

***HELLO MY
NAME IS "IF"***

ELSE



SWITCH



```
if ($somethingIsTheCase) {  
    do_something();  
}
```

```
if ($somethingIsTheCase) {  
    do_something();  
} else {  
}  
}
```

```
if ($somethingIsTheCase) {  
    do_something();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase) {  
    do_something();  
}  
  
else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase) {  
    do_something();  
} elseif ($somethingSpecialIsTheCase) {  
  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase) {  
    do_something();  
} elseif ($somethingSpecialIsTheCase) {  
    do_something_special();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase or ) {  
    do_something();  
} elseif ($somethingSpecialIsTheCase) {  
    do_something_special();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase or $somethingRareIsTheCase) {  
    do_something();  
} elseif ($somethingSpecialIsTheCase) {  
    do_something_special();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase or $somethingRareIsTheCase) {  
    do_something();  
  
} elseif ($somethingSpecialIsTheCase) {  
    do_something_special();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase or $somethingSpecialIsTheCase) {  
    do_something();  
    if ($somethingSuperRareIsTheCase) {  
        }  
    } elseif ($somethingElseIsTheCase) {  
        do_something_special();  
    } else {  
        do_something_else();  
    }
```

```
if ($somethingIsTheCase or $somethingSpecialIsTheCase) {  
    do_something();  
    if ($somethingSuperRareIsTheCase) {  
        do_something_extra();  
    }  
} elseif ($somethingElseIsTheCase) {  
    do_something_special();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase or $somethingSpecialIsTheCase) {  
    do_something();  
    if ($somethingSuperRareIsTheCase) {  
        do_something_extra();  
    }  
} elseif ($somethingElseIsTheCase) {  
    do_something_special();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsTheCase or $somethingSpecialIsTheCase) {  
    do_something();  
    if ($somethingSuperRareIsTheCase) {  
        do_something_extra();  
    }  
} elseif ($somethingElseIsTheCase) {  
    do_something_special();  
} else {  
    do_something_else();  
}
```

```
if ($somethingIsHappening) {  
    if ($somethingIsTheCase or $somethingSpecialIsTheCase) {  
        do_something();  
        if ($somethingSuperRareIsTheCase) {  
            do_something_extra();  
        }  
    } elseif ($somethingElseIsTheCase) {  
        do_something_special();  
    } else {  
        do_something_else();  
    }  
}
```

```
if ($somethingIsHappening) {  
    if ($somethingIsTheCase or $somethingSpecialIsTheCase) {  
        do_something();  
        if ($somethingSuperRareIsTheCase) {  
            do_something_extra();  
        }  
    } elseif ($somethingElseIsTheCase) {  
        do_something_special();  
    } else {  
        do_something_else();  
    }  
}
```

```
if ($somethingIsHappening) {
    if ($somethingIsTheCase or $somethingSpecialIsTheCase) {
        do_something();
        if ($somethingSuperRareIsTheCase) {
            do_something_extra();
        }
    } elseif ($somethingElseIsTheCase) {
        do_something_special();
    } else {
        do_something_else();
    }

    if ($somethingSpecialIsTheCase and $somethingElseIsTheCase) {
    }
}
```

```
if ($somethingIsHappening) {
    if ($somethingIsTheCase or $somethingSpecialIsTheCase) {
        do_something();
        if ($somethingSuperRareIsTheCase) {
            do_something_extra();
        }
    } elseif ($somethingElseIsTheCase) {
        do_something_special();
    } else {
        do_something_else();
    }

    if ($somethingSpecialIsTheCase and $somethingElseIsTheCase) {
        do_something_extra_else();
    }
}
```

```
if ($somethingIsHappening) {  
    if ($somethingIsTheCase or $somethingSpecialIsTheCase) {  
        do_something();  
        if ($somethingSuperRareIsTheCase) {  
            do_something_extra();  
        }  
    } elseif ($somethingElseIsTheCase) {  
        do_something_special();  
    } else {  
        do_something_else();  
    }  
  
    if ($somethingSpecialIsTheCase and $somethingElseIsTheCase) {  
        do_something_extra_else();  
    }  
}
```

```
if ($somethingIsHappening) {
    if ($somethingIsTheCase or $somethingSpecialIsTheCase) {
        do_something();
        if ($somethingSuperRareIsTheCase) {
            do_something_extra();
        }
    } elseif ($somethingElseIsTheCase) {
        do_something_special();
    } else {
        do_something_else();
    }

    if ($somethingSpecialIsTheCase and $somethingElseIsTheCase) {
        do_something_extra_else();
    }

    if ($somethingSpecialIsTheCase and $somethingSuperIsTheCase) {
    }
}
```

```
if ($somethingIsHappening) {
    if ($somethingIsTheCase or $somethingSpecialIsTheCase) {
        do_something();
        if ($somethingSuperRareIsTheCase) {
            do_something_extra();
        }
    } elseif ($somethingElseIsTheCase) {
        do_something_special();
    } else {
        do_something_else();
    }

    if ($somethingSpecialIsTheCase and $somethingElseIsTheCase) {
        do_something_extra_else();
    }

    if ($somethingSpecialIsTheCase and $somethingSuperIsTheCase) {
        do_something_super_special();
    }
}
```

**P
C
O
D
E
S**

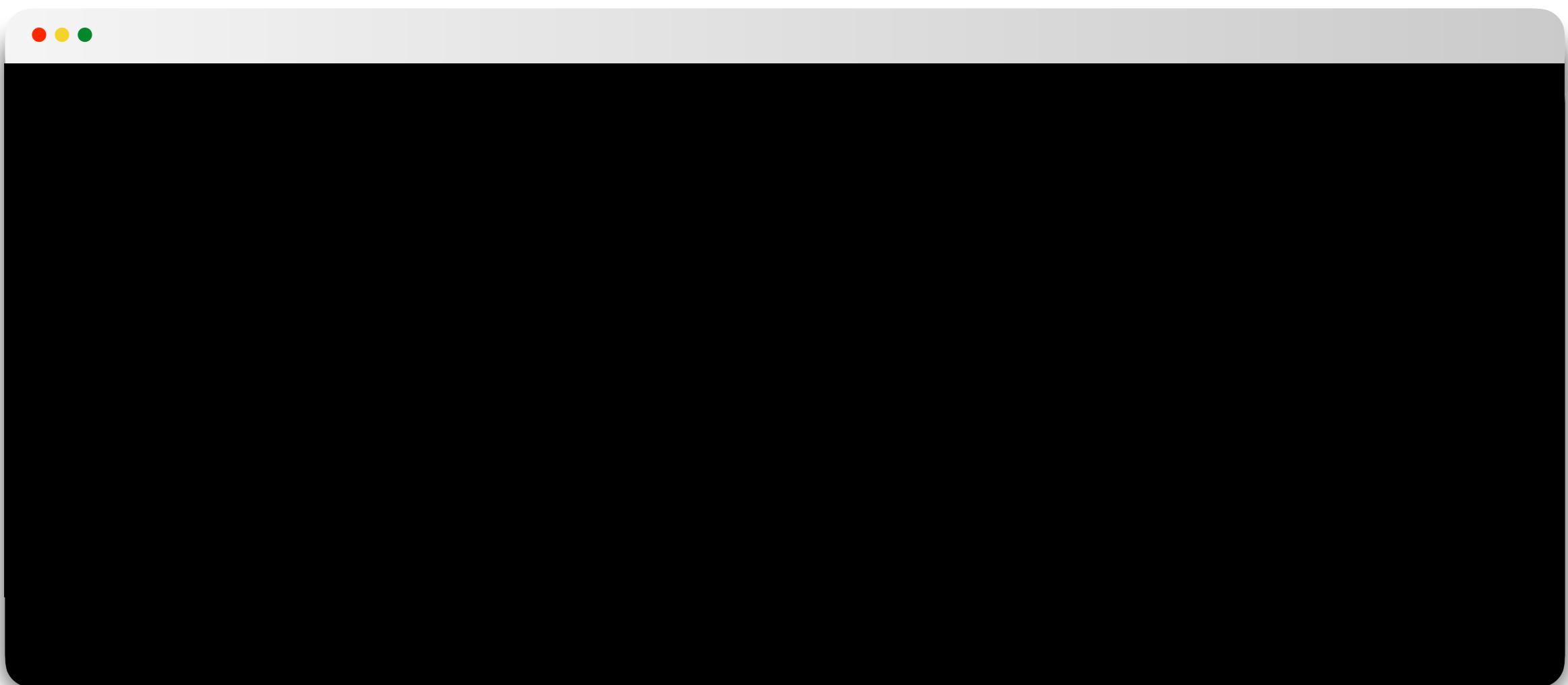
```
1 <?php  
2  
3 $condition = true;  
4
```

The screenshot shows a debugger interface with a dark theme. At the top, there are three colored window control buttons (red, yellow, green). Below that is a header bar with the text "line #* E I O op return operands". A dashed horizontal line separates this from the main table area. The table has four columns: line number, address (*), execution state (E), instruction (I), operation (O), return value, and operands. The data is as follows:

line	#*	E	I	O	op	return	operands
3	0	E >			EXT STM		
			1		ASSIGN		!0, <true>
4	2	>			RETURN		1

A large yellow callout bubble highlights the "op" column for the second row (line 3, offset 1). It contains the text "EXT STM", "ASSIGN", and "RETURN", with "ASSIGN" being the active part. The entire callout is highlighted in yellow.

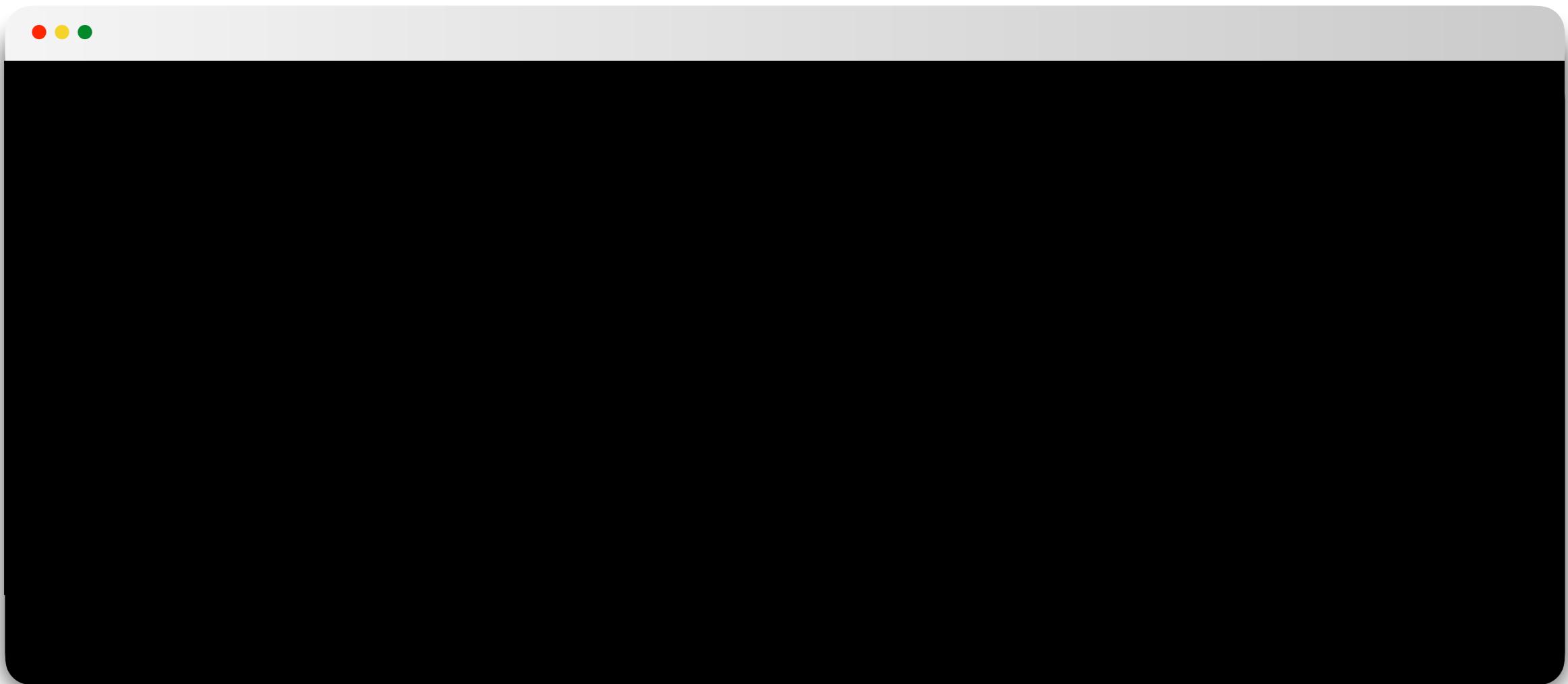
```
1 <?php  
2  
3 $condition = true;  
4  
5
```



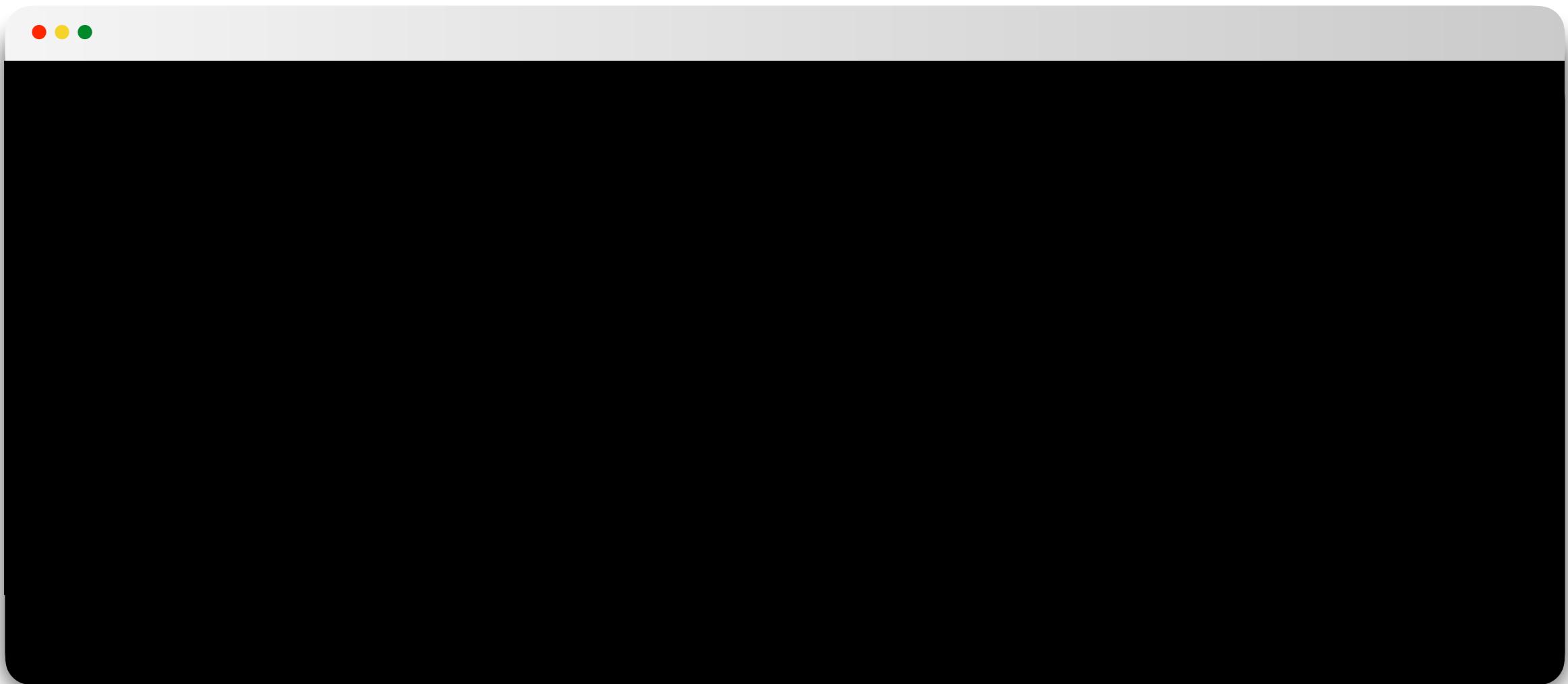
```
1 <?php  
2  
3 $condition = true;  
4 $foo        = true;  
5
```

line	#	*	E	I	O	op	return	operands
3	0		E	>		EXT_STMT		
	1					ASSIGN		!0, <true>
4	2					EXT_STMT		
	3					ASSIGN		!1, <true>
5	4					> RETURN		1

```
1 <?php  
2  
3 $condition = true;  
4 $foo        = true;  
5  
6
```



```
1 <?php  
2  
3 $condition = true;  
4  
5  
6
```



```
1 <?php  
2  
3 $condition = true;  
4 if ($condition) {  
5     echo "hello if";  
6 }  
7
```

line	#	*	E	I	O	op	return	operands
3	0		E	>		EXT_STMT		
	1					ASSIGN		!0, <true>
4	2					EXT_STMT		
	3					> JMPZ		!0, ->6
5	4					> EXI_STMT		
	5					ECHO		'hello+if'
7	6					> > RETURN		1

```
1 <?php  
2  
3 $condition = true;  
4 if (!$condition) {  
5     echo "hello if";  
6 }  
7
```

line	#	*	E	I	O	op	return	operands
3	0		E	>		EXT_STMT		
	1					ASSIGN		!0, <true>
4	2					EXT_STMT		
	3					BOOL NOT	~2	!0
	4			>		JMPZ		~2, ->7
5	5			>		EXT_STMT		
	6					ECHO		'hello+if'
7	7			>	>	RETURN		1

```
1 <?php  
2  
3 $condition = true;  
4 if ($condition) {  
5     echo "hello if";  
6 } else {  
7     echo "hello else";  
8 }  
9
```

line	#	*	E	I	O	op	return	operands
3	0		E	>		EXT_STMT		
	1					ASSIGN		!0, <true>
4	2					EXT_STMT		
	3					> JMPZ		!0, ->7
5	4					> EXT_STMT		
	5					ECHO		'hello+if'
	6					> JMP		->9
7	7					> EXT_STMT		
	8					ECHO		'hello+else'
9	9					> > RETURN		1

```

1 <?php
2
3 $condition = true;
4 switch ($condition) {
5     case true:
6         echo 'hello switch';
7         break;
8     default:
9         echo 'hello default';
10        break;
11 }
13

