

```

phpunit.xml   x PracticeTest.php   x routes.php   x statistic.php   x runner.php   x composer.json   x
1 <?php namespace Statistic;
2
3 class Statistic
4 {
5     private $data;
6     private $length;
7     private $indexMedian;
8     private $median;
9     private $average;
10    private $isOdd;
11    private $validated;
12
13    public function __construct($Data)
14    {
15        if($this->validate($Data))
16        {
17            $this->validated = $this->validate($Data);
18            $this->data = $this->radixSort($Data);
19            $this->isOdd = (count($Data)%2 == 0) ? false : true;
20        }
21    }

```

# 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:

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

- » Rumus Median:

$$\text{Median} = \frac{\text{jumlah array}}{2}$$

(jumlah bilangan ganjil)

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

$$\text{Index Median} = \frac{\text{jumlah array}}{2}$$

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

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 )

```

4 private $data = array();
5 private $length;
6 private $indexMedian;
7 private $median;
8 private $average;
9 private $isOdd;
10
11 public function __construct($data)
12 {
13     $this->$data = $data;
14 }

```

# 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.

The screenshot shows the Sublime Text interface with the following file structure:

- File: E:\Code\testing\composer.json (testing) - Sublime Text (UNREGISTERED)
- File menu: File Edit Selection Find View Goto Tools Project Preferences Help
- Folders: testing
  - src
    - statistic
  - tests
  - vendor
- Files: composer.json, composer.lock, phpunit.xml, runner.php

The content of composer.json is:

```

1  {
2      "autoload": {
3          "psr-0": {"Statistic\\": "src/"}
4      },
5      "require-dev": {
6          "phpunit/phpunit": "3.7.*"
7      }
8  }

```

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 executable 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**

The terminal window shows the following output:

```

C:\Program Files (x86)\Git\git-cheetah\..\.bin\sh.exe
Fikri@DLLPTP ~E:/Code/testing
$ vendor/bin/phpunit tests/PracticeTest
PHPUnit 3.7.27 by Sebastian Bergmann.

Configuration read from E:/Code/testing/phpunit.xml

Time: 29 ms, Memory: 1.75Mb
OK (1 test, 1 assertion)
Fikri@DLLPTP ~E:/Code/testing
$ -

