



Refactoring Unit-Tests

TDD Stoneage



Birth of **T**est **D**riven **D**evelopment ?



Anno 1957



John von Neuman



Anno 1989



Kent Beck



TDD
???

Anno
2005



webmasters.de

2005

```
def test_validate_password
  @user.password = ''
  @user.password_confirmation = ''
  assert !@user.save
  assert_equal ["is too short (minimum is 5 characters)",
               "has invalid characters", "can't be blank"],
               @user.errors.on(:password)

  @user.password = 'testtest'
  @user.password_confirmation = 'testttest'
  assert !@user.save
  assert_equal "doesn't match confirmation" , @user.errors.on(:password)

  @user.password = ''
  @user.password_confirmation = 'testttest'
  assert !@user.save
  assert_equal ["is too short (minimum is 5 characters)",
               "has invalid characters", "doesn't match confirmation",
               "can't be blank"], @user.errors.on(:password)

  @user.password = 'testttest'
  @user.password_confirmation = ''
  assert !@user.save
  assert_equal "doesn't match confirmation", @user.errors.on(:password)

  @user.password = 'test'
  @user.password_confirmation = 'test'
  assert !@user.save
  assert_equal "is too short (minimum is 5 characters)",
               @user.errors.on(:password)

  @user.password = 'testet'
  @user.password_confirmation = 'testet'
  assert @user.save
end
```



```

def test_validate_password
  @user.password = ''
  @user.password_confirmation = ''
  assert !@user.save
  assert_equal ["is too short (minimum is 5 characters)",
                "has invalid characters", "can't be blank"],
                @user.errors.on(:password)

  @user.password = 'testtest'
  @user.password_confirmation = 'testttest'
  assert !@user.save
  assert_equal "doesn't match confirmation" , @user.errors.on(:password)

  @user.password = ''
  @user.password_confirmation = 'testttest'
  assert !@user.save
  assert_equal ["is too short (minimum is 5 characters)",
                "has invalid characters", "doesn't match confirmation",
                "can't be blank"], @user.errors.on(:password)

  @user.password = 'testttest'
  @user.password_confirmation = ''
  assert !@user.save
  assert_equal "doesn't match confirmation", @user.errors.on(:password)

  @user.password = 'test'
  @user.password_confirmation = 'test'
  assert !@user.save
  assert_equal "is too short (minimum is 5 characters)",
                @user.errors.on(:password)

  @user.password = 'testet'
  @user.password_confirmation = 'testet'
  assert @user.save
end

```



There is still hope

...



Test Refactoring



Example







ONLINE Seminar-Shop



ONLINE Seminar-Shop

**Bugs
inside**







Specs



Specs

- Seminare haben **Name** und **Nettopreis**

Seminar
-name: String -net_price: Float

Ruby

Seminar
-name: String -netPrice: float

Java



Specs

- Seminare haben **Name** und **Nettopreis**
- Seminare berechnen Ihren **Bruttopreis** - außer, sie sind **steuerbefreit**

Seminar
-name: String
-net_price: Float
-tax free: Boolean
+gross_price():Float

Ruby

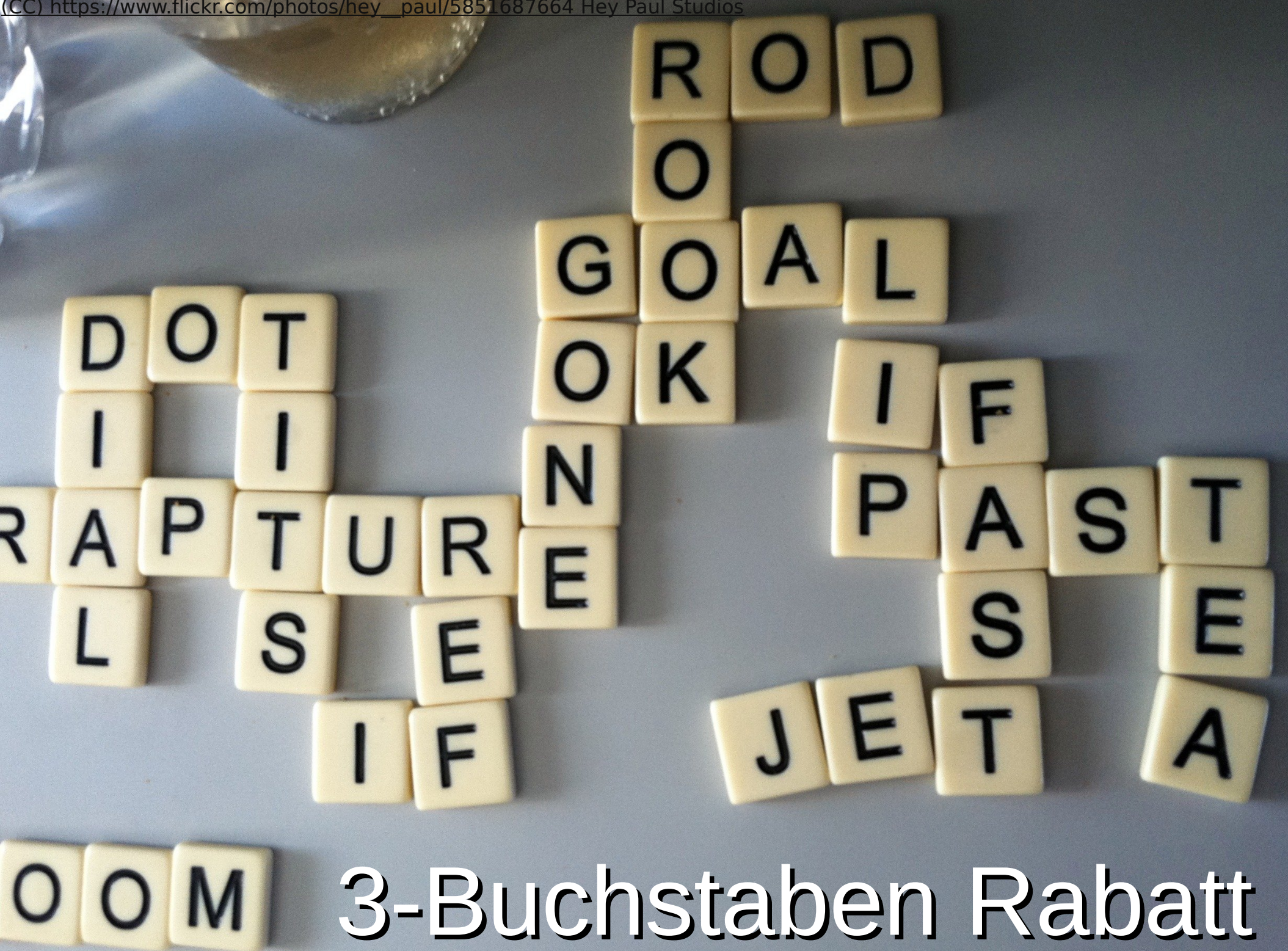
Seminar
-name: String
-netPrice: float
-taxFree: boolean
+grossPrice():float

Java/JS



Marketing Department





3-Buchstaben Rabatt

Specs

- Seminare haben **Name** und **Nettopreis**
- Seminare berechnen Ihren **Bruttopreis** - außer, sie sind **steuerbefreit**
- **3-Buchstaben-Seminare*** bekommen 5% **Rabatt**

Seminar
-name: String -net_price: Float -tax free: Boolean
+net_price(): Float +gross_price():Float +discount_rate(): Float +discount(): Float

Ruby

Seminar
-name: String -netPrice: float -taxFree: boolean
+netPrice(): float +grossPrice():float +discountRate(): float +discount(): float

Java/JS



```

class Seminar
  TAX_RATE = 1.19
  THREE_LETTER_DISCOUNT_RATE = 5

  attr_writer :net_price, :tax_free, :name

  def initialize(name, net_price, tax_free)
    @name, @net_price, @tax_free =
      name, net_price, tax_free
  end

  def gross_price
    net_price * tax_rate
  end

  def net_price
    @net_price - discount
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end

  def discount
    @net_price * discount_rate / 100
  end

```

```

  def discount_rate
    discount_granted? ?
      THREE_LETTER_DISCOUNT_RATE :
      0
  end

  def discount_granted?
    @name.size < 3
  end
end