```

line	#	*	E	I	O	op	return	operands
3	0		E	>		EXT_STMT		
	1					ASSIGN		!0, <true>
4	2					EXT_STMT		
	3					CASE	~2	!0. <true>
	4			>		JMPNZ		~2, ->6
	5			>	>	JMP		->10
6	6			>		EXT_STMT		
	7					ECHO		,hello+switch'
7	8					EXT_STMT		
	9			>		JMP		->14
9	10			>		EXT_STMT		
	11					ECHO		'hello+default'
10	12					EXT_STMT		
	13			>		JMP		->14
13	14			>	>	RETURN		1

```

1 <?php
2
3 if (isset($_GET['foo'])) {
4     $foo = $_GET['foo'];
5 } else {
6     $foo = '';
7 }
8

```

line	#*	E	I	0	op	return	operands
3	0	E	>		EXT STMT		
	1				FETCH IS	\$1	' GET'
	2				ISSET_ISEMPTY_DIM_OBJ	~2	\$1, 'foo'
	3				> JMPZ		~2, ->9
4	4		>		EXT STMT		
	5				FETCH R	\$3	' GET'
	6				FETCH DIM R	\$4	\$3, 'foo'
	7				ASSIGN		!0, \$4
	8				> JMP		->11
6	9		>		EXT STMT		
	10				ASSIGN		!0, ''
8	11		>	>	RETURN	1	

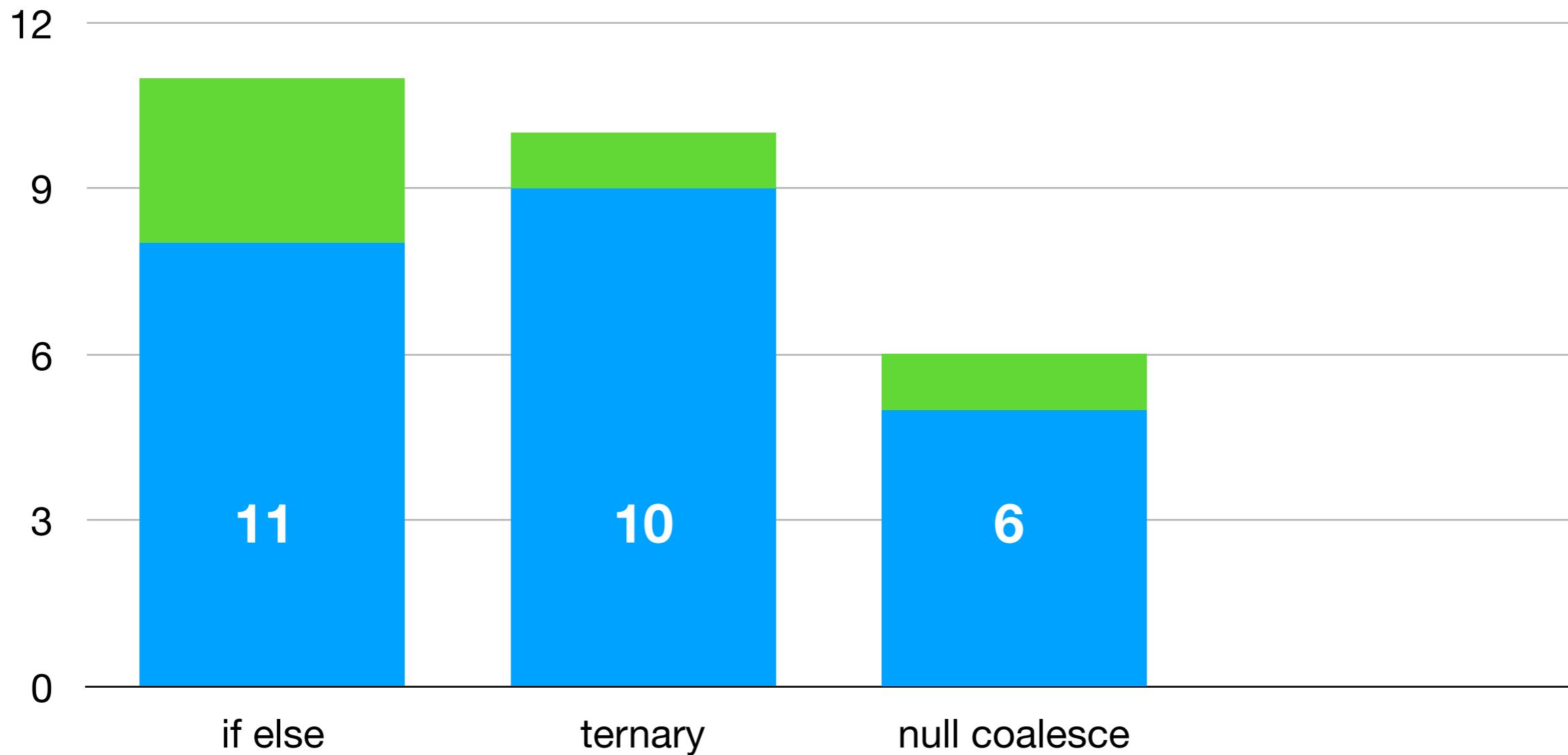
```
1 <?php  
2  
3 $foo = isset($_GET['foo']) ? $_GET['foo'] : '';  
4
```

line	#*	E	I	0	op	return	operands
3	0	E	>		EXT_STMT		
	1				FETCH_IS	\$1	'GET'
	2				ISSET_ISEMPTY_DIM_OBJ	~2	\$1, 'foo'
	3				> JMPZ	~2,	->8
	4				> FETCH_R	\$3	'GET'
	5				FETCH_DIM_R	\$4	\$3, 'foo'
	6				QM_ASSIGN	~5	\$4
	7				> JMP		->9
	8				> OM_ASSIGN	~5	''
	9				> ASSIGN		!0, ~5
4	10				> RETURN	1	

```
1 <?php  
2  
3 $foo = $_GET['foo'] ?? '';  
4
```

line	#	*	E	I	O	op	return	operands
3	0		E	>		EXT_STMT		
	1					FETCH_IS	\$1	' GET'
	2					FETCH_DIM_IS	\$2	\$1. 'foo'
	3					COALESCE	~3	\$2
	4					QM_ASSIGN	~3	''
	5					ASSIGN		{
4	6					> RETURN	1	!0, ~3

OP-Codes





Cyclomatic Complexity

... is defined as the number of
linearly independent paths within
a section of source code

```
public function doSomething()
{
    $this->foo();

    $this->bar();

    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```



```
public function doSomething()
{
    $this->foo();

    $this->bar();

    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```

```
public function doSomething()
{
    $this->foo();

    $this->bar();

    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```

```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    }
    $this->bar();

    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```

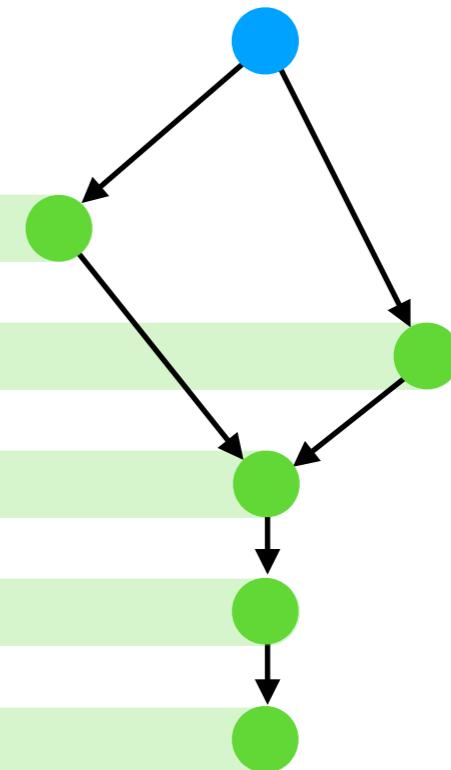
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    }
    $this->bar();

    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```

```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    $this->fiz();
    $this->baz();
    return $this->computeResponse();
}
```



```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```

```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }

    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```

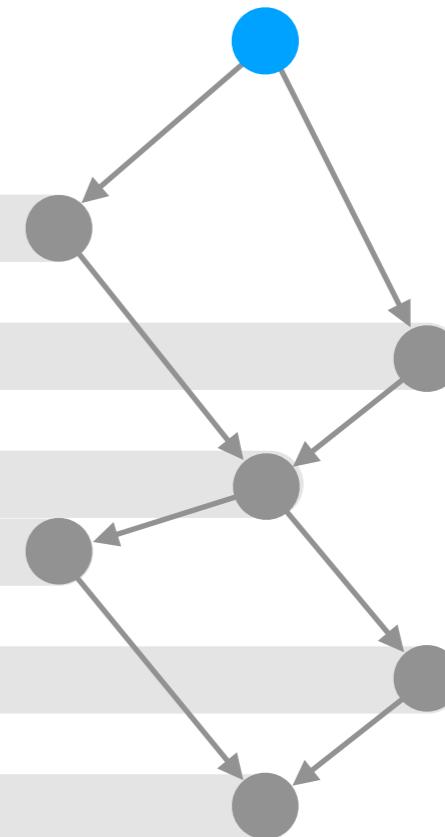
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    }
    $this->baz();

    return $this->computeResponse();
}
```

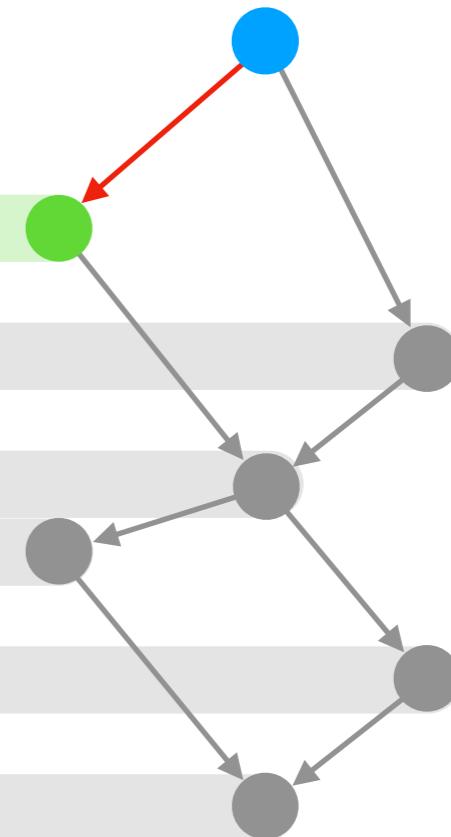
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    }
    $this->baz();

    return $this->computeResponse();
}
```

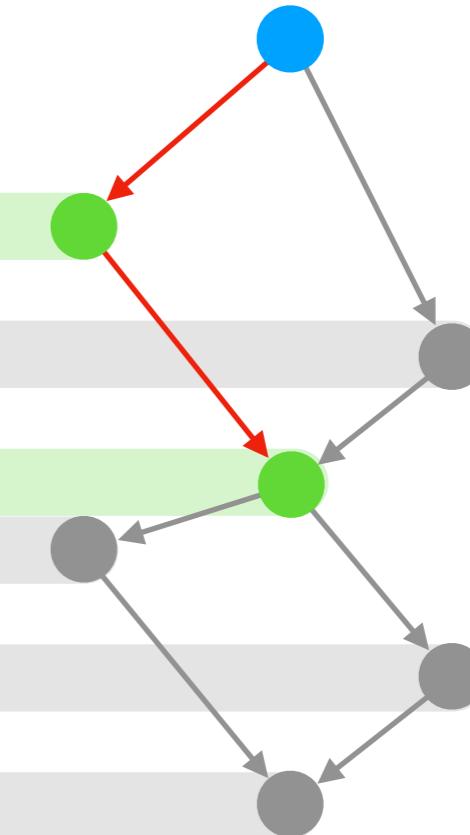
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



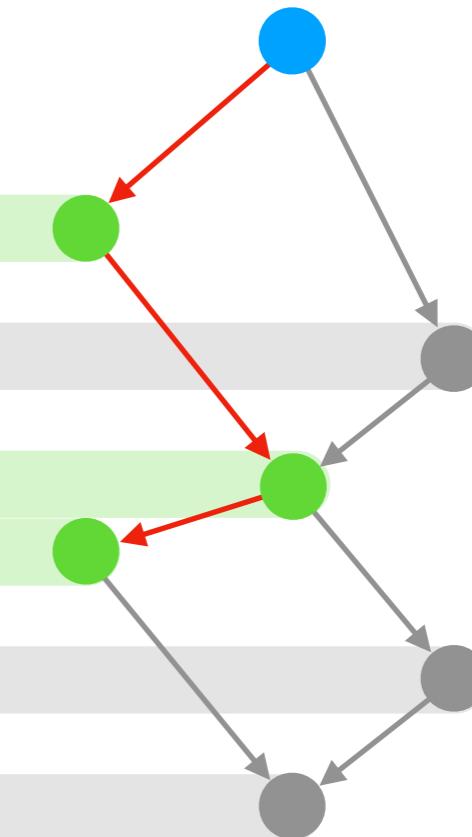
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



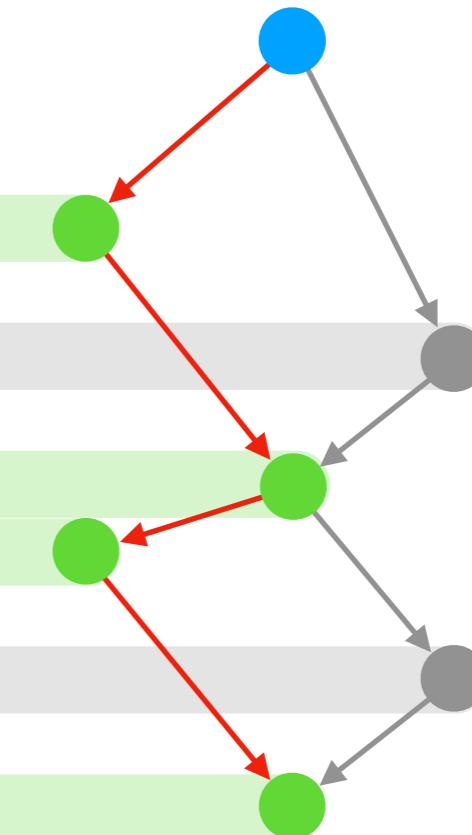
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



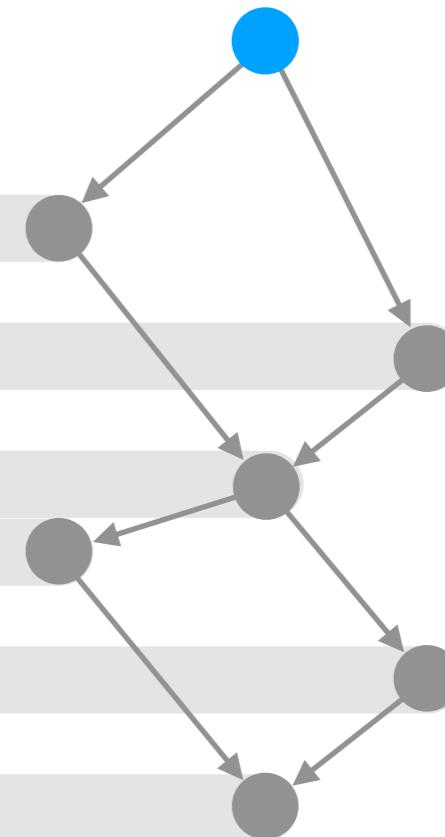
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



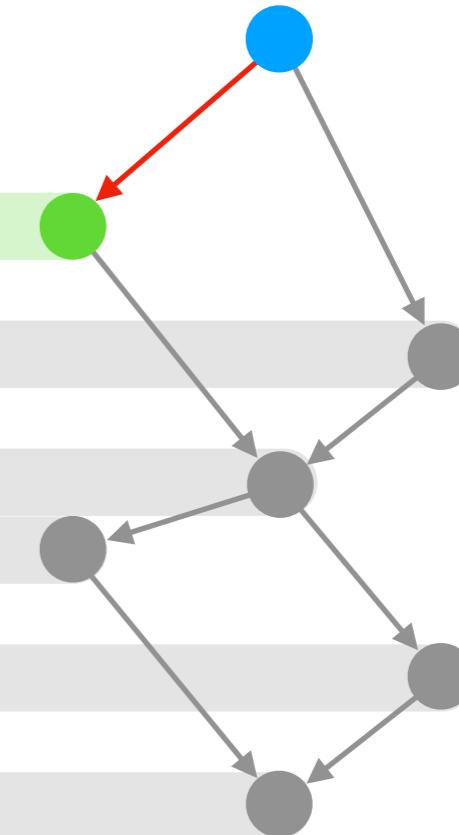
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