```

```

4   private $data = array();
5   private $length;
6   private $indexMedian;
7   private $median;
8   private $average;
9   private $isOdd;
10
11  public function __construct($data)
12  {
13      $this->setData($data);
14  }
15
16  public function getMedian()
17  {
18      if ($this->isOdd) {
19          return $this->data[$this->indexMedian];
20      } else {
21          return ($this->data[$this->indexMedian] + $this->data[$this->indexMedian - 1]) / 2;
22      }
23  }
24
25  public function getAverage()
26  {
27      $sum = 0;
28      foreach ($this->data as $value) {
29          $sum += $value;
30      }
31      return $sum / count($this->data);
32  }
33
34  public function setIsOdd($isOdd)
35  {
36      $this->isOdd = $isOdd;
37  }
38
39  public function setData($data)
40  {
41      $this->data = $data;
42  }
43
44  public function setLength($length)
45  {
46      $this->length = $length;
47  }
48
49  public function setIndexMedian($indexMedian)
50  {
51      $this->indexMedian = $indexMedian;
52  }
53
54  public function setMedian($median)
55  {
56      $this->median = $median;
57  }
58
59  public function setSum($sum)
60  {
61      $this->sum = $sum;
62  }
63
64  public function setTotal($total)
65  {
66      $this->total = $total;
67  }
68
69  public function setOdd($odd)
70  {
71      $this->odd = $odd;
72  }
73
74  public function setEven($even)
75  {
76      $this->even = $even;
77  }
78
79  public function setLength($length)
80  {
81      $this->length = $length;
82  }
83
84  public function setIndexMedian($indexMedian)
85  {
86      $this->indexMedian = $indexMedian;
87  }
88
89  public function setMedian($median)
90  {
91      $this->median = $median;
92  }
93
94  public function setSum($sum)
95  {
96      $this->sum = $sum;
97  }
98
99  public function setTotal($total)
100 {
101     $this->total = $total;
102 }
103
104 public function setOdd($odd)
105 {
106     $this->odd = $odd;
107 }
108
109 public function setEven($even)
110 {
111     $this->even = $even;
112 }
113
114 public function setLength($length)
115 {
116     $this->length = $length;
117 }
118
119 public function setIndexMedian($indexMedian)
120 {
121     $this->indexMedian = $indexMedian;
122 }
123
124 public function setMedian($median)
125 {
126     $this->median = $median;
127 }
128
129 public function setSum($sum)
130 {
131     $this->sum = $sum;
132 }
133
134 public function setTotal($total)
135 {
136     $this->total = $total;
137 }
138
139 public function setOdd($odd)
140 {
141     $this->odd = $odd;
142 }
143
144 public function setEven($even)
145 {
146     $this->even = $even;
147 }
148
149 public function setLength($length)
150 {
151     $this->length = $length;
152 }
153
154 public function setIndexMedian($indexMedian)
155 {
156     $this->indexMedian = $indexMedian;
157 }
158
159 public function setMedian($median)
160 {
161     $this->median = $median;
162 }
163
164 public function setSum($sum)
165 {
166     $this->sum = $sum;
167 }
168
169 public function setTotal($total)
170 {
171     $this->total = $total;
172 }
173
174 public function setOdd($odd)
175 {
176     $this->odd = $odd;
177 }
178
179 public function setEven($even)
180 {
181     $this->even = $even;
182 }
183
184 public function setLength($length)
185 {
186     $this->length = $length;
187 }
188
189 public function setIndexMedian($indexMedian)
190 {
191     $this->indexMedian = $indexMedian;
192 }
193
194 public function setMedian($median)
195 {
196     $this->median = $median;
197 }
198
199 public function setSum($sum)
200 {
201     $this->sum = $sum;
202 }
203
204 public function setTotal($total)
205 {
206     $this->total = $total;
207 }
208
209 public function setOdd($odd)
210 {
211     $this->odd = $odd;
212 }
213
214 public function setEven($even)
215 {
216     $this->even = $even;
217 }
218
219 public function setLength($length)
220 {
221     $this->length = $length;
222 }
223
224 public function setIndexMedian($indexMedian)
225 {
226     $this->indexMedian = $indexMedian;
227 }
228
229 public function setMedian($median)
230 {
231     $this->median = $median;
232 }
233
234 public function setSum($sum)
235 {
236     $this->sum = $sum;
237 }
238
239 public function setTotal($total)
240 {
241     $this->total = $total;
242 }
243
244 public function setOdd($odd)
245 {
246     $this->odd = $odd;
247 }
248
249 public function setEven($even)
250 {
251     $this->even = $even;
252 }
253
254 public function setLength($length)
255 {
256     $this->length = $length;
257 }
258
259 public function setIndexMedian($indexMedian)
260 {
261     $this->indexMedian = $indexMedian;
262 }
263
264 public function setMedian($median)
265 {
266     $this->median = $median;
267 }
268
269 public function setSum($sum)
270 {
271     $this->sum = $sum;
272 }
273
274 public function setTotal($total)
275 {
276     $this->total = $total;
277 }
278
279 public function setOdd($odd)
280 {
281     $this->odd = $odd;
282 }
283
284 public function setEven($even)
285 {
286     $this->even = $even;
287 }
288
289 public function setLength($length)
290 {
291     $this->length = $length;
292 }
293
