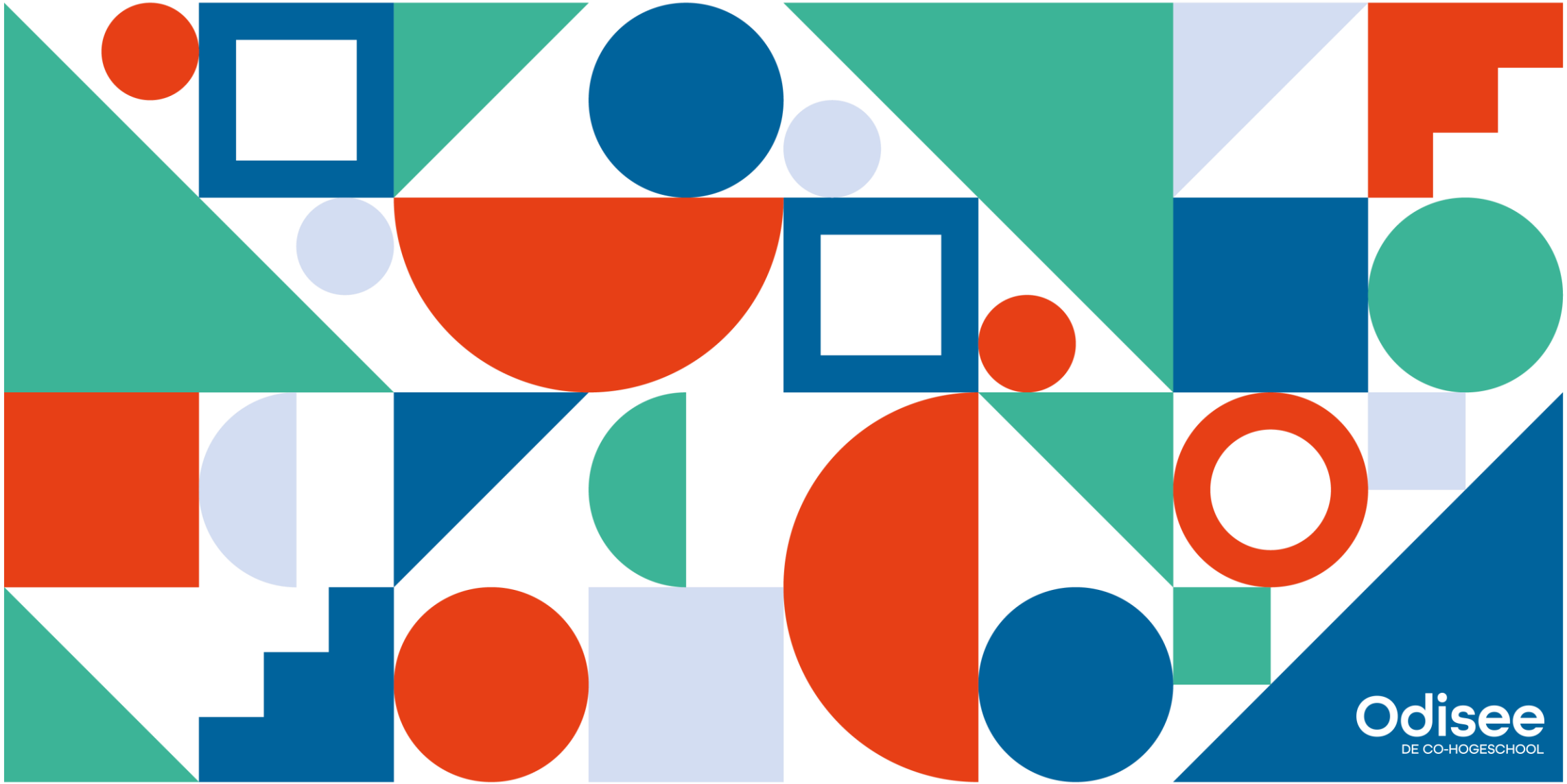

Odisee
DE CO-HOGESCHOOL

Code coverage

Jens Baetens

Wat is code coverage?

