Entwurfsmuster - Design Patterns

PROXY

JAN ANDRÉ SCHLÖSSER

DUPLEXMEDIA®

{ it4need
Making IT solutions affordable for everyone.

it4need

jan.schloesser

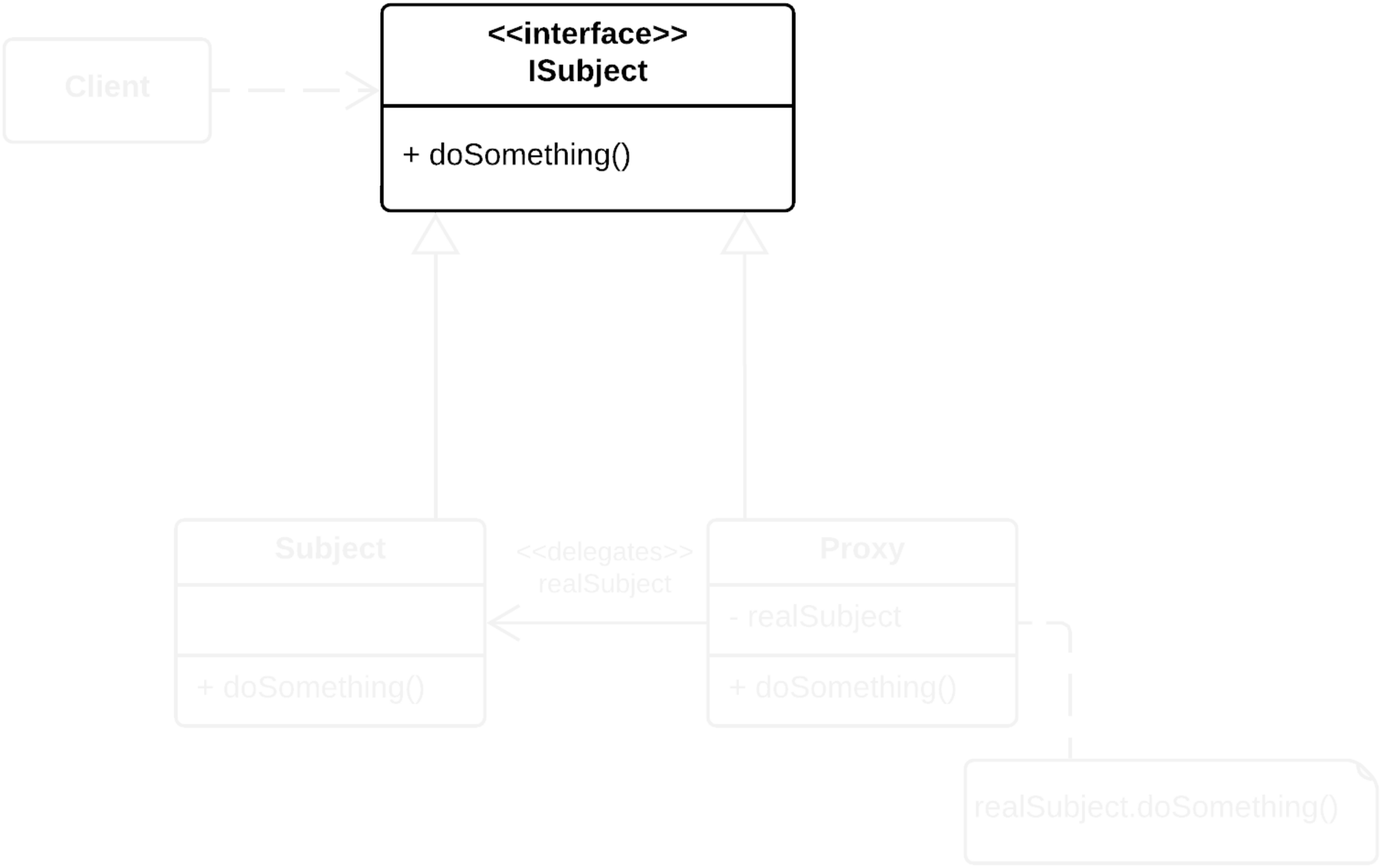