294 public function setIndexMedian($indexMedian)
295 {
296     $this->indexMedian = $indexMedian;
297 }
298
299 public function setMedian($median)
300 {
301     $this->median = $median;
302 }
303
304 public function setSum($sum)
305 {
306     $this->sum = $sum;
307 }
308
309 public function setTotal($total)
310 {
311     $this->total = $total;
312 }
313
314 public function setOdd($odd)
315 {
316     $this->odd = $odd;
317 }
318
319 public function setEven($even)
320 {
321     $this->even = $even;
322 }
323
324 public function setLength($length)
325 {
326     $this->length = $length;
327 }
328
329 public function setIndexMedian($indexMedian)
330 {
331     $this->indexMedian = $indexMedian;
332 }
333
334 public function setMedian($median)
335 {
336     $this->median = $median;
337 }
338
339 public function setSum($sum)
340 {
341     $this->sum = $sum;
342 }
343
344 public function setTotal($total)
345 {
346     $this->total = $total;
347 }
348
349 public function setOdd($odd)
350 {
351     $this->odd = $odd;
352 }
353
354 public function setEven($even)
355 {
356     $this->even = $even;
357 }
358
359 public function setLength($length)
360 {
361     $this->length = $length;
362 }
363
364 public function setIndexMedian($indexMedian)
365 {
366     $this->indexMedian = $indexMedian;
367 }
368
369 public function setMedian($median)
370 {
371     $this->median = $median;
372 }
373
374 public function setSum($sum)
375 {
376     $this->sum = $sum;
377 }
378
379 public function setTotal($total)
380 {
381     $this->total = $total;
382 }
383
384 public function setOdd($odd)
385 {
386     $this->odd = $odd;
387 }
388
389 public function setEven($even)
389 {
390     $this->even = $even;
391 }
392
393 public function setLength($length)
394 {
395     $this->length = $length;
396 }
397
398 public function setIndexMedian($indexMedian)
399 {
400     $this->indexMedian = $indexMedian;
401 }
402
403 public function setMedian($median)
404 {
405     $this->median = $median;
406 }
407
408 public function setSum($sum)
408 {
409     $this->sum = $sum;
410 }
411
412 public function setTotal($total)
413 {
414     $this->total = $total;
415 }
416
417 public function setOdd($odd)
418 {
419     $this->odd = $odd;
420 }
421
422 public function setEven($even)
423 {
424     $this->even = $even;
425 }
426
427 public function setLength($length)
428 {
429     $this->length = $length;
430 }
431
432 public function setIndexMedian($indexMedian)
433 {
434     $this->indexMedian = $indexMedian;
435 }
436
437 public function setMedian($median)
438 {
439     $this->median = $median;
440 }
441
442 public function setSum($sum)
442 {
443     $this->sum = $sum;
444 }
445
446 public function setTotal($total)
447 {
448     $this->total = $total;
449 }
450
451 public function setOdd($odd)
452 {
453     $this->odd = $odd;
454 }
455
456 public function setEven($even)
457 {
458     $this->even = $even;
459 }
460
461 public function setLength($length)
462 {
463     $this->length = $length;
464 }
465
466 public function setIndexMedian($indexMedian)
467 {
468     $this->indexMedian = $indexMedian;
469 }
470
471 public function setMedian($median)
472 {
473     $this->median = $median;
474 }
475
476 public function setSum($sum)
476 {
477     $this->sum = $sum;
478 }
479
479 public function setTotal($total)
480 {
481     $this->total = $total;
482 }
483
484 public function setOdd($odd)
485 {
486     $this->odd = $odd;
487 }
488
489 public function setEven($even)
489 {
490     $this->even = $even;
491 }
492
493 public function setLength($length)
494 {
495     $this->length = $length;
496 }
497
498 public function setIndexMedian($indexMedian)
499 {
500     $this->indexMedian = $indexMedian;
501 }
502
503 public function setMedian($median)
504 {
505     $this->median = $median;
506 }
507
508 public function setSum($sum)
508 {
509     $this->sum = $sum;
510 }
511
511 public function setTotal($total)
512 {
513     $this->total = $total;
514 }
515
516 public function setOdd($odd)
517 {
518     $this->odd = $odd;
519 }
520
521 public function setEven($even)
522 {
523     $this->even = $even;
524 }
525
526 public function setLength($length)
527 {
528     $this->length = $length;
529 }
530
531 public function setIndexMedian($indexMedian)
532 {
533     $this->indexMedian = $indexMedian;
534 }
535
536 public function setMedian($median)
537 {
538     $this->median = $median;
539 }
540
541 public function setSum($sum)
541 {
542     $this->sum = $sum;
543 }
544
544 public function setTotal($total)
545 {
546     $this->total = $total;
547 }
548
549 public function setOdd($odd)
550 {
551     $this->odd = $odd;
552 }
553
554 public function setEven($even)
555 {
556     $this->even = $even;
557 }
558
559 public function setLength($length)
559 {
560     $this->length = $length;
561 }
562
563 public function setIndexMedian($indexMedian)
564 {
565     $this->indexMedian = $indexMedian;
566 }
567
568 public function setMedian($median)
569 {
570     $this->median = $median;
571 }
572
573 public function setSum($sum)
573 {
574     $this->sum = $sum;
575 }
576
576 public function setTotal($total)
577 {
578     $this->total = $total;