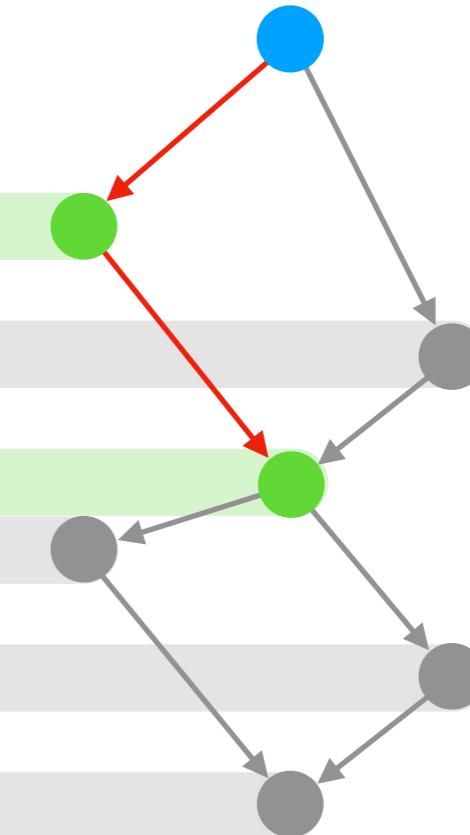
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



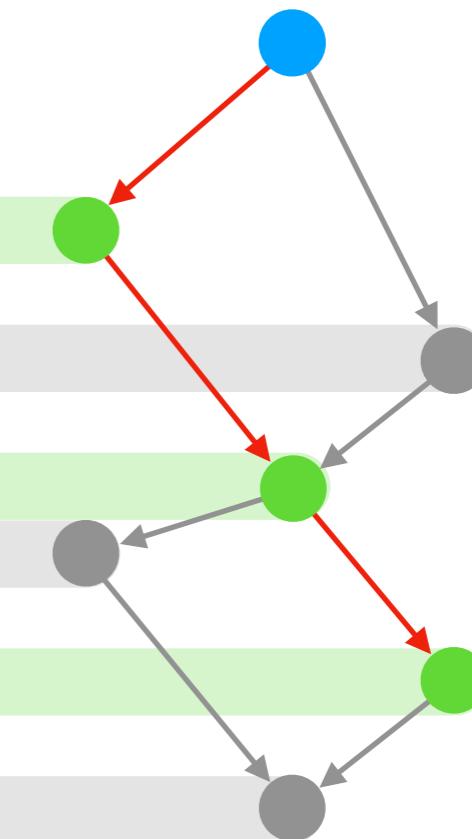
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



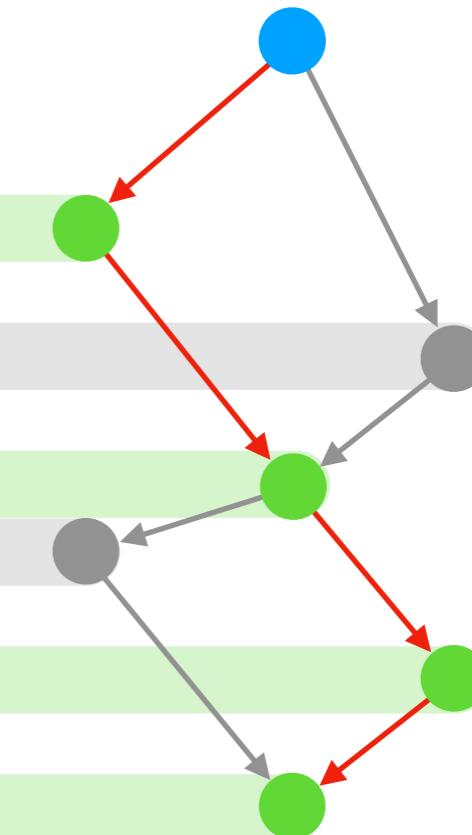
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



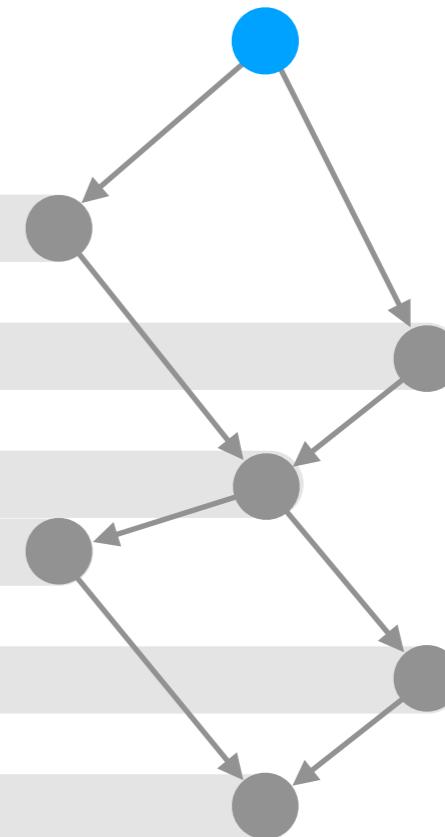
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



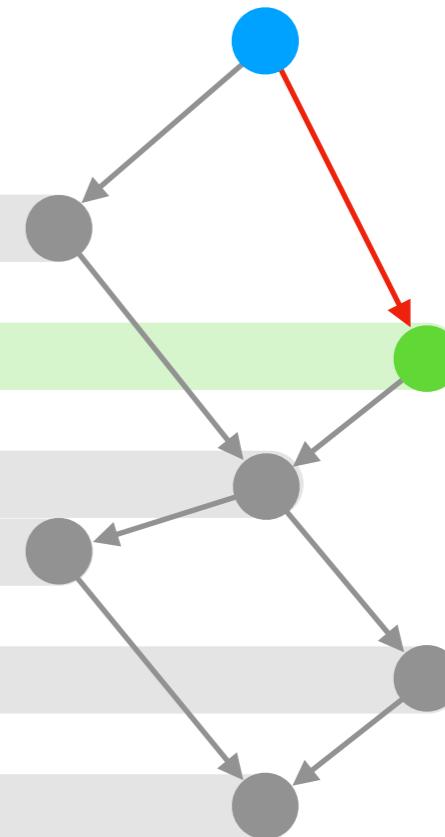
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



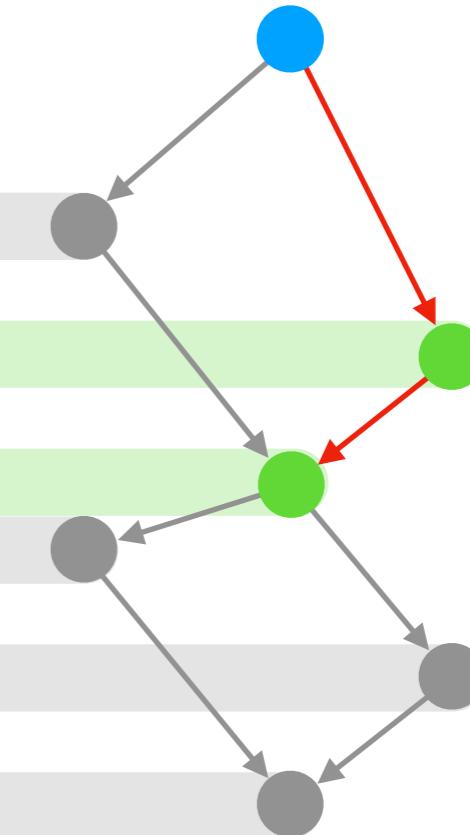
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



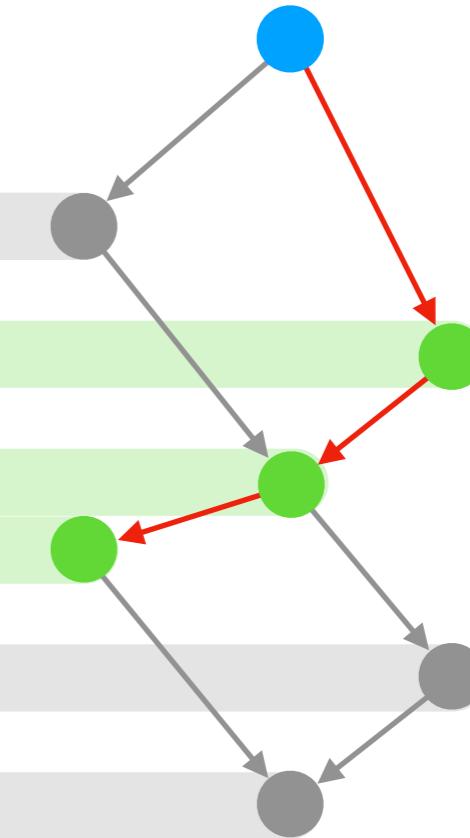
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



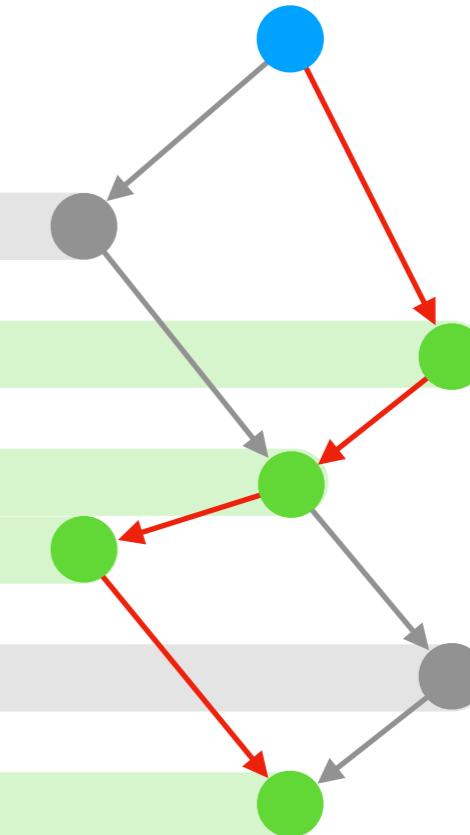
```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```

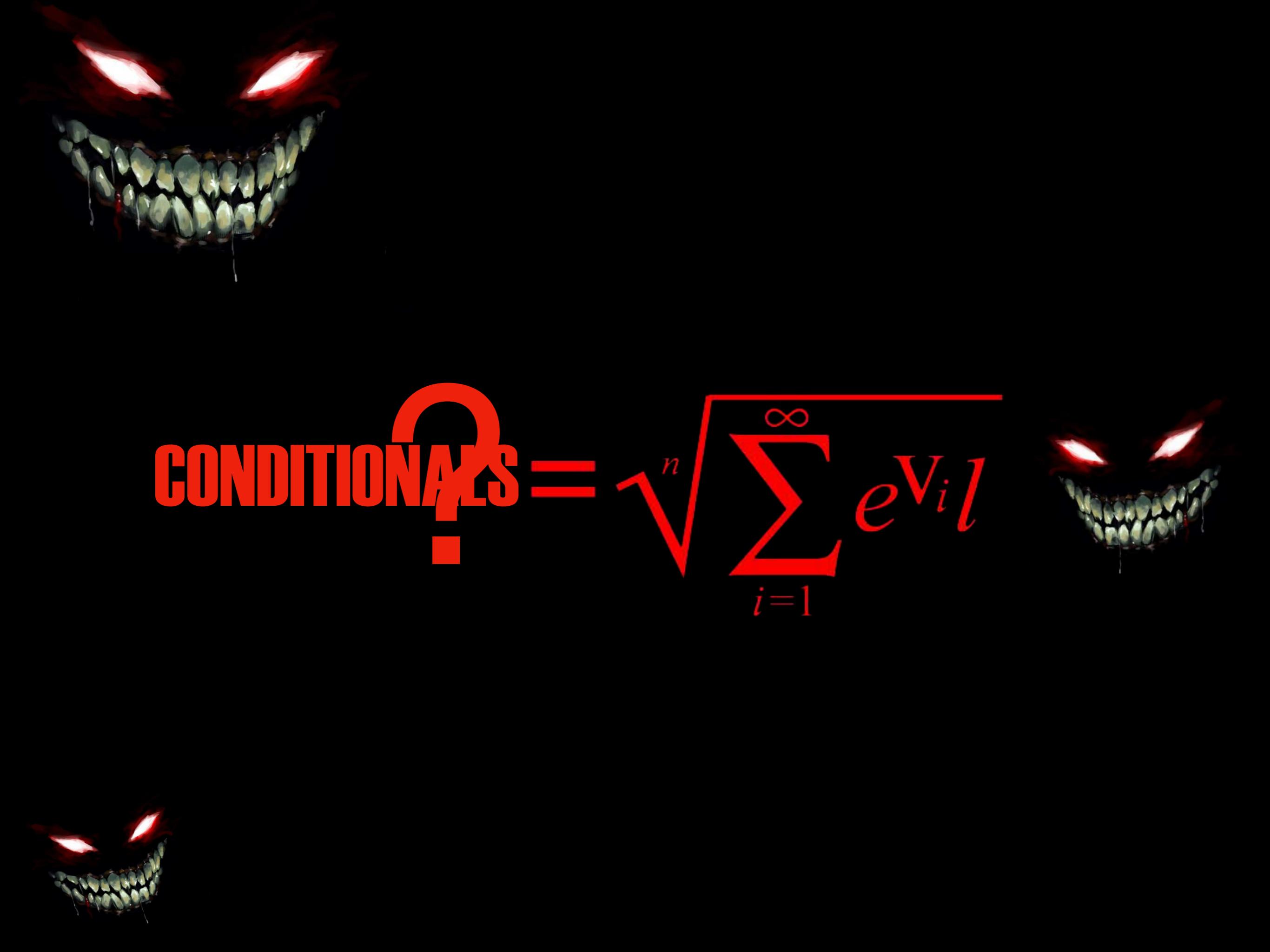


```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```

```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo(); ←
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz(); →
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```

CAUTION

**A MODULE SHOULD HAVE AT
LEAST AS MANY TESTS AS ITS
CYCLOMATIC COMPLEXITY**


$$\text{CONDITIONALS} = \sqrt{\sum_{i=1}^n e^{V_i}}$$

PHP Keywords

<code>_halt_compiler()</code>	<code>abstract</code>	<code>and</code>	<code>array()</code>	<code>as</code>
<code>break</code>	<code>callable</code>	<code>case</code>	<code>catch</code>	<code>class</code>
<code>clone</code>	<code>const</code>	<code>continue</code>	<code>declare</code>	<code>default</code>
<code>die()</code>	<code>do</code>	<code>echo</code>	<code>else</code>	<code>elseif</code>
<code>empty()</code>	<code>enddeclare</code>	<code>endfor</code>	<code>endforeach</code>	<code>endif</code>
<code>endswitch</code>	<code>endwhile</code>	<code>eval()</code>	<code>exit()</code>	<code>extends</code>
<code>final</code>	<code>finally</code>	<code>for</code>	<code>foreach</code>	<code>function</code>
<code>global</code>	<code>goto</code>	<code>if</code>	<code>implements</code>	<code>include</code>
<code>include_once</code>	<code>instanceof</code>	<code>insteadof</code>	<code>interface</code>	<code>isset()</code>
<code>list()</code>	<code>namespace</code>	<code>new</code>	<code>or</code>	<code>print</code>
<code>private</code>	<code>protected</code>	<code>switch</code>	<code>require</code>	<code>require_once</code>
<code>return</code>	<code>static</code>	<code>switch</code>	<code>throw</code>	<code>trait</code>
<code>try</code>	<code>unset()</code>	<code>use</code>	<code>var</code>	<code>while</code>
<code>xor</code>	<code>yield</code>			

Smalltalk Keywords

true
false
nil
super
thisContext





**Sweet world
where everything
is an object**

objects everywhere

`$object->method()`

objects everywhere

```
$object->method()
```

```
'some text'->append('more text')
```

objects everywhere

```
$object->method()
```

```
'some text'->append('more text')
```

```
'some text'->__call('append', 'more text')
```

objects everywhere

\$object->method()

'some text'->append('more text')

'some text'->__call('append', 'more text')

1 + 1

objects everywhere

```
$object->method()
```

```
'some text'->append('more text')
```

```
'some text'->__call('append', 'more text')
```

```
1 + 1
```

```
1->+(1)
```

objects everywhere

```
$object->method()
```

```
'some text'->append('more text')
```

```
'some text'->__call('append', 'more text')
```

```
1 + 1
```

```
1->+(1)
```

```
1->__call('+', 1)
```

objects everywhere

`1 === 1`

objects everywhere

1 === 1

1->====(1)

objects everywhere

```
1 === 1
```

```
1->===(1)
```

```
1->__call('==>', 1) : BooleanClass
```

```
1 <?php
2
3 interface BooleanClass
4 {
5     public function ifTrue($callable): BooleanClass;
6     public function ifFalse($callable): BooleanClass;
7 }
8
```

```
1 <?php
2
3 class TrueClass implements BooleanClass
4 {
5     public function ifTrue($callable): BooleanClass
6     {
7         call_user_func($callable);
8         return $this;
9     }
10
11    public function iffFalse($callable): BooleanClass
12    {
13        return $this;
14    }
15 }
16
```

```
1 <?php
2
3 class FalseClass implements BooleanClass
4 {
5     public function ifTrue($callable): BooleanClass
6     {
7         return $this;
8     }
9
10    public function iffFalse($callable): BooleanClass
11    {
12        call_user_func($callable);
13        return $this;
14    }
15 }
16
```

```
1 <?php  
2  
3 Evaluate::condition($a === $b)  
4  
5  
6  
7  
8  
9  
10
```