- Maatstaf voor hoeveel % van de code gedekt door automatische testen
- Metriek om inzicht te krijgen in de mate waarin
  - Je code getest wordt
  - De kwaliteit van de test cases
- Code coverage brengt niet geteste stukken code aan het licht
  - Helpt bij het opstellen van nieuwe test cases die nog niet getest worden

Metrics

Method	Cyclomatic complexity	NPath complexity	Sequence coverage	Branch coverage	Crap Score
Sum(...)	1	0	100%	100%	1

Vormen van code coverage

- Code coverage / test coverage
- Statement coverage
- Decision coverage
- Branch coverage
- Toggle coverage
- FSM coverage

Code coverage / Test Coverage

- Berekend als volgt:
  - $Code\ Coverage\ (test\ coverage) = \frac{Aantal\ lijnen\ code\ uitgevoerd}{Totaal\ aantal\ lijnen\ code}$
- Voorbeeld:

```
 public static bool
CanVote(int age)
  {
     if (age >= 18)
        return true;

     return false;
  }
```

```
 [Test]
 public void
Person_younger_than_18_cannot_vote()
  {
    int age = 16;

    bool canVote = Person.CanVote(age);

    Assert.IsFalse(canVote);
  }
```

Aantal lijnen: 5 (accolades tellen ook!)
Aantal uitgevoerd: 4
Code Coverage = 4/5 = 80%

Odisee
DE CO-HOGESCHOOL

Code coverage / Test Coverage

- Stel we herschrijven de functie als volgt

```
public static bool CanVote(int age)
 {
   return (age >= 18);
 }
```

- Is de code coverage erop vooruitgegaan?
- Is de kwaliteit erop vooruitgegaan?

Code coverage / Test Coverage

- Stel we herschrijven de functie als volgt

```
public static bool CanVote(int age)
 {
   return (age >= 18);
 }
```

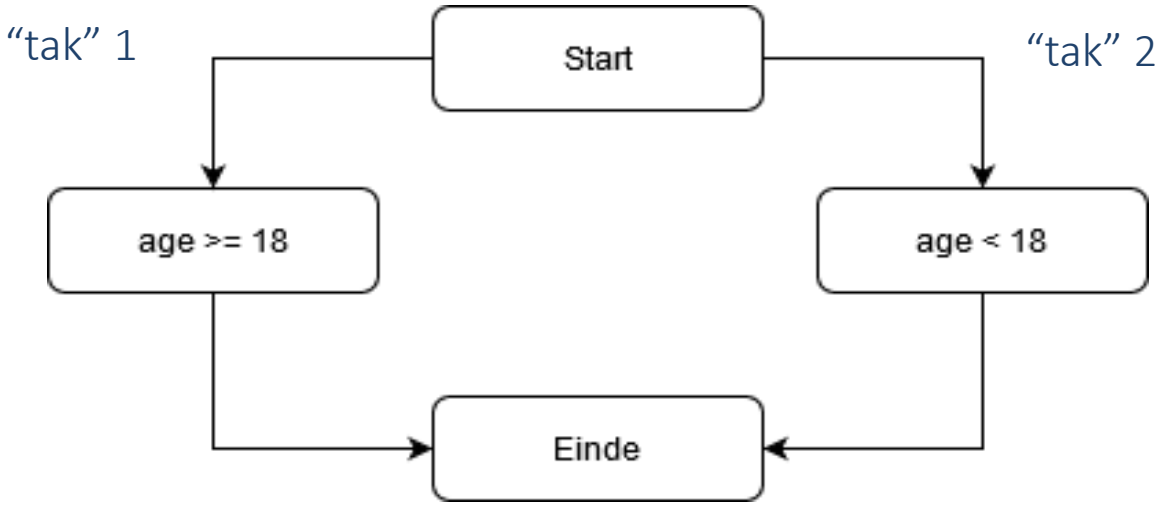
- Is de code coverage erop vooruitgegaan?
  - Ja: alle lijnen code uitgevoerd
- Is de kwaliteit erop vooruitgegaan?
  - Nee: Slechts 1 van de twee mogelijke outputs getest

Code coverage / Test Coverage

- Conclusie:
  - Zeer eenvoudig om “vals te spleen” met deze metriek
  - Hoe compacter je code, hoe hoger de code coverage
  - Maar: Compactere code, betekend niet dat je testen beter zijn!
- Gebruik het als hulpmiddel voor het vinden van nieuwe testcases
  - Niet als doel op zich
  - 100% wil niet zeggen dat er geen bugs zijn!
  - Niet omdat alle code uitgevoerd is dat je tests goed zijn.