```



```

class Seminar
  TAX_RATE = 1.19
  THREE_LETTER_DISCOUNT_RATE = 5

  attr_writer :net_price, :tax_free, :name

  def initialize(name, net_price, tax_free)
    @name, @net_price, @tax_free =
      name, net_price, tax_free
  end

  def gross_price
    net_price * tax_rate
  end

  def net_price
    @net_price - discount
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end

  def discount
    @net_price * discount_rate / 100
  end

```

```

  def discount_rate
    discount_granted? ?
      THREE_LETTER_DISCOUNT_RATE :
      0
  end

  def discount_granted?
    @name.size < 3
  end
end

```





Tests?

```
class SeminarTest < Test::Unit::TestCase

  def test_seminar_should_calculate_correct_gross_prices
    seminar = Seminar.new('OOP', 500, false)

    assert_equal 565.25, seminar.gross_price

    seminar.net_price = 300
    assert_equal 339.15, seminar.gross_price

    seminar.tax_free = true
    assert_equal 285, seminar.gross_price

    seminar.name = 'Objekt-Orientierte Programmierung'
    assert_equal 300, seminar.gross_price
  end
end
```




```
class SeminarTest < Test::Unit::TestCase


  def test_seminar_should_calculate_correct_gross_prices
    seminar = Seminar.new('OOP', 500, false)

    assert_equal 565.25, seminar.gross_price

    seminar.net_price = 300
    assert_equal 339.15, seminar.gross_price

    seminar.tax_free = true
    assert_equal 285, seminar.gross_price

    seminar.name = 'Objekt-Orientierte Programmierung'
    assert_equal 300, seminar.gross_price
  end
end
```



Failure:
<565.25> expected but was
<5.95>.



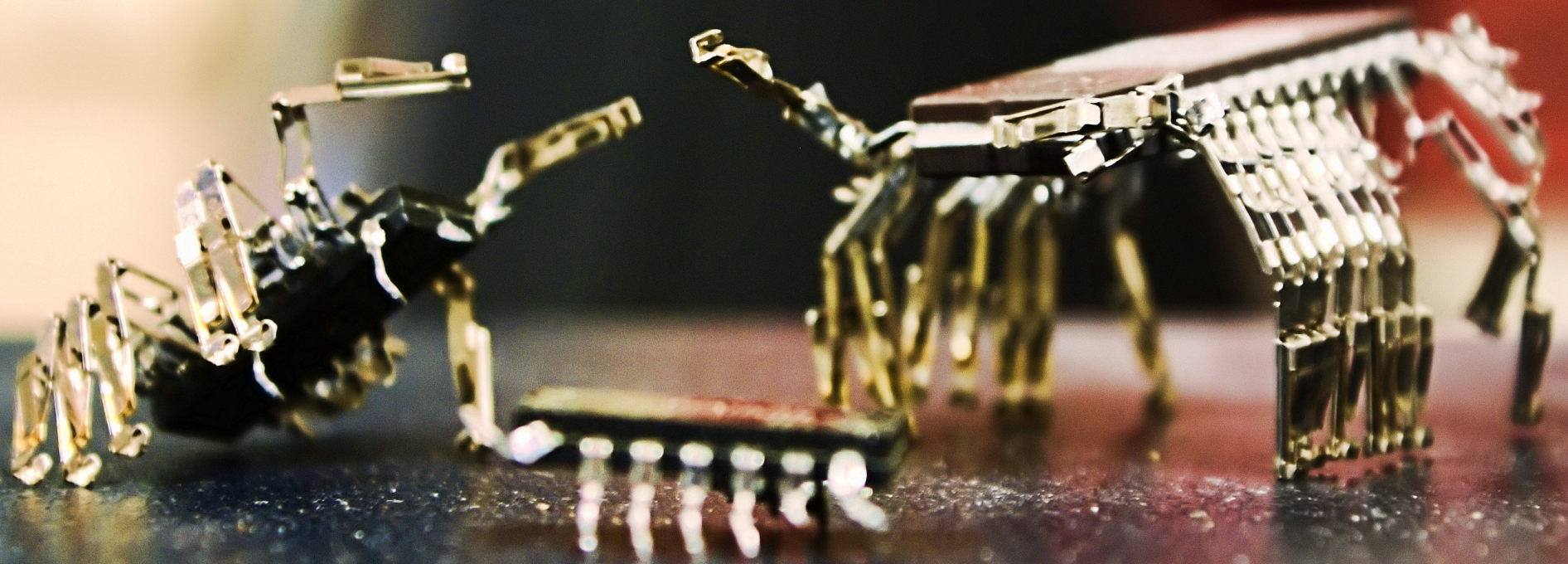
I made a Redundant Clock...



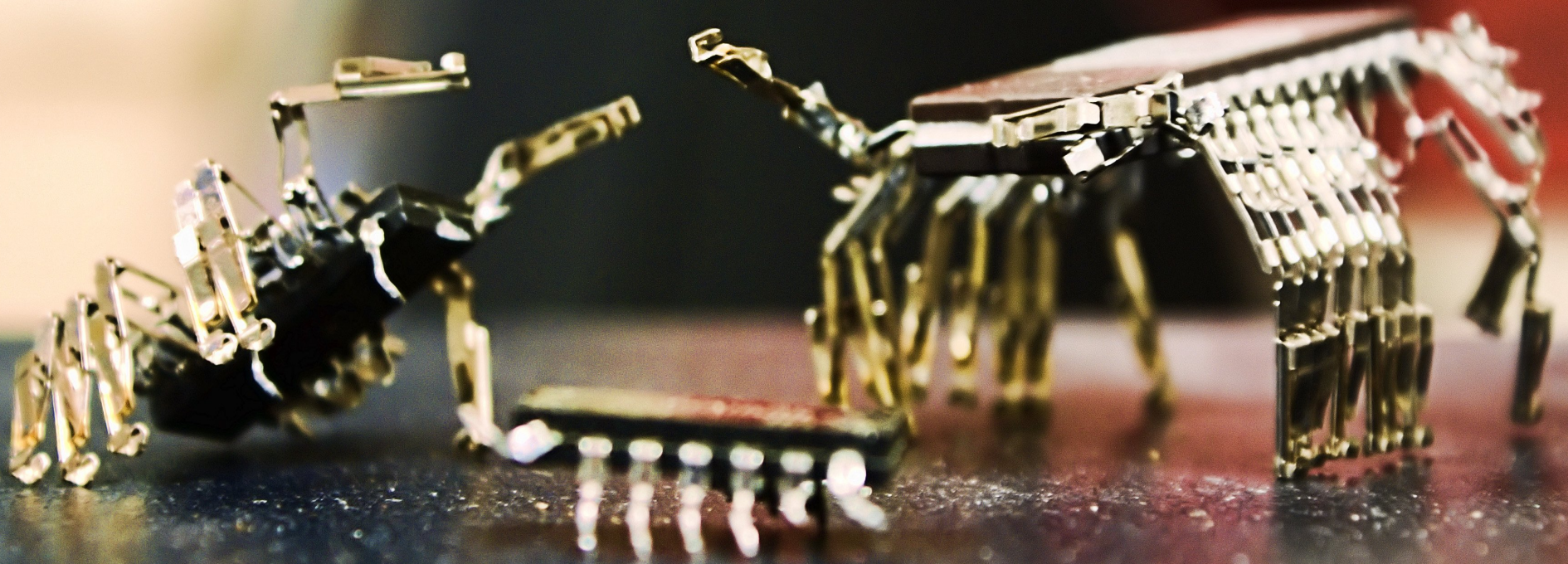
3 minutes



Bug?



2 Bugs?





alternative
test suit



9 test cases



2 failing tests


```
def test_seminar_should_have_the_german_mwst_tax_rate_if_it_is_not_tax_free
  seminar = create_seminar(tax_free: false)
  assert_equal Seminar::TAX_RATE, seminar.tax_rate
end
```

Failure:
<1.19> expected but was
<0.0119>.

```
def test_discount_granted_should_return_true_if_name_consists_of_3_letters
  seminar = create_seminar(name: 'OOP') # 3 Letters
  assert seminar.discount_granted?
end
```

Failure:
<false> is not true.