```
1 <?php  
2  
3 Evaluate::condition($a === $b)  
4     ->ifTrue(  
5         )  
6  
7  
8  
9  
10
```

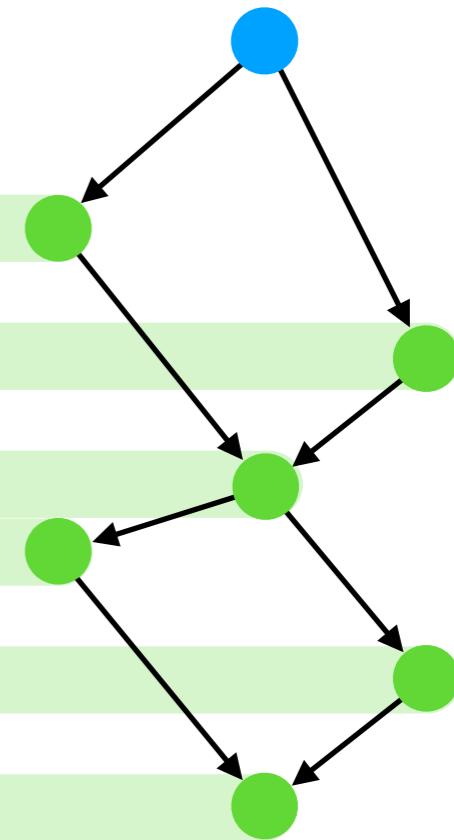
```
1 <?php  
2  
3 Evaluate::condition($a === $b)  
4     ->ifTrue(  
5         )  
6     ->ifFalse(  
7         )  
8     );  
9  
10
```

```
1 <?php
2
3 Evaluate::condition($a === $b)
4     ->ifTrue(function() {
5         echo 'hello if';
6     })
7     ->ifFalse(
8
9 );
10
```

```
1 <?php
2
3 Evaluate::condition($a === $b)
4     ->ifTrue(function() {
5         echo 'hello if';
6     })
7     ->ifFalse(function() {
8         echo 'hello else';
9     });
10
```

```
1 <?php
2
3 Evaluate::condition($a === $b)
4     ->ifTrue(function() {
5         echo 'hello if';
6     })
7     ->ifFalse(function() {
8         echo 'hello else';
9     });
10
```

```
public function doSomething()
{
    if ($this->someTypeCheck()) {
        $this->foo();
    } else {
        $this->bar();
    }
    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```



```
public function doSomething()
{
    $this->foo();

    $this->bar();

    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```

```
public function doSomething()
{
    Evaluate::condition($this->someTypeCheck())

        ->ifTrue( [$this, , foo' ] )

            $this->bar();

    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```

```
public function doSomething()
{
    Evaluate::condition($this->someTypeCheck())

        ->ifTrue([$this, 'foo'])

        ->ifFalse([$this, 'bar']);

    if ($this->someOtherTypeCheck()) {
        $this->fiz();
    } else {
        $this->baz();
    }
    return $this->computeResponse();
}
```

```
public function doSomething()
{
    Evaluate::condition($this->someTypeCheck())

        ->ifTrue( [$this, 'foo' ] )

        ->ifFalse( [$this, 'bar' ] );

    $this->fiz();

    $this->baz();

    return $this->computeResponse();
}
```

```
public function doSomething()
{
    Evaluate::condition($this->someTypeCheck())

        ->ifTrue( [$this, 'foo' ] )

        ->ifFalse( [$this, 'bar' ] );

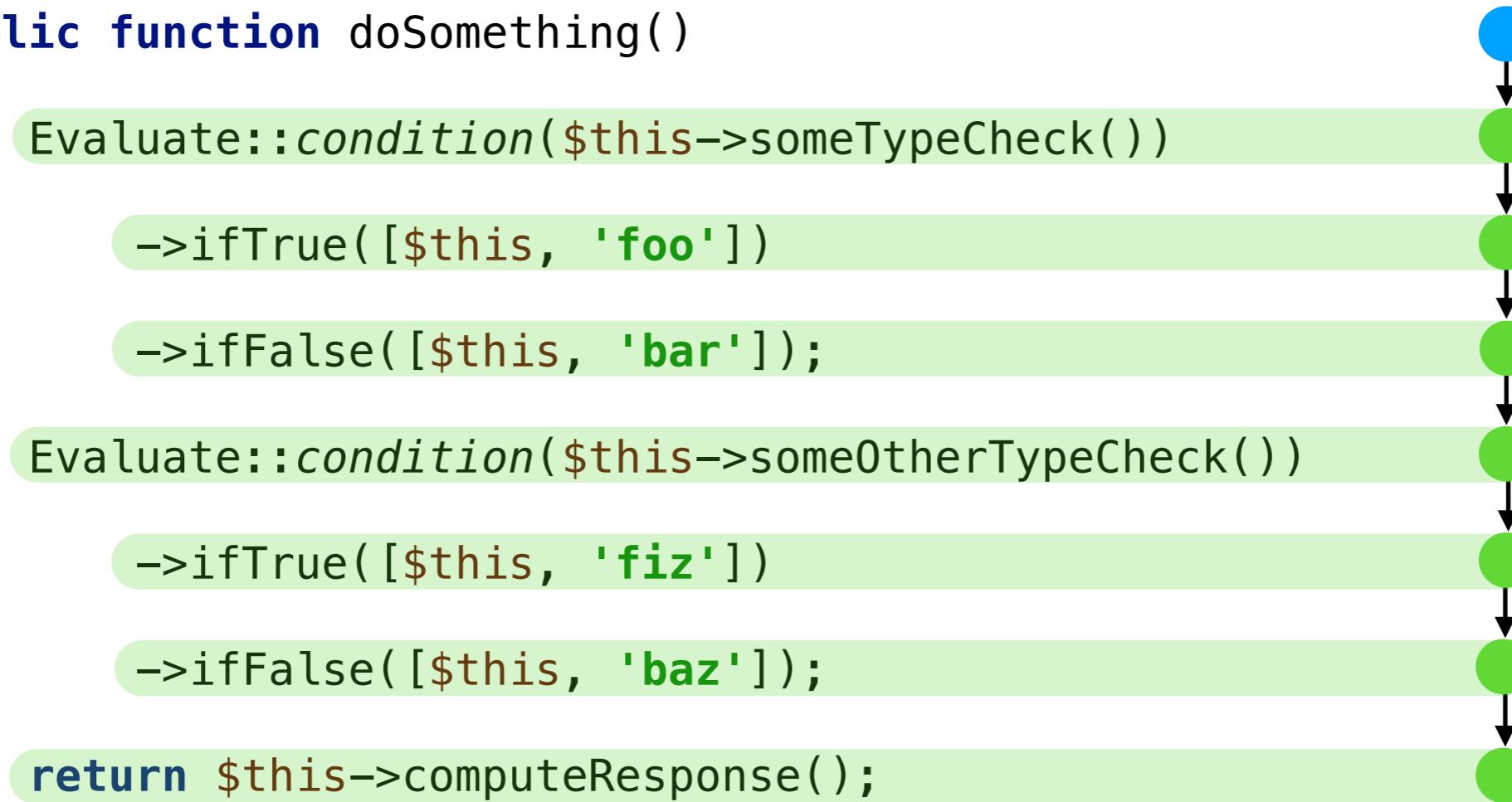
    Evaluate::condition($this->someOtherTypeCheck())

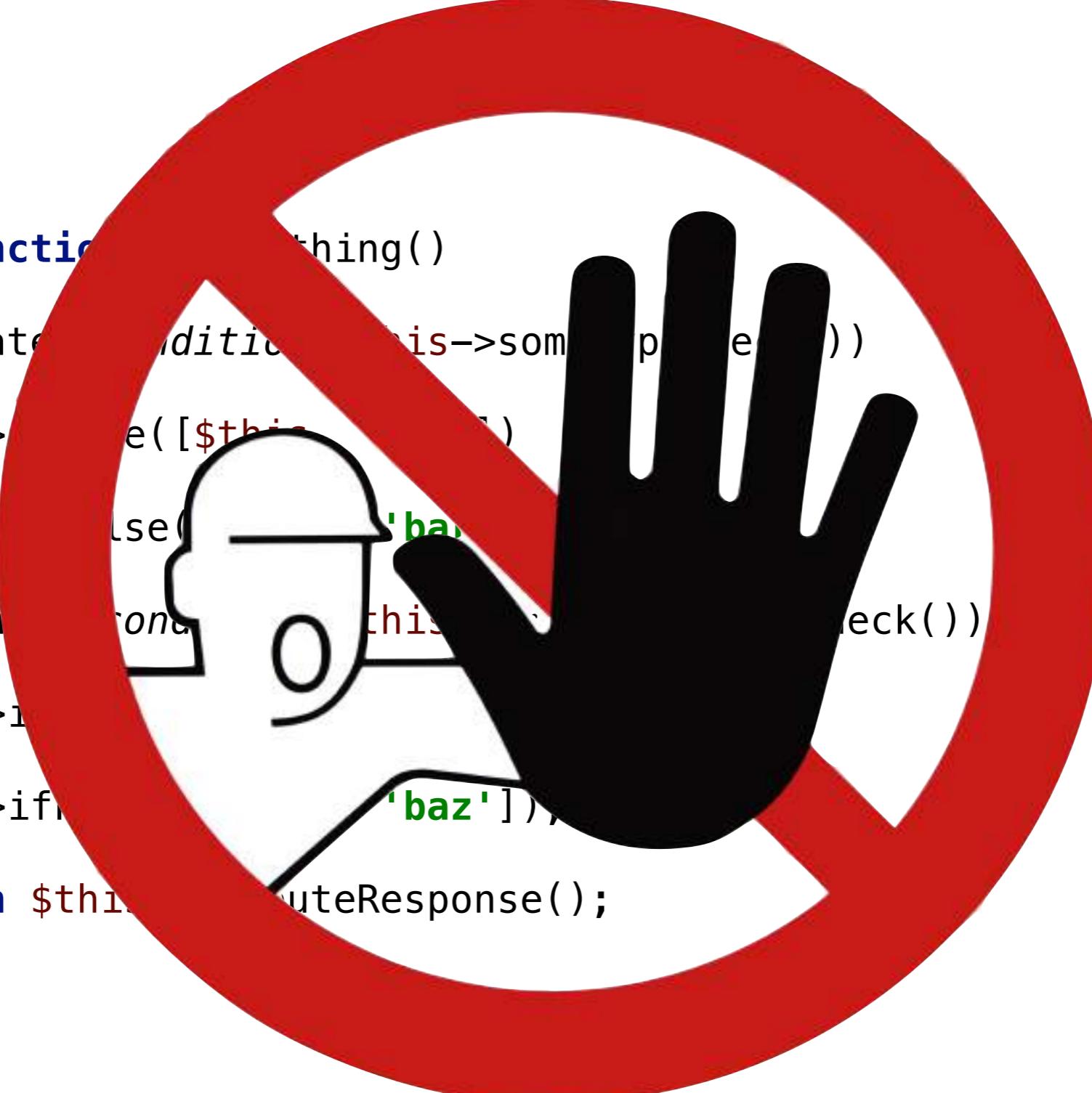
        ->ifTrue( [$this, 'fiz' ] )

        $this->baz();

    return $this->computeResponse();
}
```

```
public function doSomething()
{
    Evaluate::condition($this->someTypeCheck())
        ->ifTrue( [$this, 'foo' ] )
        ->ifFalse( [$this, 'bar' ] );
    Evaluate::condition($this->someOtherTypeCheck())
        ->ifTrue( [$this, 'fiz' ] )
        ->ifFalse( [$this, 'baz' ] );
    return $this->computeResponse();
}
```





```
public function something()
{
    Evaluate condition ($this->someProp == 1)
    -> if (true) use ('baz')
    Evaluate condition ($this->check())
    -> if (true) use ('baz')
    return $this->oluteResponse();
}
```

Gilded Rose Kata



```
class GildedRose
{
    private $name;
    private $quality;
    private $sellIn;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->name = $name;
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        # ...
    }
}
```

```
class GildedRose
{
    private $name;
    private $quality;
    private $sellIn;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->name = $name;
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

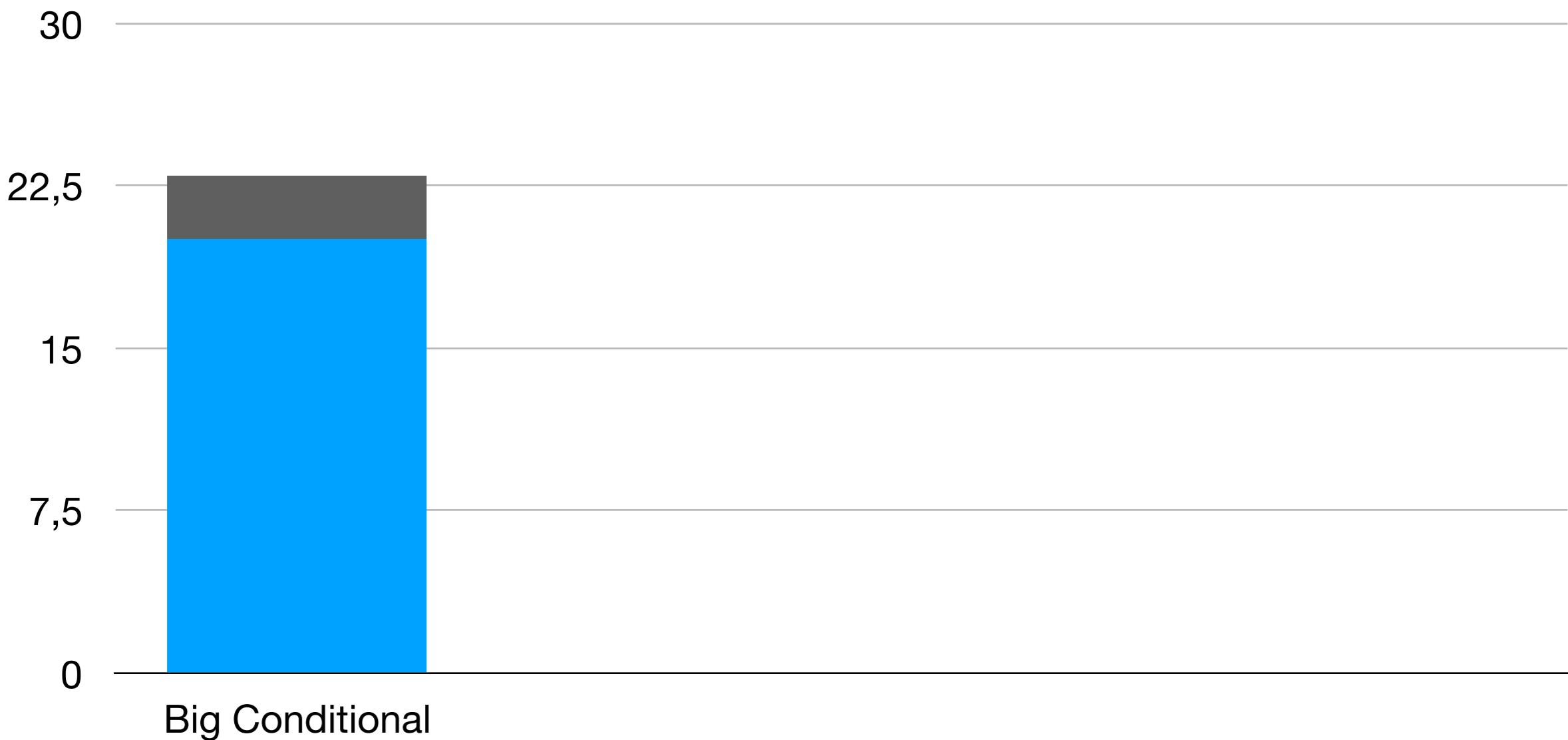
    public function tick() : void
    {
        if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
            if ($this->quality > 0) {
                if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                    $this->quality -= 1;
                }
            }
        } else {
            if ($this->quality < 50) {
                $this->quality += 1;
                if ($this->name == 'Backstage passes to a TAFKAL80ETC concert') {
                    if ($this->sellIn < 11) {
                        if ($this->quality < 50) {
                            $this->quality += 1;
                        }
                    }
                    if ($this->sellIn < 6) {
                        if ($this->quality < 50) {
                            $this->quality += 1;
                        }
                    }
                }
            }
        }
    }
} # more to come
```

```
# previous lines exist

if ($this->name != 'Sulfuras, Hand of Ragnaros') {
    $this->sellIn -= 1;
}
if ($this->sellIn < 0) {
    if ($this->name != 'Aged Brie') {
        if ($this->name != 'Backstage passes to a TAFKAL80ETC concert') {
            if ($this->quality > 0) {
                if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                    $this->quality -= 1;
                }
            }
        } else {
            $this->quality = $this->quality - $this->quality;
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
        }
    }
}
}
```

```
public function tick() : void
{
    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
            if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                $this->quality -= 1;
            }
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
            if ($this->name == 'Backstage passes to a TAFKAL80ETC concert') {
                if ($this->sellIn < 11) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
                if ($this->sellIn < 6) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
            }
        }
    }
    if ($this->name != 'Sulfuras, Hand of Ragnaros') {
        $this->sellIn -= 1;
    }
    if ($this->sellIn < 0) {
        if ($this->name != 'Aged Brie') {
            if ($this->name != 'Backstage passes to a TAFKAL80ETC concert') {
                if ($this->quality > 0) {
                    if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                        $this->quality -= 1;
                    }
                }
            } else {
                $this->quality = $this->quality - $this->quality;
            }
        } else {
            if ($this->quality < 50) {
                $this->quality += 1;
            }
        }
    }
}
```

C.R.A.P



```
public function tick() : void
{
    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
            if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                $this->quality -= 1;
            }
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
        }
        if ($this->name == 'Backstage passes to a TAFKAL80ETC concert') {
            if ($this->sellIn < 11) {
                if ($this->quality < 50) {
                    $this->quality += 1;
                }
            }
            if ($this->sellIn < 6) {
                if ($this->quality < 50) {
                    $this->quality += 1;
                }
            }
        }
    }
}
if ($this->name != 'Sulfuras, Hand of Ragnaros') {
    $this->sellIn -= 1;
}
if ($this->sellIn < 0) {
    if ($this->name != 'Aged Brie'){
        if ($this->name != 'Backstage passes to a TAFKAL80ETC concert') {
            if ($this->quality > 0){
                if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                    $this->quality -= 1;
                }
            }
        }
    } else {
        $this->quality = $this->quality - $this->quality;
    }
} else {
    if ($this->quality < 50) {
        $this->quality += 1;
    }
}
```

