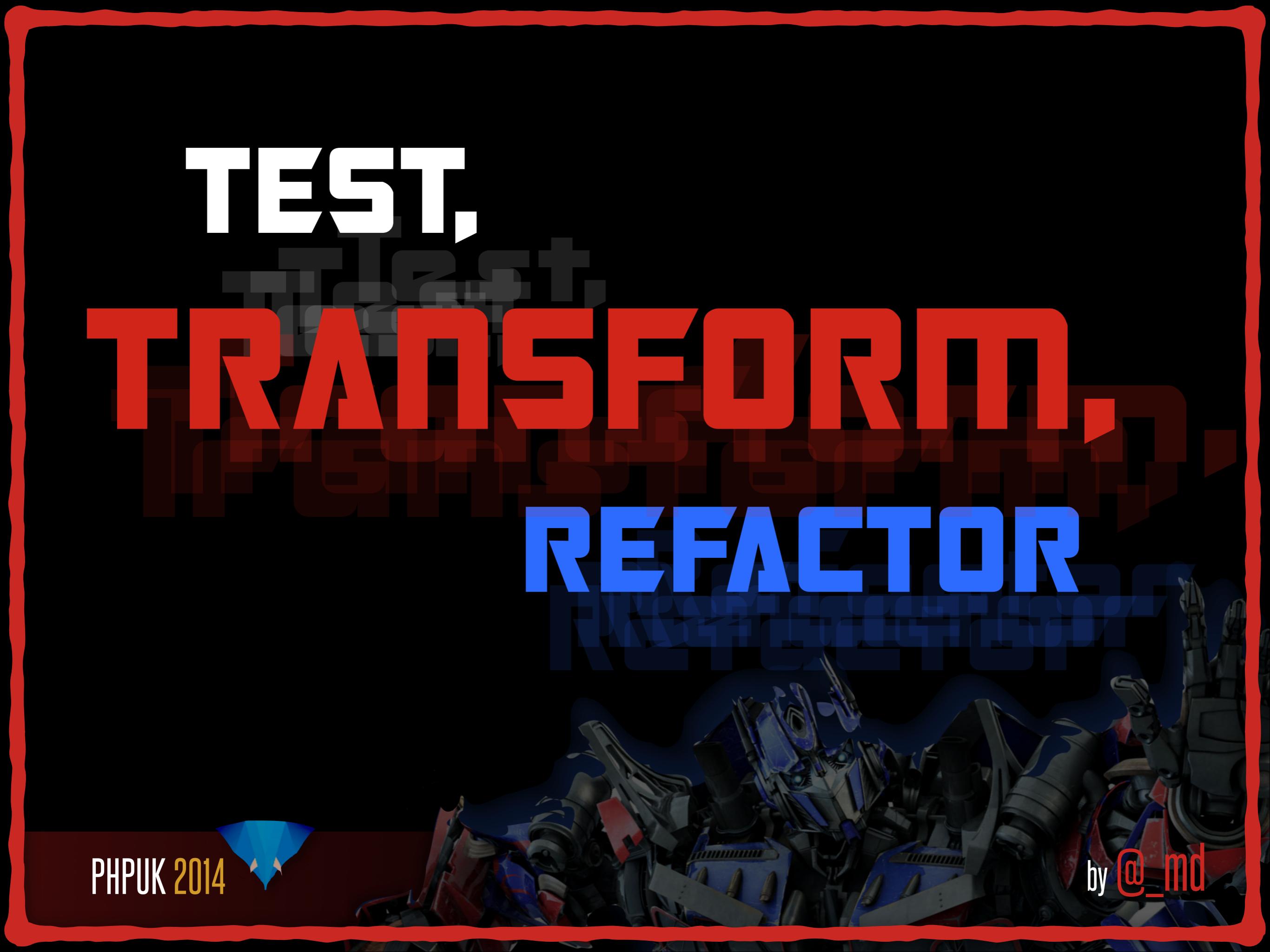


TEST, TRANSFORM, REFACTOR



PHPUK 2014

by @_md

Learning





@_md

php
spec



Marcello Duarte

Lead developer

Head of Training

Kata

form

movement

practice

deliberate

20 minutes

throw away

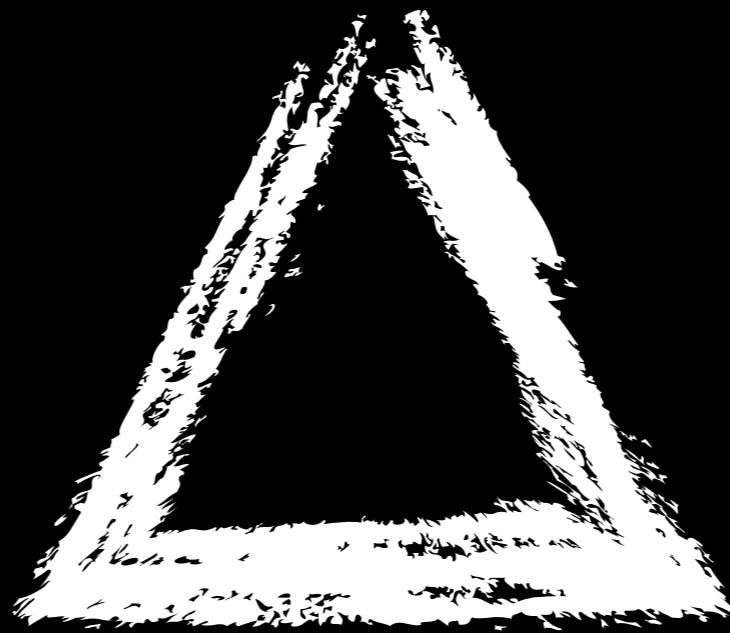
TDD

refactor



Kata

Test



Refactor

Code

TDD 3 Rules

no code without test

enough to go red

enough to go green

Test
Code

no code without test

enough to go red

enough to go green

Refactor

Simple Design

4 Rules

tests run and pass

dry

remove opacity

simplify

The Roman Numerals Kata

Converting arabic numbers into their
roman equivalent

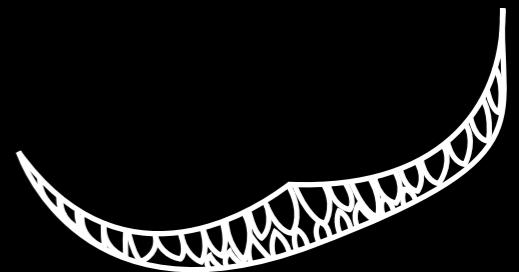
e.g. 2149 => MMCXLIX

Test Make it **red**

Code Make it **green**

Refactor Make it **right**

The Roman Numerals Kata gone wrong
demo



Introducing TPP

The image shows a YouTube video player window. The video is titled "Robert C Martin - The Transformation Priority Premise". The title card features the text "THE TRANSFORMATION PRIORITY PREMISE" in large, gold-colored letters, with "A WORKSHOP" below it. To the left of the title card, there is a small video frame showing Robert C. Martin standing on a stage next to a laptop on a stand. In the bottom left corner of the title card, there is a circular badge with the number "5" and the text "NDC 2011 NORWEGIAN DEVELOPERS CONFERENCE". The video player interface includes a progress bar at 0:10 / 53:43, a volume icon, and a settings icon. Below the video player, the channel information for "Robert C Martin - The Transformation Priority Premise" is shown, along with a video thumbnail for "Jim Coplien and Bob Martin Debate TDD" and another for "Clean Code".