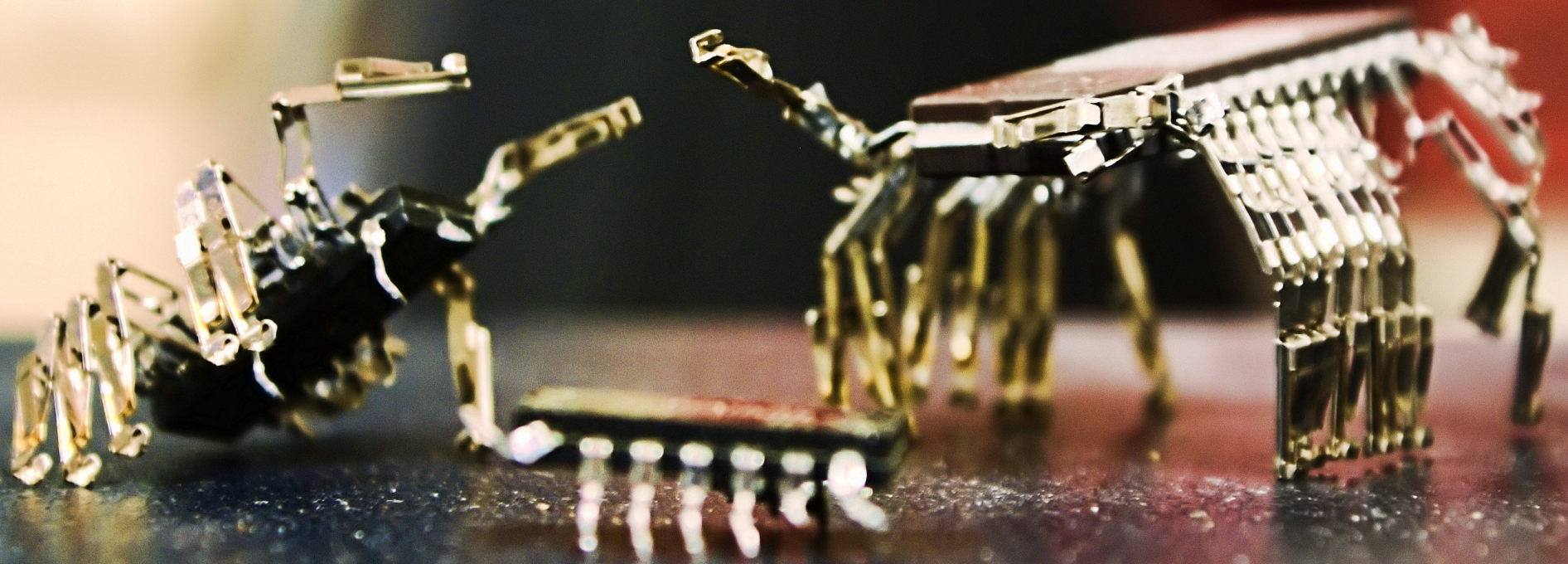
I made a Redundant Clock...



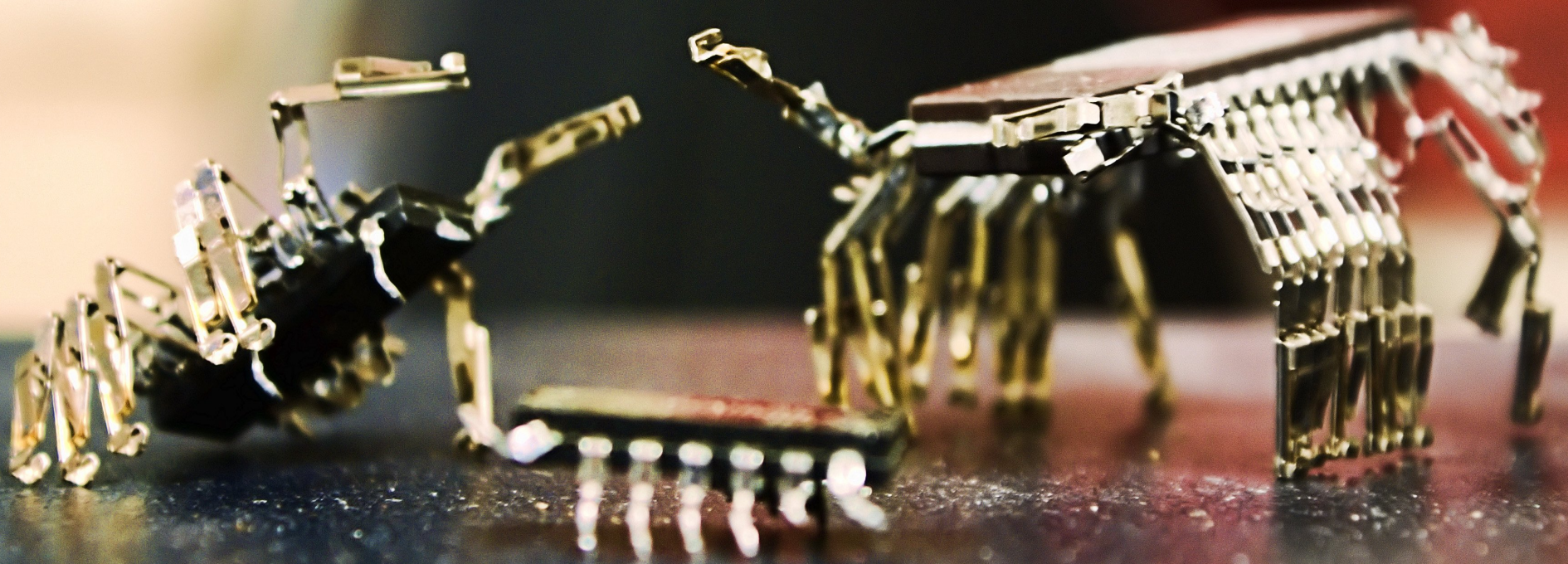
3 minutes



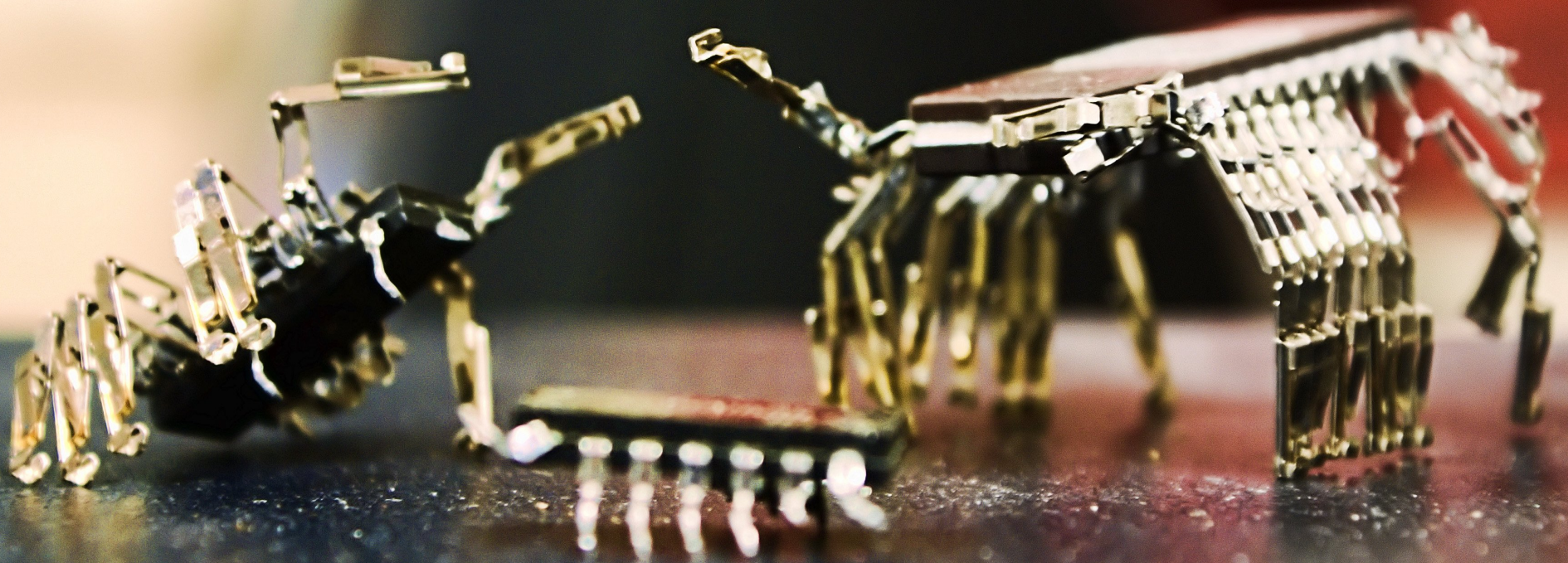
Bug?



