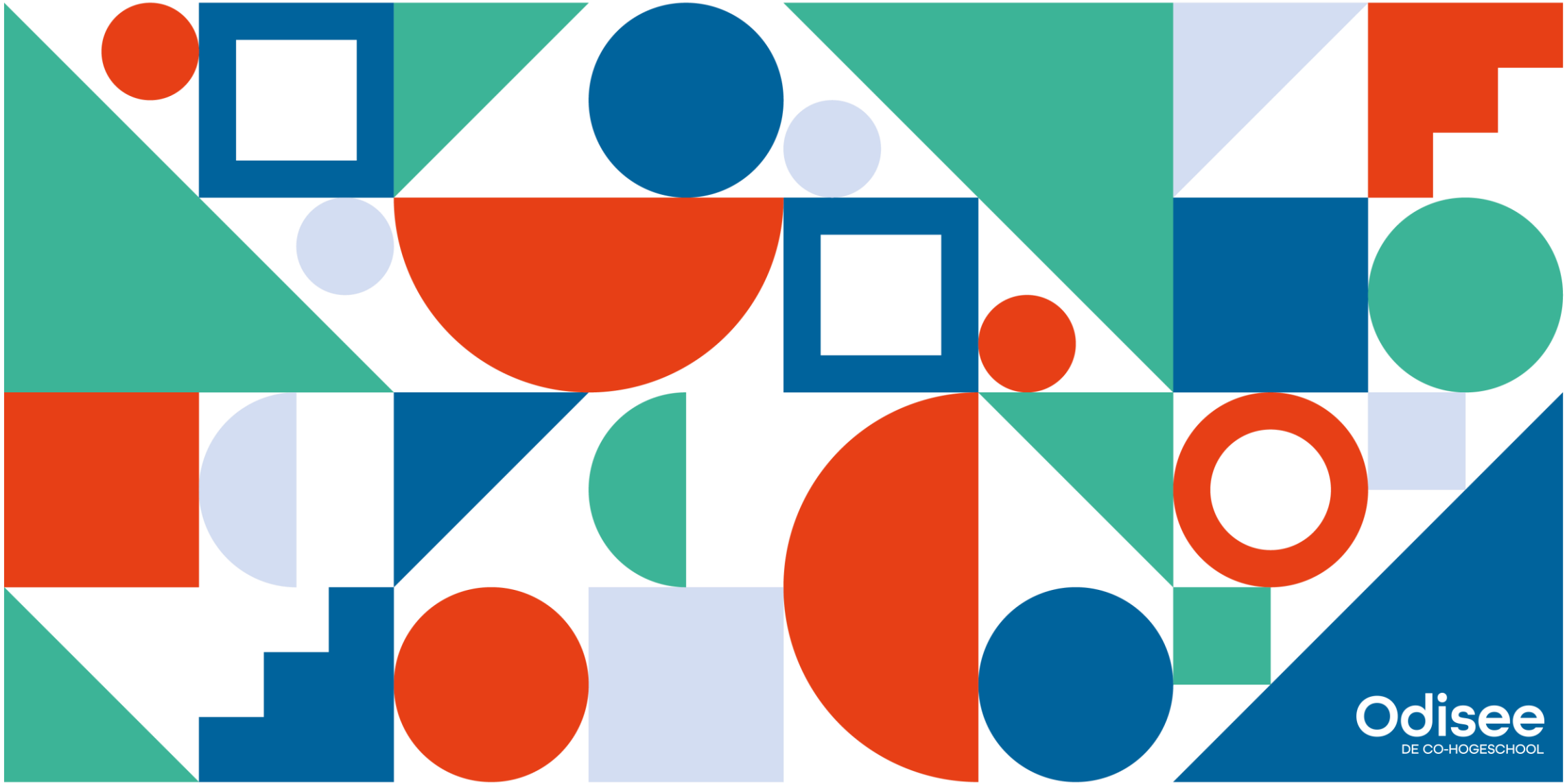

Odisee
DE CO-HOGESCHOOL

Test doubles

Jens Baetens

Situatieschets

Situatie

- Gevraagd: Schrijf unit tests voor het project PersonAgeValidator
- Volgende klasses hebben tests nodig
  - Person
  - AgeValidator

Oplossing: Wat valt op? Is er een problem?

```
[TestFixture]
0 references
class AgeValidatorTests
{
    //Age valid
    [TestCase(18)]
    [TestCase(70)]
    [TestCase(36)]
    0 references
    public void IsValidAge_ValidAges_ReturnTrue(int age)
    {
        //Arrange
        AgeValidator ageValidator = new AgeValidator();

        //Act
        bool result = ageValidator.IsValidAge(age);

        //Assert
        Assert.IsTrue(result);
    }

    //Age not valid
    [TestCase(17)]
    [TestCase(71)]
    [TestCase(10)]
    [TestCase(80)]
    0 references
    public void IsValidAge_InvalidAges_ReturnFalse(int age)
    {
        //Arrange
        AgeValidator ageValidator = new AgeValidator();

        //Act
        bool result = ageValidator.IsValidAge(age);

        //Assert
        Assert.IsFalse(result);
    }
}
```

```
[TestFixture]
0 references
public class PersonTests
{
    // Test person created (valid age)
    [TestCase(18)]
    [TestCase(70)]
    [TestCase(36)]
    0 references
    public void Ctor_ValidAges_PersonCreated(int age)
    {
        //Arrange
        string firstname = "John";
        string lastname = "Doe";

        //Act
        Person person = new Person(firstname, lastname, age);

        //Assert
        Assert.That(person.FirstName, Is.EqualTo(firstname));
        Assert.That(person.LastName, Is.EqualTo(lastname));
        Assert.That(person.Age, Is.EqualTo(age));
    }

    // Test person not created (invalid age/ exception throwed)
    [TestCase(17)]
    [TestCase(71)]
    [TestCase(10)]
    [TestCase(80)]
    0 references
    public void Ctor_InvalidAges_ThrowsException(int age)
    {
        //Arrange
        string firstname = "John";
        string lastname = "Doe";

        //Act is de new person in de assert

        //Assert
        Assert.Throws<Exception>(() => new Person(firstname, lastname, age)
    }
```

