

# Ar (reikia|verta|(ne)?galima) testuoti privačius metodus?

Povilas Balzaravičius

Vilnius PHP 0x4C 0x7E3

# Pristatymo tikslas - duoti peno pamąstymui.

Visas pristatyme esančias citatas išskaičiau arba išgirdau diskusijų metu iš pašnekovų.

# Kodėl ši tema?

- Per pristatymą Vilnius PHP užsiminiau, kad esu nuodėmingas ir testuoju privačius metodus.
- Po renginio bare BarKodas iki 1AM vis dar diskutavom ta pačia tema. Draugai mus paliko. Vieningos nuomonės nepriėjom.
- Kitą dieną ofise pakalbinau kolegas.
- 2 valandos dingo.
- Tada atsidariau #vilniusphp kanalą Usergroups.lt Slack'e...

# Kodėl ši tema?

- Nepaisant patirties, inžinieriai laikosi skirtingų pozicijų.
- Bandžiau suprasti kodėl taip yra.

Nuomonės, tiesos ir mitai.

# Mantros programinės įrangos inžinerijoje

Programinės įrangos kūrimo sritis pilna mantrų, teigiančių kaip reikia **teisingai** rašyti kodą ir ko **negalima** daryti.

# Mantros programinės įrangos inžinerijoje

Programinės įrangos kūrimo sritis pilna mantrų, teigiančių kaip reikia **teisingai** rašyti kodą ir ko **negalima** daryti.

Tuo pat metu programinės įrangos kūrimas yra tapatinamas su menu. Kiekvienas kūrinys yra unikalus su savais ir unikaliais reikalavimais.

# Mantros programinės įrangos inžinerijoje

Programinės įrangos kūrimo sritis pilna mantrų, teigiančių kaip reikia **teisingai** rašyti kodą ir ko **negalima** daryti.

Tuo pat metu programinės įrangos kūrimas yra tapatinamas su menu. Kiekvienas kūrinys yra unikalus su savais ir unikaliais reikalavimais.

Nors šios mantros atsirado remiantis daugelio projektų patirtimi, tačiau dažnai jas priimame kaip absoliučią tiesą, o ne kaip rekomendacijas.



# Mantros programinės įrangos inžinerijoje

Programinės įrangos kūrimo sritis pilna mantrų, teigiančių kaip reikia **teisingai** rašyti kodą ir ko **negalima** daryti.

Tuo pat metu programinės įrangos kūrimas yra tapatinamas su menu. Kiekvienas kūrinys yra unikalus su savais ir unikaliais reikalavimais.

Nors šios mantros atsirado remiantis daugelio projektų patirtimi, tačiau dažnai jas priimame kaip absoliučią tiesą, o ne kaip rekomendacijas.

Jei kiekvienas kūrinys yra unikalus ar visada jam galime taikyti tas pačias taisykles?

Vieša (ir kartais teisinga) nuomonė

## Test only publics\*

- "Unit" testing == "Unit Of Work Testing"
- Testing a private makes your test more brittle
- Makes you think about the design and usability of a feature
- Test-First leads to private members after Refactoring, but they are all tested!

\*Skairdė iš Roy Osherove prezentacijos [Unit Testing Good Practices and Horrible Mistakes](#).

Vieša (ir kartais teisinga) nuomonė

## Test only publics\*

- "Unit" testing == "Unit Of Work Testing"
- Testing a private makes your test more brittle
- Makes you think about the design and usability of a feature
- Test-First leads to private members after Refactoring, but they are all tested!
- **But sometimes there's no choice**

\*Skairdė iš Roy Osherove prezentacijos **Unit Testing Good Practices and Horrible Mistakes**.

Vieša (ir kartais teisinga) nuomonė

## Test only publics\*

- "Unit" testing == "Unit Of Work Testing"
- Testing a private makes your test more brittle
- Makes you think about the design and usability of a feature
- Test-First leads to private members after Refactoring, but they are all tested!
- But sometimes there's no ~~choice~~ point

\*Skairdė iš Roy Osherove prezentacijos [Unit Testing Good Practices and Horrible Mistakes](#).