2 Bugs?



3 Bugs?






```

class Seminar
  TAX_RATE = 1.19
  THREE_LETTER_DISCOUNT_RATE = 5

  attr_writer :net_price, :tax_free, :name

  def initialize(name, net_price, tax_free)
    @name, @net_price, @tax_free =
      name, net_price, tax_free
  end

  def gross_price
    net_price * tax_rate
  end

  def net_price
    @net_price - discount
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end

  def discount
    @net_price * discount_rate / 100
  end

```

```

  def discount_rate
    discount_granted? ?
      THREE_LETTER_DISCOUNT_RATE :
      0
  end

  def discount_granted?
    @name.size < 3
  end
end

```



```

class Seminar
  TAX_RATE = 1.19
  THREE_LETTER_DISCOUNT_RATE = 5

  attr_writer :net_price, :tax_free, :name

  def initialize(name, net_price, tax_free)
    @name, @net_price, @tax_free =
      name, net_price, tax_free
  end

  def gross_price
    net_price * tax_rate
  end

  def net_price
    @net_price - discount
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE
  end

  def discount
    @net_price * discount_rate / 100
  end

```

```

def discount_rate
  discount_granted? ?
    THREE_LETTER_DISCOUNT_RATE :
    0
end

def discount_granted?
  @name.size <= 3
end

end

```





comparing
test suites

Quality Criteria for test cases



Readable



...between NeXT and Apple...
...Apple executives. NeXT went first, with Avie Tevanian...
...showed how the software displayed his hypnotizing...
...the NeXT operating system was the Internet...
...praised the virtues and strengths of Olivier as Macbeth...
...afterward, but he acted as if he had the deal in his...
...Amelio said that was too high. He countered with...
...\$12 a share for NeXT. That would amount to...
...Gasser's mistake of overreaching. He was...
...Unlike Be, NeXT had an act...
...but Jobs was nevertheless...
...He accepted immediately...
...He wanted his payout to...
...Jobs wanted his skin in the game...
...that he needed to "have skin in the game...
...he would agree to hold for at least...



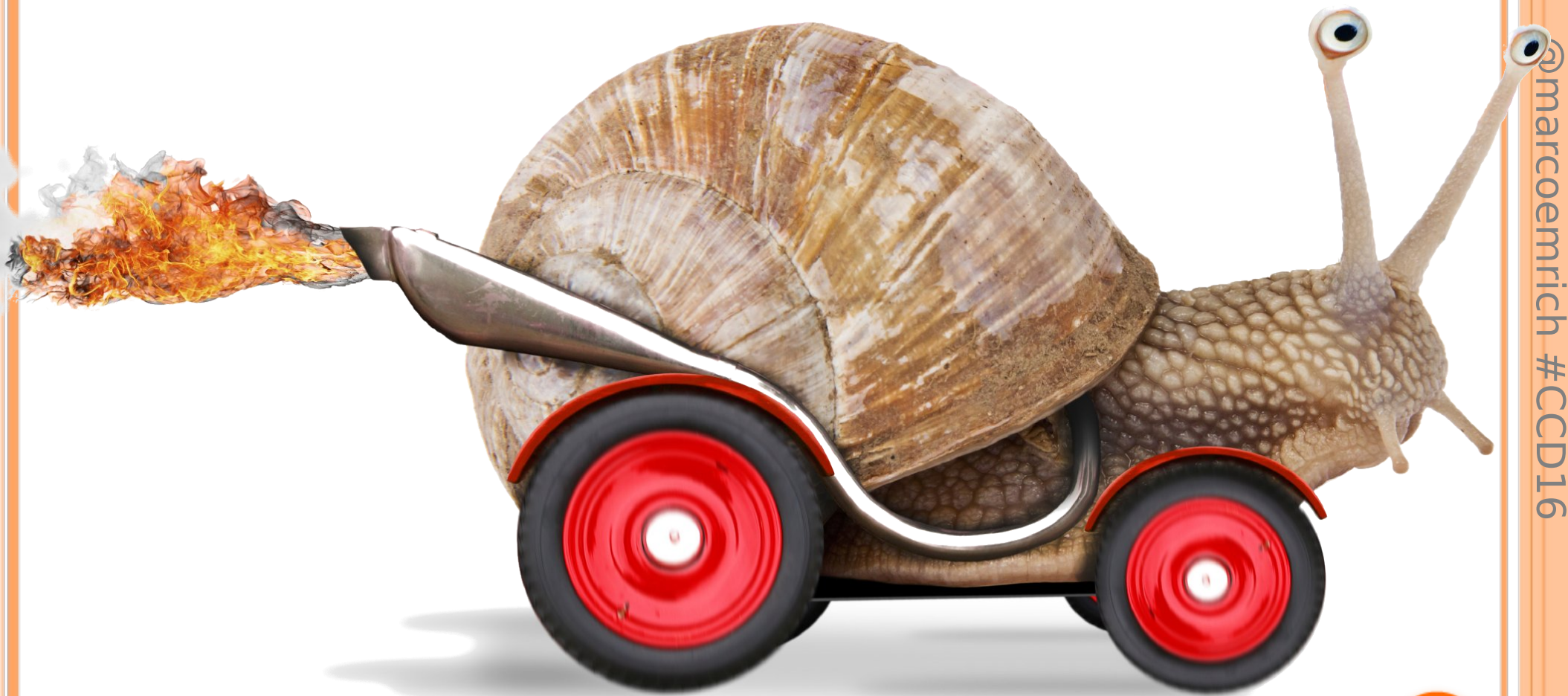
Living Documentation!



Testing the Tests?