Conclusies

- Code is dubbel getest
  - In principe niet slecht maar kan voor problemen zorgen als tests elkaar tegenspreken
- Gedrag van AgeValidator verandert
  - Ook Unit tests voor Person aanpassen
  - De bedoeling dat een test-klasse de logica van 1 klasse test en niet van de afhankelijkheden
    - Dependency onafhankelijk

Conclusie Concreet

- AgeValidatorTests
  - True: leeftijd tussen 17 en 71
  - False: leeftijd groter dan of gelijk aan 71
  - False: leeftijd kleiner dan of gelijk aan 17
- UserTests
  - User object aangemaakt wanneer leeftijd correct
  - Exception gethrowed wanneer leeftijd incorrect

Test Doubles

- Worden gebruikt om afhankelijkheden na te bootsen
  - Om logica in isolatie te testen
  - Manipuleren van afhankelijkheden om externe factoren te negeren
- Types
  - Stubs
  - Fakes
  - Mocks
  - Spies
  - Dummies

Test doubles in real life

- Brandweer
  - Bij brand stuur brandweerwagen
  - Bij ongeval stuur ziekenwagen
  - Bij wespennest wordt het opgenomen in de planning
- Bij een oefening zal er geen echte brand of ongeval zijn en wordt de telefoon en planning gesimuleerd om verschillende scenario's te testen

- Gelijkaardig in code, we bootsen de externe factoren na (dependencies) om alle outcomes te testen

Wat is een Test Double?

- Een klasse die dezelfde publieke interface heeft als een andere klasse
  - Vandaar double
  - Hiervoor worden interfaces gebruikt

- Zien er hetzelfde uit vanuit een andere klasse gekeken maar bevatten een andere logica
  - Doel: meer controle tijdens testen

Wanneer test doubles gebruiken

- Wanneer er andere systemen aangesproken worden
  - Unit tests zijn kleine en korte tests
  - Niet de bedoeling dat een database, externe server gecontacteerd wordt
- Logica van een klasse afhankelijk van logica in een andere klasse

- Niet nodig om voor elke afhankelijkheid een test double te maken
  - Bijvoorbeeld een data klasse die frequent gebruikt wordt

Casus

Project: GradesHelper

- Klasse met enkele veel gebruikte methodes
  - Van toepassing op scores van studenten

- Maakt reeds gebruik van dependency injection

```
2 references
public class GradesHelper
{
    private readonly IGradeRepository gradeRepository;

    0 references
    public GradesHelper(IGradeRepository gradeRepository)
    {
        this.gradeRepository = gradeRepository;
    }

    0 references
    public GradesHelper()
    {
        this.gradeRepository = new GradeRepository();
    }
```

Project: GradesHelper

- Maakt gebruik van interface IGradeRepository voor scores
- Standaard data klasse voor student

```
3 references
public interface IGradeRepository
{
    2 references
    List<int> GetGrades(Student student);
    3 references
    void AddScore(Student student, int score);
    1 reference
    int GetTotalScore(Student student);
    2 references
    void ClearScore(Student student);
}
```

```
public class Student
{
    [Key]

    public int Id { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }


    public List<int> Scores { get; set; }
}
```