jan-schloesser.de

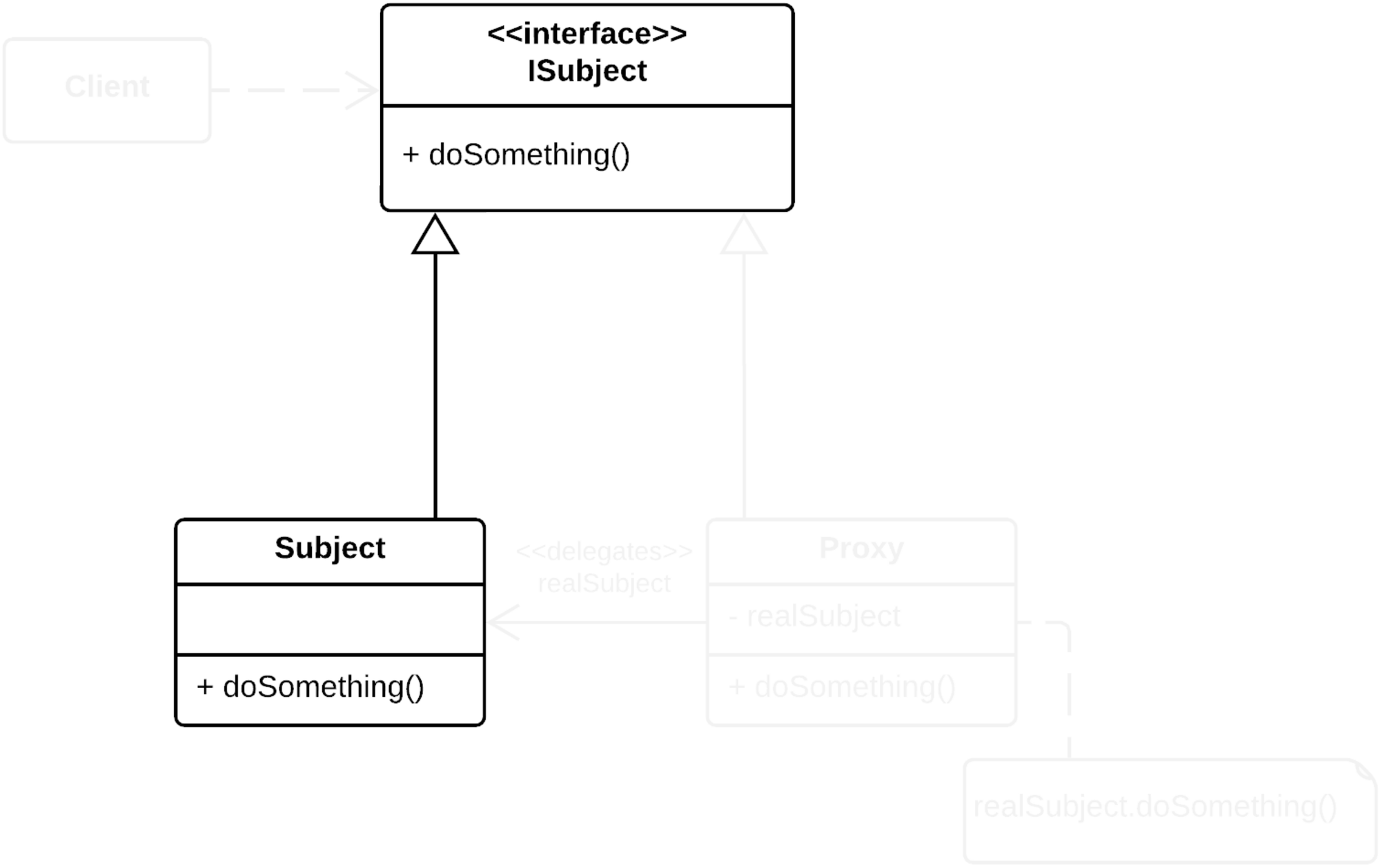
„EIN PROXY KONTROLLIERT DEN ZUGRIFF AUF EIN OBJEKT MIT HILFE EINES VORGELAGERTEN STELLVERTRETERS.“

