Voorbeeld van testgestuurd ontwikkelen

We willen een functie *truncate* maken die een gegeven tekst afkapt op een gegeven lengte. Voorbeeld: `truncate('een stuk tekst', 8)` levert op: `'een s...'`. We gebruiken de object-georiënteerde taal Ruby, waar standaard een testraamwerk meegeleverd wordt.

De verschillende testgevallen maken deel uit van een klasse die afgeleid wordt van de standaardklasse TestCase. We gaan verder niet in op de details van het raamwerk. Elk testgeval wordt geschreven als functie op die klasse.

```
require 'test/unit'

class TruncateTest < Test::Unit::TestCase

        def test_truncate_emptyString
                assert_equal '', truncate('', 4)
        end

end
```

We hebben hier het eenvoudigste testgeval geformuleerd: een lege tekst en een lengte van 4 moet een lege tekst opleveren. De functie `assert_equal` controleert of beide argumenten gelijk zijn.

De eenvoudigste implementatie die dit testgeval laat slagen is:

```
def truncate text, length
        return ''
end
```

Uitvoeren van de test levert op:

```
Started
......
Finished in 0.0 seconds.

1 tests, 1 assertions, 0 failures, 0 errors
```

We voegen een test toe voor het geval dat de tekst korter is dan de opgegeven maximale lengte. De tekst moet onveranderd worden opgeleverd:

```
def test_no_truncation
        assert_equal 'een tekst', truncate('een tekst', 12)
end
```

De test faalt:

```
Started
F.
Finished in 0.031 seconds.

  1) Failure:
test_no_truncation(TruncateTest)
<"een tekst"> expected but was
<"">.

2 tests, 2 assertions, 1 failures, 0 errors
```

We moeten de implementatie van de `truncate` functie dus aanpassen en we kiezen weer voor de eenvoudigste oplossing. Deze werkwijze lijkt op het eerste gezicht wellicht simplistisch, maar werkt effectief om onnodige complexiteit te voorkomen.

```
def truncate text, length
        return text
end
```

De test slaagt nu:

```
Started
..
Finished in 0.0 seconds.

2 tests, 2 assertions, 0 failures, 0 errors
```

Werkt de huidige implementatie ook voor het randgeval waarbij de lengte van de tekst precies gelijk is aan de opgegeven lengte?

```
def test_text_as_long_as_length
        assert_equal 'meer tekst', truncate('meer tekst', 10)
end
```

Het werkt naar behoren:

```
3 tests, 3 assertions, 0 failures, 0 errors
```

Als de tekst langer is dan de gegeven lengte, moet deze correct worden afgebroken:

```
def test_truncate
        assert_equal 'nog meer...', truncate('nog meer tekst', 11)
end
```

Deze test faalt in eerste instantie, dus we passen de implementatie aan.

```
def truncate text, length
        if text.length > length then
                return text[0 .. length-4] + '...'
        end

        return text
end
```

De tests slagen nu. Er is nog een aantal randgevallen te onderscheiden. Werkt de functie bijvoorbeeld correct als de gegeven lengte net kleiner is dan de lengte van de tekst?

```
def test_text_and_truncation_as_long_as_length
        assert_equal 'meer t...', truncate('meer tekst', 9)
end
```

Wat als de opgegeven lengte kleiner is dan de lengte van de afbreektekst? We kiezen ervoor dat dan alleen de betreffende tekens van de tekst opgeleverd moeten worden, zonder afbreektekst.

```
def test_truncation_text_larger_than_length
        assert_equal 'vo', truncate('voorbeeld', 2)
end
```

De huidige implementatie blijkt correct te werken voor het eerste geval maar niet voor het tweede geval. De implementatie van truncate wordt hiervoor aangepast:

```
def truncate text, length
        if length < 3 then
                return text[0 .. length-1]
        end

        if text.length > length then
                return text[0 .. length-4] + '...'
        end

        return text
end
```

In de `truncate` functie wordt op drie plekken verwezen naar de afbreektekst of de lengte ervan, zonder dat de afhankelijkheden tussen deze drie lokaties expliciet is. Deze duplicatie kunnen we verwijderen door een constante te introduceren:

```
$truncation_text = '...'

def truncate text, length
        if length < $truncation_text.length then
                return text[0 .. length-1]
        end

        if text.length > length then
                return text[0..length-$truncation_text.length-1] + $truncation_text
        end

        return text
end
```

We voeren de tests nog een keer uit zodat eventuele gemaakte fouten zichtbaar worden.

```
6 tests, 6 assertions, 0 failures, 0 errors
```

Alles werkt zoals het hoort.