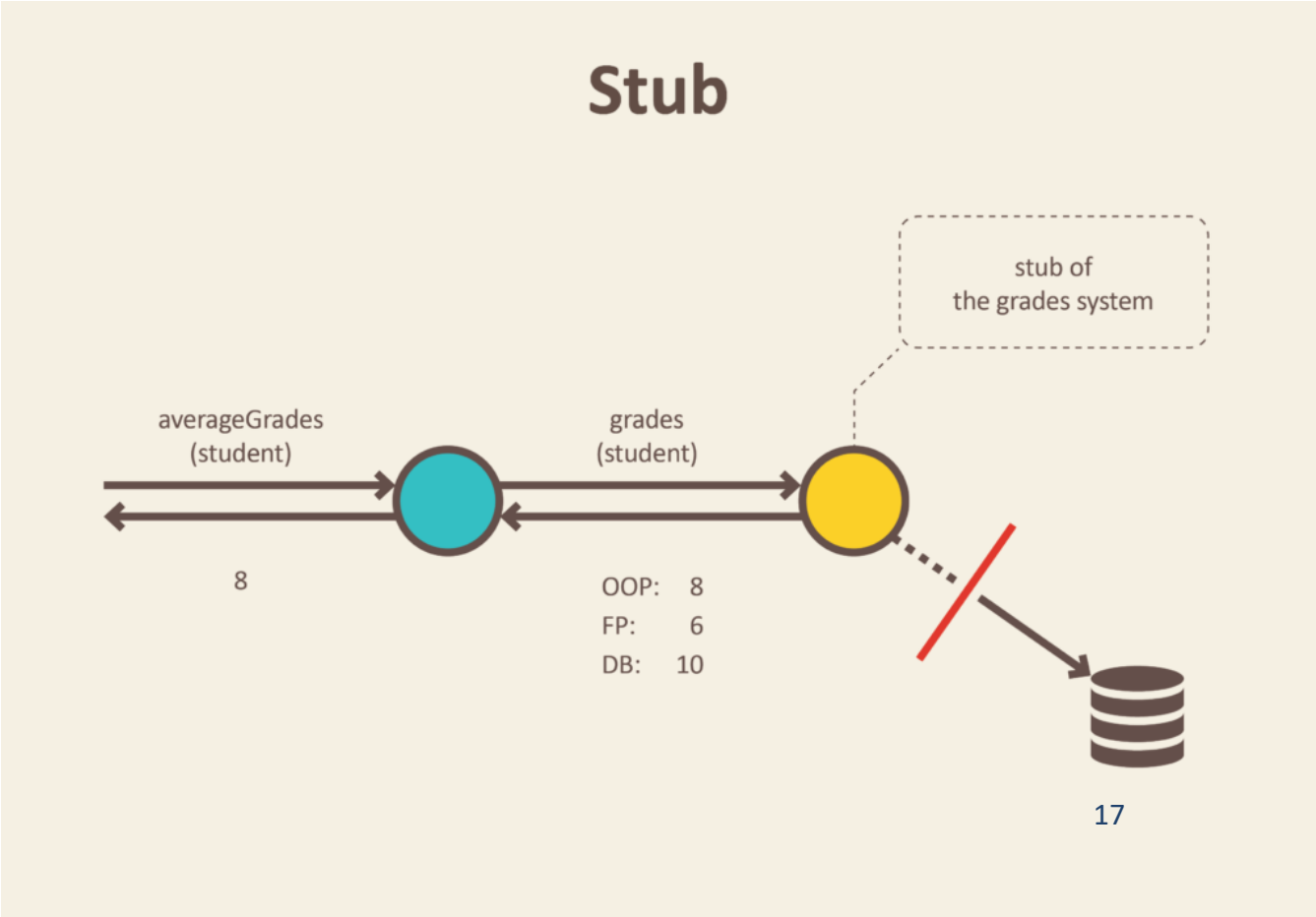
Project: GradesHelper

- GradeRepository
  - De klasse om scores op te halen uit de database
  - Testen voor gradeshelper mogen deze klasse niet gebruiken
    - Moet systeem onafhankelijk zijn
  - Test double nodig

Stubs

Stub

- Stub is een implementatie van een interface die hard-coded waarden teruggeeft
  - Maakt het mogelijk specifieke scenario's te testen



Stubs voorbeeld

- Te testen methode: CalcAverageGrade(Student student)

```
2 references
public double CalcAverageGrade(Student student)
{
    List<int> grades = gradeRepository.GetGrades(student);

    double totalScore = 0;

    foreach (var grade in grades)
    {
        totalScore += grade;
    }

    return totalScore / grades.Count;
}
```

Voorbeeld Stub

- Hard coded list

```
1 reference
class GradeRepositoryStub : IGradeRepository
{
    private List<int> scores = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

    3 references
    public void AddScore(Student student, int score)
    {
        throw new NotImplementedException();
    }

    2 references
    public void ClearScore(Student student)
    {
        throw new NotImplementedException();
    }

    2 references
    public List<int> GetGrades(Student student)
    {
        return scores;
    }

    1 reference
    public int GetTotalScore(Student student)
    {
        throw new NotImplementedException();
    }
}
```

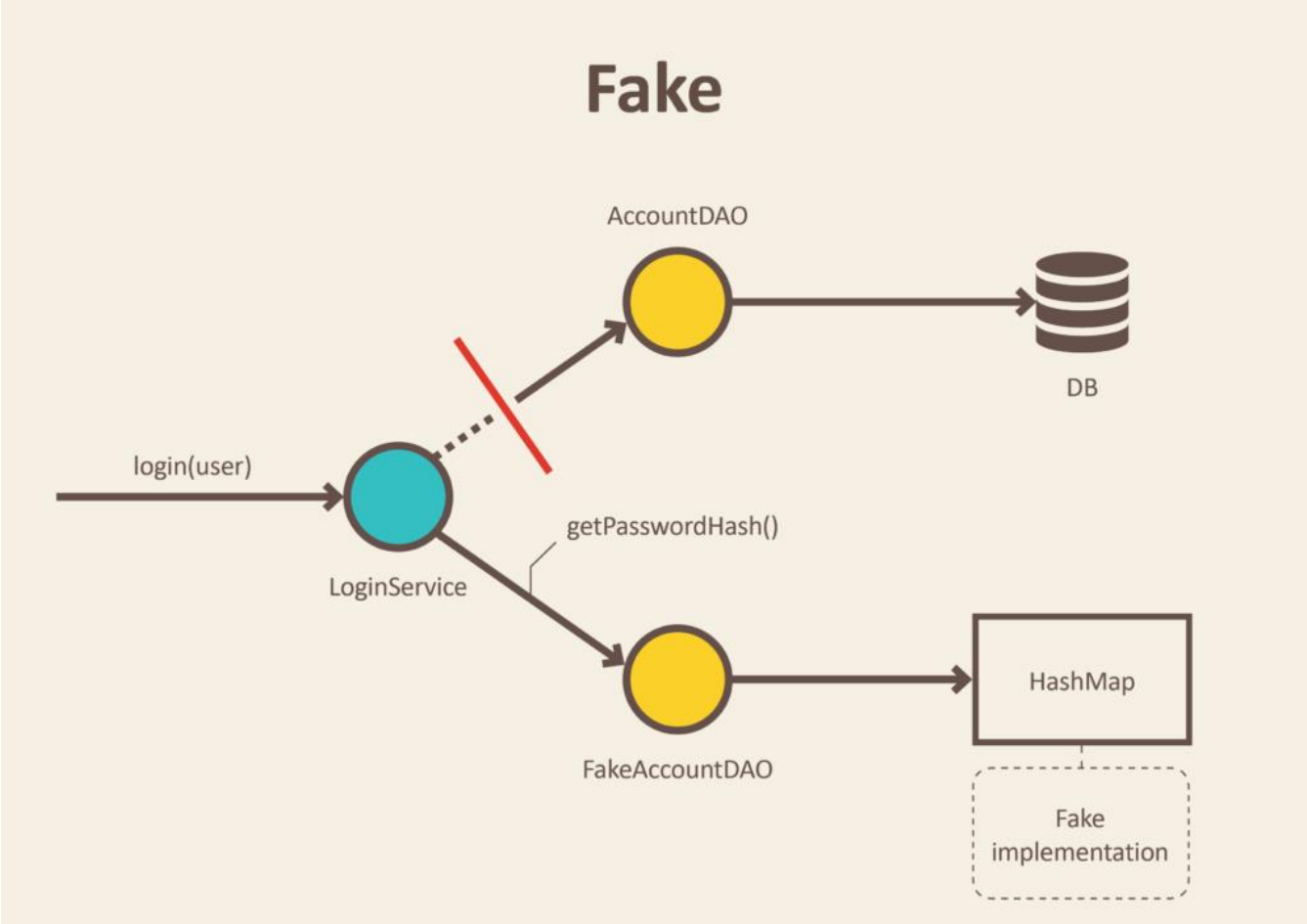
Gebruik stub

```
[Test]
0 references
public void CalcAverageGrade_SomeNumbersEntered_ReturnsAverage()
{
    //Arrange
    //Stub: fixed hard coded values that cannot be changed
    IGradeRepository stub = new GradeRepositoryStub();
    GradesHelper sut = new GradesHelper(stub);
    //Student s can be zero because s.GetGrades() or s.Scores is not done (stubbed)
    Student s = null;

    //Act
    double average = sut.CalcAverageGrade(s);

    //Assert
    Assert.AreEqual(average, 5);
}
```