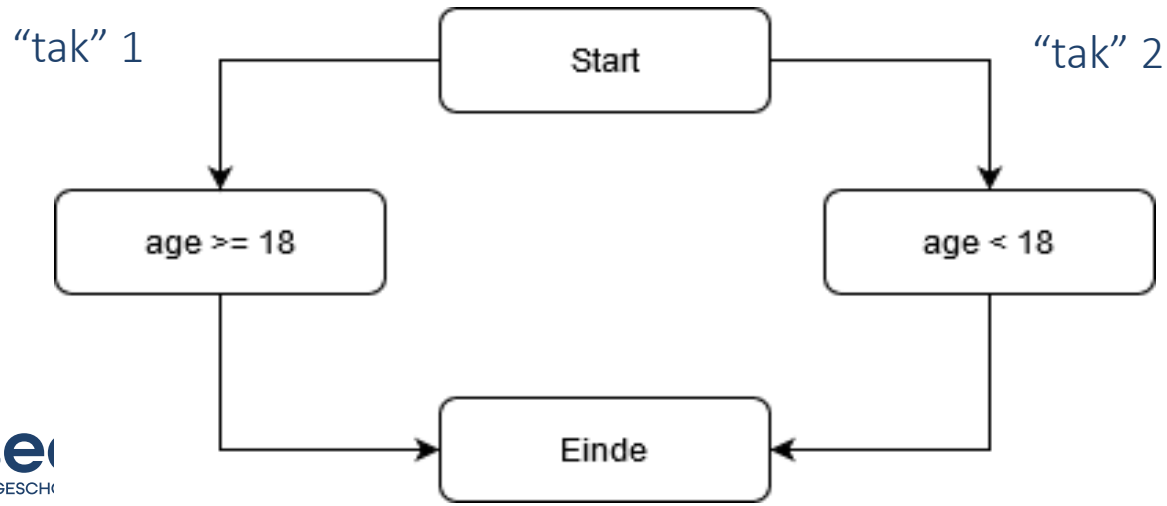
Branch Coverage

- Nauwkeuriger dan code coverage
  - In plaats van lijnen te tellen, tel taken in de code
  - Focus op selectie-structuren (if-else en switch-case)



Branch Coverage

- Metriek dat aangeeft hoeveel van de takken minstens 1 keer uitgevoerd warden
  - $Branch\ Coverage = \frac{Aantal\ uitgevoerde\ branches}{Totaal\ aantal\ branches}$
- Toegepast op het voorbeeld



Totaal aantal branches: 2
Aantal uitgevoerde branches: 1 *(enkel “tak 2”)*
Branch Coverage: 1/2 = 0,5 = 50%

Oefening – branch coverage

- Bereken de branch coverage van volgende test

```
 public double
CalculateReduction(double price)
 {
    if(price > 100)
    {
      return (price * 0.15);
    }
    else if(price > 50)
    {
      return (price * 0.10);
    }else if(price > 20)
    {
      return (price * 0.05);
    }

    return 0;
 }
```