# Ką mano Vilnius PHP bendruomenė?

Kai kurias citatas leidau sau pageduoti nekeisdamas konteksto, kad aiškiau skambėtų.

## Nuomonė #1

# Privatūs metodai "neegzistuoja"

"Mano kuklia nuomone, private metodų testuoti nereikia, nes jie "neegzistuoja". Tai kad PHP ar kitos kalbos leidžia prieiti prie jų per reflection ar kitokius API, nereiškia kad juos reikia testuoti."

"Unit'as nėra metodas, tai turėtų būti objektas, kuris naudoja enkapsuliaciją, o tai reiškia, kad testuojam objekto elgseną, o ne private metodų."

## Nuomonė #1

# Privatūs metodai "neegzistuoja"

- Teisinga kai kalbama apie viešo API testavimą.
- Ar Reflection egzistuoja kalboje - nesusiję su tuo ar reikia ar nereikia testuoti kodą.
- Integracinio testavimo požymis.

## Nuomonė #2

# Iškelti metodą į naują klasę

"Jei yra poreikis testuoti `private` metodus - kodėl juos slėpti? Galima <...> funkcionalumą iškelti į kitą klasę. Toje klasėje metodas jau bus `public` ir buvusioje klasėje per `private` kintamąjį jį bus galima pasiekti."



## Nuomonė #2

# Iškelti metodą į naują klasę

- Teisingas požiūris. Užsimanius testuoti privatų metodą pirmiausiai reiktų savęs paklausti - "gal vertėtų jį iškelti į atskirą klasę?"