Fakes

Fakes

- Gelijkaardig aan een stub
- lets meer logica
- Vooral indien er meerdere acties uitgevoerd worden op 1 item



Fakes voorbeeld

- Te testen methode: DidStudentPerformBetterWithNewScore(…)

```
2 references
public bool DidStudentPerformBetterWithNewScore(Student s, int score)
{
    double oldAvgScore = CalcAverageGrade(s);

    gradeRepository.AddScore(s, score);

    double newAvgScore = CalcAverageGrade(s);

    return newAvgScore > oldAvgScore;
}
```

Fakes voorbeeld

```
2 references
class GradeRepositoryFake : IGradeRepository
{
    private List<int> scores = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

    3 references
    public void AddScore(Student student, int score)
    {
        scores.Add(score);
    }

    2 references
    public void ClearScore(Student student)
    {
        throw new NotImplementedException();
    }

    2 references
    public List<int> GetGrades(Student student)
    {
        return scores;
    }

    1 reference
    public int GetTotalScore(Student student)
    {
        throw new NotImplementedException();
    }
}
```

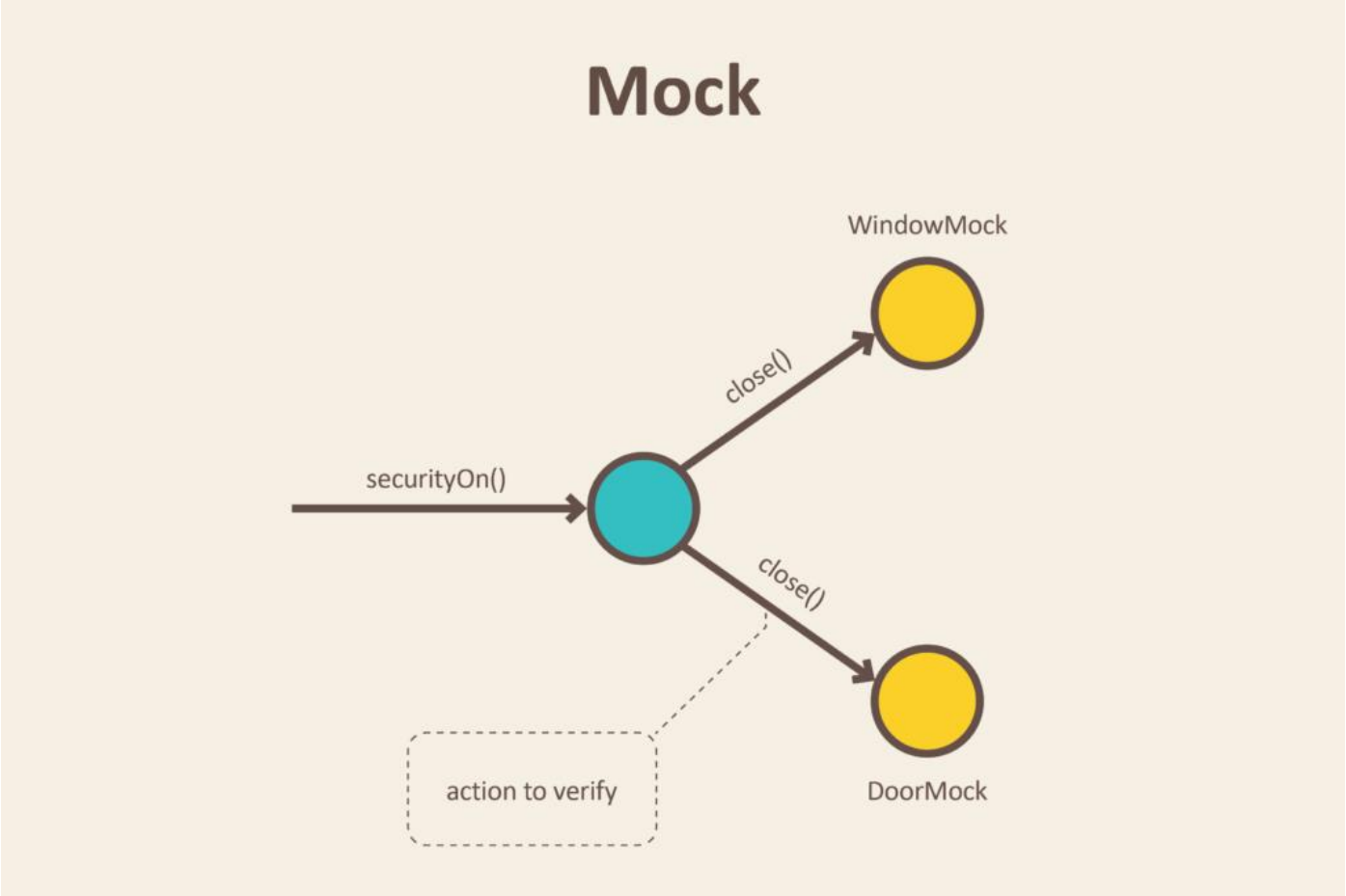
Fakes voorbeeld

```
[Test]
0 references
public void DidStudentPerformBetterWithNewScore_WithBadScore_ReturnsFalse()
{
    //Arrange
    //The function DidStudentPerformBetterWithNewScore voegt iets toe aan de score lijst
    //Dit vereist extra logica om de state bij te houden en aan te passen (AddScore)
    //Dit wordt fake genoemd
    IGradeRepository fake = new GradeRepositoryFake();
    GradesHelper sut = new GradesHelper(fake);
    Student s = null;

    //Act
    bool result = sut.DidStudentPerformBetterWithNewScore(s, 0);

    //Assert
    Assert.IsFalse(result);
}
```

