10,15);
119
120         //hasil yang diharapkan ((5+7)/2 = 6)
121         $expected = 6;
122
123         //assert
124         $test = new \Statistic\Statistic($data);
125         $this->assertEquals($expected,$test->getMedian());
126     }
127
128 }
```


RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

```
    private $data = array();
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;

    public function __construct($data)
```

Implementasi Kode

Berikut kode yang diperlukan untuk membuat test case - test case tadi lolos uji.

E:\Code\testing\src\statistic\statistic.php (testing) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

phpunit.xml | PracticeTest.php | routes.php | statistic.php | runner.php | composer.json

```php
<?php namespace Statistic;

class Statistic
{
    //deklarasi variabel
    private $data;
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;
    public $validated;

    public function __construct($Data)
    {
        if($this->validate($Data))
        {
            $this->validated = $this->validate($Data); //validasi datanya
            $this->data = $this->radixSort($Data); //jalankan sortir kemudian simpan data
            $this->isOdd = (count($Data)%2 == 0) ? false : true; //cek genap atau ganjil
        }
    }

    public function getData()
    {
        return $this->data; //kembalikan data yang telah di sortir
    }

    public function validate($Data)
    {
        if(!is_array($Data))
        {
            $this->data = NULL; //klo bukan array harusnya data jadi NULL
            return FALSE; //kemudian return false
        }

        if(count($Data)<1)
        {
            return FALSE; //klo data kurang dari 1 juga return false
        }
        else
        {
            foreach($Data as $data)
            {
                if(!is_numeric($data)) return FALSE; //klo data bukan angka juga return false
            }

            $this->length = count($Data); //klo angka semua set panjang datanya berapa
            return TRUE; //kemudian return true
        }
    }

    public function getAverage()
    {
        if((!$this->validated)){ return false; } //klo validasi lolos

        $hasil = 0;
        foreach($this->data as $angka)
        {
            $hasil = $hasil+$angka; //tambahkan seluruh anggota array
        }
        return $this->average = $hasil/$this->length; //hasil tambah dibagi jumlah
    }
```


RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

```
{
    private $data = array();
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;

    public function __construct($data)
```

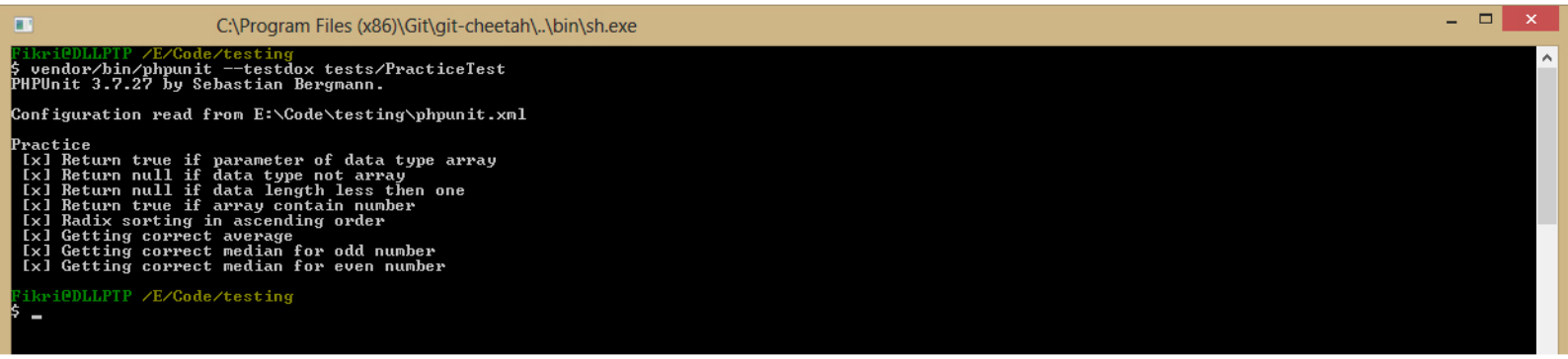
Lanjutan dari halaman sebelumnya.

```
    public function getAverage()
    {
        if((!$this->validated)){ return false; } //klo validasi lolos

        $hasil = 0;
        foreach($this->data as $angka)
        {
            $hasil = $hasil+$angka; //tambahkan seluruh anggota array
        }
        return $this->average = $hasil/$this->length; //hasil tambah dibagi jumlah
    }

    public function getMedian()
    {
        if((!$this->validated)){ return false; } //klo validasi lolos

        $this->indexMedian = ceil($this->length/2); //hitung nilai tengah, dibulatkan keatas

        if($this->isOdd)
        {
            //klo ganjil
            return $this->median = $this->data[$this->indexMedian-1];
        }
        else
        {
            //klo genap
            return $this->median = ($this->data[$this->indexMedian-1]+$this->data[$this->indexMedian])/2;
        }
    }

    //fungsi sorting nya nih
    public function radixSort($input)
    {
        $temp = $output = array();
        $len = count($input);

        for ($i = 0; $i < $len; $i++) {
            $temp[$input[$i]] = ($temp[$input[$i]] > 0)
                ? ++$temp[$input[$i]]
                : 1;
        }

        ksort($temp);

        foreach ($temp as $key => $val) {
            if ($val == 1) {
                $output[] = $key;
            } else {
                while ($val--) {
                    $output[] = $key;
                }
            }
        }

        return $output;
    }
}
```


RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name

```
    private $data = array();
    private $length;
    private $indexMedian;
    private $median;
    private $average;
    private $isOdd;

    public function __construct($data)
```

Berikut hasil jika test dijalankan



Semua kode pada dokumen ini bisa didownload dari github penulis dengan alamat:

https://github.com/desertlion/ugm-testcase

RIJALUL FIKRI
13/356469/PPA/04428
fikri.desertlion@gmail.com
http://rijalulfikri.name