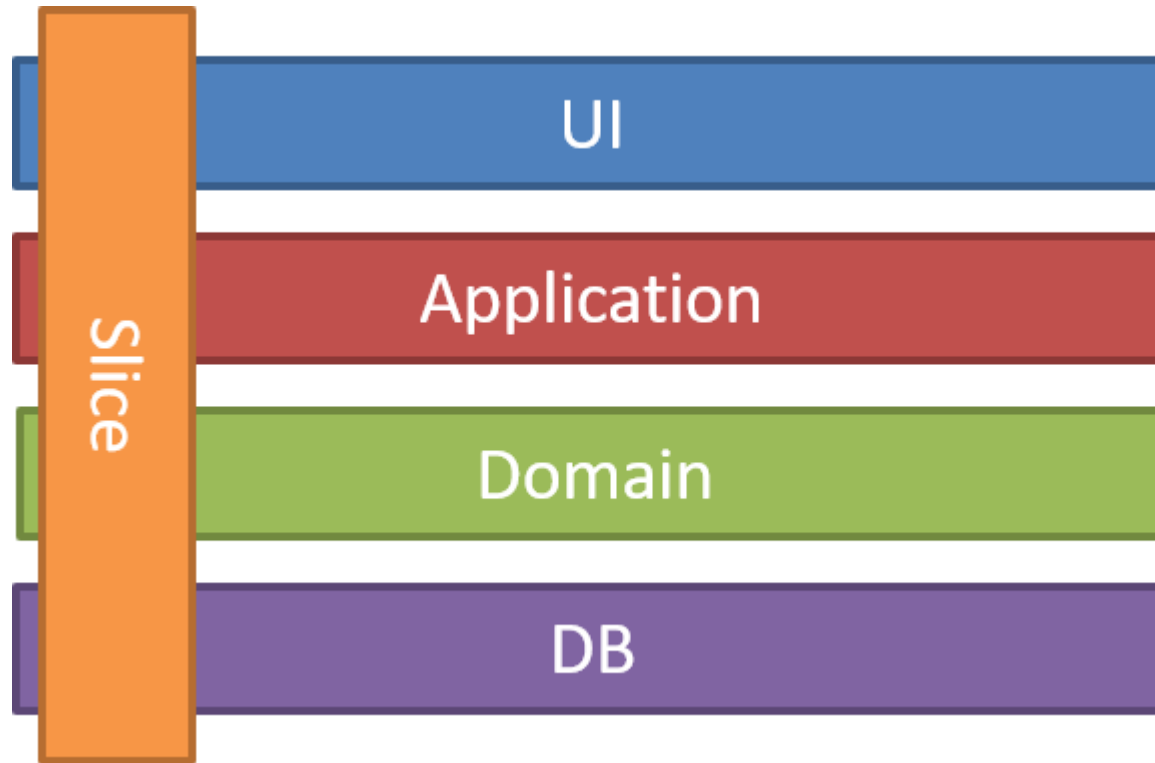
## Nuomonė #2

# Iškelti metodą į naują klasę

- Teisingas požiūris. Užsimančius testuoti privatų metodą pirmiausiai reiktų savęs paklausti - "gal vertėtų jį iškelti į atskirą klasę?"
- Bet ką daryti, jei vidinis "aš" sako - "Mielas, Povilai, žinok, nevertėtų"?
- Ir taip yra ne dėl to, kad tingima, tačiau iškėlimas į atskirą klasę didina disponuojamų objektų kiekį. Dėl to gali nukentėti efektyvumas rašant kodą.
- Ar visada verta kurti naują klasę naujame faile ir galbūt naujame namespace tam, kad pasakytume sau jog teisingai testuojam kodą?

# Vertical Slice Architecture

Efektyvumo didinimo pavyzdys, visiškai nesusijęs su testavimu



Šaltinis: <https://jimmybogard.com/vertical-slice-architecture/>

## Nuomonė #3

# Privačius metodus gali testuoti tik patyrę programuotojai

"Mano labai stiprus argumentas prieš `private` metodų testavimą - reikia išmanyti taisykles prieš pradedant jas laužyti."

"Negalima pradedantiesiems programuotojams rodyti blogų pavyzdžių, kaip pvz. privačių metodų testavimas."

Nuomonė #3

# Privačius metodus gali testuoti tik patyrę programuotojai

- Kaip išmokti dirbti su technologija, jei neleidžiama dirbti su technologija?


## Nuomonė #3

# Privačius metodus gali testuoti tik patyrę programuotojai

- Kaip išmokti dirbti su technologija, jei neleidžiama dirbti su technologija?
- Kaip tik geriau leisti rašyti testus ir suteikti progą pačiam užlipti ant savų grėblių.
- "Kodėl šitas testas nepagavo klaidos?" arba "Kodėl 6 testai lūžta pakeitus vieną eilutę?" yra labai geros pamokos kurias reikia išmokti.
- Ant tų pačių grėblių galima užlipti testuojant tik `public` metodus.

## Nuomonė #4

# Tik apleistų projektų testavimui



"Privačius metodus testuoti priverčia darbas su senais (*legacy*) projektais."

## Nuomonė #4

# Tik apleistų projektų testavimui

```
// https://github.com/golang/go/blob/master/src/os/user/lookup_unix.go
func findGroupName(name string, r io.Reader) (*Group, error) {
    if v, err := readColonFile(r, matchGroupIndexValue(name, 0)); err != nil {
        return nil, err
    } else if v != nil {
        return v.(*Group), nil
    }
    return nil, UnknownGroupError(name)
}
```

```
// https://github.com/golang/go/blob/master/src/os/user/lookup_unix_test.go
func TestFindGroupName(t *testing.T) {
    for _, tt := range groupTests {
        got, err := findGroupName(tt.name, strings.NewReader(tt.in))
        // ...
    }
}
```



## Nuomonė #4

# Tik apleistų projektų testavimui

- Senus projektus kaip tik sudėtinga padengti testais.
- Juos dažniausiai patogiausia testuoti rašant integracinius testus, kurie yra daug aukštesnio lygmens, nei privačių metodų testavimas.
- Pabandykit **parašyt phpSpec testą** senam kodui :-)

## Nuomonė #5

# 100% padengimas testais

"... Pradėjom nuo klausimo ar reikia Private metodus testuotis (**kas greičiausiai iškyla, kai norim padengti “code coverage 100%”**) ..."