16 if statements

7 !=

2 != with &&

3 strings

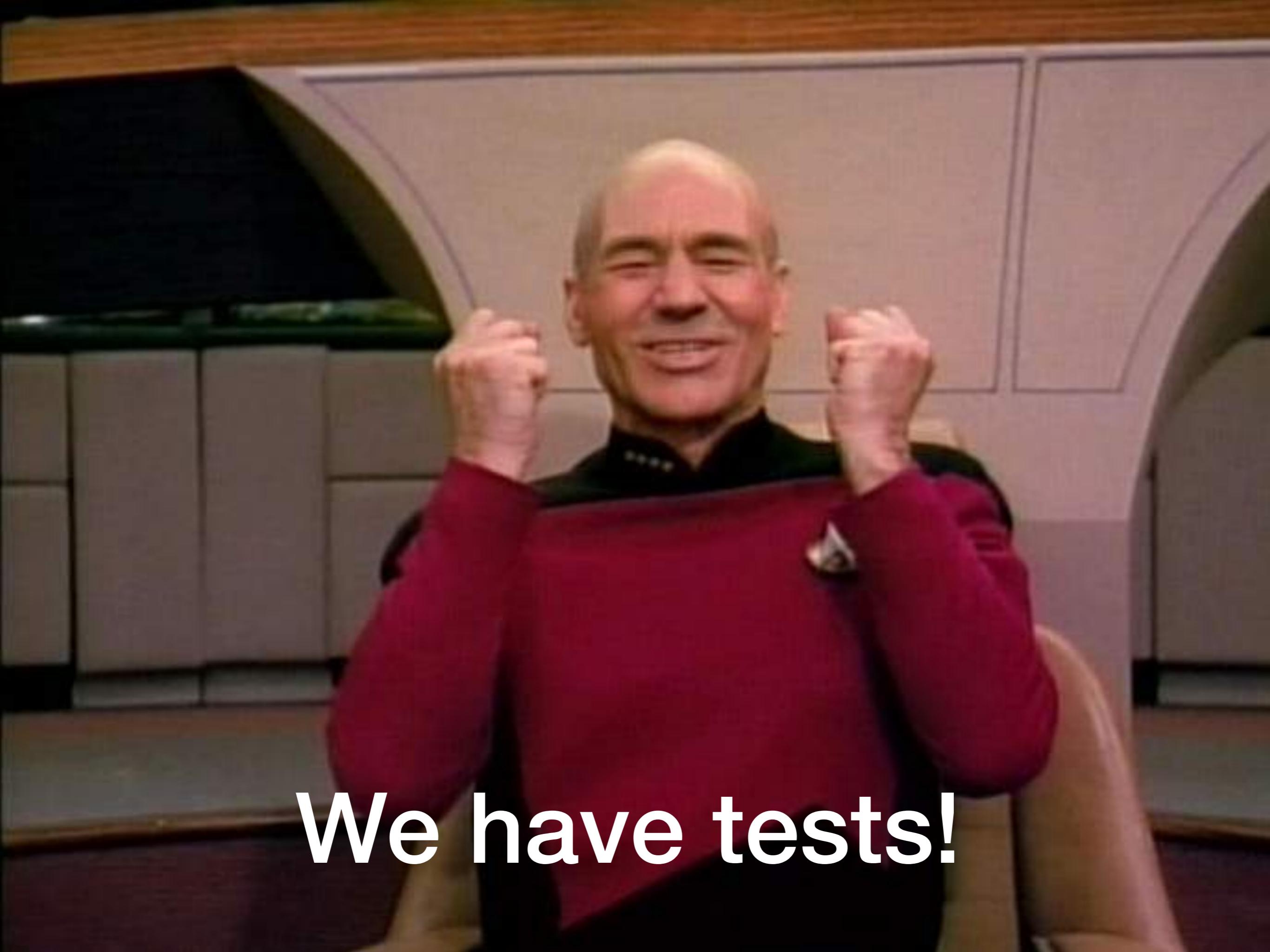
lot of numbers

```
public function tick() : void
{
    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
            if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                $this->quality -= 1;
            }
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
            if ($this->name == 'Backstage passes to a TAFKAL80ETC concert') {
                if ($this->sellIn < 11) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
                if ($this->sellIn < 6) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
            }
        }
    }
    if ($this->name != 'Sulfuras, Hand of Ragnaros') {
        $this->sellIn -= 1;
    }
    if ($this->sellIn < 0) {
        if ($this->name == 'Aged Brie') {
            if ($this->name != 'Backstage passes to a TAFKAL80ETC concert') {
                if ($this->quality > 0) {
                    if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                        $this->quality -= 1;
                    }
                }
            }
        } else {
            $this->quality = $this->quality - $this->quality;
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
        }
    }
}
```

'Aged Brie'

'Backstage passes to a TAFKAL80ETC concert'

'Sulfuras, Hand of Ragnaros'



We have tests!



```
$ phpunit  
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.
```

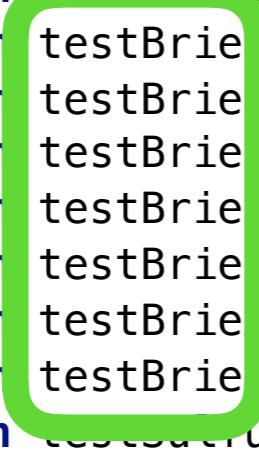
```
.....SSSS
```

```
29 / 29 (100%)
```

```
Time: 369 ms, Memory: 12.00MB
```

```
OK, but incomplete, skipped, or risky tests!  
Tests: 29, Assertions: 50, Skipped: 4.
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }

    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

'Aged Brie'

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

'Sulfuras, Hand of Ragnaros'

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
}
```

'Backstage passes to a TAFKAL80ETC concert'

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalIfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

'Normal'

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose( name 'normal', quality 10, sellIn 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }

    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose( name 'normal', quality 10, sellIn 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }

    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    # more tests above

    public function testConjuredBeforeSellDate()
    public function testConjuredOnSellDate()
    public function testConjuredAfterSellDate()
    public function testConjuredOfZeroQuality()
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    # more tests above

    public function testConjuredBeforeSellDate()
    public function testConjuredOnSellDate()
    public function testConjuredAfterSellDate()
    public function testConjuredToZeroQuality()
}
```

'Conjured'



```
public function tick() : void
{
    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
            if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                $this->quality -= 1;
            }
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
            if ($this->name == 'Backstage passes to a TAFKAL80ETC concert') {
                if ($this->sellIn < 11) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
                if ($this->sellIn < 6) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
            }
        }
    }
}
if ($this->name != 'Sulfuras, Hand of Ragnaros') {
    $this->sellIn -= 1;
}
if ($this->sellIn < 0) {
    if ($this->name != 'Aged Brie') {
        if ($this->name != 'Backstage passes to a TAFKAL80ETC concert') {
            if ($this->quality > 0) {
                if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                    $this->quality -= 1;
                }
            }
        }
    } else {
        $this->quality = $this->quality - $this->quality;
    }
}
```

```
public function tick() : void
{
    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
            if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                $this->quality -= 1;
            }
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
            if ($this->name == 'Backstage passes to a TAFKAL80ETC concert') {
                if ($this->sellIn < 11) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
                if ($this->sellIn < 6) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
            }
        }
    }
}

if ($this->name != 'Sulfuras, Hand of Ragnaros') {
    $this->sellIn -= 1;
}
if ($this->sellIn < 0) {
    if ($this->name != 'Aged Brie') {
        if ($this->name != 'Backstage passes to a TAFKAL80ETC concert') {
            if ($this->quality > 0) {
                if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                    $this->quality -= 1;
                }
            }
        }
    }
}
```

```
public function tick() : void
{
    if ($this->name == 'normal') {
        return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
            if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                $this->quality -= 1;
            }
        }
    } else {
        if ($this->quality < 50) {
            $this->quality += 1;
            if ($this->name == 'Backstage passes to a TAFKAL80ETC concert') {
                if ($this->sellIn < 11) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
                if ($this->sellIn < 6) {
                    if ($this->quality < 50) {
                        $this->quality += 1;
                    }
                }
            }
        }
    }
}

if ($this->name != 'Sulfuras, Hand of Ragnaros') {
    $this->sellIn -= 1;
}

if ($this->sellIn < 0) {
    if ($this->name != 'Aged Brie') {
        if ($this->name != 'Backstage passes to a TAFKAL80ETC concert') {
            if ($this->quality > 0) {
                if ($this->name != 'Sulfuras, Hand of Ragnaros') {
                    $this->quality -= 1;
                }
            }
        }
    }
}
```



```
$ phpunit
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.

FFFF.....SSSS                                         29 / 29 (100%)

Time: 383 ms, Memory: 12.00MB

There were 4 failures:

1) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalBeforeSellDate
Failed asserting that 10 matches expected 9.

/kata.gilded-rose/tests/Kata/GildedRoseTest.php:12

2) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalOnSellDate
Failed asserting that 10 matches expected 8.

/kata.gilded-rose/tests/Kata/GildedRoseTest.php:21

3) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalAfterSellDate
Failed asserting that 10 matches expected 8.

/kata.gilded-rose/tests/Kata/GildedRoseTest.php:30

4) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalOfZeroQuality
Failed asserting that 5 matches expected 4.

/kata.gilded-rose/tests/Kata/GildedRoseTest.php:40

FAILURES!
Tests: 29, Assertions: 47, Failures: 4, Skipped: 4.
```

```
public function tick() : void
{
    if ($this->name == 'normal') {
        return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
            if ($this->name != 'Sulfuras, Hand of Ragnaros') {
```

```
private function tickNormal() : void
{
```

```
public function tick() : void
{
    if ($this->name == 'normal') {
        $this->tickNormal();
        return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
```

```
private function tickNormal() : void
{
```



```
$ phpunit
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.

FFFF.....SSSS                                         29 / 29 (100%)

Time: 383 ms, Memory: 12.00MB

There were 4 failures:

1) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalBeforeSellDate
Failed asserting that 10 matches expected 9.

2) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalOnSellDate
Failed asserting that 10 matches expected 8.

3) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalAfterSellDate
Failed asserting that 10 matches expected 8.

4) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalOfZeroQuality
Failed asserting that 5 matches expected 4.

FAILURES!
Tests: 29, Assertions: 47, Failures: 4, Skipped: 4.
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose('normal', 10, 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose('normal', 10, 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose('normal', 10, 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        $this->quality--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose('normal', 10, 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        $this->quality--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose('normal', 10, 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        $this->quality--;
        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose('normal', 10, 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        $this->quality--;
        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate()
    {
        $rose = new GildedRose('normal', 10, 5);
        $rose->tick();

        $this->assertEquals(9, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        $this->quality--;
        $this->sellIn--;
    }
}
```



```
$ phpunit
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.

.FFF.....SSSS                                         29 / 29 (100%)

Time: 380 ms, Memory: 12.00MB

There were 3 failures:

1) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalOnSellDate
Failed asserting that 9 matches expected 8.

2) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalAfterSellDate
Failed asserting that 9 matches expected 8.

3) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalOfZeroQuality
Failed asserting that -1 matches expected 0.

FAILURES!
Tests: 29, Assertions: 47, Failures: 3, Skipped: 4.
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalOnSellDate()
    {
        $rose = new GildedRose('normal', 10, 0);
        $rose->tick();

        $this->assertEquals(8, $rose->getQuality());
        $this->assertEquals(-1, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        $this->quality--;
        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalOnSellDate()
    {
        $rose = new GildedRose('normal', 10, 0);
        $rose->tick();

        $this->assertEquals(8, $rose->getQuality());
        $this->assertEquals(-1, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        $this->quality--;
        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalOnSellDate()
    {
        $rose = new GildedRose('normal', 10, 0);
        $rose->tick();

        $this->assertEquals(8, $rose->getQuality());
        $this->assertEquals(-1, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {

        $this->quality--;

        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalOnSellDate()
    {
        $rose = new GildedRose('normal', 10, 0);
        $rose->tick();

        $this->assertEquals(8, $rose->getQuality());
        $this->assertEquals(-1, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
        $this->sellIn--;
    }
}
```



```
$ phpunit
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.

...F.....SSSS                                         29 / 29 (100%)

Time: 365 ms, Memory: 12.00MB

There was 1 failure:

1) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testNormalOfZeroQuality
Failed asserting that -1 matches expected 0.

FAILURES!
Tests: 29, Assertions: 49, Failures: 1, Skipped: 4.
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalOfZeroQuality()
    {
        $rose = new GildedRose('normal', 0, 5);
        $rose->tick();

        $this->assertEquals(0, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormal0fZeroQuality()
    {
        $rose = new GildedRose('normal', 0, 5);
        $rose->tick();

        $this->assertEquals(0, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormal0fZeroQuality()
    {
        $rose = new GildedRose('normal', 0, 5);
        $rose->tick();

        $this->assertEquals(0, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        if ($this->quality != 0) {
            if ($this->sellIn > 0 ) {
                $this->quality -= 1;
            } else {
                $this->quality -= 2;
            }
        }
        $this->sellIn--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormal0fZeroQuality()
    {
        $rose = new GildedRose('normal', 0, 5);
        $rose->tick();

        $this->assertEquals(0, $rose->getQuality());
        $this->assertEquals(4, $rose->getDaysRemaining());
    }
}
```

```
class GildedRose
{
    # ...

    private function tickNormal() : void
    {
        if ($this->quality != 0) {
            if ($this->sellIn > 0 ) {
                $this->quality -= 1;
            } else {
                $this->quality -= 2;
            }
        }
        $this->sellIn--;
    }
}
```



```
$ phpunit  
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.
```

```
.....SSSS
```

```
29 / 29 (100%)
```

```
Time: 369 ms, Memory: 12.00MB
```

```
OK, but incomplete, skipped, or risky tests!  
Tests: 29, Assertions: 50, Skipped: 4.
```



```
private function tickNormal() : void
{
    if ($this->quality != 0) {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
    }
    $this->sellIn--;
}
```

```
private function tickNormal() : void
{
    if ($this->quality != 0) {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
    }
    $this->sellIn--;
}
```

```
private function tickNormal() : void
{
```

```
}
```

```
private function tickNormal() : void
{
    if ($this->quality != 0) {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
    }
    $this->sellIn--;
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;
}
```

```
private function tickNormal() : void
{
    if ($this->quality != 0) {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
    }
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;
}
```

```
private function tickNormal() : void
{
    if ($this->quality != 0) {
        if ($this->sellIn > 0) {
            $this->quality -= 1;
        } else {
            $this->quality -= 2;
        }
    }
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }
}
```

```
private function tickNormal() : void
{
    if ($this->sellIn > 0) {
        $this->quality -= 1;
    } else {
        $this->quality -= 2;
    }
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }
}
```

```
private function tickNormal() : void
{
    if ($this->sellIn > 0 ) {
        $this->quality -= 1;
    } else {
        $this->quality -= 2;
    }
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }

    $this->quality--;
}
```

```
private function tickNormal() : void
{
    if ($this->sellIn > 0 ) {
        $this->quality -= 1;
    } else {
        $this->quality -= 2;
    }
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }

    $this->quality--;
    if ($this->sellIn <= 0) {
        $this->quality--;
    }
}
```

```
private function tickNormal() : void
{
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }

    $this->quality--;
    if ($this->sellIn <= 0) {
        $this->quality--;
    }
}
```

```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }

    $this->quality--;
    if ($this->sellIn <= 0) {
        $this->quality--;
    }
}
```



```
$ phpunit  
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.
```

```
.....SSSS
```

```
29 / 29 (100%)
```

```
Time: 369 ms, Memory: 12.00MB
```

```
OK, but incomplete, skipped, or risky tests!
```

```
Tests: 29, Assertions: 50, Skipped: 4.
```

```
public function tick() : void
{
    if ($this->name == 'normal') {
        $this->tickNormal();
        return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        if ($this->quality > 0) {
```

```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }

    $this->quality--;
    if ($this->sellIn <= 0) {
        $this->quality--;
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        // ...
    }
}
```

```
private function tickAgedBrie()
```

```
}
```



```
$ phpunit
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.

....FFFFFFFFFF.....SSSS                                     29 / 29 (100%)

Time: 386 ms, Memory: 12.00MB

There were 7 failures:

1) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testBrieBeforeSellDate
Failed asserting that 10 matches expected 11.

2) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testBrieBeforeSellDateWithMaxQuality
Failed asserting that 5 matches expected 4.

3) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testBrieOnSellDate
Failed asserting that 10 matches expected 12.

4) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testBrieOnSellDateNearMaxQuality
Failed asserting that 49 matches expected 50.

5) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testBrieOnSellDateWithMaxQuality
Failed asserting that 0 matches expected -1.

6) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testBrieAfterSellDate
Failed asserting that 40 matches expected 42.

7) SebastianFeldmann\Kata\GildedRose\GildedRoseTest::testBrieAfterSellDateWithMaxQuality
Failed asserting that -1 matches expected -2.