Mocks

Mocks

- Gebruikt om te controleren of gewenst gedrag uitgevoerd is



Mocks voorbeeld

- Te testen methode: RemoveAllScores(Student student)

```
1 reference
public void RemoveAllScores(Student student)
{
    gradeRepository.ClearScore(student);
}
```

Mocks voorbeeld

```
1 reference
class GradeRepositoryMock : IGradeRepository
{
    3 references
    public void AddScore(Student student, int score)
    {
        throw new NotImplementedException();
    }

    2 references
    public void ClearScore(Student student)
    {
        //als we hier komen is het goed, test stopt hier
        Assert.Pass();
    }

    2 references
    public List<int> GetGrades(Student student)
    {
        throw new NotImplementedException();
    }

    1 reference
    public int GetTotalScore(Student student)
    {
        throw new NotImplementedException();
    }
}
```

Mocks voorbeeld

```
[Test]
0 references
public void RemoveAllScores_ClearScoreIsCalled()
{
    //Arrange
    IGradeRepository mock = new GradeRepositoryMock();
    GradesHelper sut = new GradesHelper(mock);
    Student s = null;

    //Act
    sut.RemoveAllScores(s);

    //Assert
    Assert.Fail();
}
```

Spies

Spies

- Verschil tussen spies en mocks is klein
- Mocks testen direct of iets correct is
- Spies houden data bij en controleren of gedrag correct was in de unit test

Spies voorbeeld

- Te testen methode: AddScore(…)

```
2 references
public void AddScore(Student student, int score)
{
    if (score < 0)
    {
        throw new Exception();
    }

    gradeRepository.AddScore(student, score);
}
```

Spies voorbeeld

```
5 references
class GradeRepositorySpy : IGradeRepository
{
    4 references
    public bool AddScoreIsCalled { get; private set; }
    2 references
    public int LatestAddedScoreIs { get; private set; }

    2 references
    public GradeRepositorySpy()
    {
        AddScoreIsCalled = false;
    }

    3 references
    public void AddScore(Student student, int score)
    {
        AddScoreIsCalled = true;
        LatestAddedScoreIs = score;
    }

    2 references
    public void ClearScore(Student student)
    {
        throw new NotImplementedException();
    }

    2 references
    public List<int> GetGrades(Student student)
    {
        throw new NotImplementedException();
    }

    1 reference
    public int GetTotalScore(Student student)
    {
        throw new NotImplementedException();
    }
}
```

Odisee
DE CO-HOGESCHOOL

Spies voorbeeld

```
[Test]
0 references
public void AddScore_WithValidData_AddScoreIsCalled()
{
    //Arrange
    //Spy must be GradeRepositorySpy, otherwise we cannot do spy.AddScoreIsCalled
    GradeRepositorySpy spy = new GradeRepositorySpy();
    GradesHelper sut = new GradesHelper(spy);
    Student s = null;

    //Act
    sut.AddScore(s, 5);

    //Assert
    Assert.IsTrue(spy.AddScoreIsCalled);
    Assert.That(spy.LatestAddedScoreIs, Is.EqualTo(5));
}
```

Spies voorbeeld

```
[Test]
0 references
public void AddScore_WithInvalidData_AddScoreNotCalled()
{
    //Arrange
    //Spy must be GradeRepositorySpy, otherwise we cannot do spy.AddScoreIsCalled
    GradeRepositorySpy spy = new GradeRepositorySpy();
    GradesHelper sut = new GradesHelper(spy);
    Student s = null;

    //Act
    Assert.Throws<Exception>( () =>
        { sut.AddScore(s, -5); }
    );

    //Assert
    Assert.IsFalse(spy.AddScoreIsCalled);
}
```

Dummies

Dummies

- Waarden waar we niet veel om geven
- Reeds gebruikt in de Arrange sectie van een unit test
  - Student variabele die null is
- Meestal meegegeven als argument

Status valideren?