Robert C Martin - The Transformation Priority Premise



Senad Mehic · 6 videos

Subscribe

8

2,310

21 1



Jim Coplien and Bob Martin Debate
TDD

by toalexsmall
7,045

FEATURED



Clean Code

by David Denchue

Test

Make it more specific

Transform

Make it more generic

Refactor

Put on the design hat

}{

- null to constant
- constant to constant+
- constant to scalar
- statement to statements
- unconditional to if
- if to while

Transformations (just the most basic ones)

Jim Weirich's Roman Numeral Kata*

* in PHP

```
1 <?php~  
2 ~  
3 namespace spec;~  
4 ~  
5 use PhpSpec\ObjectBehavior;~  
6 use Prophecy\Argument;~  
7 ~  
8 class RomanNumeralSpec extends ObjectBehavior~  
9 {~  
10 ~  
11 }~  
12
```

```
1 <?php-
2 
3 namespace spec;
4 
5 use PhpSpec\ObjectBehavior;
6 use Prophecy\Argument;
7 
8 class RomanNumeralSpec extends ObjectBehavior
9 {
10     function it_returns_an_empty_string_for_zero()
11     {
12         $this->fromArabic(0)->shouldReturn("");
13     }
14 }
15 |
```

Walking skeleton

[Cockburn 94]



```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
10 ! it returns an empty string for zero
    class RomanNumeral does not exist.
```

100%

1

1 specs

1 example (1 broken)

6ms

Do you want me to create `RomanNumeral` for you?