FAILURES!
Tests: 29, Assertions: 46, Failures: 7, Skipped: 4.
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        // ...
    }
}
```

```
private function tickAgedBrie()
```

```
}
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {
        // ...
    }
}
```

```
private function tickAgedBrie()
{
    $this->sellIn -= 1;

    if ($this->quality >= 50) {
        return;
    }

    $this->quality += 1;
    if ($this->sellIn <= 0) {
        $this->quality += 1;
    }
}
```



```
private function tickNormal() : void
{
    $this->sellIn--;

    if ($this->quality == 0) {
        return;
    }

    $this->quality--;
    if ($this->sellIn <= 0) {
        $this->quality--;
    }
}
```

```
private function tickAgedBrie()
{
    $this->sellIn -= 1;

    if ($this->quality >= 50) {
        return;
    }

    $this->quality += 1;
    if ($this->sellIn <= 0) {
        $this->quality += 1;
    }
}
```



STOP

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
    }
}
```

```
private function tickSulfuras()
{}
```



```
$ phpunit  
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.
```

```
.....SSSS
```

```
29 / 29 (100%)
```

```
Time: 369 ms, Memory: 12.00MB
```

```
OK, but incomplete, skipped, or risky tests!  
Tests: 29, Assertions: 50, Skipped: 4.
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testSulfurasBeforeSellDate()
    {
        $rose = new GildedRose('Sulfuras, Hand of Ragnaros', 10, 5);
        $rose->tick();

        $this->assertEquals(10, $rose->getQuality());
        $this->assertEquals(5, $rose->getDaysRemaining());
    }
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testSulfurasBeforeSellDate()
    {
        $rose = new GildedRose('Sulfuras, Hand of Ragnaros', 10, 5);
        $rose->tick();

        $this->assertEquals(10, $rose->getQuality());
        $this->assertEquals(5, $rose->getDaysRemaining());
    }
}
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
    }
}
```

```
private function tickSulfuras()
{}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testNormalBeforeSellDate() { ... }
    public function testNormalOnSellDate() { ... }
    public function testNormalAfterSellDate() { ... }
    public function testNormalOfZeroQuality() { ... }
    public function testBrieBeforeSellDate() { ... }
    public function testBrieBeforeSellDateWithMaxQuality() { ... }
    public function testBrieOnSellDate() { ... }
    public function testBrieOnSellDateNearMaxQuality() { ... }
    public function testBrieOnSellDateWithMaxQuality() { ... }
    public function testBrieAfterSellDate() { ... }
    public function testBrieAfterSellDateWithMaxQuality() { ... }
    public function testSulfurasBeforeSellDate() { ... }
    public function testSulfurasOnSellDate() { ... }
    public function testSulfurasAfterSellDate() { ... }
    public function testBackstageLongBeforeSellDate() { ... }
    public function testBackstageMediumCloseToSellDateUpperBound() { ... }
    public function testBackstageMediumCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageMediumCloseToSellDateLowerBound() { ... }
    public function testBackstageMediumCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateUpperBound() { ... }
    public function testBackstageVeryCloseToSellDateUpperBoundAtMaxQuality() { ... }
    public function testBackstageVeryCloseToSellDateLowerBound() { ... }
    public function testBackstageVeryCloseToSellDateLowerBoundAtMaxQuality() { ... }
    public function testBackstageOnSellDate() { ... }
    public function testBackstageAfterSellDate() { ... }
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
        case 'Backstage passes to a TAFKAL80ETC concert':
            $this->tickBackstage();
            return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {

private function tickBackstage()
{
}
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
        case 'Backstage passes to a TAFKAL80ETC concert':
            $this->tickBackstage();
            return;
    }

    if ($this->name != 'Aged Brie' && $this->name != 'Backstage passes to a TAFKAL80ETC concert') {

private function tickBackstage()
{
    $this->sellIn -= 1;

    if ($this->quality >= 50) {
        return;
    }
    if ($this->sellIn < 0) {
        $this->quality = 0;
        return;
    }

    $this->quality += 1;
    if ($this->sellIn < 10) {
        $this->quality += 1;
    }
    if ($this->sellIn < 5) {
        $this->quality += 1;
    }
}
```



```
$ phpunit  
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.
```

```
.....SSSS
```

```
29 / 29 (100%)
```

```
Time: 369 ms, Memory: 12.00MB
```

```
OK, but incomplete, skipped, or risky tests!  
Tests: 29, Assertions: 50, Skipped: 4.
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
        case 'Backstage passes to a TAFKAL80ETC concert':
            $this->tickBackstage();
            return;
    }

    public function tickNormal() {...}

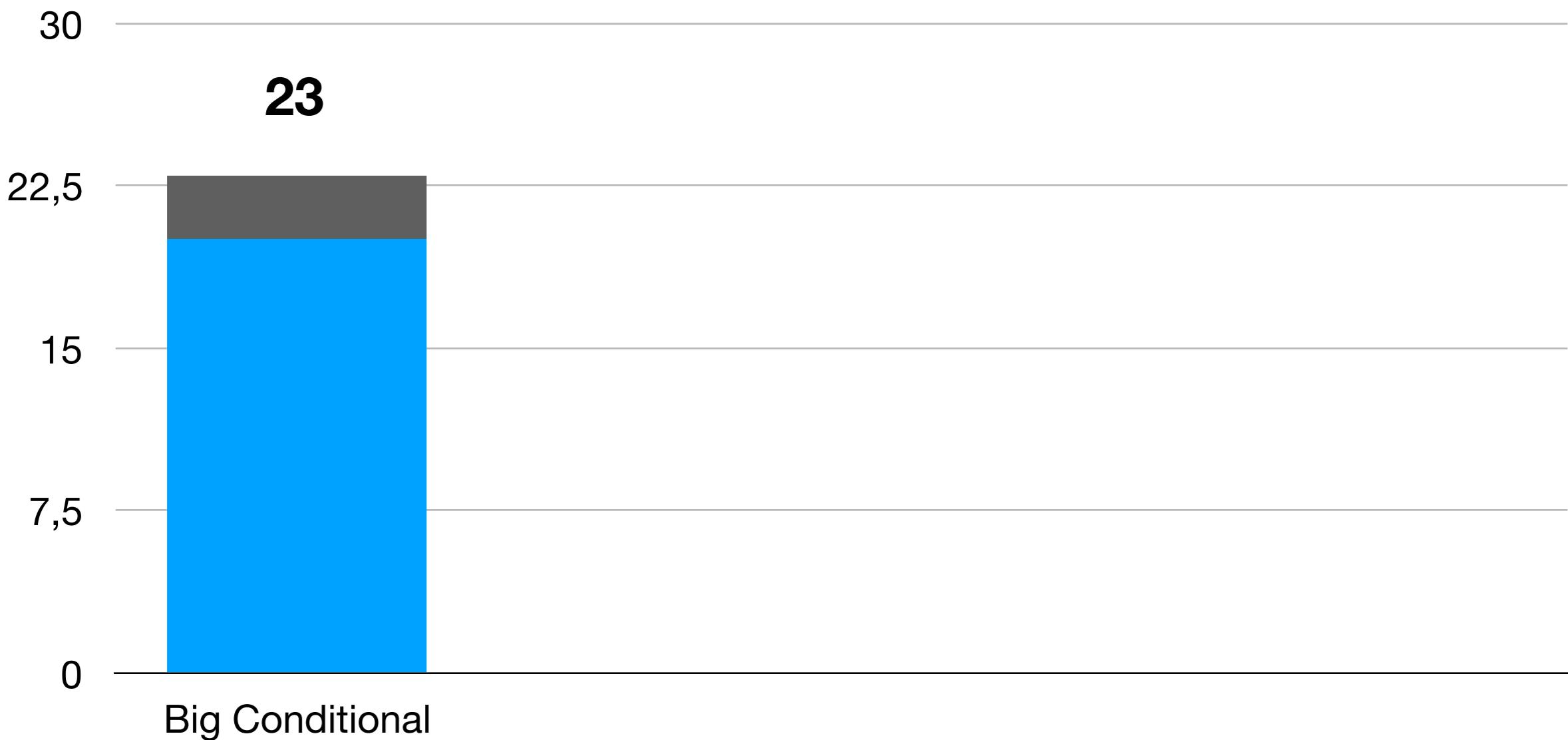
    public function tickAgedBrie() {...}

    public function tickSulfuras() {...}

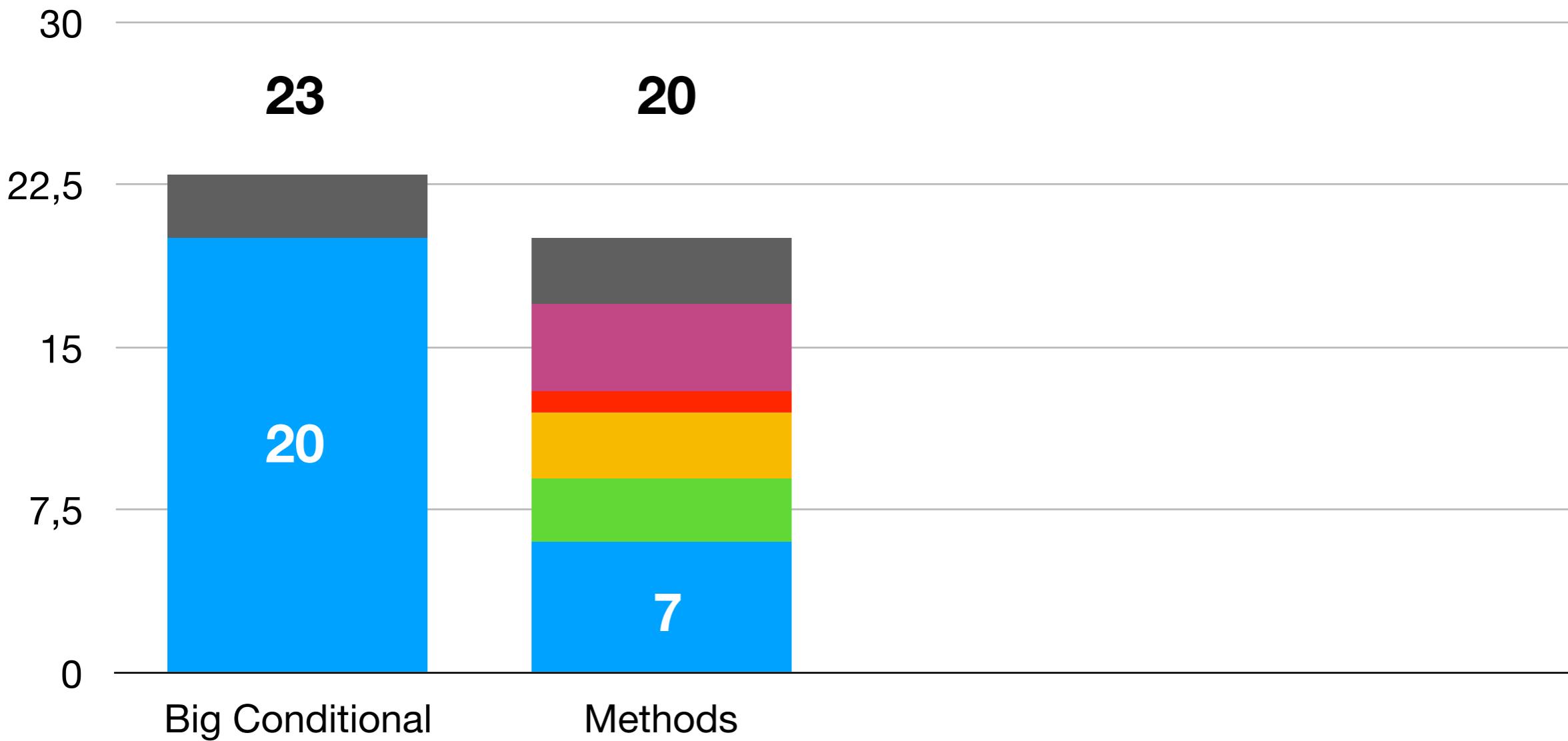
    public function tickBackstage() {...}
}
```



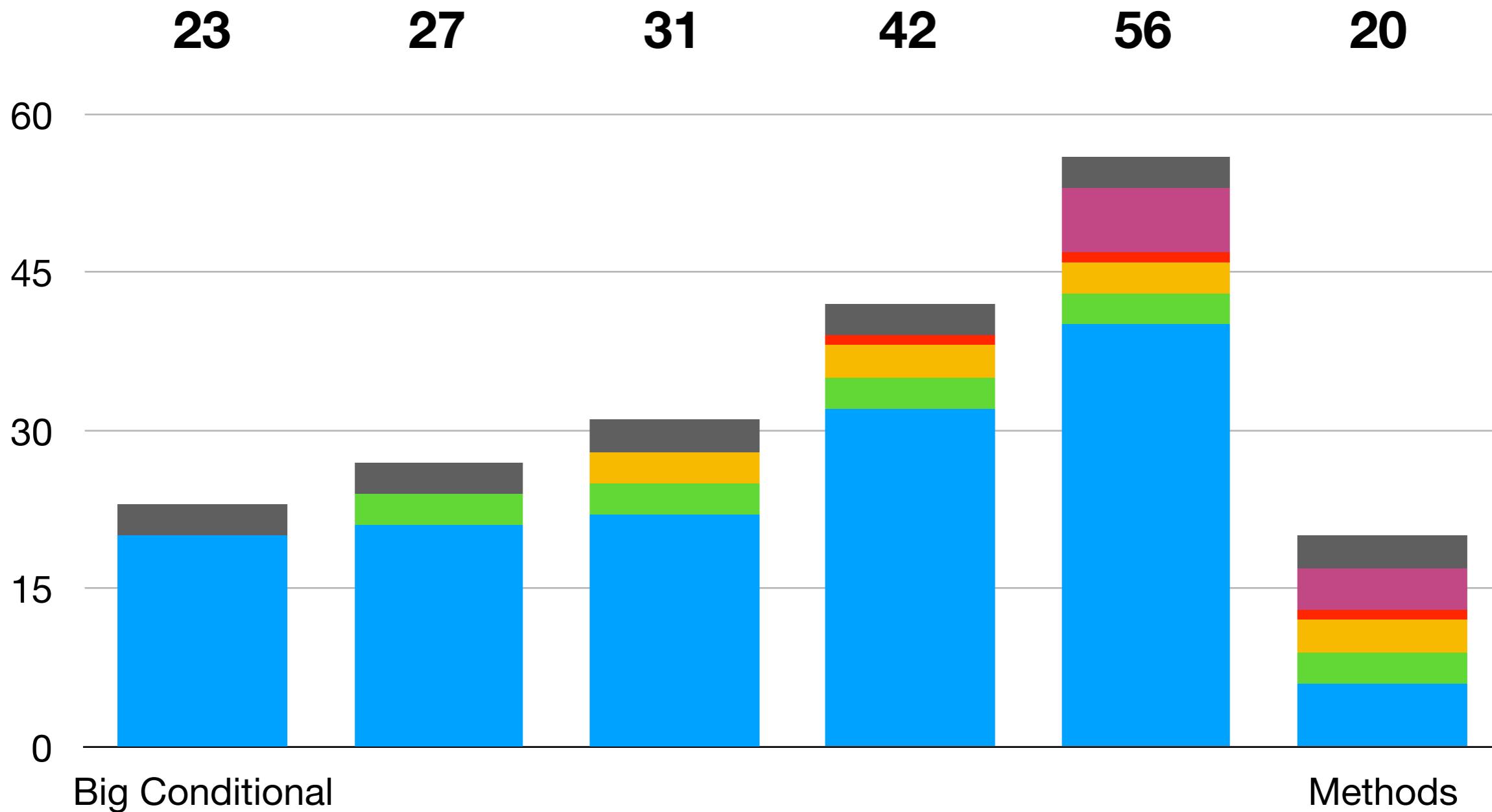
C.R.A.P



C.R.A.P



C.R.A.P



```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    # more tests above

    public function testConjuredBeforeSellDate()
    public function testConjuredOnSellDate()
    public function testConjuredAfterSellDate()
    public function testConjuredToZeroQuality()
}
```

'Conjured'