- Een waarde controleren wanneer we iets gedaan hebben
- Status bestaat uit meerdere waarden
  - Oppassen dat je niet de verkeerde zaken test/controleert

```
given:
Car testedCar = new Car()

when:
testedCar.setSpeed(40)

then:
testedCar.getSpeed() == 40
```

```
given:
Car testedCar = new Car()

when:
testedCar.setSpeed(40)

then:
testedCar.getSpeed() == 40
testedCar.getGear() == 2
testedCar.getTachometer().getValue() == 2000
```

Gedrag valideren?

- Controleren of een bepaald gedrag uitgevoerd is
- Vaak synoniem voor gebruik mocks
- Pas op dat je niet te gedetailleerd valideert

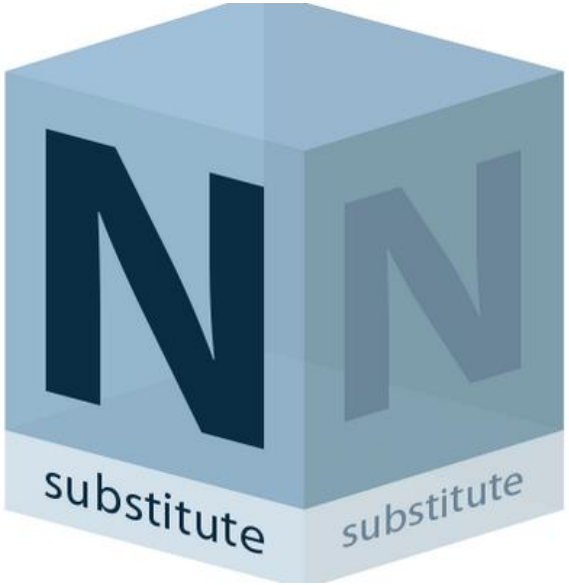
```
public void ChargeCustomer(CustomerId customerId,
                           IList<Product> products)
{
    var customer = customerRepository.Find(customerId);
    var invoice = CreateInvoice(customer, products);
    invoiceRepository.Save(invoice);
    mailService.SendInvoice(customer, invoice);
}

private Invoice CreateInvoice(Customer customer, IList<Product> products)
{
    var invoice = new Invoice(customer.Id);

    // Do something interesting with products here...

    return invoice;
}
```

NSubstitute

NSubstitute

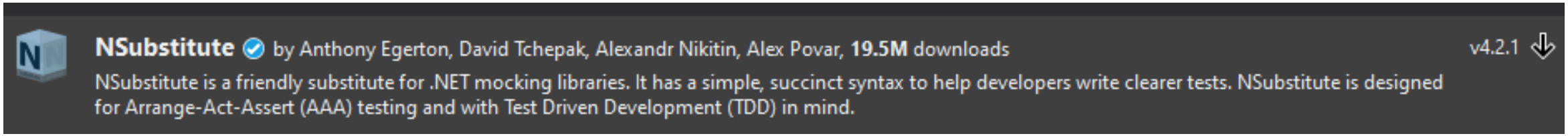
- Mocking framework
  - Steeds aanmaken mocks, spies, stubs, fakes zorgt voor veel overhead
  - Doen dit in de achtergrond zodat wij het niet moeten doen

- Andere mogelijkheden
  - Moq
  - RhinoMocks
  - FakeItEasy

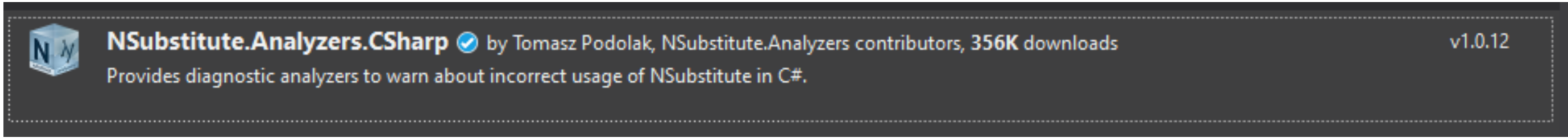


NSubstitute

- Gebruik NSubstitute door de volgende NuGet packages toe te voegen
  - Nsubstitute
    - Helpt het aanmaken van de test doubles



  - Nsubstitute.Analyzers.Csharp
    - Analyseren of we NSubstitute correct gebruiken



NSubstitute

- Mock, Spy, fake of stub?
  - Alles is een Substitute
  - Een voorbeeld voor de verschillende types gaan we geven

- Aanmaken substitute

```
IGradeRepository stub = Substitute.For<IGradeRepository>();
GradesHelper gradesHelper = new GradesHelper(stub);
```

Gebruik NSubstitute

- Subsitute.For<T>
  - T is interface of klasse
    - Bij klasse kan je enkel werken met virtual methodes (zie overerving)
    - Probeer klassen te vermijden en geef de voorkeur aan interfaces

```
IGradeRepository stub = Substitute.For<IGradeRepository>();
GradesHelper gradesHelper = new GradesHelper(stub);
```

Stub

- Teruggeven default values voor bepaalde methoden

```
//Arrange
IGradeRepository stub = Substitute.For<IGradeRepository>();
GradesHelper gradesHelper = new GradesHelper(stub);

Student s = null;
stub.GetGrades(s).Returns(new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 });
```

- Telkens wanneer GetGrades opgeroepen wordt met de parameter student wordt de lijst teruggegeven