[Y/n] y

```
Class RomanNumeral created in /Users/mduarte/Developer/kata/roman_numerals/src/RomanNumeral.php.
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
10 ! it returns an empty string for zero
    method RomanNumeral::fromArabic not found.
```

100%

1

1 specs

1 example (1 broken)

8ms

Do you want me to create `RomanNumeral::fromArabic()` for you?

[Y/n] y

```
Method RomanNumeral::fromArabic() has been created.
```

```
md@capoeira:roman_numerals $
```

```
1 <?php~  
2 ~  
3 class RomanNumeral~  
4 {~  
5     public function fromArabic($arabic)~  
6     {~  
7         ~  
8     }~  
9 }~  
10
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
10 ✘ it returns an empty string for zero  
expected "", but got null.
```

100%

1

1 specs

1 example (1 failed)

28ms

```
md@capoeira:roman_numerals $ █
```

Transformation
null -> constant



```
1 <?php  
2  
3 class RomanNumeral  
4 {  
5     public function fromArabic($arabic)  
6     {  
7         return "";  
8     }  
9 }  
10 |
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```



100%

1

1 specs

1 example (1 passed)

11ms

```
md@capoeira:roman_numerals $ █
```

```
1 <?php
2 -
3 namespace spec;
4 -
5 use PhpSpec\ObjectBehavior;
6 use Prophecy\Argument;
7 -
8 class RomanNumeralSpec extends ObjectBehavior
9 {
10     function it_returns_an_empty_string_for_zero()
11     {
12         $this->fromArabic(0)->shouldReturn("");
13     }
14 -
15     function it_returns_I_for_1()
16     {
17         $this->fromArabic(1)->shouldReturn("I");
18     }
19 }
20
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
15 ✘ it returns I for 1  
expected "I", but got "".
```

A horizontal progress bar consisting of two colored segments: green on the left and red on the right. The green segment covers approximately half of the bar's length, with the text "50%" centered above it in white.

50%

A horizontal progress bar consisting of two colored segments: green on the left and red on the right. The red segment covers approximately half of the bar's length, with the text "50%" centered above it in white.

50%

2

1 specs

2 examples (1 passed, 1 failed)

32ms

```
md@capoeira:roman_numerals $ █
```

```
1 <?php  
2  
3 class RomanNumeral {  
4     public function fromArabic($arabic) {  
5         if ($arabic) {  
6             return "I";  
7         }  
8         return "";  
9     }  
10 }  
11  
12 }  
13
```

Transformation
unconditional -> if



```
1 <?php  
2  
3 class RomanNumeral {  
4     public function fromArabic($arabic) {  
5         if ($arabic) {  
6             return "I";  
7         }  
8         return "";  
9     }  
10 }  
11  
12 }  
13
```