579 }
580
581 public function setOdd($odd)
580 {
581     $this->odd = $odd;
582 }
583
584 public function setEven($even)
585 {
586     $this->even = $even;
587 }
588
589 public function setLength($length)
589 {
590     $this->length = $length;
591 }
592
593 public function setIndexMedian($indexMedian)
594 {
595     $this->indexMedian = $indexMedian;
596 }
597
598 public function setMedian($median)
599 {
600     $this->median = $median;
601 }
602
603 public function setSum($sum)
603 {
604     $this->sum = $sum;
605 }
606
606 public function setTotal($total)
607 {
608     $this->total = $total;
609 }
610
611 public function setOdd($odd)
610 {
611     $this->odd = $odd;
612 }
613
614 public function setEven($even)
615 {
616     $this->even = $even;
617 }
618
619 public function setLength($length)
619 {
620     $this->length = $length;
621 }
622
623 public function setIndexMedian($indexMedian)
624 {
625     $this->indexMedian = $indexMedian;
626 }
627
628 public function setMedian($median)
629 {
630     $this->median = $median;
631 }
632
633 public function setSum($sum)
633 {
634     $this->sum = $sum;
635 }
636
636 public function setTotal($total)
637 {
638     $this->total = $total;
639 }
640
641 public function setOdd($odd)
640 {
641     $this->odd = $odd;
642 }
643
644 public function setEven($even)
645 {
646     $this->even = $even;
647 }
648
649 public function setLength($length)
649 {
650     $this->length = $length;
651 }
652
653 public function setIndexMedian($indexMedian)
654 {
655     $this->indexMedian = $indexMedian;
656 }
657
658 public function setMedian($median)
659 {
660     $this->median = $median;
661 }
662
663 public function setSum($sum)
663 {
664     $this->sum = $sum;
665 }
666
666 public function setTotal($total)
667 {
668     $this->total = $total;
669 }
670
671 public function setOdd($odd)
670 {
671     $this->odd = $odd;
672 }
673
674 public function setEven($even)
675 {
676     $this->even = $even;
677 }
678
679 public function setLength($length)
679 {
680     $this->length = $length;
681 }
682
683 public function setIndexMedian($indexMedian)
684 {
685     $this->indexMedian = $indexMedian;
686 }
687
688 public function setMedian($median)
689 {
690     $this->median = $median;
691 }
692
693 public function setSum($sum)
693 {
694     $this->sum = $sum;
695 }
696
696 public function setTotal($total)
697 {
698     $this->total = $total;
699 }
700
701 public function setOdd($odd)
700 {
701     $this->odd = $odd;
702 }
703
704 public function setEven($even)
705 {
706     $this->even = $even;
707 }
708
709 public function setLength($length)
709 {
710     $this->length = $length;
711 }
712
713 public function setIndexMedian($indexMedian)
714 {
715     $this->indexMedian = $indexMedian;
716 }
717
718 public function setMedian($median)
719 {
720     $this->median = $median;
721 }
722
723 public function setSum($sum)
723 {
724     $this->sum = $sum;
725 }
726
726 public function setTotal($total)
727 {
728     $this->total = $total;
729 }
730
731 public function setOdd($odd)
730 {
731     $this->odd = $odd;
732 }
733
734 public function setEven($even)
735 {
736     $this->even = $even;
737 }
738
739 public function setLength($length)
739 {
740     $this->length = $length;
741 }
742
743 public function setIndexMedian($indexMedian)
744 {
745     $this->indexMedian = $indexMedian;
746 }
747
748 public function setMedian($median)
749 {
750     $this->median = $median;
751 }
752
753 public function setSum($sum)
753 {
754     $this->sum = $sum;
755 }
756
756 public function setTotal($total)
757 {
758     $this->total = $total;
759 }
760
761 public function setOdd($odd)
760 {
761     $this->odd = $odd;
762 }
763
764 public function setEven($even)
765 {
766     $this->even = $even;
767 }
768
769 public function setLength($length)
769 {
770     $this->length = $length;
771 }
772
773 public function setIndexMedian($indexMedian)
774 {
775     $this->indexMedian = $indexMedian;
776 }
777
778 public function setMedian($median)
779 {
780     $this->median = $median;
781 }
782
783 public function setSum($sum)
783 {
784     $this->sum = $sum;
785 }
786
786 public function setTotal($total)
787 {
788     $this->total = $total;
789 }
790
791 public function setOdd($odd)
790 {
791     $this->odd = $odd;
792 }
793
794 public function setEven($even)
795 {
796     $this->even = $even;
797 }
798
799 public function setLength($length)
799 {
800     $this->length = $length;
801 }
802
803 public function setIndexMedian($indexMedian)
804 {
805     $this->indexMedian = $indexMedian;
806 }
807
808 public function setMedian($median)
809 {
810     $this->median = $median;
811 }
812
813 public function setSum($sum)
813 {
814     $this->sum = $sum;
815 }
816
816 public function setTotal($total)
817 {
818     $this->total = $total;
819 }
820
821 public function setOdd($odd)
820 {
821     $this->odd = $odd;
822 }
823
824 public function setEven($even)
825 {
826     $this->even = $even;
827 }
828
829 public function setLength($length)
829 {
830     $this->length = $length;
831 }
832
833 public function setIndexMedian($indexMedian)
834 {
835     $this->indexMedian = $indexMedian;