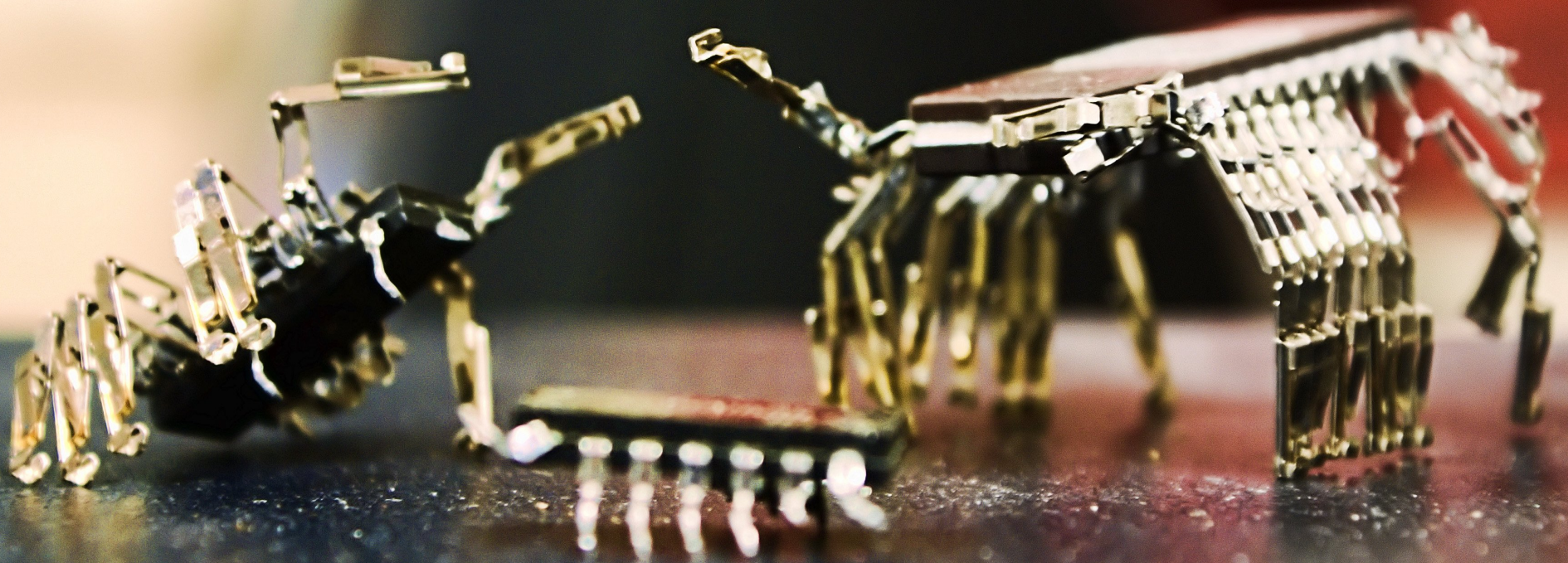
Fast

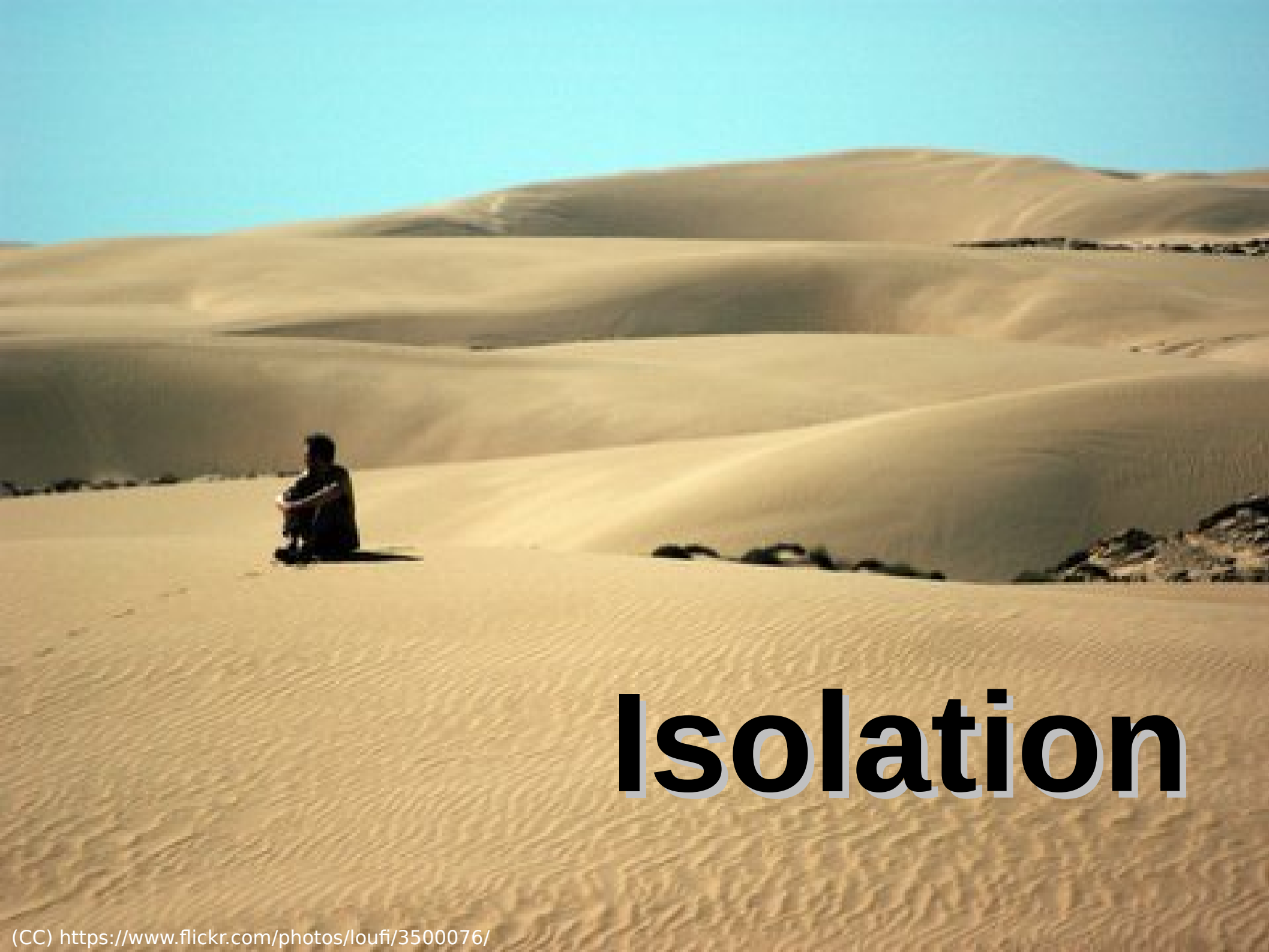


Maintainable



Defect Localization





Isolation

Focus



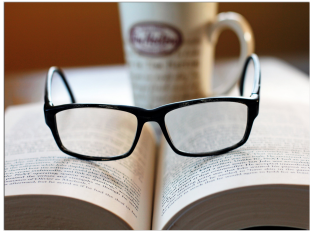


J.B. Rainsberger

*"Jeder Test-Case prüft
genau eine interessante
Verhaltensweise"*

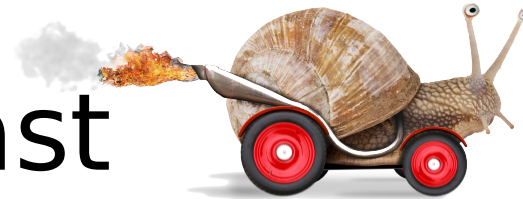


Quality Criteria



Readable

Fast



Maintainable

Defect Localization



Conflicts?



Optimaler Kompromiss?





Kent Beck

TDD



BDD



Dan North



TDD/BDD führen zu guten Tests!



TDD/BDD führen zu guten
Tests!
(nicht unbedingt
automatisch)



Verbesserungen möglich!



Legacy Test Suites





Testcode muss gewartet werden



Test Refactoring Example




```
class SeminarTest < Test::Unit::TestCase

  def test_seminar_should_calculate_correct_gross_prices
    seminar = Seminar.new('OOP', 500, false)

    assert_equal 565.25, seminar.gross_price

    seminar.net_price = 300
    assert_equal 339.15, seminar.gross_price

    seminar.tax_free = true
    assert_equal 285, seminar.gross_price

    seminar.name = 'Objekt-Orientierte Programmierung'
    assert_equal 300, seminar.gross_price
  end
end
```



```
class SeminarTest < Test::Unit::TestCase

  def test_seminar_should_calculate_correct_gross_prices
    seminar = Seminar.new('OOP', 500, false)

    assert_equal 565.25, seminar.gross_price

    seminar.net_price = 300
    assert_equal 339.15, seminar.gross_price

    seminar.tax_free = true
    assert_equal 285, seminar.gross_price

    seminar.name = 'Objekt-Orientierte Programmierung'
    assert_equal 300, seminar.gross_price
  end
end
```

Remove Redudant Tests
(gleiche Äquivalenzklasse)




```
class SeminarTest < Test::Unit::TestCase

  def test_seminar_should_calculate_correct_gross_prices
    seminar = Seminar.new('OOP', 500, false)

    assert_equal 565.25, seminar.gross_price

seminar.net_price = 300
assert_equal 339.15, seminar.gross_price

    seminar.tax_free = true
    assert_equal 475, seminar.gross_price

    seminar.name = 'Objekt-Orientierte Programmierung'
    assert_equal 500, seminar.gross_price
  end
end
```

Remove Redudant Tests



```
class SeminarTest < Test::Unit::TestCase

  def test_seminar_should_calculate_correct_gross_prices
    seminar = Seminar.new('OOP', 500, false)

    assert_equal 565.25, seminar.gross_price

    seminar.tax_free = true
    assert_equal 475, seminar.gross_price

    seminar.name = 'Objekt-Orientierte Programmierung'
    assert_equal 500, seminar.gross_price
  end
end
```




```
class SeminarTest < Test::Unit::TestCase

  def test_seminar_should_calculate_correct_gross_prices
    seminar = Seminar.new('Objekt-Orientierte Programmierung', 500, true)

    assert_equal 500, seminar.gross_price

    seminar.name = 'OOP'
    assert_equal 475, seminar.gross_price

    seminar.tax_free = false
    assert_equal 565.25, seminar.gross_price
  end
end
```