```
1 <?php
2
3 namespace spec;
4
5 use PhpSpec\ObjectBehavior;
6 use Prophecy\Argument;
7
8 class RomanNumeralSpec extends ObjectBehavior
9 {
10     function it_returns_an_empty_string_for_zero()
11     {
12         $this->fromArabic(0)->shouldReturn("");
13     }
14
15     function it_returns_I_for_1()
16     {
17         $this->fromArabic(1)->shouldReturn("I");
18     }
19
20     function it_returns_II_for_2()
21     {
22         $this->fromArabic(2)->shouldReturn("II");
23     }
24 }
25
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
20  ✘ it returns II for 2
    expected "II", but got "I".
```

67%

33%

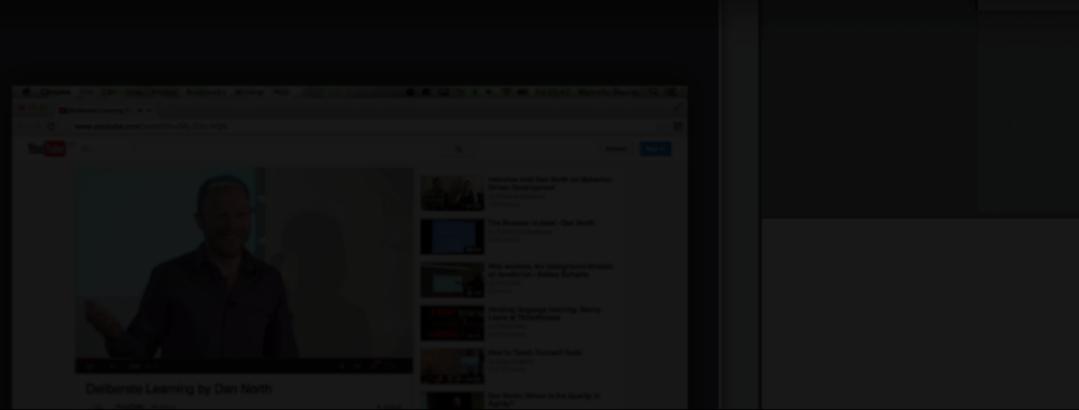
3

1 specs

3 examples (2 passed, 1 failed)

12ms

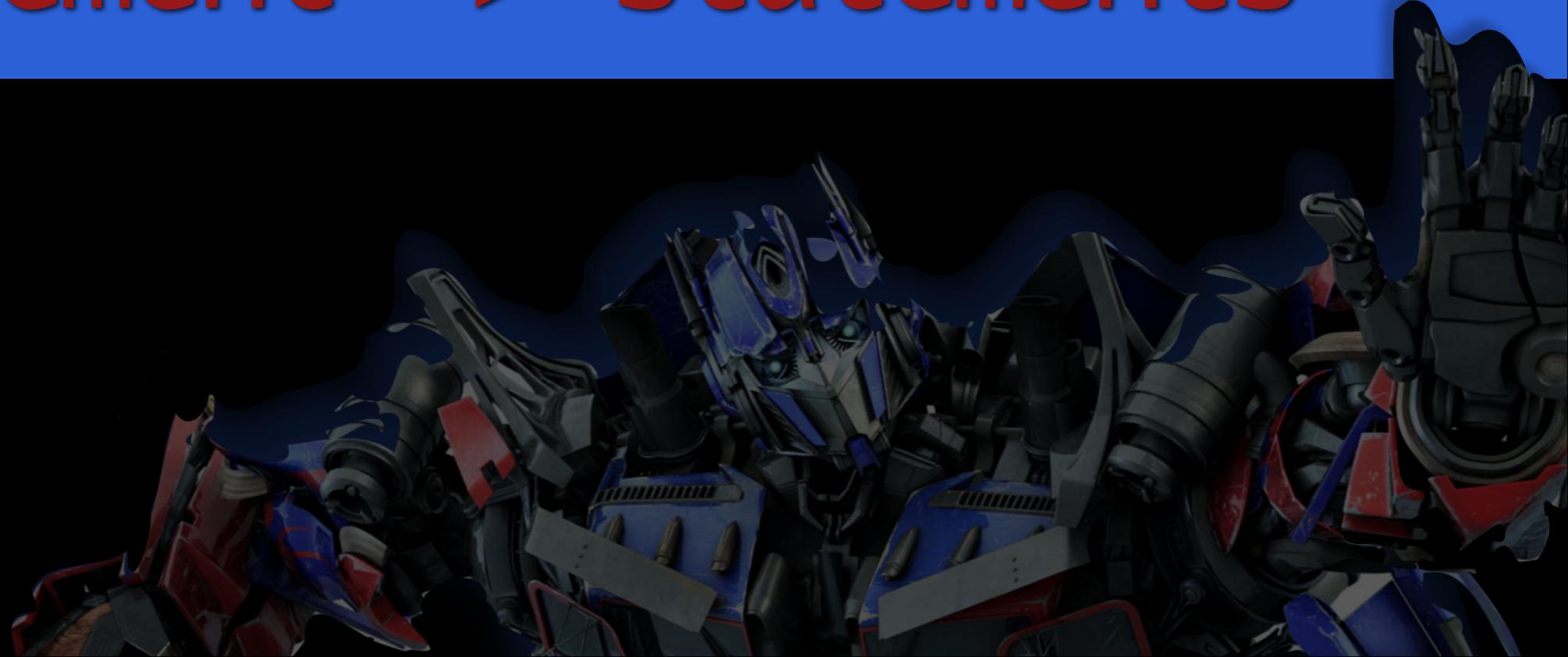
```
md@capoeira:roman_numerals $ █
```



```
1 <?php  
2  
3 class RomanNumeral {  
4     public function fromArabic($arabic) {  
5         if ($arabic) {  
6             return "I";  
7         }  
8         return "";  
9     }  
10 }  
11  
12 }  
13
```

Transformation

statement -> statements



```
1 <?php  
2  
3 class RomanNumeral {  
4     public function fromArabic($arabic) {  
5         if ($arabic) {  
6             return str_repeat("I", $arabic);  
7         }  
8         return "";  
9     }  
10 }  
11  
12 }  
13 |
```