## Nuomonė #5

# 100% padengimas testais

```
class SomeService {  
    public function handle($data) {  
        // ...  
        $magic = $this->doMagic($data);  
        // ...  
    }  
  
    private function doMagic($data) {  
        // Some magic here.  
    }  
}
```

- Testuojant tik `handle` galimai bus testuojamas ir `doMagic` metodas.

## Nuomonė #6

# Privataus metodo pakeitimas į viešą

Kartais `private` metodą padarau specialiai `public`, kad galėčiau paprastai ir greitai testus parašyti. Nes tiesiog taip paprasčiau.

Jei yra poreikis testuoti `private` metodus - kodėl juos slėpti? Galima juos daryti `public` <...>

I just make it public.

**Sveikinu, testuoji privatų metodą!**

Sveikinu, testuoji privatų metodą!

## Nuomonė #6

# Privataus metodo pakeitimas į viešą

- Bloga praktika praplėsti klasės *API* metodais, kurių ten neturėtų būti.
- Kalbėjom, kad privačių metodų testavimas pradedančiuosius inžinierius gali privesti prie blogo kodo rašymo. Tai atvirkščias pavyzdys, kaip vengimas testuoti privačius metodus priveda prie analogiškų dalykų.

## Nuomonė #7

# Testing a private makes your test more brittle

In computer programming and software engineering, software brittleness is the increased difficulty in fixing older software that may appear reliable, but fails badly when presented with unusual data or altered in a seemingly minor way. The phrase is derived from analogies to brittleness in metalworking.

Šaltinis: [Wikipedia - Software Brittleness](#)



## Nuomone #7

# Testing a private makes your test more brittle

```
class SomeService {  
    private $doer;  
  
    public function __construct(MagicDoer $doer) {  
        $this->doer = $doer;  
    }  
  
    public function handle($data) {  
        // ...  
        $magic = $this->doer->doMagic($data);  
        // ...  
    }  
}  
  
class MagicDoer {  
    public function doMagic($data) {  
        // Some magic here.  
    }  
}
```

## Nuomonė #7

# Testing a private makes your test more brittle

```
class SomeServiceTestCase extends TestCase {
    public function testHandle() {
        $doer = new MagicDoer(); // Gali būti stub'as.
        $service = new SomeService($doer);
        $result = $service->handle("input");
        $this->assertEquals("expected output", $result);
    }
}

class MagicDoerTestCase extends TestCase {
    public function testDoMagic() {
        $doer = new MagicDoer();
        $result = $doer->doMagic("input");
        $this->assertEquals("expected output", $result);
    }
}
```

- "Netrapus" testas?

## Nuomone #7

# Testing a private makes your test more brittle

```
class SomeServiceTestCase extends TestCase {  
    public function testHandle() {  
        $service = new SomeService();  
        $result = $service->handle("input");  
        $this->assertEquals("expected output", $result);  
    }  
  
    public function testDoMagic() {  
        $service = new SomeService();  
        $result = $this->invoke($service, "doMagic", "input");  
        $this->assertEquals("expected output", $result);  
    }  
}
```

- "Trapus" testas? Bet Kodėl?

## Nuomonė #7

# Testing a private makes your test more brittle

Galimi šio kodo testavimo variantai:

- Testuojamas `handle()` metodas atskirai.
- Testuojamas `doMagic()` metodas atskirai.
- Ir bendrai, testuojamas `handle()` ir iškviečiamas `doMagic()`.

## Nuomonė #7

# Testing a private makes your test more brittle

Galimi šio kodo testavimo variantai:

- Testuojamas `handle()` metodas atskirai.
- Testuojamas `doMagic()` metodas atskirai.
- Ir bendrai, testuojamas `handle()` ir iškviečiamas `doMagic()`.

Visus šiuos testus parašyti galima tiek turint dvi klases su `public` metodais, tiek ir vieną klasę su `private` metodu.

Tai kaip privačių metodų testavimas gali įtakoti kodo trapumą?