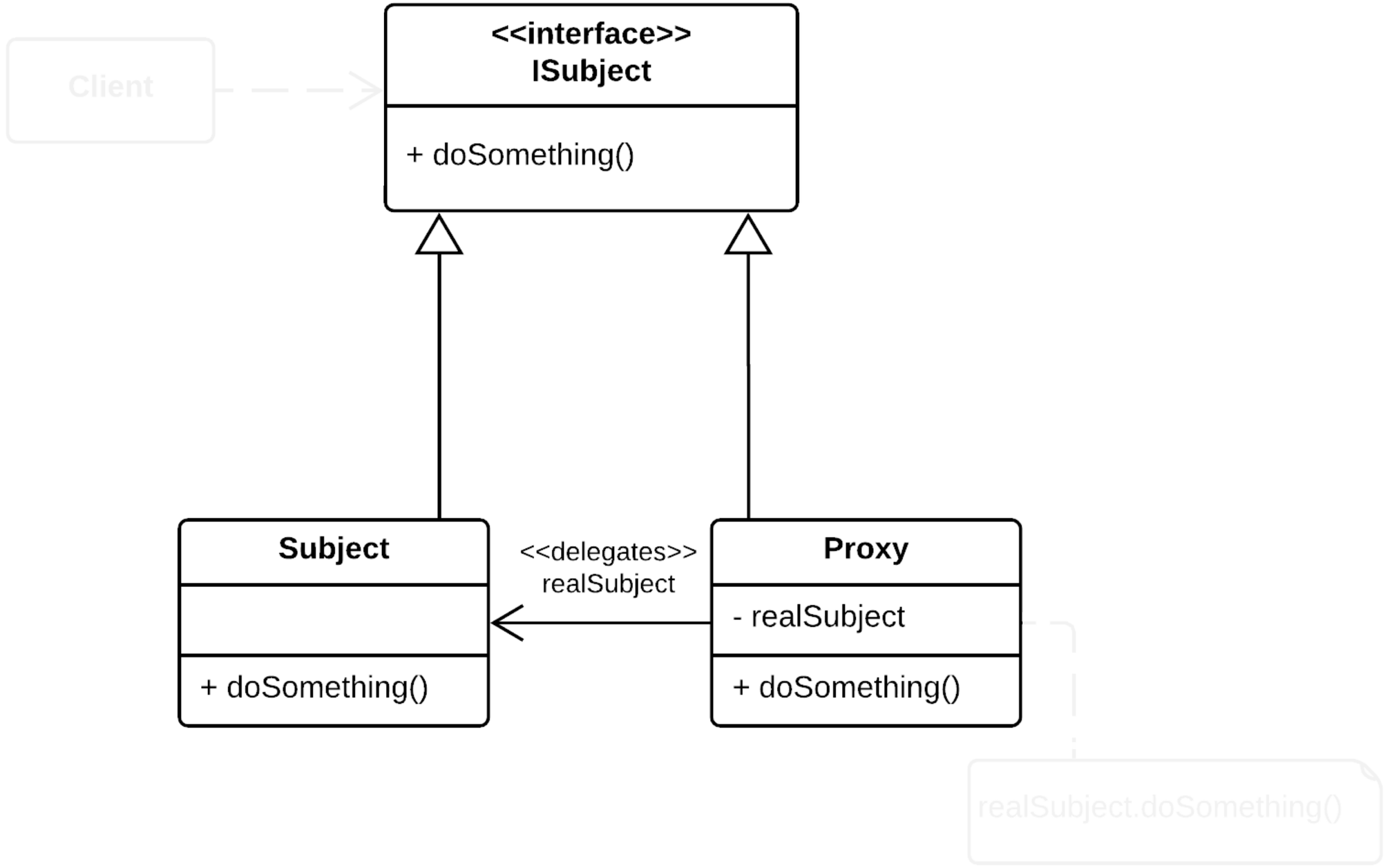
Entwurfmuster von Kopf bis Fuß

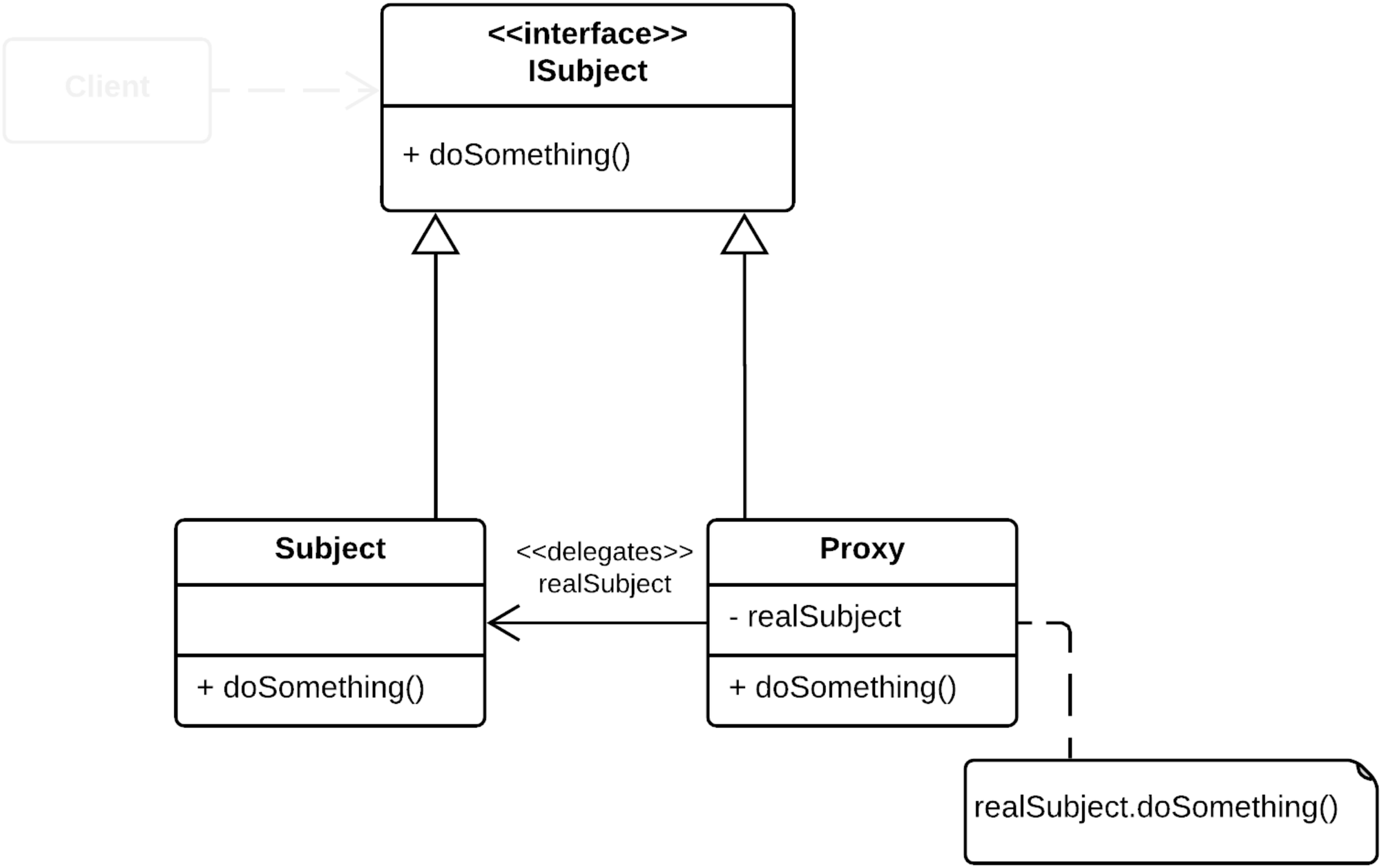
Proxy ist ein **GoF-Muster** im Bereich der **Strukturmuster**.