Use **neutral fixture**
(Build up!)



```
class SeminarTest < Test::Unit::TestCase

  def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
    seminar = Seminar.new('Objekt-Orientierte Programmierung', 500, true)
    assert_equal 500, seminar.gross_price
  end

  def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
    seminar = Seminar.new('Objekt-Orientierte Programmierung', 500, false)
    assert_equal 595, seminar.gross_price
  end

  def test_a_3letter_seminar_should_return_a_gross_price_with_discount
    seminar = Seminar.new('OOP', 500, true)
    assert_equal 475, seminar.gross_price
  end

end
```

Split test methods
Fresh fixture
Arrange Act Assert: **AAA**-Pattern




```
class SeminarTest < Test::Unit::TestCase
```

```
  def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
```

```
    seminar = Seminar.new('Objekt-Orientierte Programmierung', 500, true)
```

```
    assert_equal 500, seminar.gross_price
```

```
  end
```

```
  def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
```

```
    seminar = Seminar.new('Objekt-Orientierte Programmierung', 500, false)
```

```
    assert_equal 595, seminar.gross_price
```

```
  end
```

```
  def test_a_3letter_seminar_should_return_a_gross_price_with_discount
```

```
    seminar = Seminar.new('OOP', 500, true)
```

```
    assert_equal 475, seminar.gross_price
```

```
  end
```

```
end
```

Use test data factories



Test data factories

- **Ruby-Frameworks**

- Factory-Girl
- Machinist

- **Java-Frameworks**

- Usurper
- PojoBuilder

- **Patterns**

- Object Mothers
- Test Data Builders/Factories
- Example Factories



```
class SeminarTest < Test::Unit::TestCase
```

```
  def create_seminar(args = {})
```

```
    new_args = {
```

```
      :name => 'Object Oriented Programming'
```

```
      :net_price => 500,
```

```
      :tax_free => true
```

```
    }.merge(args)
```

```
    Seminar.new(new_args[:name], new_args[:net_price], new_args[:tax_free])
```

```
  end
```

```
end
```

Defaults
(neutral Fixture)



Use factories




```
class SeminarTest < Test::Unit::TestCase
```

```
  def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
```

```
    seminar = Seminar.new('Objekt-Orientierte Programmierung', 500, true)
```

```
    seminar = create_seminar(tax_free: true)
```

```
    assert_equal 500, seminar.gross_price
```

```
  end
```

```
  def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
```

```
    seminar = Seminar.new('Objekt-Orientierte Programmierung', 500, false)
```

```
    seminar = create_seminar(tax_free: false)
```

```
    assert_equal 595, seminar.gross_price
```

```
  end
```

```
  def test_a_3letter_seminar_should_return_a_gross_price_with_discount
```

```
    seminar = Seminar.new('OOP', 500, true)
```

```
    seminar = create_seminar(name: 'OOP')
```

```
    assert_equal 475, seminar.gross_price
```

```
  end
```

```
end
```

Use factories



```
class SeminarTest < Test::Unit::TestCase

  def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
    seminar = create_seminar(tax_free: true)
    assert_equal(500, seminar.gross_price)
  end

  def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
    seminar = create_seminar(tax_free: false)
    assert_equal(595, seminar.gross_price)
  end

  def test_a_3letter_seminar_should_return_a_gross_price_with_discount
    seminar = create_seminar(name: 'OOP')
    assert_equal(475, seminar.gross_price)
  end

end
```

Use factories



```
class SeminarTest < Test::Unit::TestCase
```

```
  def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
```

```
    seminar = create_seminar(tax_free: true)
```

```
    assert_equal seminar.net_price, seminar.gross_price
```

```
  end
```

```
  def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
```

```
    seminar = create_seminar(tax_free: false)
```

```
    assert_equal seminar.net_price * Seminar::TAX_RATE, seminar.gross_price
```

```
  end
```

```
  def test_a_3letter_seminar_should_return_a_gross_price_with_discount
```

```
    seminar = create_seminar(name: 'OOP', net_price: 500)
```

```
    assert_equal 500 * 0.95, seminar.gross_price
```

```
  end
```

```
end
```

Use factories




```
class SeminarTest < Test::Unit::TestCase

  def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
    seminar = create_seminar(tax_free: true)
    assert_equal seminar.net_price, seminar.gross_price
  end

  def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
    seminar = create_seminar(tax_free: false)
    assert_equal seminar.net_price * Seminar::TAX_RATE, seminar.gross_price
  end

  def test_a_3letter_seminar_should_return_a_gross_price_with_discount
    seminar = create_seminar(name: 'OOP', net_price: 500)
    assert_equal 500 * 0.95, seminar.gross_price
  end

end
```

add missing test



```
class SeminarTest < Test::Unit::TestCase

  def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
    seminar = create_seminar(tax_free: true)
    assert_equal seminar.net_price, seminar.gross_price
  end

  def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
    seminar = create_seminar(tax_free: false)
    assert_equal seminar.net_price * Seminar::TAX_RATE, seminar.gross_price
  end

  def test_a_3letter_seminar_should_return_a_gross_price_with_discount
    seminar = create_seminar(name: 'OOP', net_price: 500)
    assert_equal 500 * 0.95, seminar.gross_price
  end

  def test_a_more_letters_seminar_should_return_a_net_price_without_discount
    seminar = create_seminar(name: 'Object 0. Programming', net_price: 500)
    assert_equal 500, seminar.gross_price
  end

end
```

add missing test



Discount

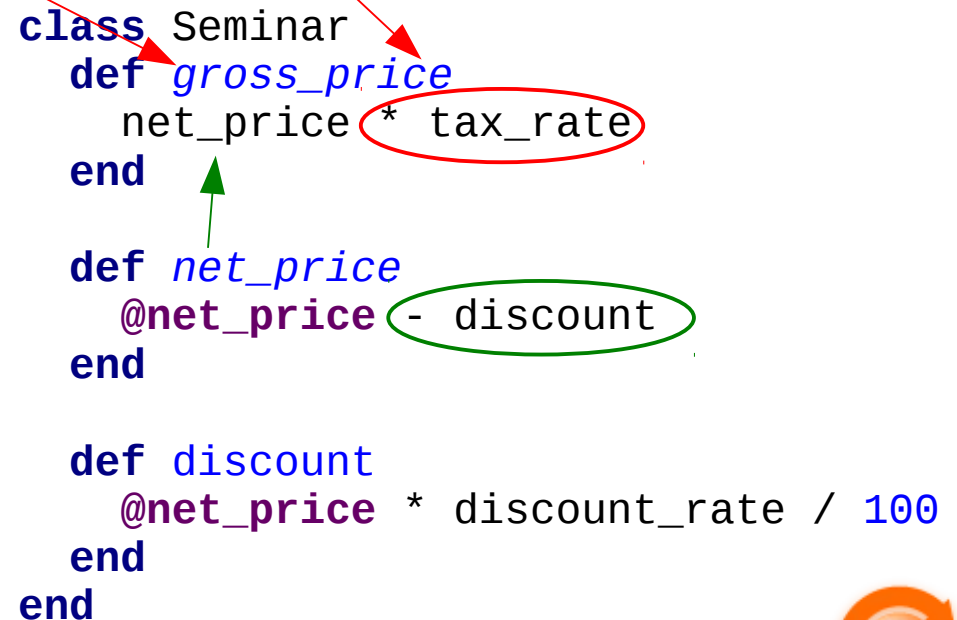
```
def test_a_3letter_seminar_should_return_a_gross_price_with_discount
  seminar = create_seminar(name: 'OOP')
  assert_equal 500 * 0.95, seminar.gross_price
end

def test_a_more_letters_seminar_should_return_a_net_price_without_discount
  seminar = create_seminar(name: 'Object Oriented Programming')
  assert_equal 500, seminar.gross_price
end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end

  def net_price
    @net_price - discount
  end

  def discount
    @net_price * discount_rate / 100
  end
end
```