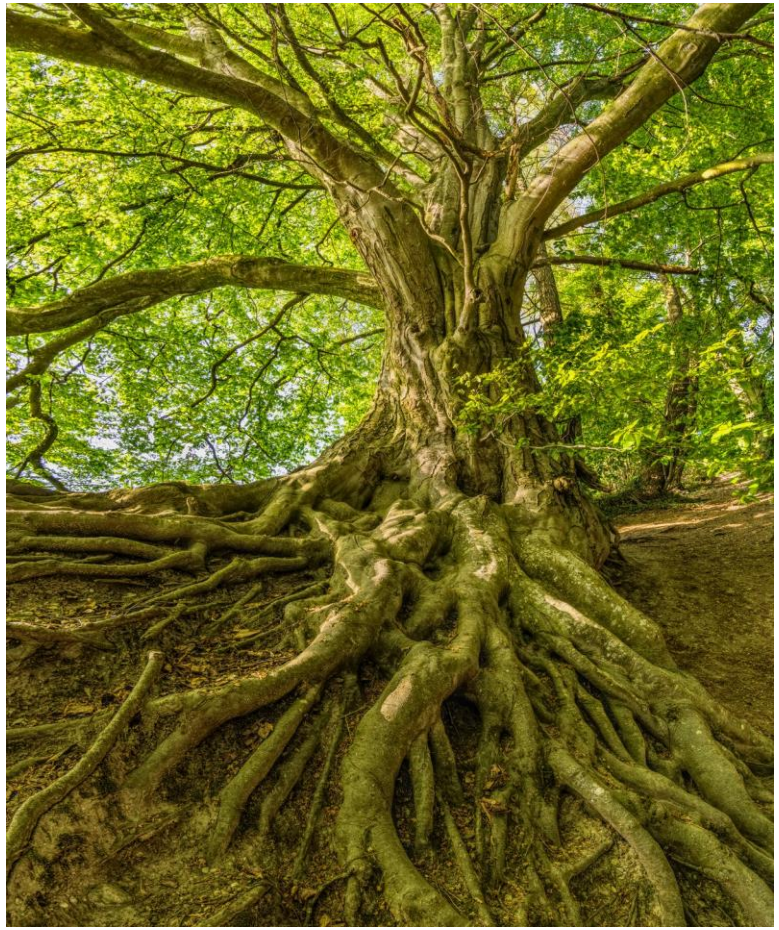
```
 [TestCase(200.0, ExpectedResult = 30.0)]
 [TestCase(150.0, ExpectedResult = 22.5)]
 [TestCase(100.0, ExpectedResult = 10.0)]
 public double Test_reduction_calculation(double
price)
 {
   ReductionCalculator sut = new
ReductionCalculator();

   double reduction =
sut.CalculateReduction(price);

   return reduction;
 }
```

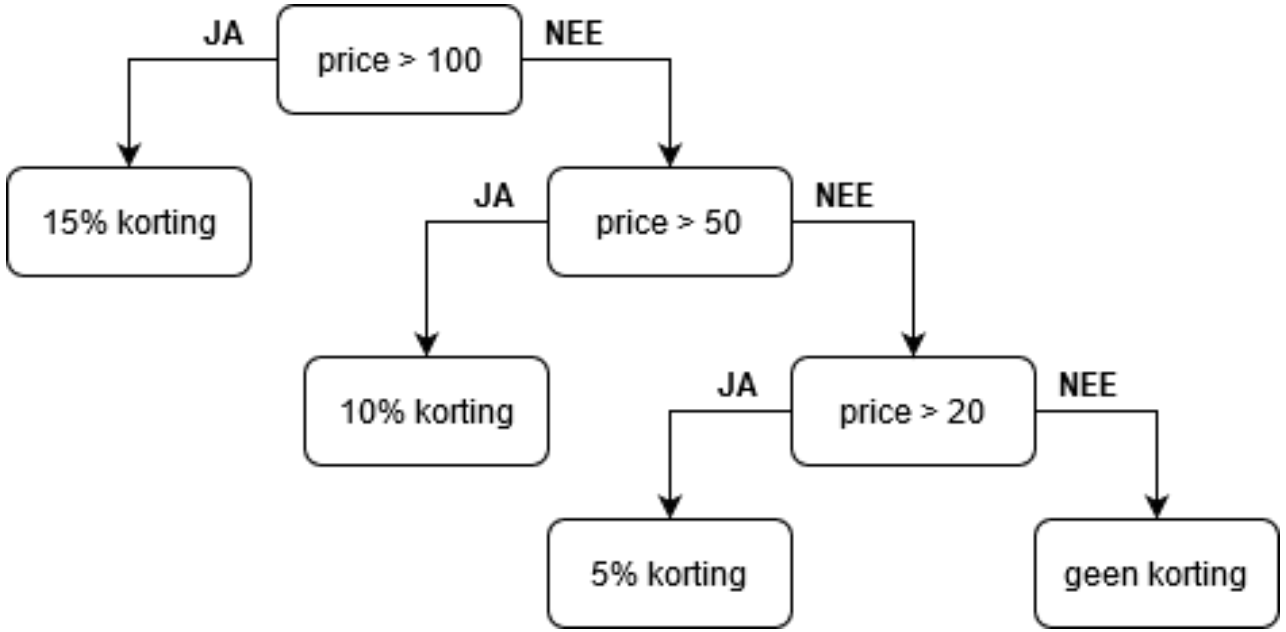
Branch coverage

- Stappenplan:
  - Teken een beslissingsboom die de verschillende branches weergeeft
  - Overloop de code manueel en duid de branches aan die door de code uitgevoerd worden
  - Tel het aantal uitgevoerde branches en deel dit door het totaal aantal branches