Stubs voorbeeld

```
2 references
public double CalcAverageGrade(Student student)
{
    List<int> grades = gradeRepository.GetGrades(student);

    double totalScore = 0;

    foreach (var grade in grades)
    {
        totalScore += grade;
    }

    return totalScore / grades.Count;
}
```

```
[Test]
0 references
public void CalcAverageGrade_SomeNumbersEntered_ReturnsAverage()
{
    //Arrange
    IGradeRepository stub = Substitute.For<IGradeRepository>();
    GradesHelper gradesHelper = new GradesHelper(stub);
    Student student = null;
    List<int> grades = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    stub.GetGrades(student).Returns(grades);

    //Act
    double average = gradesHelper.CalcAverageGrade(student);

    //Assert
    Assert.AreEqual(average, 5);
}
```

Stubs – more specific returns

```
[Test]
0 references
public void CalcAverageGrade_SomeNumbersEntered_ReturnsAverage()
{
    //Arrange
    IGradeRepository stub = Substitute.For<IGradeRepository>();
    GradesHelper gradesHelper = new GradesHelper(stub);

    Student student = new Student();
    Student student2 = new Student();
    List<int> grades = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    List<int> grades2 = new List<int>() {8,8,8,8,8 };
    stub.GetGrades(student).Returns(grades);
    stub.GetGrades(student2).Returns(grades2);

    //Act
    double average = gradesHelper.CalcAverageGrade(student);
    double average2 = gradesHelper.CalcAverageGrade(student2);

    //Assert
    Assert.AreEqual(average, 5);
    Assert.AreEqual(average2, 8);
}
```

Stubs – return for all values

- Default parameter met ReturnsForAnyArgs
- Arg.Any<Student> parameter

```
[Test]
0 references
public void CalcAverageGrade_SomeNumbersEntered_ReturnsAverage()
{
    //Arrange
    IGradeRepository stub = Substitute.For<IGradeRepository>();
    GradesHelper gradesHelper = new GradesHelper(stub);

    List<int> grades = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    stub.GetGrades(default).ReturnsForAnyArgs(grades);
    stub.GetGrades(Arg.Any<Student>()).Returns(grades);

    //Act
    double average = gradesHelper.CalcAverageGrade(null);

    //Assert
    Assert.AreEqual(average, 5);
}
```

Stubs – multiple return values

- Meerdere return values ook mogelijk door meerdere parameters terug te geven in de Returns methode
  - Laatste wordt steeds herhaald

```
[Test]
0 references
public void CalcAverageGrade_SomeNumbersEntered_ReturnsAverage()
{
    //Arrange
    IGradeRepository stub = Substitute.For<IGradeRepository>();
    GradesHelper gradesHelper = new GradesHelper(stub);

    List<int> grades = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    List<int> grades2 = new List<int>() { 8, 8, 8, 8 };
    stub.GetGrades(default).ReturnsForAnyArgs(grades, grades2);

    //Act
    double average = gradesHelper.CalcAverageGrade(null);
    double average2 = gradesHelper.CalcAverageGrade(null);
    double average3 = gradesHelper.CalcAverageGrade(null);

    //Assert
    Assert.AreEqual(average, 5);
    Assert.AreEqual(average2, 8);
    Assert.AreEqual(average3, 8);
}
```

Spies

- Met de Received methode controleren we of een methode is uitgevoerd en hoe veel keer
- DidNotReceive() test of een methode niet is uitgevoerd

Spy voorbeeld

```
[Test]
0 references
public void AddScore_WithValidData_AddScoreIsCalled()
{
    //Arrange
    IGradeRepository spy = Substitute.For<IGradeRepository>();
    GradesHelper sut = new GradesHelper(spy);
    Student s = null;
    int score = 5;

    //Act
    sut.AddScore(s, score);

    //Assert
    spy.Received(1).AddScore(s, score);
}
```

```
[Test]
0 references
public void AddScore_WithInvalidData_AddScoreNotCalled()
{
    //Arrange
    IGradeRepository spy = Substitute.For<IGradeRepository>();
    GradesHelper sut = new GradesHelper(spy);
    Student s = null;
    int score = -5;

    //Act
    Assert.Throws<Exception>(() =>
        { sut.AddScore(s, score); }
    );

    //Assert
    spy.DidNotReceive().AddScore(s, score);
}
```

Spy parameters

- Net zoals bij stubs zijn er parameters voor default waarden:
  - Default of Arg.Any<Student>
  - Indien je niet geinteresseerd bent met welke parameters de functie is opgeroepen

```
[Test]
0 references
public void AddScore_WithValidData_AddScoreIsCalled()
{
    //Arrange
    IGradeRepository spy = Substitute.For<IGradeRepository>();
    GradesHelper sut = new GradesHelper(spy);
    Student s = null;
    int score = 5;

    //Act
    sut.AddScore(s, score);

    //Assert
    spy.Received(1).AddScore(Arg.Any<Student>(), default);
}
```

Mocks

- Maakt gebruik van
  - .When() een functie uitgevoerd
  - .Do() laat het slagen
- In de praktijk kies je rapper voor een spy

```
[Test]
0 references
public void RemoveAllScores_ClearScoreIsCalled()
{
    //Arrange
    IGradeRepository mock = Substitute.For<IGradeRepository>();
    GradesHelper sut = new GradesHelper(mock);
    Student s = null;
    mock.When(x => x.ClearScore(s)).Do(x => Assert.Pass());

    //Act
    sut.RemoveAllScores(s);

    //Assert
    Assert.Fail();
}
```

Fakes

- Met behulp van functies/ lambdas in .Returns() kan je fakes maken
- State in de Substitute kan ook aangepast worden met
  - .When() en .Do()

Fakes

```
[Test]
0 references
public void DidStudentPerformBetterWithNewScore_WithBadScore_ReturnsTrue()
{
    //Arrange
    IGradeRepository fake = Substitute.For<IGradeRepository>();
    GradesHelper sut = new GradesHelper(fake);
    Student s = null;
    int score = 10;
    List<int> grades = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

    fake.When(x => x.AddScore(s, score)).Do(x => grades.Add(x.ArgAt<int>(1)));
    fake.GetGrades(s).Returns(x=> grades);

    //Act
    bool result = sut.DidStudentPerformBetterWithNewScore(s, score);

    //Assert
    Assert.IsTrue(result);
}
```