```
7
8 class RomanNumeralSpec extends ObjectBehavior
9 {
10     function it_returns_an_empty_string_for_zero()
11     {
12         $this->fromArabic(0)->shouldReturn("");
13     }
14
15     function it_returns_I_for_1()
16     {
17         $this->fromArabic(1)->shouldReturn("I");
18     }
19
20     function it_returns_II_for_2()
21     {
22         $this->fromArabic(2)->shouldReturn("II");
23     }
24
25     function it_returns_V_for_5()
26     {
27         $this->fromArabic(5)->shouldReturn("V");
28     }
29 }
30
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
25 ✘ it returns V for 5  
expected "V", but got "IIIII".
```

75%

25%

4

1 specs

4 examples (3 passed, 1 failed)

11ms

```
md@capoeira:roman_numerals $ █
```

```
1 <?php  
2  
3 class RomanNumeral {  
4     public function fromArabic($arabic) {  
5         if ($arabic) {  
6             return str_repeat("I", $arabic);  
7         }  
8         return "";  
9     }  
10 }  
11  
12 }  
13 |
```

Transformation
unconditional -> if



```
1 <?php
2
3 class RomanNumeral{
4 {
5     public function fromArabic($arabic)
6     {
7         if ($arabic === 5) {
8             return "V";
9         }
10
11        if ($arabic) {
12            return str_repeat("I", $arabic);
13        }
14        return "";
15    }
16 }
17
```

```
8 class RomanNumeralSpec extends ObjectBehavior
9 {
10     function it_returns_an_empty_string_for_zero()
11     {
12         $this->fromArabic(0)->shouldReturn("");
13     }
14
15     function it_returns_I_for_1()
16     {
17         $this->fromArabic(1)->shouldReturn("I");
18     }
19
20     function it_returns_II_for_2()
21     {
22         $this->fromArabic(2)->shouldReturn("II");
23     }
24
25     function it_returns_V_for_5()
26     {
27         $this->fromArabic(5)->shouldReturn("V");
28     }
29
30     function it_returns_VI_for_6()
31     {
32         $this->fromArabic(6)->shouldReturn("VI");
33     }
34 }
35 }
```

```
md@capoeira:roman_numerals $ r
```

RomanNumeral

```
30 ✘ it returns VI for 6  
expected "VI", but got "IIIIII".
```

80%

20%

5

1 specs

5 examples (4 passed, 1 failed)

11ms

```
md@capoeira:roman_numerals $ █
```

```
1 <?php
2
3 class RomanNumeral{
4 {
5     public function fromArabic($arabic)
6     {
7         if ($arabic === 5) {
8             return "V";
9         }
10
11        if ($arabic) {
12            return str_repeat("I", $arabic);
13        }
14        return "";
15    }
16 }
17
```

Refactor time?

```
29
30     function it_returns_VI_for_6()
31 {
32     // $this->fromArabic(6)->shouldReturn("VI");
33 }
```

```
md@capoeira:roman_numerals $ r
```

RomanNumeral

```
30 - it returns VI for 6
```

```
todo: write pending example
```

20%

80%

5

1 specs

5 examples (4 passed, 1 pending)

10ms

```
md@capoeira:roman_numerals $ █
```

```
1 <?php  
2  
3 class RomanNumeral  
4 {  
5     public function fromArabic($arabic)  
6     {  
7         if ($arabic === 5) {  
8             return "V";  
9         }  
10        if ($arabic) {  
11            return str_repeat("I", $arabic);  
12        }  
13        return "";  
14    }  
15 }  
16 }  
17 }
```

```
1 <?php
2
3 class RomanNumeral{
4 {
5     public function fromArabic($arabic)
6 {
7         $roman = "";
8
9         if ($arabic === 5) {
10             $roman = "V";
11             $arabic -= 5;
12         }
13
14         if ($arabic) {
15             $roman .= str_repeat("I", $arabic);
16         }
17         return $roman;
18     }
19 }
20
```

```
8 class RomanNumeralSpec extends ObjectBehavior
9 {
10     function it_returns_an_empty_string_for_zero()
11     {
12         $this->fromArabic(0)->shouldReturn("");
13     }
14
15     function it_returns_I_for_1()
16     {
17         $this->fromArabic(1)->shouldReturn("I");
18     }
19
20     function it_returns_II_for_2()
21     {
22         $this->fromArabic(2)->shouldReturn("II");
23     }
24
25     function it_returns_V_for_5()
26     {
27         $this->fromArabic(5)->shouldReturn("V");
28     }
29
30     function it_returns_VI_for_6()
31     {
32         $this->fromArabic(6)->shouldReturn("VI");
33     }
34 }
35 }
```