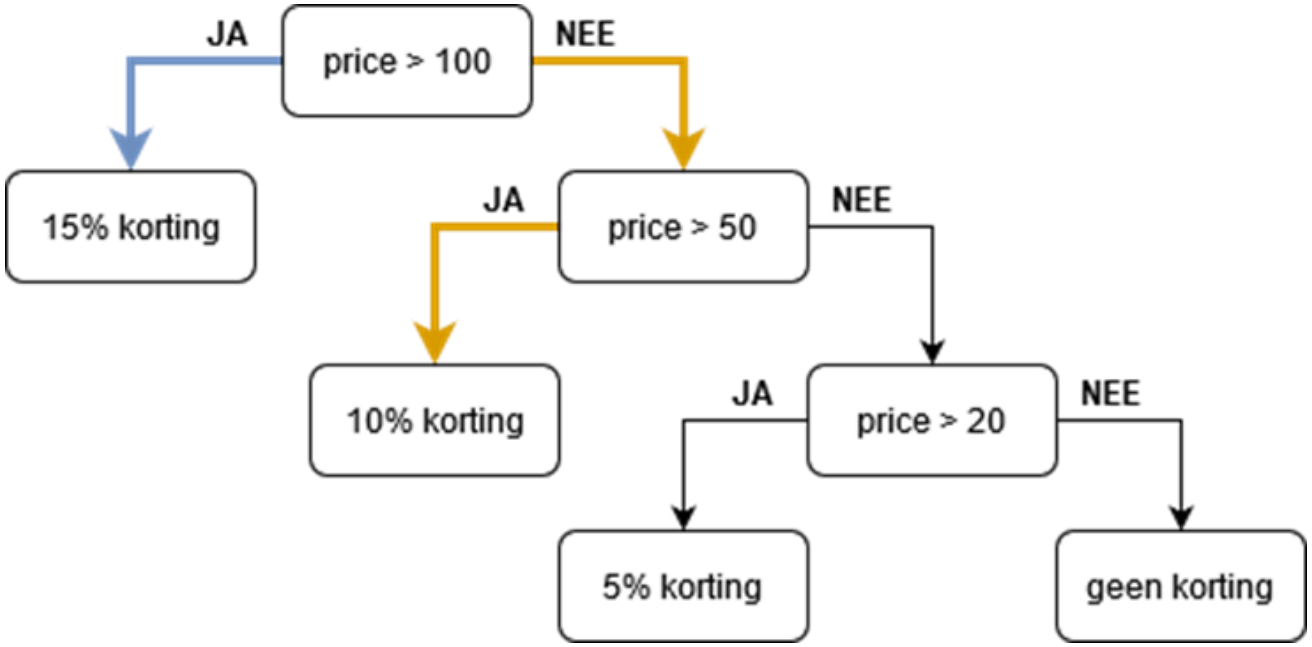
Branch Coverage

- Stap 1: Beslissingsboom



Branch Coverage

- Stap 2: uitgevoerde branches aanduiden



Branch coverage

- 3 branches uitgevoerd van de 6
  - Branch coverage is 50%

Class:	ConsoleCalculator.ReductionCalculator
Assembly:	ConsoleCalculator
File(s):	C:\Users\sam.vanbuggenhout\source\repos\ConsoleCalculator\ConsoleCalculator\Person.cs
Covered lines:	8
Uncovered lines:	4
Coverable lines:	12
Total lines:	37
Line coverage:	66.6% (8 of 12)
Covered branches:	3
Total branches:	6
Branch coverage:	50% (3 of 6)

Tips

- Enkel uitgaan van Code Cerage-technieken niet voldoende om de kwaliteit van de testen te beoordelen
- Stel geen percentage als doel
  - De technieken zijn een hulpmiddel, geen doel
- Lage code coverage kan de slechte kwaliteit van een test naar boven brengen
  - Hoge code coverage betekent echter niet dat de kwaliteit goed is