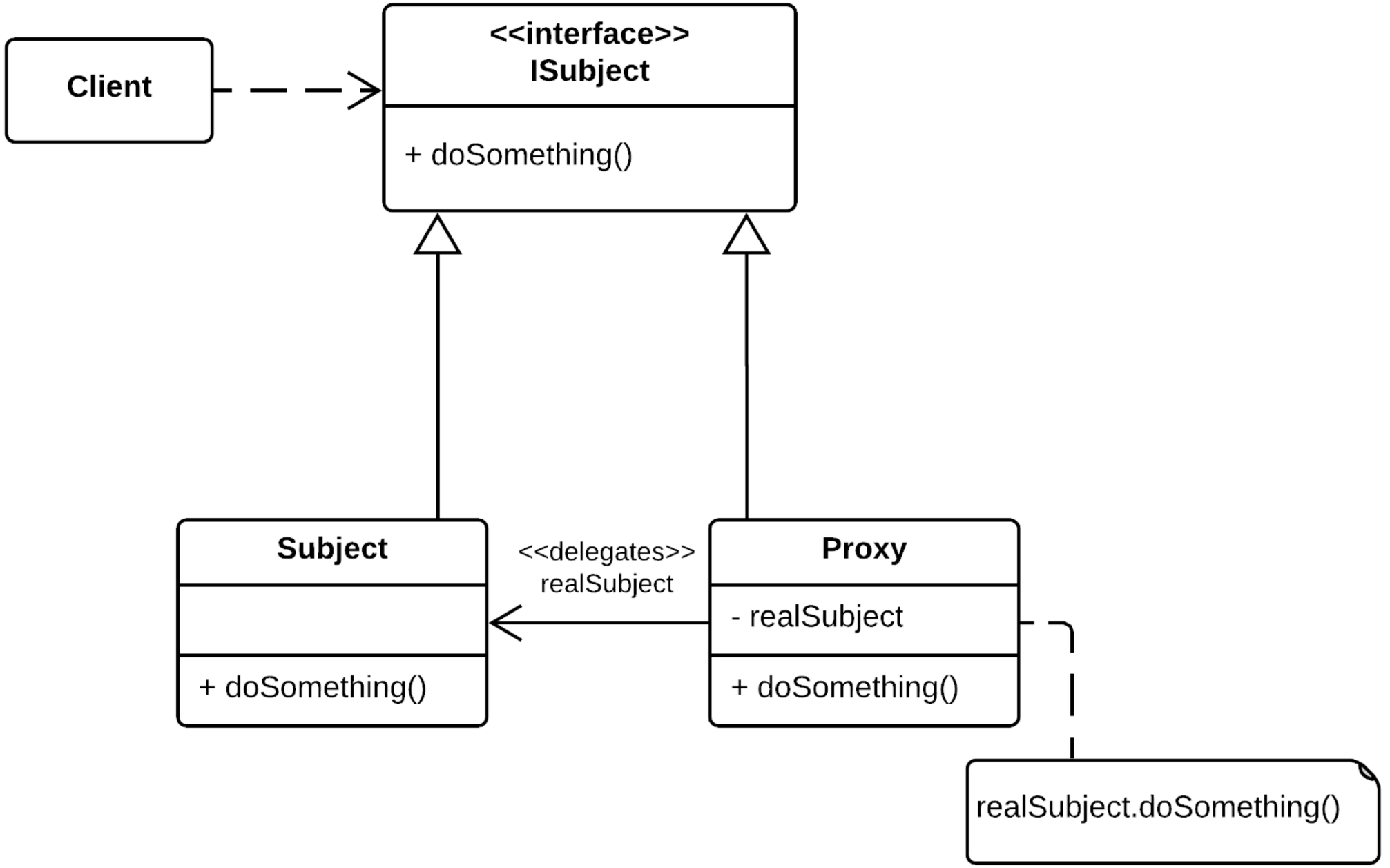
Substituiere Objekt oder stelle **Platzhalter**
zur **Zugriffskontrolle eines anderes Objekt** zur Verfügung.


Client
<<interface>>
ISubject
+ doSomething()
Subject
+ doSomething()
<<delegates>>
realSubject
Proxy
- realSubject
+ doSomething()
realSubject.doSomething()


Client
<<interface>>
ISubject
+ doSomething()
Subject
+ doSomething()
<<delegates>>
realSubject
Proxy
- realSubject
+ doSomething()
realSubject.doSomething()


Client
<<interface>>
ISubject
+ doSomething()
Subject
+ doSomething()
<<delegates>>
realSubject
Proxy
- realSubject
+ doSomething()
realSubject.doSomething()