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
        case 'Backstage passes to a TAFKAL80ETC concert':
            $this->tickBackstage();
            return;
    }

    public function tickNormal() {...}

    public function tickAgedBrie() {...}

    public function tickSulfuras() {...}

    public function tickBackstage() {...}
}
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
        case 'Backstage passes to a TAFKAL80ETC concert':
            $this->tickBackstage();
            return;
    }

    public function tickNormal() {...}

    public function tickAgedBrie() {...}

    public function tickSulfuras() {...}

    public function tickBackstage() {...}

    public function tickConjured() {...}
}
```

???

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
        case 'Backstage passes to a TAFKAL80ETC concert':
            $this->tickBackstage();
            return;
    }

    public function tickNormal() {...}

    public function tickAgedBrie() {...}

    public function tickSulfuras() {...}

    public function tickBackstage() {...}
}
```

```
public function tick() : void
{
    switch ($this->name) {
        case 'normal':
            $this->tickNormal();
            return;
        case 'Aged Brie':
            $this->tickAgedBrie();
            return;
        case 'Sulfuras, Hand of Ragnaros':
            $this->tickSulfuras();
            return;
        case 'Backstage passes to a TAFKAL80ETC concert':
            $this->tickBackstage();
            return;
    }

    public function tickNormal() {...}
    public function tickAgedBrie() {...}
    public function tickSulfuras() {...}
    public function tickBackstage() {...}
}
```



```
public function tickNormal() : void
{
    $this->sellIn -= 1;

    if ($this->quality == 0) {
        return;
    }

    $this->quality -= 1;
    if ($this->sellIn <= 0) {
        $this->quality -= 1;
    }
}
```

```
public function tick() : void
{
    $this->sellIn -= 1;

    if ($this->quality == 0) {
        return;
    }

    $this->quality -= 1;
    if ($this->sellIn <= 0) {
        $this->quality -= 1;
    }
}
```

```
class Normal
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }
}
```

```
class Normal
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }
}
```

```
class Normal
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }

    public function getQuality() : int
    public function getDaysRemaining() : int
}
```

```
class GildedRose
{
    private function tickNormal() : void
    {
        $this->rose = new Rose\Normal($this->quality, $this->sellIn);
        $this->rose->tick();
    }
    # ...
}
```

```
class Normal
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }

    public function getQuality() : int
    public function getDaysRemaining() : int
}
```

```
class GildedRose
{
    private $name;
    private $quality;
    private $sellIn;

    public function getQuality() : int
    {
        return $this->quality;
    }

    public function getDaysRemaining() : int
    {
        return $this->sellIn;
    }

    # ...
}
```

```
class GildedRose
{
    private $name;
    private $quality;
    private $sellIn;
    private $rose;

    public function getQuality() : int
    {
        return $this->quality;
    }

    public function getDaysRemaining() : int
    {
        return $this->sellIn;
    }

    # ...
}
```

```
class GildedRose
{
    private $name;
    private $quality;
    private $sellIn;
    private $rose;

    public function getQuality() : int
    {
        return $this->rose ? $this->rose->getQuality() : $this->quality;
    }

    public function getDaysRemaining() : int
    {
        return $this->sellIn;
    }

    # ...
}
```

```
class GildedRose
{
    private $name;
    private $quality;
    private $sellIn;
    private $rose;

    public function getQuality() : int
    {
        return $this->rose ? $this->rose->getQuality() : $this->quality;
    }

    public function getDaysRemaining() : int
    {
        return $this->rose ? $this->rose->getDaysRemaining() : $this->sellIn;
    }

    # ...
}
```

```
public function tickAgedBrie() : void
{
    $this->sellIn -= 1;

    if ($this->quality >= 50) {
        return;
    }

    $this->quality += 1;
    if ($this->sellIn <= 0 && $this->quality < 50) {
        $this->quality += 1;
    }
}
```

```
public function tick() : void
{
    $this->sellIn -= 1;

    if ($this->quality >= 50) {
        return;
    }

    $this->quality += 1;
    if ($this->sellIn <= 0 && $this->quality < 50) {
        $this->quality += 1;
    }
}
```

```
class AgedBrie
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }
}
```

```
class AgedBrie
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }
}
```

```
class AgedBrie
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class GildedRose
{
    private function tickAgedBrie()
    {
        $this->rose = new Rose\AgedBrie($this->quality, $this->sellIn);
        $this->rose->tick();
    }
}
```

```
class AgedBrie
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
public function tickSulfuras() : void
{
}
```

```
public function tick() : void
{
}
```

```
class Sulfuras
{
    public function tick() : void
    {
    }
}
```

```
class Sulfuras
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
    }
}
```

```
class GildedRose
{
    private function tickSulfuras()
    {
        $this->rose = new Rose\Sulfuras($this->quality, $this->sellIn);
        $this->rose->tick();
    }
}
```

```
class Sulfuras
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
public function tickBackstage() : void
{
    $this->sellIn -= 1;
    if ($this->quality >= 50) {
        return;
    }
    if ($this->sellIn < 0) {
        $this->quality = 0;
        return;
    }
    $this->quality += 1;
    if ($this->sellIn < 10) {
        $this->quality += 1;
    }
    if ($this->sellIn < 5) {
        $this->quality += 1;
    }
}
```

```
public function tick() : void
{
    $this->sellIn -= 1;
    if ($this->quality >= 50) {
        return;
    }
    if ($this->sellIn < 0) {
        $this->quality = 0;
        return;
    }
    $this->quality += 1;
    if ($this->sellIn < 10) {
        $this->quality += 1;
    }
    if ($this->sellIn < 5) {
        $this->quality += 1;
    }
}
```

```
class Backstage
{
    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
}
```

```
class Backstage
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn) {...}

    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
}
```

```
class Backstage
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn) {...}

    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class GildedRose
{
    private function tickBackstage()
    {
        $this->rose = new Rose\Backstage($this->quality, $this->sellIn);
        $this->rose->tick();
    }
}
```

```
class Backstage
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn) {...}

    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class GildedRose
{
    private $name;
    private $quality;
    private $sellIn;
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->name      = $name;
        $this->quality   = $quality;
        $this->sellIn    = $sellIn;
    }

    # ...
}
```

```
class GildedRose
{
    private $name;
    private $quality;

    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->name      = $name;
        $this->quality   = $quality;

    }

    # ...
}
```

```
class GildedRose
{
    private $name;

    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->name      = $name;

    }

    # ...
}
```

```
class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
    }

    # ...
}
```

```
class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        switch ($name) {
            case 'normal':
                $this->rose = new Rose\Normal($quality, $sellIn);
                return;
            case 'Aged Brie':
                $this->rose = new Rose\AgedBrie($quality, $sellIn);
                return;
            case 'Sulfuras, Hand of Ragnaros':
                $this->rose = new Rose\Sulfuras($quality, $sellIn);
                return;
            case 'Backstage passes to a TAFKAL80ETC concert':
                $this->rose = new Rose\Backstage($quality, $sellIn);
                return;
        }
    }
}

# ...
```

```
class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        switch ($name) {
            case 'normal':
                $this->rose = new Rose\Normal($quality, $sellIn);
                return;
            case 'Aged Brie':
                $this->rose = new Rose\AgedBrie($quality, $sellIn);
                return;
            case 'Sulfuras, Hand of Ragnaros':
                $this->rose = new Rose\Sulfuras($quality, $sellIn);
                return;
            case 'Backstage passes to a TAFKAL80ETC concert':
                $this->rose = new Rose\Backstage($quality, $sellIn);
                return;
        }
    }
}

# ...
```

```
class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        switch ($name) {
            case 'normal':
                $this->rose = new Rose\Normal($quality, $sellIn);
                return;
            case 'Aged Brie':
                $this->rose = new Rose\AgedBrie($quality, $sellIn);
                return;
            case 'Sulfuras, Hand of Ragnaros':
                $this->rose = new Rose\Sulfuras($quality, $sellIn);
                return;
            case 'Backstage passes to a TAFKAL80ETC concert':
                $this->rose = new Rose\Backstage($quality, $sellIn);
                return;
        }
    }

    public function getQuality() : int
    {
        return $this->rose ? $this->rose->getQuality() : $this->quality;
    }

    public function getDaysRemaining() : int
    {
        return $this->rose ? $this->rose->getDaysRemaining() : $this->sellIn;
    }

    # ...
}
```

```
class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        switch ($name) {
            case 'normal':
                $this->rose = new Rose\Normal($quality, $sellIn);
                return;
            case 'Aged Brie':
                $this->rose = new Rose\AgedBrie($quality, $sellIn);
                return;
            case 'Sulfuras, Hand of Ragnaros':
                $this->rose = new Rose\Sulfuras($quality, $sellIn);
                return;
            case 'Backstage passes to a TAFKAL80ETC concert':
                $this->rose = new Rose\Backstage($quality, $sellIn);
                return;
        }
    }

    public function getQuality() : int
    {
        return $this->rose->getQuality();
    }

    public function getDaysRemaining() : int
    {
        return $this->rose ? $this->rose->getDaysRemaining() : $this->sellIn;
    }

    # ...
}
```

```
class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        switch ($name) {
            case 'normal':
                $this->rose = new Rose\Normal($quality, $sellIn);
                return;
            case 'Aged Brie':
                $this->rose = new Rose\AgedBrie($quality, $sellIn);
                return;
            case 'Sulfuras, Hand of Ragnaros':
                $this->rose = new Rose\Sulfuras($quality, $sellIn);
                return;
            case 'Backstage passes to a TAFKAL80ETC concert':
                $this->rose = new Rose\Backstage($quality, $sellIn);
                return;
        }
    }

    public function getQuality() : int
    {
        return $this->rose->getQuality();
    }

    public function getDaysRemaining() : int
    {
        return $this->rose->getDaysRemaining();
    }

    # ...
}
```

```
class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        switch ($name) {
            case 'normal':
                $this->rose = new Rose\Normal($quality, $sellIn);
                return;
            case 'Aged Brie':
                $this->rose = new Rose\AgedBrie($quality, $sellIn);
                return;
            case 'Sulfuras, Hand of Ragnaros':
                $this->rose = new Rose\Sulfuras($quality, $sellIn);
                return;
            case 'Backstage passes to a TAFKAL80ETC concert':
                $this->rose = new Rose\Backstage($quality, $sellIn);
                return;
        }
    }

    public function getQuality() : int
    {
        return $this->rose->getQuality();
    }

    public function getDaysRemaining() : int
    {
        return $this->rose->getDaysRemaining();
    }

    public function tick() : void
    {
        $this->rose->tick();
    }

    # ...
}
```

```
class GildedRose
```

```
{
```

```
# ...
```

```
switch ($name) {
    case 'normal':
        $this->rose = new Rose\Normal($quality, $sellIn);
        return;
    case 'Aged Brie':
        $this->rose = new Rose\AgedBrie($quality, $sellIn);
        return;
    case 'Sulfuras, Hand of Ragnaros':
        $this->rose = new Rose\Sulfuras($quality, $sellIn);
        return;
    case 'Backstage passes to a TAFKAL80ETC concert':
        $this->rose = new Rose\Backstage($quality, $sellIn);
        return;
```

```
# ...
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class
    ];

    public static function createRose(string $name, int $quality, int $sellIn)
    {
        return new self::$roses[$name]($quality, $sellIn);
    }
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class
    ];

    public static function createRose(string $name, int $quality, int $sellIn)
    {
        return new self::$roses[$name]($quality, $sellIn);
    }
}

class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        switch ($name) {
            case 'normal':
                $this->rose = new Rose\Normal($quality, $sellIn);
                return;
            case 'Aged Brie':
                $this->rose = new Rose\AgedBrie($quality, $sellIn);
                return;
            case 'Sulfuras, Hand of Ragnaros':
                $this->rose = new Rose\Sulfuras($quality, $sellIn);
                return;
            case 'Backstage passes to a TAFKAL80ETC concert':
                $this->rose = new Rose\Backstage($quality, $sellIn);
                return;
        }
    }

    # ...
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class
    ];

    public static function createRose(string $name, int $quality, int $sellIn)
    {
        return new self::$roses[$name]($quality, $sellIn);
    }
}

class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->rose = Rose\Factory::createRose($name, $quality, $sellIn);
    }

    # ...
}
```

```
interface Rose
{
    public function getQuality();

    public function getDaysRemaining();

    public function tick();
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class
    ];

    public static function createRose(string $name, int $quality, int $sellIn)
    {
        return new self::$roses[$name]($quality, $sellIn);
    }
}

class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->rose = Rose\Factory::createRose($name, $quality, $sellIn);
    }

    # ...
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class
    ];

    public static function createRose(string $name, int $quality, int $sellIn) : RoseInterface
    {
        return new self::$roses[$name]($quality, $sellIn);
    }
}

class GildedRose
{
    private $rose;

    public function __construct(string $name, int $quality, int $sellIn)
    {
        $this->rose = Rose\Factory::createRose($name, $quality, $sellIn);
    }

    # ...
}
```

```
class Normal
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }

    public function getQuality() : int
    public function getDaysRemaining() : int
}
```

```
class AgedBrie
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class Sulfuras
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class Backstage
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn) {...}

    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
abstract class Rose implements RoseInterface  
{  
}
```

```
abstract class Rose implements RoseInterface
{
    protected $quality;
    protected $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }
}
```

```
abstract class Rose implements RoseInterface
{
    protected $quality;
    protected $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function getQuality() : int
    {
        return $this->quality;
    }
}
```

```
abstract class Rose implements RoseInterface
```

```
{
```

```
    protected $quality;  
    protected $sellIn;
```

```
    public function __construct(int $quality, int $sellIn)
```

```
{
```

```
        $this->quality = $quality;  
        $this->sellIn = $sellIn;
```

```
}
```

```
    public function getQuality() : int
```

```
{
```

```
        return $this->quality;
```

```
}
```

```
    public function getDaysRemaining() : int
```

```
{
```

```
        return $this->sellIn;
```

```
}
```

```
}
```

```
abstract class Rose implements RoseInterface
{
    protected $quality;
    protected $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function getQuality() : int
    {
        return $this->quality;
    }

    public function getDaysRemaining() : int
    {
        return $this->sellIn;
    }

    public abstract function tick();
}
```

```
class Normal
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }

    public function getQuality() : int
    public function getDaysRemaining() : int
}
```

```
class Normal extends Rose
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }

    public function getQuality() : int
    public function getDaysRemaining() : int
}
```

```
class Normal extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }

    public function getQuality() : int
    public function getDaysRemaining() : int
}
```

```
class Normal extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 1;
        if ($this->sellIn <= 0) {
            $this->quality -= 1;
        }
    }
}
```

```
class AgedBrie
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class AgedBrie extend Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }
    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class AgedBrie extend Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality >= 50) {
            return;
        }

        $this->quality += 1;
        if ($this->sellIn <= 0 && $this->quality < 50) {
            $this->quality += 1;
        }
    }
}
```

```
class Sulfuras
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class Sulfuras extends Rose
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn)
    {
        $this->quality = $quality;
        $this->sellIn = $sellIn;
    }

    public function tick() : void
    {
    }

    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class Sulfuras extends Rose
{
    public function tick() : void
    {
    }
}
```

```
class Backstage
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn) {...}

    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

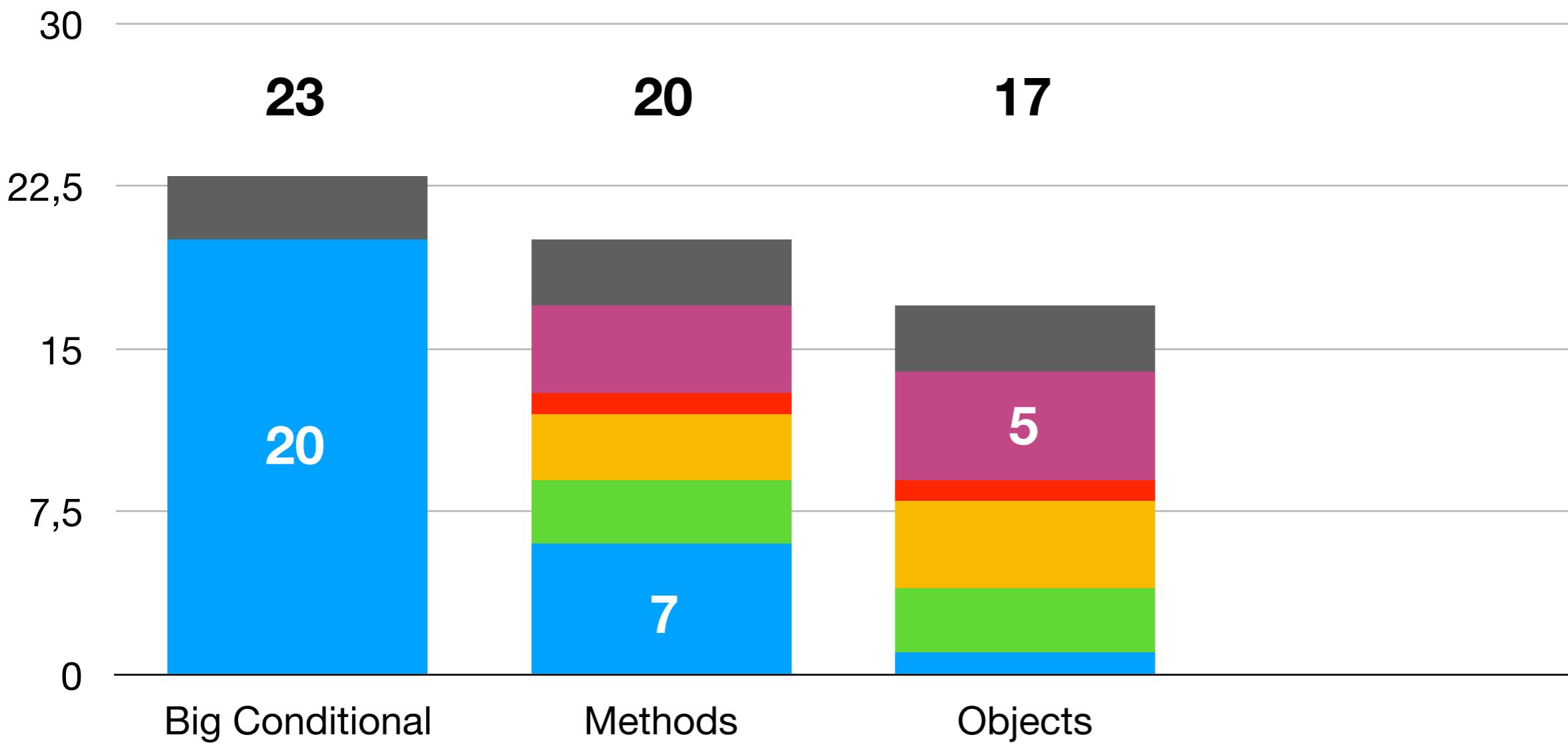
```
class Backstage extends Rose
{
    private $quality;
    private $sellIn;