836 }
837
838 public function setMedian($median)
839 {
840     $this->median = $median;
841 }
842
843 public function setSum($sum)
843 {
844     $this->sum = $sum;
845 }
846
846 public function setTotal($total)
847 {
848     $this->total = $total;
849 }
850
851 public function setOdd($odd)
850 {
851     $this->odd = $odd;
852 }
853
854 public function setEven($even)
855 {
856     $this->even = $even;
857 }
858
859 public function setLength($length)
859 {
860     $this->length = $length;
861 }
862
863 public function setIndexMedian($indexMedian)
864 {
865     $this->indexMedian = $indexMedian;
866 }
867
868 public function setMedian($median)
869 {
870     $this->median = $median;
871 }
872
873 public function setSum($sum)
873 {
874     $this->sum = $sum;
875 }
876
876 public function setTotal($total)
877 {
878     $this->total = $total;
879 }
880
881 public function setOdd($odd)
880 {
881     $this->odd = $odd;
882 }
883
884 public function setEven($even)
885 {
886     $this->even = $even;
887 }
888
889 public function setLength($length)
889 {
890     $this->length = $length;
891 }
892
893 public function setIndexMedian($indexMedian)
894 {
895     $this->indexMedian = $indexMedian;
896 }
897
898 public function setMedian($median)
899 {
900     $this->median = $median;
901 }
902
903 public function setSum($sum)
903 {
904     $this->sum = $sum;
905 }
906
906 public function setTotal($total)
907 {
908     $this->total = $total;
909 }
910
911 public function setOdd($odd)
910 {
911     $this->odd = $odd;
912 }
913
914 public function setEven($even)
915 {
916     $this->even = $even;
917 }
918
919 public function setLength($length)
919 {
920     $this->length = $length;
921 }
922
923 public function setIndexMedian($indexMedian)
924 {
925     $this->indexMedian = $indexMedian;
926 }
927
928 public function setMedian($median)
929 {
930     $this->median = $median;
931 }
932
933 public function setSum($sum)
933 {
934     $this->sum = $sum;
935 }
936
936 public function setTotal($total)
937 {
938     $this->total = $total;
939 }
940
941 public function setOdd($odd)
940 {
941     $this->odd = $odd;
942 }
943
944 public function setEven($even)
945 {
946     $this->even = $even;
947 }
948
949 public function setLength($length)
949 {
950     $this->length = $length;
951 }
952
953 public function setIndexMedian($indexMedian)
954 {
955     $this->indexMedian = $indexMedian;
956 }
957
958 public function setMedian($median)
959 {
960     $this->median = $median;
961 }
962
963 public function setSum($sum)
963 {
964     $this->sum = $sum;
965 }
966
966 public function setTotal($total)
967 {
968     $this->total = $total;
969 }
970
971 public function setOdd($odd)
970 {
971     $this->odd = $odd;
972 }
973
974 public function setEven($even)
975 {
976     $this->even = $even;
977 }
978
979 public function setLength($length)
979 {
980     $this->length = $length;
981 }
982
983 public function setIndexMedian($indexMedian)
984 {
985     $this->indexMedian = $indexMedian;
986 }
987
988 public function setMedian($median)
989 {
990     $this->median = $median;
991 }
992
993 public function setSum($sum)
993 {
994     $this->sum = $sum;
995 }
996
996 public function setTotal($total)
997 {
998     $this->total = $total;
999 }
1000
1001 public function setOdd($odd)
1000 {
1001     $this->odd = $odd;
1002 }
1003
1004 public function setEven($even)
1005 {
1006     $this->even = $even;
1007 }
1008
1009 public function setLength($length)
1009 {
1010     $this->length = $length;
1011 }
1012
1013 public function setIndexMedian($indexMedian)
1014 {
1015     $this->indexMedian = $indexMedian;
1016 }
1017
1018 public function setMedian($median)
1019 {
1020     $this->median = $median;
1021 }
1022
1023 public function setSum($sum)
1023 {
1024     $this->sum = $sum;
1025 }
1026
1026 public function setTotal($total)
1027 {
1028     $this->total = $total;
1029 }
1030
1031 public function setOdd($odd)
1030 {
1031     $this->odd = $odd;
1032 }
1033
1034 public function setEven($even)
1035 {
1036     $this->even = $even;
1037 }
1038
1039 public function setLength($length)
1039 {
1040     $this->length = $length;
1041 }
1042
1043 public function setIndexMedian($indexMedian)
1044 {
1045     $this->indexMedian = $indexMedian;
1046 }
1047
1048 public function setMedian($median)
1049 {
1050     $this->median = $median;
1051 }
1052
1053 public function setSum($sum)
1053 {
1054     $this->sum = $sum;
1055 }
1056
1056 public function setTotal($total)
1057 {
1058     $this->total = $total;
1059 }
1060
1061 public function setOdd($odd)
1060 {
1061     $this->odd = $odd;
1062 }
1063
1064 public function setEven($even)
1065 {
1066     $this->even = $even;
1067 }
1068
1069 public function setLength($length)
1069 {
1070     $this->length = $length;
1071 }
1072
1073 public function setIndexMedian($indexMedian)
1074 {
1075     $this->indexMedian = $indexMedian;
1076 }
1077
1078 public function setMedian($median)
1079 {
1080     $this->median = $median;
1081 }
1082
1083 public function setSum($sum)
1083 {
1084     $this->sum = $sum;
1085 }
1086
1086 public function setTotal($total)