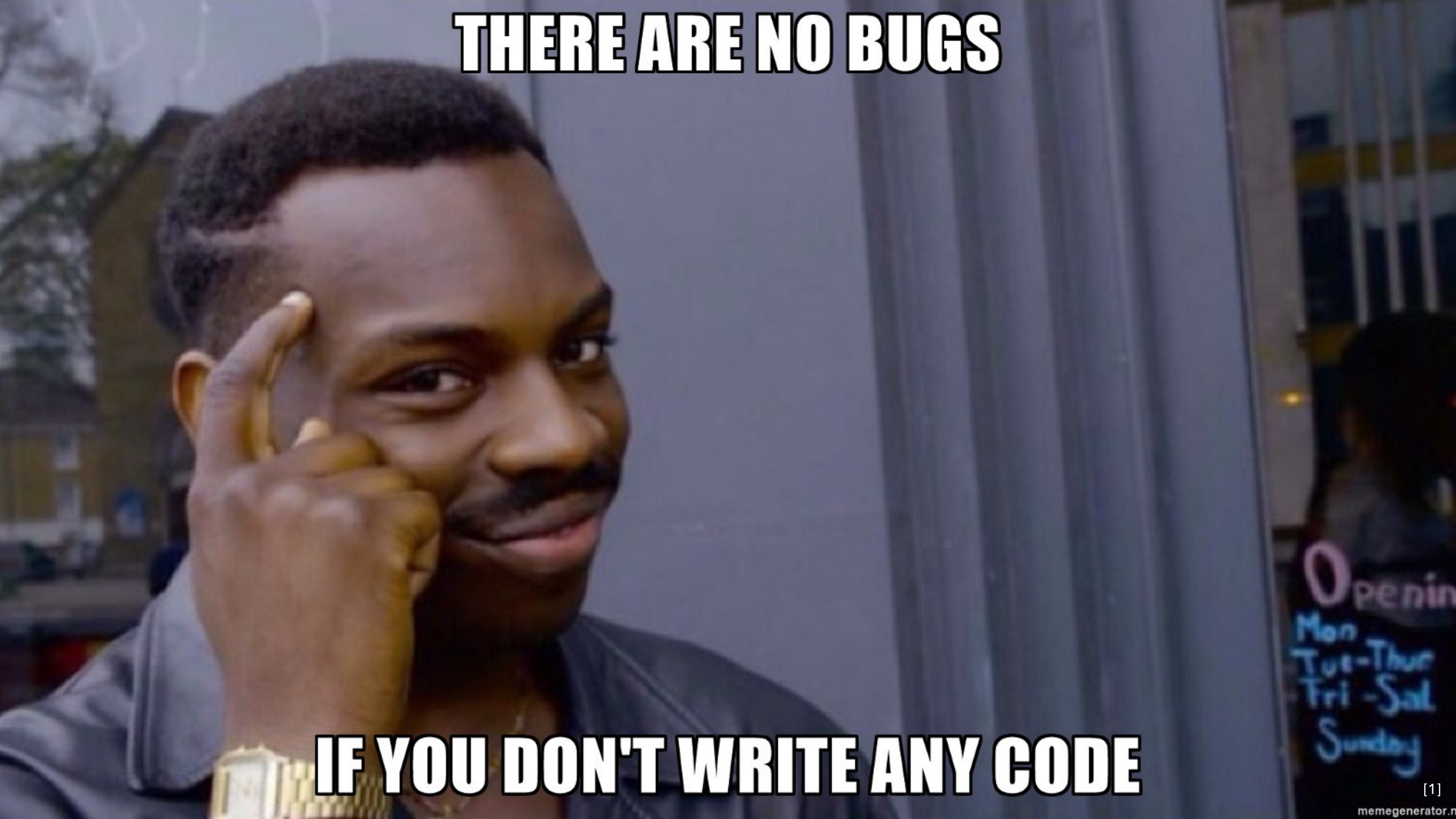

Client
<<interface>>
ISubject
+ doSomething()
Subject
+ doSomething()
<<delegates>>
realSubject
Proxy
- realSubject
+ doSomething()
realSubject.doSomething()


Client
<<interface>>
ISubject
+ doSomething()
Subject
+ doSomething()
<<delegates>>
realSubject
Proxy
- realSubject
+ doSomething()
realSubject.doSomething()

Das **Proxy** ersetzt also das **Subjekt** aufgrund des **LSP**.

ARTEN DES PROXY-MUSTERS

- **Remote Proxy**
  lokaler Repräsentant für ein Objekt in einem anderen Adressraum
- **Virtual Proxy**
  verzögert das Laden eines großen Objekts auf notwendigen Zeitpunkt (Lazy Loading)
- **Protection Proxy**
  limitiert den Zugriff auf das Subjekt durch bestimmte Regeln (z.B. ACL, Limits)
- **Cache Proxy**
  speichert Informationen und gibt diese bei erneuten Aufruf ohne Neuberechnung zurück



[1]