    public function __construct(int $quality, int $sellIn) {...}

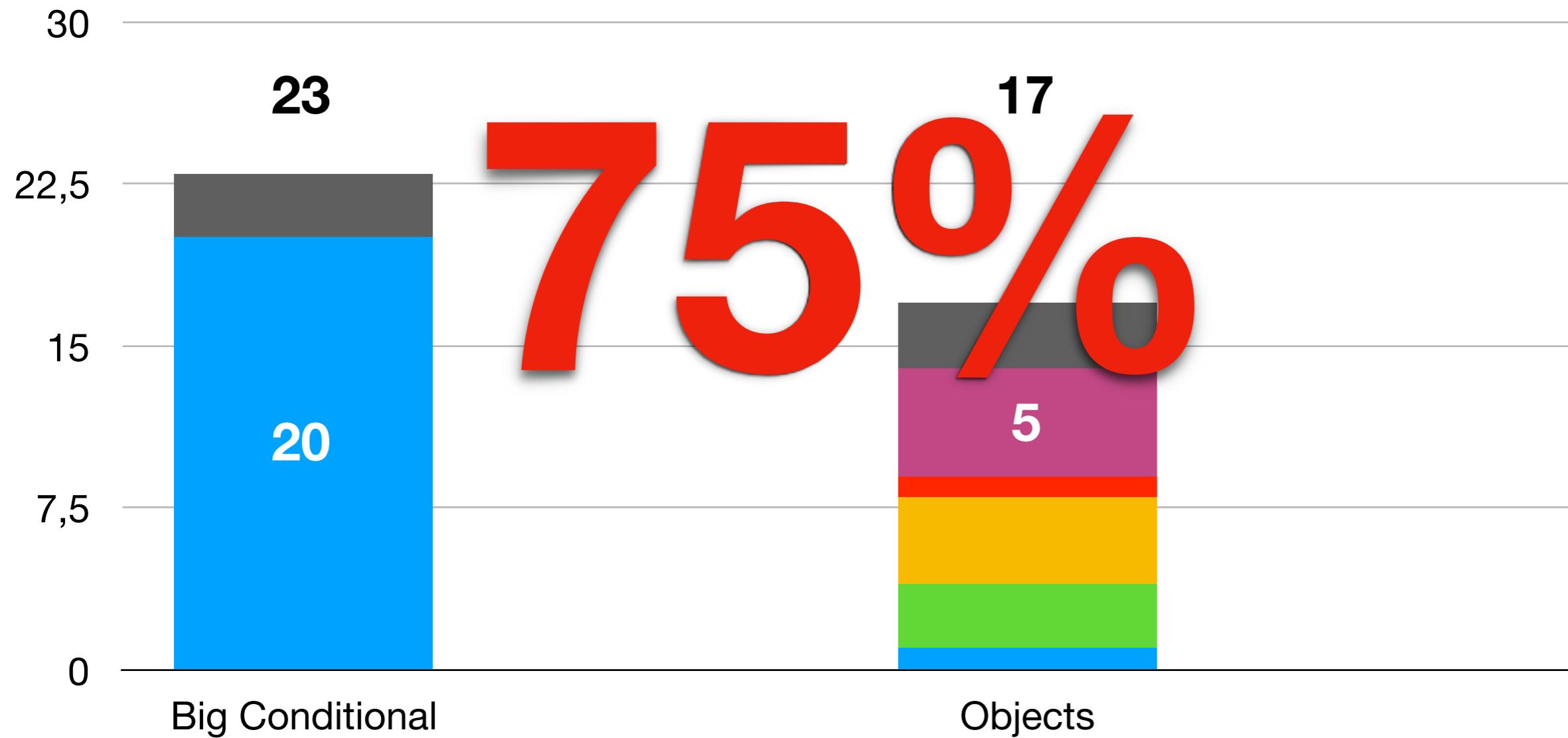
    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
    public function getQuality() : int {...}
    public function getDaysRemaining() : int {...}
}
```

```
class Backstage extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;
        if ($this->quality >= 50) {
            return;
        }
        if ($this->sellIn < 0) {
            $this->quality = 0;
            return;
        }
        $this->quality += 1;
        if ($this->sellIn < 10) {
            $this->quality += 1;
        }
        if ($this->sellIn < 5) {
            $this->quality += 1;
        }
    }
}
```

C.R.A.P



C.R.A.P



NOW

WHAT?

```
class Conjured extends Rose  
{  
}
```

```
class Conjured extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 2;
        if ($this->sellIn <= 0 && $this->quality > 0) {
            $this->quality -= 2;
        }
    }
}
```

```
class Conjured extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 2;
        if ($this->sellIn <= 0 && $this->quality > 0) {
            $this->quality -= 2;
        }
    }
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class
    ];
    # ...
}
```

```
class Conjured extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 2;
        if ($this->sellIn <= 0 && $this->quality > 0) {
            $this->quality -= 2;
        }
    }
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class,
        'Conjured'                               => Conjured::class
    ];
    # ...
}
```

```
class Conjured extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 2;
        if ($this->sellIn <= 0 && $this->quality > 0) {
            $this->quality -= 2;
        }
    }
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class,
        'Conjured'                               => Conjured::class
    ];
    # ...
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testConjuredBeforeSellDate()
    {
        $this->markTestSkipped();

        $rose = new GildedRose('Conjured', 10, 10);
        $rose->tick();

        # ...
    }
}
```

```
class Conjured extends Rose
{
    public function tick() : void
    {
        $this->sellIn -= 1;

        if ($this->quality == 0) {
            return;
        }

        $this->quality -= 2;
        if ($this->sellIn <= 0 && $this->quality > 0) {
            $this->quality -= 2;
        }
    }
}
```

```
abstract class Factory
{
    private static $roses = [
        'normal'                                => Normal::class,
        'Aged Brie'                             => AgedBrie::class,
        'Sulfuras, Hand of Ragnaros'           => Sulfuras::class,
        'Backstage passes to a TAFKAL80ETC concert' => Backstage::class,
        'Conjured'                               => Conjured::class
    ];
    # ...
}
```

```
class GildedRoseTest extends \PHPUnit\Framework\TestCase
{
    public function testConjuredBeforeSellDate()
    {
        $rose = new GildedRose('Conjured', 10, 10);
        $rose->tick();

        # ...
    }
}
```



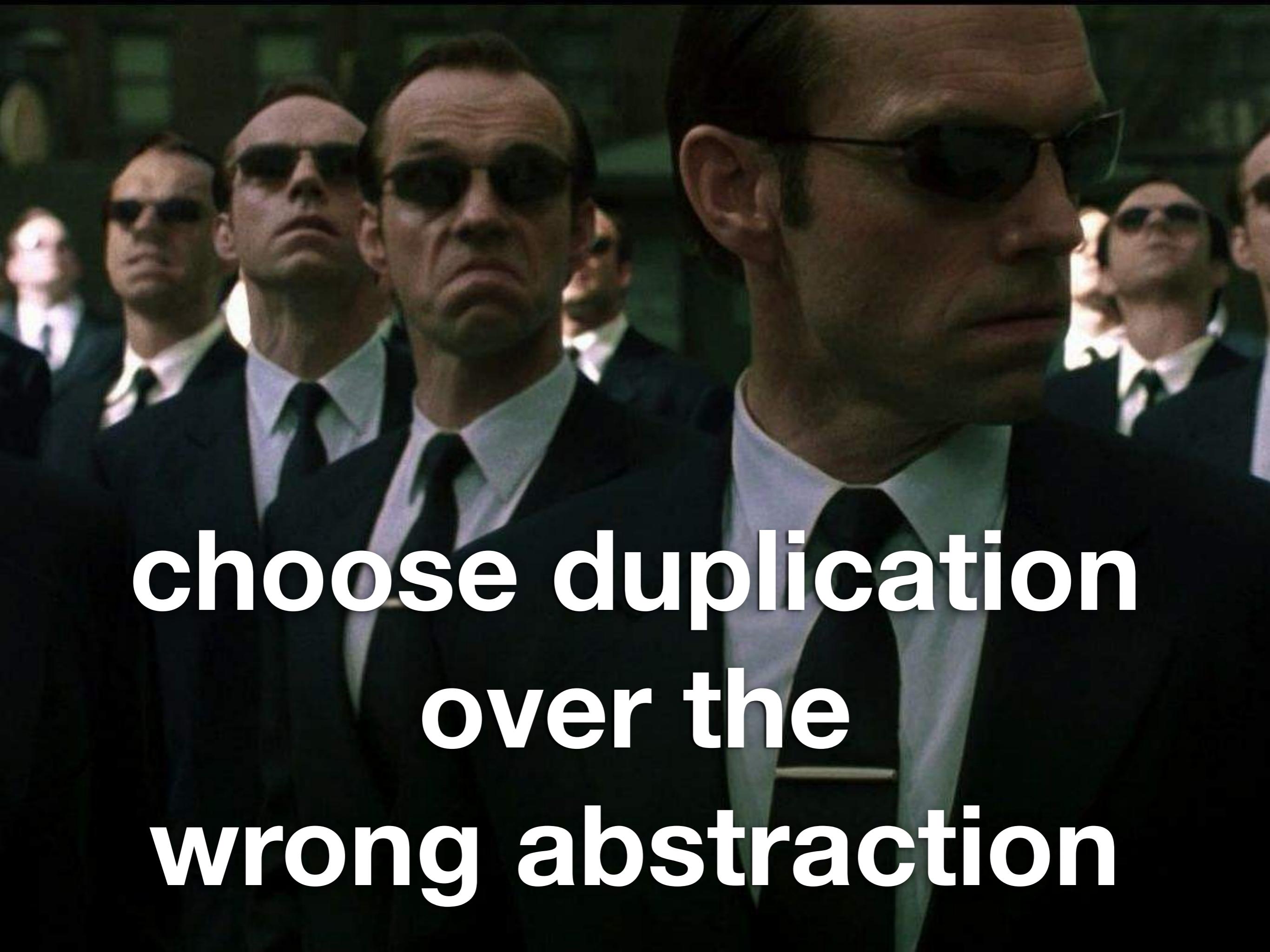
```
$ phpunit  
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.
```

```
.....
```

```
29 / 29 (100%)
```

```
Time: 393 ms, Memory: 12.00MB
```

```
OK (29 tests, 58 assertions)
```



choose duplication
over the
wrong abstraction

S.O.L.I.D.





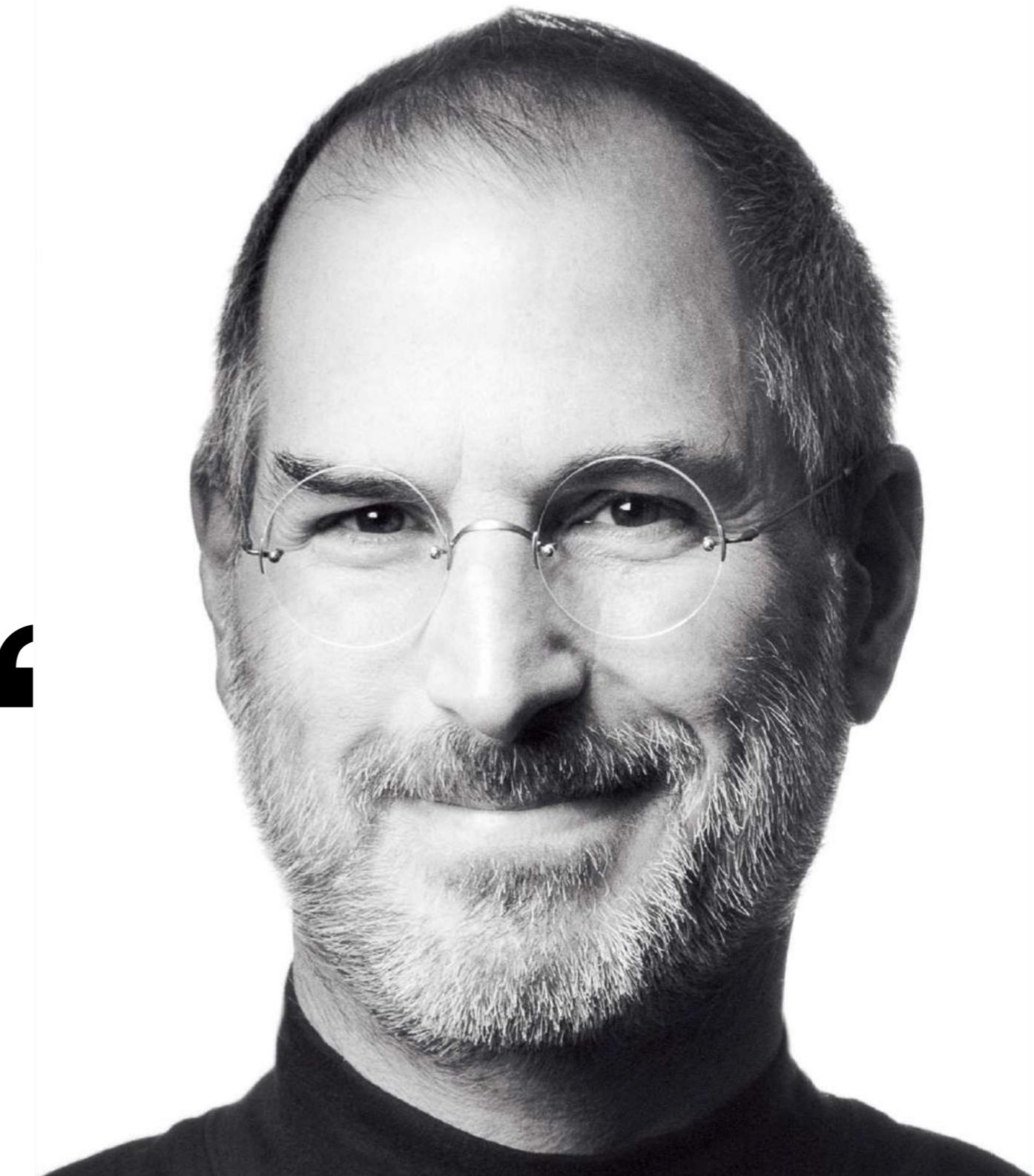
**BUILD
SMALL-
THINGS**

~~Fcar~~

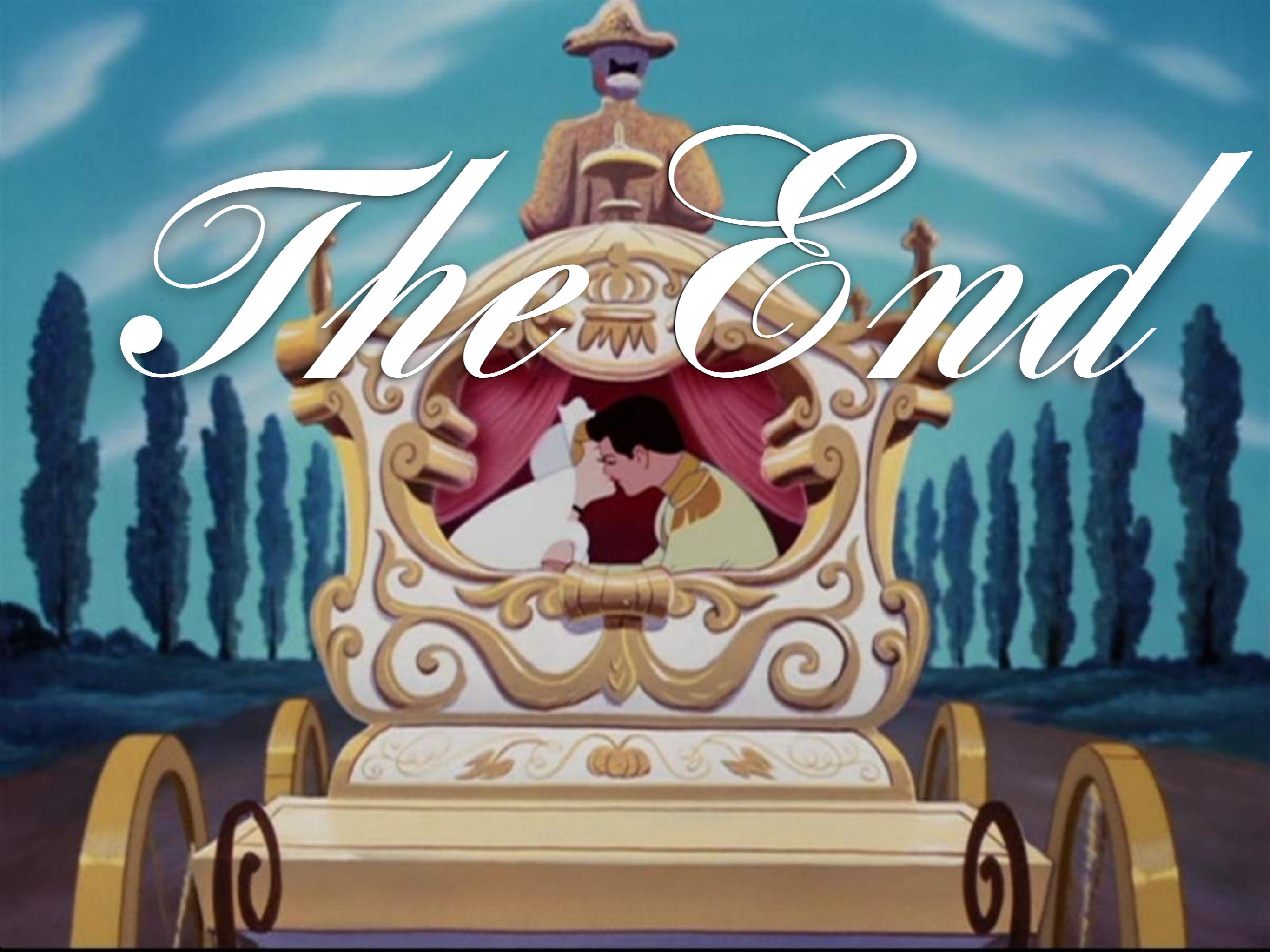
intermediate complexity

**“Simple can be harder
than complex. You have
to work hard to get
your thinking clean
to make it simple.
But it's worth it in the end
because once you get there,
you can move mountains.”**

— Steve Jobs —



The End





Sebastian Feldmann

phpbu

<https://phpbu.de>



CHECK24

Captain  *Hook*



@movetodevnull



sebastianfeldmann