1087 {
1088     $this->total = $total;
1089 }
1090
1091 public function setOdd($odd)
1090 {
1091     $this->odd = $odd;
1092 }
1093
1094 public function setEven($even)
1095 {
1096     $this->even = $even;
1097 }
1098
1099 public function setLength($length)
1099 {
1100     $this->length = $length;
1101 }
1102
1103 public function setIndexMedian($indexMedian)
1104 {
1105     $this->indexMedian = $indexMedian;
1106 }
1107
1108 public function setMedian($median)
1109 {
1110     $this->median = $median;
1111 }
1112
1113 public function setSum($sum)
1113 {
1114     $this->sum = $sum;
1115 }
1116
1116 public function setTotal($total)
1117 {
1118     $this->total = $total;
1119 }
1120
1121 public function setOdd($odd)
1120 {
1121     $this->odd = $odd;
1122 }
1123
1124 public function setEven($even)
1125 {
1126     $this->even = $even;
1127 }
1128
1129 public function setLength($length)
1129 {
1130     $this->length = $length;
1131 }
1132
1133 public function setIndexMedian($indexMedian)
1134 {
1135     $this->indexMedian = $indexMedian;
1136 }
1137
1138 public function setMedian($median)
1139 {
1140     $this->median = $median;
1141 }
1142
1143 public function setSum($sum)
1143 {
1144     $this->sum = $sum;
1145 }
1146
1146 public function setTotal($total)
1147 {
1148     $this->total = $total;
1149 }
1150
1151 public function setOdd($odd)
1150 {
1151     $this->odd = $odd;
1152 }
1153
1154 public function setEven($even)
1155 {
1156     $this->even = $even;
1157 }
1158
1159 public function setLength($length)
1159 {
1160     $this->length = $length;
1161 }
1162
1163 public function setIndexMedian($indexMedian)
1164 {
1165     $this->indexMedian = $indexMedian;
1166 }
1167
1168 public function setMedian($median)
1169 {
1170     $this->median = $median;
1171 }
1172
1173 public function setSum($sum)
1173 {
1174     $this->sum = $sum;
1175 }
1176
1176 public function setTotal($total)
1177 {
1178     $this->total = $total;
1179 }
1180
1181 public function setOdd($odd)
1180 {
1181     $this->odd = $odd;
1182 }
1183
1184 public function setEven($even)
1185 {
1186     $this->even = $even;
1187 }
1188
1189 public function setLength($length)
1189 {
1190     $this->length = $length;
1191 }
1192
1193 public function setIndexMedian($indexMedian)
1194 {
1195     $this->indexMedian = $indexMedian;
1196 }
1197
1198 public function setMedian($median)
1199 {
1200     $this->median = $median;
1201 }
1202
1203 public function setSum($sum)
1203 {
1204     $this->sum = $sum;
1205 }
1206
1206 public function setTotal($total)
1207 {
1208     $this->total = $total;
1209 }
1210
1211 public function setOdd($odd)
1210 {
1211     $this->odd = $odd;
1212 }
1213
1214 public function setEven($even)
1215 {
1216     $this->even = $even;
1217 }
1218
1219 public function setLength($length)
1219 {
1220     $this->length = $length;
1221 }
1222
1223 public function setIndexMedian($indexMedian)
1224 {
1225     $this->indexMedian = $indexMedian;
1226 }
1227
1228 public function setMedian($median)
1229 {
1230     $this->median = $median;
1231 }
1232
1233 public function setSum($sum)
1233 {
1234     $this->sum = $sum;
1235 }
1236
1236 public function setTotal($total)
1237 {
1238     $this->total = $total;
1239 }
1240
1241 public function setOdd($odd)
1240 {
1241     $this->odd = $odd;
1242 }
1243
1244 public function setEven($even)
1245 {
1246     $this->even = $even;
1247 }
1248
1249 public function setLength($length)
1249 {
1250     $this->length = $length;
1251 }
1252
1253 public function setIndexMedian($indexMedian)
1254 {
1255     $this->indexMedian = $indexMedian;
1256 }
1257
1258 public function setMedian($median)
1259 {
1260     $this->median = $median;
1261 }
1262
1263 public function setSum($sum)
1263 {
1264     $this->sum = $sum;
1265 }
1266
1266 public function setTotal($total)
1267 {
1268     $this->total = $total;
1269 }
1270
1271 public function setOdd($odd)
1270 {
1271     $this->odd = $odd;
1272 }
1273
1274 public function setEven($even)
1275 {
1276     $this->even = $even;
1277 }
1278
1279 public function setLength($length)
1279 {
1280     $this->length = $length;
1281 }
1282
1283 public function setIndexMedian($indexMedian)
1284 {
1285     $this->indexMedian = $indexMedian;
1286 }
1287
1288 public function setMedian($median)
1289 {
1290     $this->median = $median;
1291 }
1292
1293 public function setSum($sum)
1293 {
1294     $this->sum = $sum;
1295 }
1296
1296 public function setTotal($total)
1297 {
1298     $this->total = $total;
1299 }
1300
1301 public function setOdd($odd)
1300 {
1301     $this->odd = $odd;
1302 }
1303
1304 public function setEven($even)
1305 {
1306     $this->even = $even;
1307 }
1308
1309 public function setLength($length)
1309 {
1310     $this->length = $length;
1311 }
1312
1313 public function setIndexMedian($indexMedian)
1314 {
1315     $this->indexMedian = $indexMedian;
1316 }
1317
1318 public function setMedian($median)
1319 {
1320     $this->median = $median;
1321 }
1322
1323 public function setSum($sum)
1323 {
1324     $this->sum = $sum;
1325 }
1326
1326 public function setTotal($total)
1327 {
1328     $this->total = $total;
1329 }