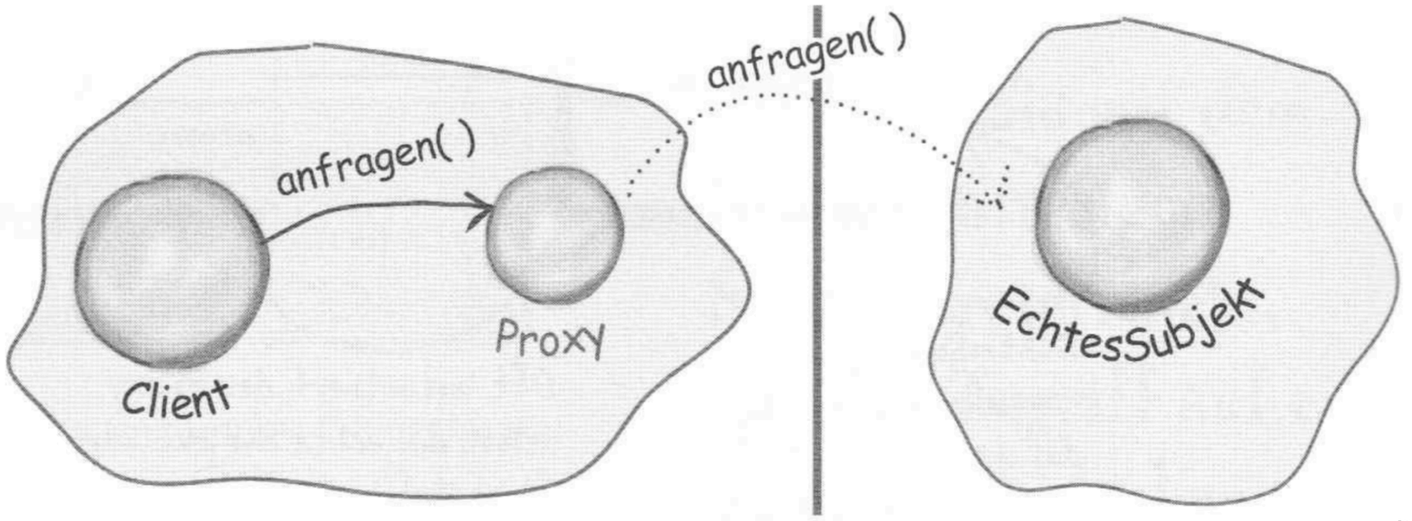
```
$ cd ~/Sites/

$ git clone https://github.com/it4need/php-proxy-pattern
```

EINBLICK IN DIE WELT EINES PROXY-PATTERNS

REMOTE-PROXY

EINFÜHRUNG

- Lokaler Stellvertreter für ein Objekt außerhalb des eigenen Adressbereich
- Client denkt, er kommuniziert mit echtem Objekt
- Reales Objekt = Subjekt
- Stellvertreter = Proxy



[2]

SIMULATE REMOTE REQUEST

```php
<?php

sleep(2); // simulate extensive calculations...
ResponseAsJson(['id' => 'B_93756', 'score' => 81.9, 'owner' => 'server_b']);

function ResponseAsJson($data)
{
    header('Content-Type: application/json');
    echo json_encode($data);
    exit;
}
```

INTERFACE

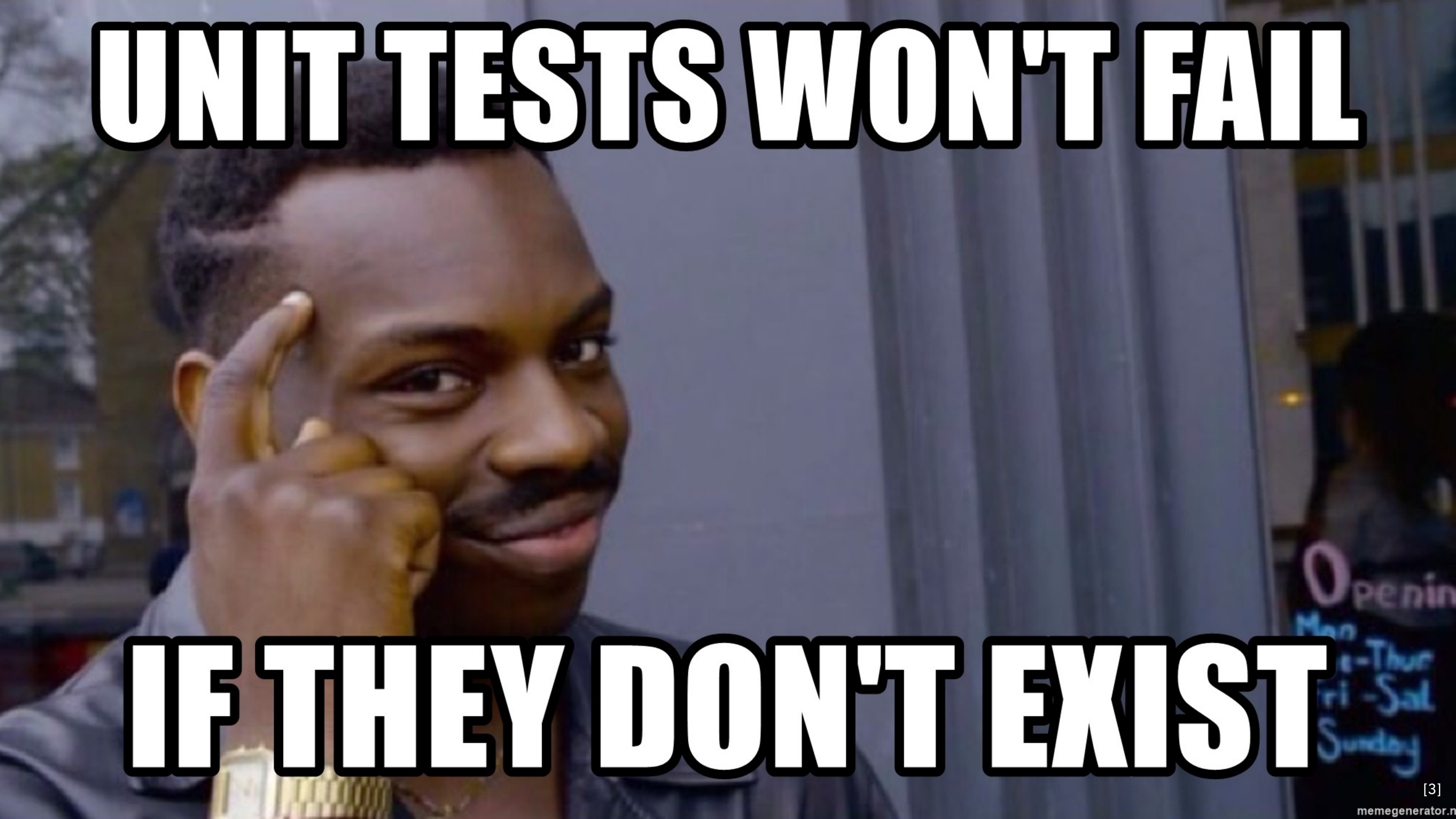
```php
<?php

namespace ProxyPatterns\RemoteProxy;


interface IReport
{
    public function generateReport();

    public function getId();

    public function getOwner();

    public function getScore();
}
```