Fakes – Dynamische waarden

- Let op dat je een lambda functie gebruikt!
  - Anders de originele waarde gebruikt

```
List<int> grades = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
fake.When(x => x.AddScore(s, score)).Do(x => grades.Add(x.ArgAt<int>(1)));
fake.GetGrades(s).Returns(grades);
```

Returns originele grades

```
fake.When(x => x.AddScore(s, score)).Do(x => grades.Add(x.ArgAt<int>(1)));
fake.GetGrades(s).Returns(x=> grades);
```

Returns geüpdatete grades

Multiple interfaces

- Een klasse kan meerdere interfaces implementeren
  - Een Substitute kan dit dus ook

```
var substitute = Substitute.For<IGradeRepository, IList>();
```

- Deze substitute heeft “implementaties” voor de methodes van zowel
  - IGradeRepository
  - IList

Oefening

Oefening - uitdaging

- Schrijf een test voor de methode
  - MakeAGuess

```
3 references
public class HigherLower
{
    int number;

    1 reference
    public HigherLower()
    {
        Random random = new Random();
        number = random.Next(100);
    }

    1 reference
    public string MakeAGuess(int guess)
    {
        if (number == guess)
        {
            return "Correct";
        }
        else if (number < guess)
        {
            return "Lower";
        }
        else
        {
            return "Higher";
        }
    }
}
```

Oplossing

- Voor random bestaat geen interface of test-doubles van te maken
  - Onze tests kunnen niet afhangen van een willekeurige waarde
  - Oplossing: Maak een wrapper klasse met een interface die we wel kunnen controleren

```
2 references
public interface IRandom
{
    2 references
    int Next(int maxValue);
}
```

```
1 reference
class RandomWrapper : IRandom
{
    Random random = new Random();
    2 references
    public int Next(int maxValue)
    {
        return random.Next(maxValue);
    }
}
```

Resulterende test

```
[Test]
0 references
public void MakeAGuess_WithLowNumber_ReturnsHigher()
{
    //Arrange
    IRandom random = Substitute.For<IRandom>();
    random.Next(Arg.Any<int>()).Returns(50);
    HigherLower higherLower = new HigherLower(random);
    int guess = 10;

    //Act
    string result = higherLower.MakeAGuess(guess);

    //Assert
    Assert.That(result, Is.EqualTo("Higher"));

}
```

DbContext

Mocken/Stubben DbContext

- Testen dataRepositories en geen data willen wegschrijven naar een database
  - DbContext mocken/stubben maar dit is geen interface
- Maak gebruik van virtual properties en methods
  - Kunnen bij overerving overschreven worden
  - Kunnen gemockt/gestubt worden

```
5 references
public class StudentsDbContext: DbContext
{
    1 reference
    public StudentsDbContext():base("students")
    {

    }

    5 references
    public virtual DbSet<Student> Students { get; set; }
}
```

Odisee
DE CO-HOGESCHOOL

DbContext mocken

- Substitute voor DbContext kan gemaakt worden
  - Geef een stub terug voor users
  - Probleem: Constructor DbSet lukt niet
    - Maak hier ook een substitute voor

Deze code werkt niet.

```
//Arrange
var dbContext = Substitute.For<StudentsDbContext>();
var studentDbSset  new DbSet<Student>();
dbContext.Students.Returns(studentDbSet);
```

```
//Arrange
var dbContext = Substitute.For<StudentsDbContext>();
var studentdbSet = Substitute.For<DbSet<Student>, IQueryable<Student>>();
((IQueryable<Student>)studentdbSet).Provider.Returns(students.AsQueryable().Provider);
((IQueryable<Student>)studentdbSet).Expression.Returns(students.AsQueryable().Expression);
((IQueryable<Student>)studentdbSet).GetEnumerator().Returns(students.AsQueryable().GetEnumerator());
```

Deze code werkt.

Goede gewoonte

- Maak code om mocken van DbSet in een klasse te verbergen: NSubstituteUtils
  - Herbruikbaar
  - Verhoogt leesbaarheid testen
  - Bron: http://sinairv.github.io/blog/2015/10/04/mock-entity-framework-dbset-with-nsubstitute/

Resultaat en gebruik

```
0 references
internal class NSubstituteUtils
{
    //functie waar T een willekeurige klasse is (hier bijvoorbeeld student)
    //dit gaat eigenlijk de Substitute voor de DbSet gaan aanmaken.
    0 references
    public static DbSet<T> GenerateMockDbSet<T>(List<T> data) where T : class
    {
        var dbSet = Substitute.For<DbSet<T>, IQueryable<T>>();

        ((IQueryable<T>)dbSet).Provider.Returns(data.AsQueryable().Provider);
        ((IQueryable<T>)dbSet).Expression.Returns(data.AsQueryable().Expression);
        ((IQueryable<T>)dbSet).GetEnumerator().Returns(data.AsQueryable().GetEnumerator());
        ((IQueryable<T>)dbSet).ElementType.Returns(data.AsQueryable().ElementType);

        return dbSet;
    }
}
```

```
var dbContext = Substitute.For<StudentsDbContext>();
studentdbSet = NSubstituteUtils.GenerateMockDbSet<Student>(students);
```

Nog meer informatie over NSubstitute nodig?

- Op dit moment hebben we de voornaamste zaken behandeld.
- Indien je meer informatie wil kan je steeds terecht in de documentatie van NSubstitute.
- https://nsubstitute.github.io/help.html