```

```
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.

```
composer.json  
phpunit.xml  
runner.php
```

```
36     public function testReturnNullIfDataLengthLessThanOne()  
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     }
```

```
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     }
84
```

**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    }
```

```
4 private $data = array();
5 private $length;
6 private $indexMedian;
7 private $median;
8 private $average;
9 private $isOdd;
10
11 public function __construct($data)
12 {
13 }
```

## 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

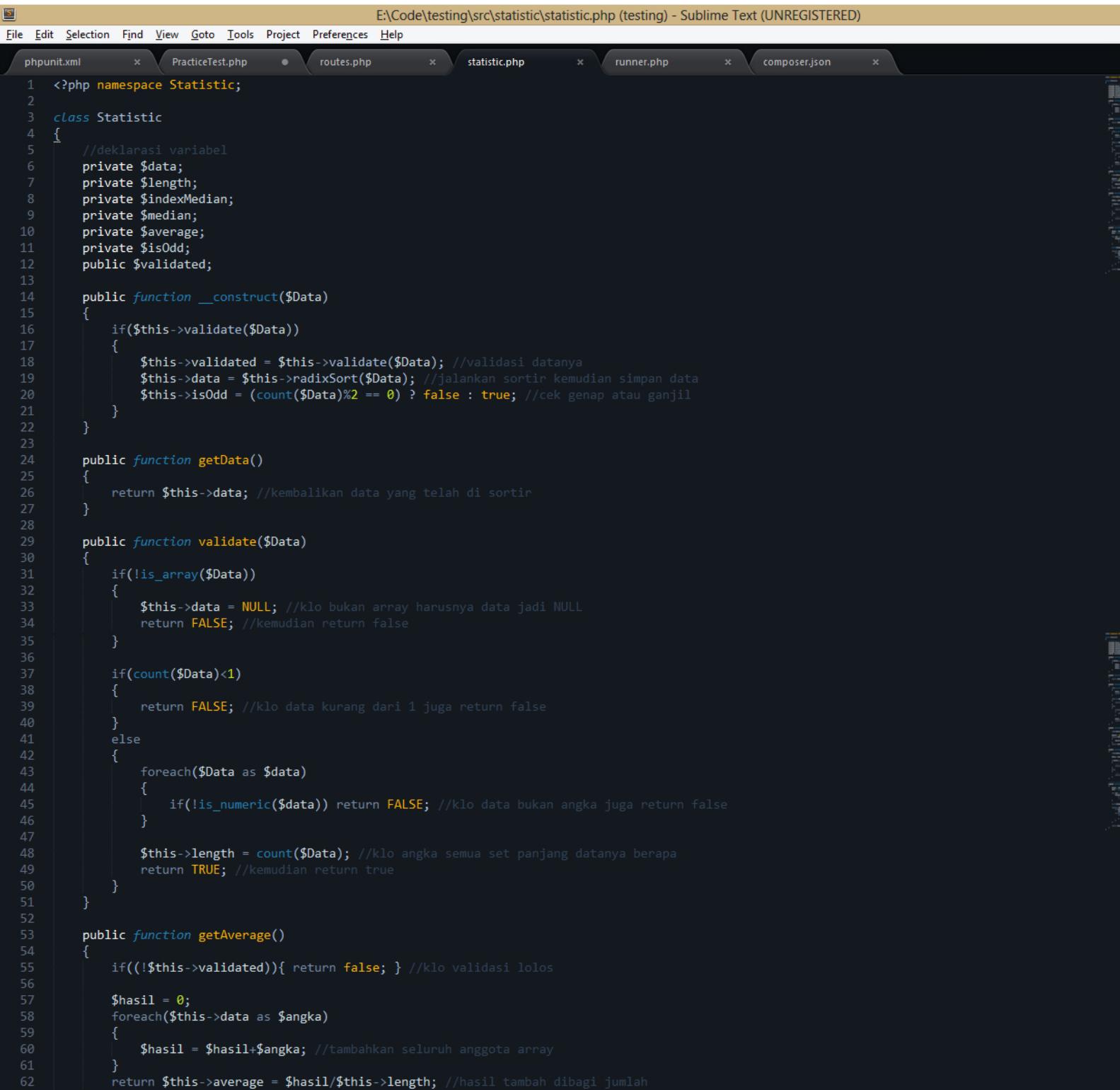
Sesuai dengan rumus median di halaman pertama kalau datanya genap maka setelah dibagi dua, akan ditambahkan elemen sebelum dan sesudah nilai tengah tersebut kemudian dibagi dengan dua. Berikut test case nya.

```
▶ 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 }
```

```
4 private $data = array();
5 private $length;
6 private $indexMedian;
7 private $median;
8 private $average;
9 private $isOdd;
10
11 public function __construct($data)
12 {
13 }
```