SUBJEKT

```php
<?php

namespace ProxyPatterns\RemoteProxy;

class ReportSubject implements IReport
{
    private $owner;
    private $id;
    private $score;

    public function __construct($id, $owner, $score)
    {
        $this->setId($id);
        $this->setOwner($owner);
        $this->setScore($score);
    }

    private function setScore($score)
    {
        $this->score = $score;
    }

    public function generateReport()
    {
        $string = '';
        foreach (['id', 'owner', 'score'] as $property) {
            $uc_property = ucfirst($property);
            $string .= $uc_property . ': ' . $this->{"get{$uc_property}"}() . "\n";
        }
        return rtrim($string);
    }

    public function getScore()
    {
        return $this->score;
    }
}
```

PROXY

```php
<?php

namespace ProxyPatterns\RemoteProxy;

use ProxyPatterns\RemoteProxy\Exceptions\ReportNotFoundException;
use Zttp\Zttp;

class ReportProxy implements IReport
{
    private $report;


    public function __construct($report_url)
    {
        $this->setReport($report_url);
    }

    private function setReport($report_url)
    {
        $json_report = Zttp::get($report_url);
        if ($json_report->status() != 200) {
            throw new ReportNotFoundException();
        }
        $report = json_decode($json_report->getBody());
        $this->report = new ReportSubject($report->id, $report->owner, $report->score);
    }

    public function generateReport()
    {
        return $this->report->generateReport();
    }

    public function getScore()
    {
        return $this->report->getScore();
    }
}
```



[3]

AUSSCHNITT UNIT-TEST (HOW TO USE)

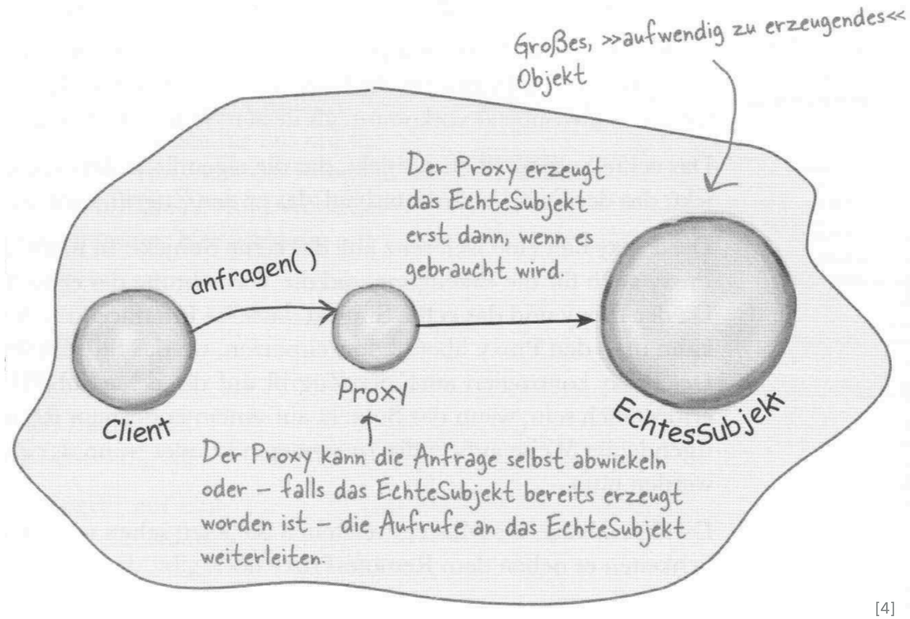
```php
<?php

use ProxyPatterns\RemoteProxy\Exceptions\ReportNotFoundException;
use ProxyPatterns\RemoteProxy\ReportProxy;

class RemoteProxyTest extends PHPUnit\Framework\TestCase
{
    /** @test */
    public function get_correct_from_remote_report()
    {
        $report = new \ProxyPatterns\RemoteProxy\ReportProxy($this->test_server . '/SimulateRemoteRequest.php');

        $this->assertEquals('B_93756', $report->getId());
        $this->assertEquals("server_b", $report->getOwner());
        $this->assertEquals(81.9, $report->getScore());
    }

    /** @test */
    public function get_correct_report_from_server_remote()
    {
        $report = new \ProxyPatterns\RemoteProxy\ReportProxy($this->test_server . '/SimulateRemoteRequest.php');

        $this->assertInternalType("string", $report->generateReport());
        $this->assertEquals("Id: B_93756\nOwner: server_b\nScore: 81.9", $report->generateReport());

    }

    /** @test */
    public function get_correct_report_from_local()
    {
        $report = new \ProxyPatterns\RemoteProxy\ReportSubject('A_352', 'server_a', 17.53);

        $this->assertInternalType("string", $report->generateReport());
        $this->assertEquals("Id: A_352\nOwner: server_a\nScore: 17.53", $report->generateReport());
    }
}
```