## Nuomonė #7

# Testing a private makes your test more brittle

- Blogus testus be problemų galima rašyti testuojant tik viešai prieinamus metodus.
- Jei rašomi geri testai, privačių metodų testavimas situacijos neturėtų pabloginti.
- Galime grįžti prie Nuomonės #3: "Privačius metodus gali testuoti tik patyrę programuotojai".
- Matematiškai, testuojant privačius metodus netgi gali prireikti mažiau testavimo atvejų (*test case*) nei testuojant tik viešus metodus. Tai "trapumą" kaip tik turėtų sumažinti.

# Kodėl tokios skirtingos nuomonės?

Pozicija priklauso nuo šių faktorių:

# Kodėl tokios skirtingos nuomonės?

Pozicija priklauso nuo šių faktorių:

1. **Patirtis su kokiais projektais yra tekę dirbti.** Skirtingi komandų dydžiai, projektų amžius, ilgaamžiškumas ir tipai įtakoja kaip struktūrizuojamas kodas.



# Kodėl tokios skirtingos nuomonės?

Pozicija priklauso nuo šių faktorių:

1. **Patirtis su kokiais projektais yra tekę dirbti.** Skirtingi komandų dydžiai, projektų amžius, ilgaamžiškumas ir tipai įtakoja kaip struktūrizuojamas kodas.
2. **Įrankiai su kuriais dirbama.** Su vienomis programavimo kalbomis ir testavimo karkasais testuoti paslėptą kodą yra lengviau, kitos stengiasi niekaip tam nepasiduoti.

# Tai visgi kada testuoti? Ir kiti trupiniai...

- Rašykite testą kuris pridės pasitikėjimo kodu.
- Naudokit testus kaip dokumentaciją kitiems programuotojams.
- Nereikia testuoti visų privačių metodų (kaip ir visų viešų metodų).
- Privačių metodų testavimas gali padaryti testus sunkiai palaikomus. Bet tą patį galima pasiekti ir testuojant tik viešus metodus :-)
- Ir visa tai galioja ne tik privatiems metodams.
- Galiausiai, jūs patys geriausiai žinote su kokiais projektais dirbate ir kas jiems labiausiai tinka.

# Ačiū!\*

\* Ypač Antanui Končiui, Mažvydui Skuodui, Sergėjui Kurakinui, Tomui Urbonaičiui, Žilvinui Gasiūnui, Žilvinui Kuusui ir kitiems kurių vardų nepamenu už idėjas ir diskusijas.

Mažvydas Skuodas [12:39]

> na į filosofinį klausimą galima atsakyti paprastai: You Should Not, but if you Must, you Can.

Povilas Balzaravičius [12:41]

> arba "never say never" :slightly\_smiling\_face:

# Appendix A - Resursai

- [https://en.wikipedia.org/wiki/Spherical\\_cow](https://en.wikipedia.org/wiki/Spherical_cow)
- <https://inviqa.com/blog/my-top-ten-favourite-phpspec-limitations>
- <http://carloshschults.net/en/are-private-methods-a-code-smell/>
- <https://medium.com/stuart-engineering/on-testing-private-methods-34109c90a72a>
- <https://martinfowler.com/bliki/UnitTest.html>

# Appendix B - Kaip testuoti?

Parodyk kol mama nemato...

## Privataus metodo kvietimas

```
function invoke($object, $name, ...$args) {  
    $refl = new ReflectionMethod($object, $name);  
    if (!$refl->isPublic()) {  
        $refl->setAccessible(true);  
    }  
  
    return $refl->invokeArgs($object, $args);  
}
```

# Appendix B - Kaip testuoti?

Parodyk kol mama nemato...

## Privataus kintamojo skaitymas ir perrašymas

```
protected function getProperty($object, $property) {  
    $refl = new ReflectionProperty($object, $property);  
    $refl->setAccessible(true);  
    return $refl->getValue($object);  
}  
  
protected function setProperty($object, $property, $value) {  
    $refl = new ReflectionProperty($object, $property);  
    $refl->setAccessible(true);  
    $refl->setValue($object, $value);  
}
```