```
md@capoeira:roman_numerals $ r
```

RomanNumeral

```
30 - it returns VI for 6  
todo: write pending example
```



1 specs

5 examples (4 passed, 1 pending)

10ms

```
md@capoeira:roman_numerals $ r
```



1 specs

5 examples (5 passed)

8ms

```
md@capoeira:roman_numerals $ █
```

```
10 function it_returns_an_empty_string_for_zero()~
11 {~
12     $this->fromArabic(0)->shouldReturn("");~
13 }~
14 ~
15 function it_returns_I_for_1()~
16 {~
17     $this->fromArabic(1)->shouldReturn("I");~
18 }~
19 ~
20 function it_returns_II_for_2()~
21 {~
22     $this->fromArabic(2)->shouldReturn("II");~
23 }~
24 ~
25 function it_returns_V_for_5()~
26 {~
27     $this->fromArabic(5)->shouldReturn("V");~
28 }~
29 ~
30 function it_returns_VI_for_6()~
31 {~
32     $this->fromArabic(6)->shouldReturn("VI");~
33 }~
34 ~
35 function it_returns_X_for_10()~
36 {~
37     $this->fromArabic(10)->shouldReturn("X");~
38 }
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
35 x it returns X for 10  
expected "X", but got "VIIII".
```

83%

17%

6

1 specs

6 examples (5 passed, 1 failed)

11ms

```
md@capoeira:roman_numerals $
```

```
1 <?php-
2 
3 class RomanNumeral-
4 {-
5     public function fromArabic($arabic)-
6     {-
7         $roman = "";-
8 
9         if ($arabic === 5) {-
10             $roman = "V";-
11             $arabic -= 5;-}
12 
13 
14         if ($arabic) {-
15             $roman .= str_repeat("I", $arabic);-
16         }-
17         return $roman;-}
18     }-
19 }-
```

```
1 <?php
2
3 class RomanNumeral{
4 {
5     public function fromArabic($arabic)
6 {
7         $roman = "";
8
9         if ($arabic >= 10) {
10            $roman = "X";
11            $arabic -= 10;
12        }
13
14         if ($arabic >= 5) {
15            $roman = "V";
16            $arabic -= 5;
17        }
18
19         if ($arabic) {
20             $roman .= str_repeat("I", $arabic);
21         }
22         return $roman;
23     }
24 }
25 |
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

```
100%
```

```
6
```

```
1 specs
```

```
6 examples (6 passed)
```

```
33ms
```

```
18 }-
19 -
20     function it_returns_II_for_2()-
21     {-
22         $this->fromArabic(2)->shouldReturn("II");-
23     }-
24 -
25     function it_returns_V_for_5()-
26     {-
27         $this->fromArabic(5)->shouldReturn("V");-
28     }-
29 -
30     function it_returns_VI_for_6()-
31     {-
32         $this->fromArabic(6)->shouldReturn("VI");-
33     }-
34 -
35     function it_returns_X_for_10()-
36     {-
37         $this->fromArabic(10)->shouldReturn("X");-
38     }-
39 -
40     function it_returns_XX_for_20()-
41     {-
42         $this->fromArabic(20)->shouldReturn("XX");-
43     }-
44 }
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
40 ✘ it returns XX for 20
    expected "XX", but got "VIIII".
```

86%

14%

7

1 specs

7 examples (6 passed, 1 failed)

12ms

```
md@capoeira:roman_numerals $ █
```

```
1 <?php
2
3 class RomanNumeral{
4 {
5     public function fromArabic($arabic)
6 {
7         $roman = "";
8
9         if ($arabic >= 10) {
10            $roman = "X";
11            $arabic -= 10;
12        }
13
14         if ($arabic >= 5) {
15            $roman = "V";
16            $arabic -= 5;
17        }
18
19         if ($arabic) {
20             $roman .= str_repeat("I", $arabic);
21         }
22         return $roman;
23     }
24 }
25 |
```

Transformation

if -> while



Condition to loop transformation

is a great indicator you are in the right track of solving the algorithm

```
1 <?php
2
3 class RomanNumeral{
4 {
5     public function fromArabic($arabic){
6     {
7         $roman = "";
8
9         while ($arabic >= 10) {
10            $roman .= "X";
11            $arabic -= 10;
12        }
13
14        if ($arabic >= 5) {
15            $roman = "V";
16            $arabic -= 5;
17        }
18
19        if ($arabic) {
20            $roman .= str_repeat("I", $arabic);
21        }
22        return $roman;
23    }
24 }
25
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