# 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 *
1 <?php namespace Statistic;
2
3 class Statistic
4 {
5     //deklarasi variabel
6     private $data;
7     private $length;
8     private $indexMedian;
9     private $median;
10    private $average;
11    private $isOdd;
12    public $validated;
13
14    public function __construct($Data)
15    {
16        if($this->validate($Data))
17        {
18            $this->validated = $this->validate($Data); //validasi datanya
19            $this->data = $this->radixSort($Data); //jalankan sortir kemudian simpan data
20            $this->isOdd = (count($Data)%2 == 0) ? false : true; //cek genap atau ganjil
21        }
22    }
23
24    public function getData()
25    {
26        return $this->data; //kembalikan data yang telah di sortir
27    }
28
29    public function validate($Data)
30    {
31        if(!is_array($Data))
32        {
33            $this->data = NULL; //klo bukan array harusnya data jadi NULL
34            return FALSE; //kemudian return false
35        }
36
37        if(count($Data)<1)
38        {
39            return FALSE; //klo data kurang dari 1 juga return false
40        }
41        else
42        {
43            foreach($Data as $data)
44            {
45                if(!is_numeric($data)) return FALSE; //klo data bukan angka juga return false
46            }
47
48            $this->length = count($Data); //klo angka semua set panjang datanya berapa
49            return TRUE; //kemudian return true
50        }
51    }
52
53    public function getAverage()
54    {
55        if((!$this->validated)){ return false; } //klo validasi lolos
56
57        $hasil = 0;
58        foreach($this->data as $angka)
59        {
60            $hasil = $hasil+$angka; //tambahkan seluruh anggota array
61        }
62        return $this->average = $hasil/$this->length; //hasil tambah dibagi jumlah
63    }
64 }
```

```
4 private $data = array();
5 private $length;
6 private $indexMedian;
7 private $median;
8 private $average;
9 private $isOdd;
10
11 public function __construct($data)
12 {
```

Lanjutan dari halaman sebelumnya.

```
52
53     public function getAverage()
54     {
55         if((!$this->validated)){ return false; } //klo validasi lolos
56
57         $hasil = 0;
58         foreach($this->data as $angka)
59         {
60             $hasil = $hasil+$angka; //tambahkan seluruh anggota array
61         }
62         return $this->average = $hasil/$this->length; //hasil tambah dibagi jumlah
63     }
64
65     public function getMedian()
66     {
67         if((!$this->validated)){ return false; } //klo validasi lolos
68
69         $this->indexMedian = ceil($this->length/2); //hitung nilai tengah, dibulatkan keatas
70
71         if($this->isOdd)
72         {
73             //klo ganjil
74             return $this->median = $this->data[$this->indexMedian-1];
75         }
76         else
77         {
78             //klo genap
79             return $this->median = ($this->data[$this->indexMedian-1]+$this->data[$this->indexMedian])/2;
80         }
81     }
82
83     //fungsi sorting nya nih
84     public function radixSort($input)
85     {
86         $temp = $output = array();
87         $len = count($input);
88
89         for ($i = 0; $i < $len; $i++) {
90             $temp[$input[$i]] = ($temp[$input[$i]] > 0)
91                 ? ++$temp[$input[$i]]
92                 : 1;
93         }
94
95         ksort($temp);
96
97         foreach ($temp as $key => $val) {
98             if ($val == 1) {
99                 $output[] = $key;
100            } else {
101                while ($val--) {
102                    $output[] = $key;
103                }
104            }
105        }
106
107        return $output;
108    }
109 }
```

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

public function __construct($data)
{
```

# **SOFTWARE TESTING**

Berikut hasil jika test dijalankan

```
C:\Program Files (x86)\Git\git-cheetah\.\bin\sh.exe
Fikri@DLLPTP /E/Code/testing
$ vendor/bin/phpunit --testdox tests/PracticeTest
PHPUnit 3.7.27 by Sebastian Bergmann.

Configuration read from E:\Code\testing\phpunit.xml

Practices
[x] Return true if parameter of data type array
[x] Return null if data type not array
[x] Return null if data length less than one
[x] Return true if array contain number
[x] Radix sorting in ascending order
[x] Getting correct average
[x] Getting correct median for odd number
[x] Getting correct median for even number

Fikri@DLLPTP /E/Code/testing
$ -
```

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

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