Isolate



Discount

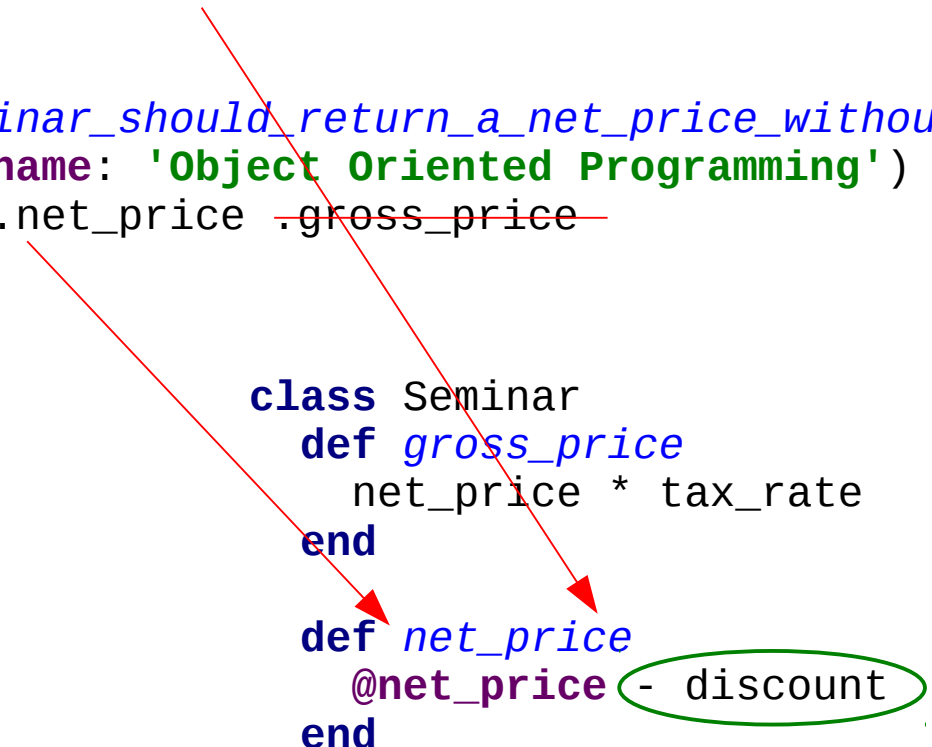
```
def test_a_3letter_seminar_should_return_a_gross_price_with_discount
  seminar = create_seminar(name: 'OOP')
  assert_equal 500 * 0.95, seminar.net_price .gross_price
end

def test_a_more_letters_seminar_should_return_a_net_price_without_discount
  seminar = create_seminar(name: 'Object Oriented Programming')
  assert_equal 500, seminar.net_price .gross_price
end

class Seminar
  def gross_price
    net_price * tax_rate
  end

  def net_price
    @net_price - discount
  end

  def discount
    @net_price * discount_rate / 100
  end
end
```



Isolate



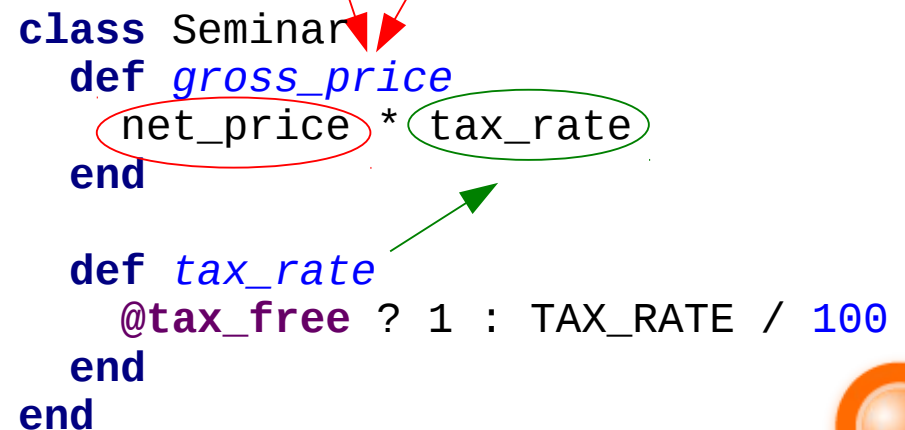
Tax

```
def test_a_tax_free_seminar_should_return_a_gross_price_without_tax
  seminar = create_seminar(tax_free: true)
  assert_equal seminar.net_price, seminar.gross_price
end
```

```
def test_a_not_tax_free_seminar_should_return_gross_price_with_correct_tax
  seminar = create_seminar(tax_free: false)
  assert_equal seminar.net_price * Seminar::TAX_RATE, seminar.gross_price
end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end
end
```



Isolate



Tax

```
def test_a_tax_free_seminar_should_have_a_tax_rate_of_1
  seminar = create_seminar(tax_free: true)
  assert_equal seminar.net_price, seminar.gross_price
  assert_equal 1, seminar.tax_rate
end
```

```
def test_a_not_tax_free_seminar_should_have_the_correct_tax_rate
  seminar = create_seminar(tax_free: false)
  assert_equal seminar.net_price * Seminar::TAX_RATE, seminar.gross_price
  assert_equal Seminar::TAX_RATE, seminar.tax_rate
end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end
  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end
end
```

Isolate



Tax

```
def test_a_tax_free_seminar_should_have_a_tax_rate_of_1
  seminar = create_seminar(tax_free: true)
  assert_equal 1, seminar.tax_rate
end

def test_a_not_tax_free_seminar_should_have_the_correct_tax_rate
  seminar = create_seminar(tax_free: false)
  assert_equal Seminar::TAX_RATE, seminar.tax_rate
end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end
end
```

Isolate



Tax

```
def test_a_tax_free_seminar_should_have_a_tax_rate_of_1
  seminar = create_seminar(tax_free: true)
  assert_equal 1, seminar.tax_rate
end

def test_a_not_tax_free_seminar_should_have_the_correct_tax_rate
  seminar = create_seminar(tax_free: false)
  assert_equal Seminar::TAX_RATE, seminar.tax_rate
end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end
end
```

Lost Coverage



Tax

```
def test_a_tax_free_seminar_should_have_a_tax_rate_of_1
  seminar = create_seminar(tax_free: true)
  assert_equal 1, seminar.tax_rate
end

def test_a_not_tax_free_seminar_should_have_the_correct_tax_rate
  seminar = create_seminar(tax_free: false)
  assert_equal Seminar::TAX_RATE, seminar.tax_rate
end

def test_seminar_should_use_tax_rate_to_calculate_gross_price

end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end

  def net_price
    @net_price - discount
  end
end
```

Lost Coverage



Tax

```
def test_a_tax_free_seminar_should_have_a_tax_rate_of_1
  seminar = create_seminar(tax_free: true)
  assert_equal 1, seminar.tax_rate
end
```

```
def test_a_not_tax_free_seminar_should_have_the_correct_tax_rate
  seminar = create_seminar(tax_free: false)
  assert_equal Seminar::TAX_RATE, seminar.tax_rate
end
```

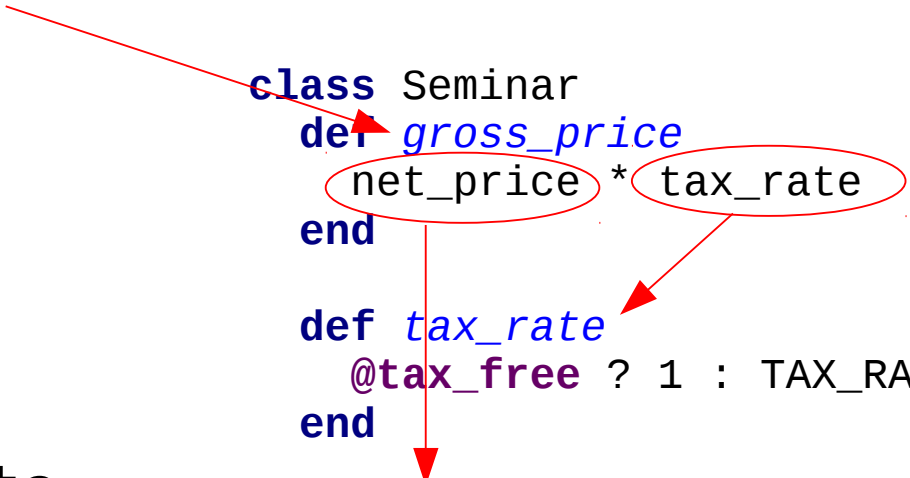
```
def test_seminar_should_use_tax_rate_to_calculate_gross_price
  seminar = create_seminar(tax_free: false)

  assert_equal ?, seminar.gross_price
end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end

  def net_price
    @net_price - discount
  end
end
```