```
100%
```

```
7
```

```
1 specs
```

```
7 examples (7 passed)
```

```
9ms
```

```
md@capoeira:roman_numerals $ █
```

```
22     $this->fromArabic(2)->shouldReturn("II");-
23 }-
24 -
25 function it_returns_V_for_5()-{
26     $this->fromArabic(5)->shouldReturn("V");-
27 }-
28 -
29 function it_retunrs_VI_for_6()-{
30     $this->fromArabic(6)->shouldReturn("VI");-
31 }-
32 -
33 function it_retunrs_X_for_10()-{
34     $this->fromArabic(10)->shouldReturn("X");-
35 }-
36 -
37 function it_retunrs_XX_for_20()-{
38     $this->fromArabic(20)->shouldReturn("XX");-
39 }-
40 -
41 function it_retunrs_4_for_4()-{
42     $this->fromArabic(4)->shouldReturn("IV");-
43 }-
44 -
45 }
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

RomanNumeral

```
45  ✘ it returns 4 for 4
    expected "IV", but got "IIII".
```

88%

13%

8

1 specs

8 examples (7 passed, 1 failed)

12ms

```
md@capoeira:roman_numerals $ █
```

```
2
3 class RomanNumeral {
4 {
5     public function fromArabic($arabic) {
6     {
7         $roman = "";
8
9     while ($arabic >= 10) {
10        $roman .= "X";
11        $arabic -= 10;
12    }
13
14    while ($arabic >= 5) {
15        $roman .= "V";
16        $arabic -= 5;
17    }
18
19    while ($arabic >= 4) {
20        $roman .= "IV";
21        $arabic -= 4;
22    }
23
24    if ($arabic) {
25        $roman .= str_repeat("I", $arabic);
26    }
27
28    return $roman;
29 }
30 }
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

```
100%
```

```
8
```

```
1 specs
```

```
8 examples (8 passed)
```

```
11ms
```

```
md@capoeira:roman_numerals $ █
```

```
3 class RomanNumeral {
4     public function fromArabic($arabic) {
5         $roman = "";
6
7         while ($arabic >= 10) {
8             $roman .= "X";
9             $arabic -= 10;
10        }
11
12        while ($arabic >= 5) {
13            $roman = "V";
14            $arabic -= 5;
15        }
16
17        while ($arabic >= 4) {
18            $roman = "IV";
19            $arabic -= 4;
20        }
21
22        if ($arabic) {
23            $roman .= str_repeat("I", $arabic);
24        }
25
26        return $roman;
27    }
28}
29}
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

```
100%
```

```
8
```

```
1 specs
```

```
8 examples (8 passed)
```

```
11ms
```

```
md@capoeira:roman_numerals $ █
```

```
3 class RomanNumeral {
4     public function fromArabic($arabic) {
5         $roman = "";
6
7         while ($arabic >= 10) {
8             $roman .= "X";
9             $arabic -= 10;
10        }
11
12        while ($arabic >= 5) {
13            $roman = "V";
14            $arabic -= 5;
15        }
16
17        while ($arabic >= 4) {
18            $roman = "IV";
19            $arabic -= 4;
20        }
21
22        if ($arabic) {
23            $roman .= str_repeat("I", $arabic);
24        }
25
26        return $roman;
27    }
28}
29}
```

```
5     public function fromArabic($arabic) {  
6         $roman = "";  
7  
8         while ($arabic >= 10) {  
9             $roman .= "X";  
10            $arabic -= 10;  
11        }  
12  
13        while ($arabic >= 5) {  
14            $roman = "V";  
15            $arabic -= 5;  
16        }  
17  
18        while ($arabic >= 4) {  
19            $roman = "IV";  
20            $arabic -= 4;  
21        }  
22  
23        while ($arabic >= 1) {  
24            $roman .= "I";  
25            $arabic -= 1;  
26        }  
27  
28        return $roman;  
29    }
```

```
5     private $romans = array(-
6         10 => 'X'-
7     );-
8 
9     public function fromArabic($arabic)-
10    {-
11        $roman = "";-
12 
13        foreach ($this->romans as $digit => $glyph) {-
14            while ($arabic >= $digit) {-
15                $roman .= $glyph;-
16                $arabic -= $digit;-
17            }-
18        }-
19 
20        while ($arabic >= 5) {-
21            $roman = "V";-
22            $arabic -= 5;-
23        }-
24 
25        while ($arabic >= 4) {-
26            $roman = "IV";-
27            $arabic -= 4;-
28        }-
29 
30        while ($arabic >= 1) {-
31            $roman .= "I";-
32            $arabic -= 1;-
33        }-
34        return $roman;-
35    }-
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