EINBLICK IN DIE WELT EINES PROXY-PATTERNS

VIRTUAL-PROXY

EINFÜHRUNG

- ermöglicht Lazy Loading (Laden bei Bedarf)
- Aufwendig zu erzeugtes Objekt = Subjekt
- Proxy instanziiert Subjekt erst, wenn benötigt
- falls bereits erzeugtes Subjekt:
  einfache Delegation an Subjekt



[4]

INTERFACE

```php
<?php

namespace ProxyPatterns\VirtualProxy;


interface IImage
{
    public function getSize();
}
```

SUBJEKT

```php
<?php

namespace ProxyPatterns\VirtualProxy;

class ImageSubject implements IImage
{
    protected $img_data;


    public function __construct($image)
    {
        $this->img_data = file_get_contents($image);
    }

    public function getSize()
    {
        $img_properties = getimagesizefromstring($this->img_data);

        return ['width' => $img_properties[0], 'height' => $img_properties[1]];
    }

}
```

PROXY

```php
<?php

namespace ProxyPatterns\VirtualProxy;

class ImageProxy implements IImage
{

    private $image;
    private $subject;

    public function __construct($image)
    {
        $this->image = $image;
    }

    public function getSize()
    {
        if ($this->subject == null) {
            $this->subject = new ImageSubject($this->image);
        }

        return $this->subject->getSize();
    }

}
```

AUSSCHNITT UNIT-TEST (HOW TO USE)

```php
<?php

class VirtualProxyTest extends PHPUnit\Framework\TestCase
{

    const TEST_IMAGE = __DIR__ . '/../../assets/images/big_image.jpg';


    /** @test */
    public function image_subject_consumes_two_times_memory_after_creating_two_instances_of_the_image()
    {
        $first_memory = $this->memory_usage_in_mb();
        $image = new ProxyPatterns\VirtualProxy\ImageSubject(self::TEST_IMAGE); // image: ≈ 4.8MB, after: first_memory+image
        $second_memory = $this->memory_usage_in_mb();
        $image2 = new ProxyPatterns\VirtualProxy\ImageSubject(self::TEST_IMAGE); // image: ≈ 4.8MB, after: second_memory+image
        $third_memory = $this->memory_usage_in_mb();

        $memory_usage_of_image = $second_memory - $first_memory;
        $memory_usage_of_image_after_second_load = $third_memory - $second_memory;

        $this->assertEquals($memory_usage_of_image, $memory_usage_of_image_after_second_load);
        $this->assertEquals($memory_usage_of_image_after_second_load, $this->get_test_file_size_in_mb());
        $this->assertEquals($image->getSize(), $image2->getSize());
    }

    /** @test */
    public function image_proxy_doesnt_consumes_memory_right_directly_after_creating_one_instance()
    {
        $first_memory = $this->memory_usage_in_mb();
        $image = new                              ImageProxy(self::TEST_IMAGE); // image: ≈ 4.8MB
        $second_memory = $this->memory_usage_in_mb();

        $this->assertEquals($second_memory, $first_memory);
        $this->assertEquals($this->get_test_image_size(), $image->getSize());
    }

    /** @test */
    public function image_proxy_only_consumes_memory_after_called_getSize_method()
    {
        $first_memory = $this->memory_usage_in_mb();
        $image = new ProxyPatterns\VirtualProxy\ImageProxy(self::TEST_IMAGE); // image: ≈ 4.8MB
        $second_memory = $this->memory_usage_in_mb();

        $this->assertEquals($second_memory, $first_memory);
        $this->assertEquals($this->get_test_image_size(), $image->getSize());

        $third_memory = $this->memory_usage_in_mb(); // image is actually loaded by $image->getSize() above
        $memory_usage_of_image = $third_memory - $second_memory;

        $this->assertEquals($memory_usage_of_image, $this->get_test_file_size_in_mb());
    }

}
```

Weitere **Beispiele** des Proxy-Patterns im **Repository**.

it4need/php-proxy–pattern

VOR- UND NACHTEILE

- Vorteile:
  - Client bemerkt die Kontrolle des Objekts nicht
  - Funktioniert auch für Objekte, welche noch nicht geladen sind
  - Ermöglicht je nach Implementierung weiterhin Sicherheitspolicen, Lazy-Loading, Caching, Objektzugang außerhalb eigenem Adressraums
- Nachteile:
  - Weitere Abstraktionssicht kann zu Problemen bei falscher Verwendung führen

DANKESCHÖN!

it4need/php-proxy–pattern

ABBILDUNGSVERZEICHNIS