Isolate
Use **Stubs**



Tax

```
def test_a_tax_free_seminar_should_have_a_tax_rate_of_1
  seminar = create_seminar(tax_free: true)
  assert_equal 1, seminar.tax_rate
end
```

```
def test_a_not_tax_free_seminar_should_have_the_correct_tax_rate
  seminar = create_seminar(tax_free: false)
  assert_equal Seminar::TAX_RATE, seminar.tax_rate
end
```

```
def test_seminar_should_use_tax_rate_to_calculate_gross_price
  seminar = create_seminar(tax_free: false)
  seminar.stubs(net_price: 100)
  seminar.stubs(tax_rate: 1.5)
  assert_equal 150, seminar.gross_price
end
```

```
class Seminar
  def gross_price
    net_price * tax_rate
  end

  def tax_rate
    @tax_free ? 1 : TAX_RATE / 100
  end

  def net_price
    @net_price - discount
  end
end
```

Isolate
Use **Stubs**



Mocks & Stubs

- **Ruby**

- Rspec-Mocks
- Mocha
- FlexMock

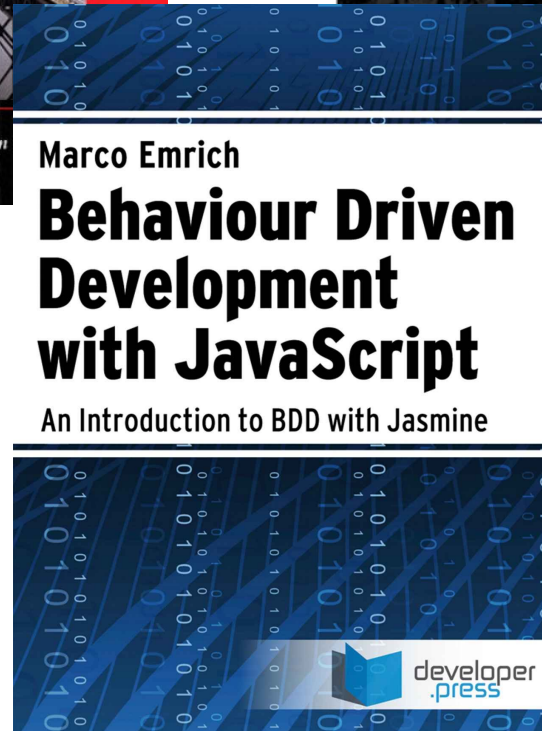
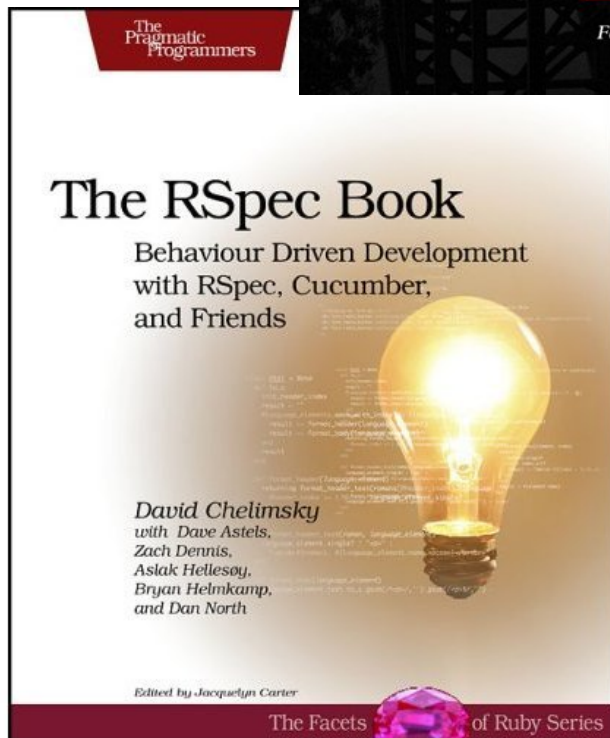
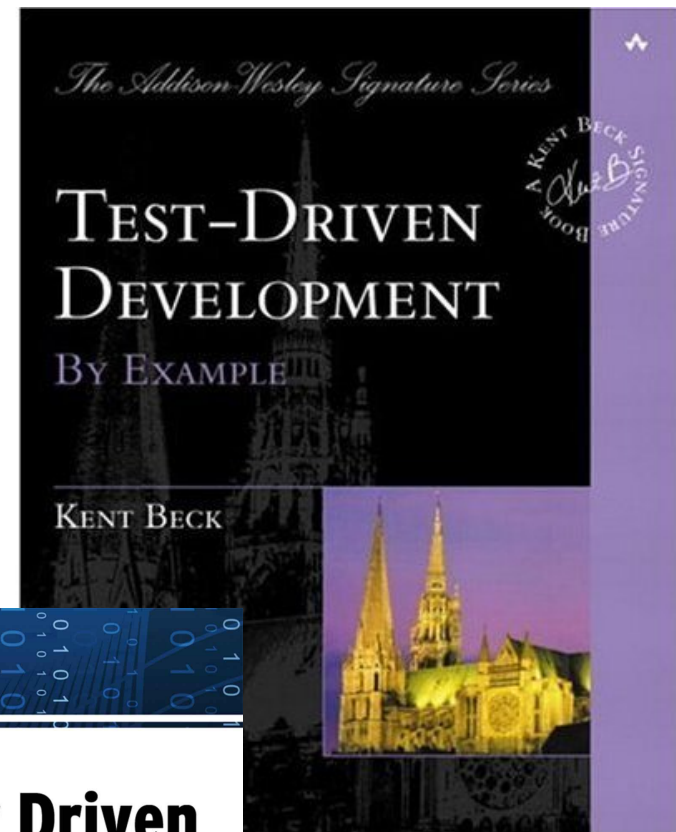
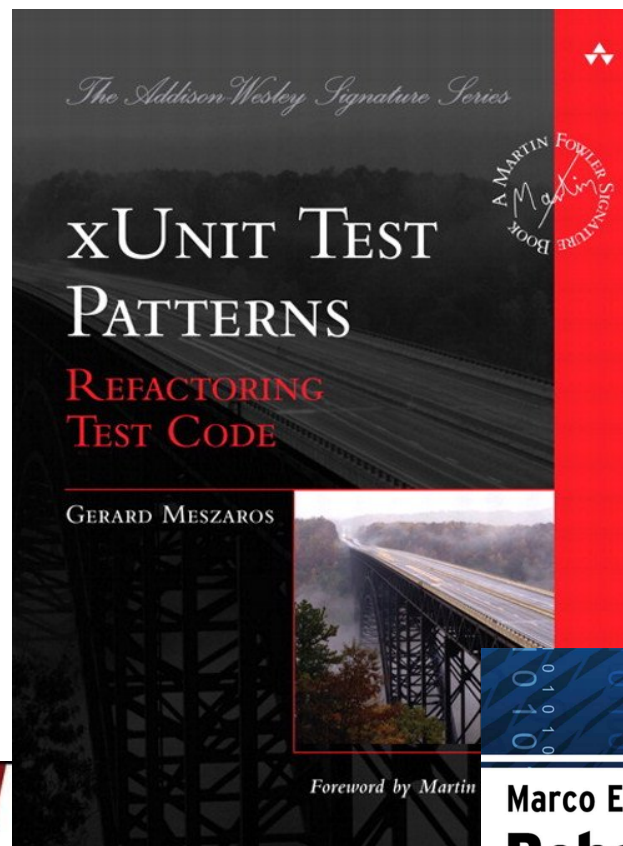
- **Java**

- Mockito
- EasyMock
- Jmockit

- **JavaScript**

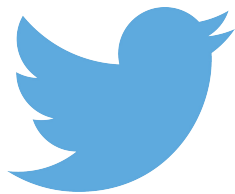
- SinonJS
- Jasmine Spies







<https://github.com/marcoemrich>



[@marcoemrich](https://twitter.com/marcoemrich)