```
100%
```

```
8
```

```
1 specs
```

```
8 examples (8 passed)
```

```
9ms
```

```
md@capoeira:roman_numerals $ █
```

```
1 <?php
2
3 class RomanNumeral{
4 {
5     private $romans = array(
6         10 => 'X',
7         5 => 'V',
8         4 => 'IV',
9         1 => 'I',
10    );
11
12     public function fromArabic($arabic)
13 {
14         $roman = "";
15
16         foreach ($this->romans as $digit => $glyph) {
17             while ($arabic >= $digit) {
18                 $roman .= $glyph;
19                 $arabic -= $digit;
20             }
21         }
22
23         return $roman;
24     }
25 }
26
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

```
100%
```

```
8
```

```
1 specs
```

```
8 examples (8 passed)
```

```
9ms
```

```
md@capoeira:roman_numerals $ █
```

```
function it_returns_the_roman_equivalent_of_any_arabic_number()
{
    $this->fromArabic(2465)->shouldReturn("MMCDLXV");
}
```

```
3 class RomanNumeral{  
4 {  
5     private $romans = array(  
6         1000 => 'M',  
7               
8         900 => 'CM',  
9               
10        500 => 'D',  
11               
12        400 => 'CD',  
13               
14        100 => 'C',  
15               
16               
17               
18        90 => 'XC',  
19               
20        50 => 'L',  
21               
22        40 => 'XL',  
23               
24        10 => 'X',  
25               
26               
27               
28     );  
29       
30     public function fromArabic($arabic){  
31     {  
32         $roman = "";  
33               
34               
35         foreach ($this->romans as $digit => $glyph) {  
36             while ($arabic >= $digit) {  
37                 $roman .= $glyph;  
38                 $arabic -= $digit;  
39             }  
40         }  
41               
42         return $roman;  
43     }  
44 }
```

```
md@capoeira:roman_numerals $ bin/phpspec run
```

```
100%
```

```
8
```

```
1 specs
```

```
8 examples (8 passed)
```

```
9ms
```

```
md@capoeira:roman_numerals $ █
```

Katas and Learning

Chrome File Edit View History Bookmarks Window Help Sat 05:43 Marcello Duarte

Deliberate Learning by | www.youtube.com/watch?v=SPj-23z-hQA

YouTube PT Upload Sign in

The screenshot shows a YouTube video player for "Deliberate Learning by Dan North". The video is at 2:02 / 22:47. Below the video, the title "Deliberate Learning by Dan North" is displayed, along with the channel logo for FooCafe (38 videos) and a "Subscribe" button (104 subscribers). The video has 1,006 views, 13 likes, and 0 dislikes. Below the video player are buttons for "Like", "Share", "Add to", and "Flag". The video content shows a man in a purple shirt smiling and gesturing.

Interview with Dan North on Behavior-Driven Development
by Vladimir Gitlevich
27:22
1,819 views

The Browser is dead - Dan North
by GOTO Conferences
50:05
2,551 views

Web workers, the background threads of JavaScript - Balázs Suhajda
by FooCafe
30:30
53 views

Hacking language learning: Benny Lewis at TEDxWarsaw
by TEDxTalks
15:11
219,770 views

How to Teach Yourself Code
by InternetWeek
44:08
116,139 views

Dan North: Where is the Quality in Agility?
by Trine Mathorne
13:35
693 views

Jane McGonigal on Productivity
by schooloflifechannel
45:21
3,604 views

Facebook för företag - Daniel Herstedt
by FooCafe
21 views

Published on 20 Sep 2012
Some programmers look at programming as more than their day job. They want to deepen their understanding, develop skills and learn techniques that will make them better programmers. Many

Kata

Deliberate Practice

- Optimising for performance
- Minimising variance

Koan

Deliberate Learning

- Optimising for discovery
- Maximising variance

Learning TDD is learning
all that TDD does for you



"The beauty of a move lies not in its' appearance but in the thought behind it."

My System

elements of chess

positional play

illustrative games



First learn the elements of design
Then learn how to drive it with test

Marcello Duarte

we are hiring



try phpspec!



follow @md



inviqa.com/tdd-immersion-day/

Thank you !



Questions or Comments?

joind.in/10711

@ md

Want to learn more? inviqa.com/tdd-immersion-day/

