VDAB

samen sterk voor werk

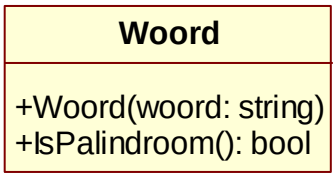
C# 2015
TEST DRIVEN DEVELOPMENT
TAKENBUNDEL

Inhoudsopgave

1 TAKEN

1.1 Palindroom

Je maakt een class `Woord`



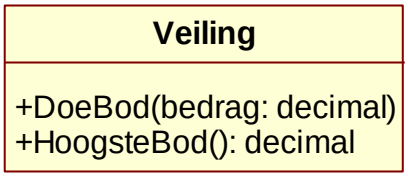
Je geeft aan de constructor een woord mee.

De method `IsPalindroom` geeft enkel `true` terug als dit woord een palindroom is: een woord dat hetzelfde is als je het van voor naar achter leest en als je het van achter naar voor leest (bvb. lepel).

Je schrijft in een class `WoordTest` de nodige tests voor de class `Woord`

1.2 VEILING

Je maakt met de voorgeschreven stappen van TDD
een class die een veiling (verkoop) voorstelt en de bijbehorende unit test.



Je kan op een `Veiling`-object meerdere keren de method `DoeBod` oproepen. Je geeft als parameter het bedrag van het bod mee.

Je kan op ieder moment de readonly property `HoogsteBod` oproepen.
Deze method geeft je het hoogst geboden bedrag terug.

```
var veiling = new Veiling();
veiling.DoeBod(1000);
var hoogsteBod = veiling.HoogsteBod;  // hoogsteBod bevat 1000
veiling.DoeBod(2000);
hoogsteBod = veiling.HoogsteBod; // hoogsteBod bevat 2000
```

Uit de analyse blijkt dat

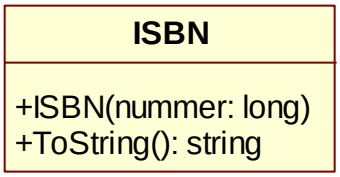
- Als nog geen enkel bod werd uitgevoerd, het hoogste bod gelijk is aan 0.
- Als een eerste bod werd uitgevoerd,
  het hoogste bod gelijk is aan het bedrag van dit bod.
- Al meerdere keren een bod werd uitgevoerd,
  het hoogste bod gelijk is aan bedrag van het hoogste bod.

1.3 Test fixtures

Je gebruikt een testinitialize method in de unit test van de class `Veiling`.

1.4 ISBN

Je maakt met de voorgeschreven stappen van TDD een class die een ISBN (Internationaal Standaard Boeknummer) voorstelt en de bijbehorende unit test.



De regels van ISBN zijn als volgt

- Het bestaat uit 13 cijfers
- Een ISBN bevat een controlemechanisme
    - Je maakt de som van de cijfers op de eerste 6 oneven posities
    - Je maakt de som van de cijfers op de eerste 6 even posities
      en je vermenigvuldigt deze som maal 3.
    - Je maakt de som van deze twee tussenresultaten.
    - Je maakt het verschil van deze som en het naastgelegen hoger gelegen tiental.
    - Het dertiende cijfer moet gelijk zijn aan dit verschil,
      tenzij het verschil 10 is. Dan moet het dertiende cijfer gelijk zijn aan 0.

Voorbeeld het ISBN `9789027439642`

- de som van de cijfers op de eerste 6 oneven posities
  `9 + 8 + 0 + 7 + 3 + 6 = 33`
- de som van de cijfers op de eerste 6 even posities, maal 3
  `7 + 9 + 2 + 4 + 9 + 4 = 35 x 3 = 105`
- de som van deze twee tussenresultaten
  `33 + 105 = 138`
- het verschil van deze som en het naastgelegen hoger gelegen tiental
  2
- het dertiende cijfer is gelijk aan dit verschil

1.5 Stub

Je maakt een class `WinstService`.

Deze class heeft een dependency, uitgedrukt in een interface `IOpbrengstDAO`.
Deze interface bevat één method-declaratie: `decimal TotaleOpbrengst();`

De class `WinstService` heeft een tweede dependency, uitgedrukt in een interface `IKostDAO`.
Deze interface bevat één method-declaratie: `decimal TotaleKost();`

De class `WinstService` bevat read only property: `decimal Winst`
De berekening van de winst is: totale opbrengst – totale kost

Je schrijft de class `WinstService` en de bijbehorende unit test.
In deze unit test gebruik je stubs voor `IOpbrengstDAO` en `IKostDAO`.

1.6 Mock

Je vervangt de stubs in `WinstService` door mocks, aangemaakt met Mockito.

Je doet ook verificaties op deze stubs.

2 VOORBEELDOPLOSSINGEN

2.1 Palindroom

2.1.1 Woord

```
using System.Linq;

namespace TDDCursusLibrary
{
  public class Woord
  {
    private readonly string woord;
    public Woord(string woord)
    {
      this.woord = woord;
    }
    public bool IsPalindroom()
    {
        var omgekeerd = new string(woord.ToArray().Reverse().ToArray());
        return woord == omgekeerd;
    }
  }
}
```

2.1.2 WoordTest

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TDDCursusLibrary;

namespace TDDCursusLibraryTest
{
  [TestClass]
  public class WoordTest
  {
    [TestMethod]
    public void LepelIsEenPalindroom()
    {
      Assert.IsTrue(new Woord("lepel").IsPalindroom());
    }
    [TestMethod]
    public void VorkIsGeenPalindroom()
    {
      Assert.IsFalse(new Woord("vork").IsPalindroom());
    }
    [TestMethod]
    public void eenLegeStringIsEenPalindroom()
    {
      Assert.IsTrue(new Woord(String.Empty).IsPalindroom());
    }
  }
}
```

2.2 Veiling

2.2.1 De class Veiling zonder echte code in zijn methods

```
using System;

namespace TDDCursusLibrary
{
  public class Veiling
  {
    public void DoeBod(decimal bedrag)
    {
      throw new NotImplementedException();
```

```
    }
    public decimal HoogsteBod
    {
      get
      {
        throw new NotImplementedException();
      }
    }
  }
}
```

2.2.2 De unit test VeilingTest

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TDDCursusLibrary;

namespace TDDCursusLibraryTest
{
  [TestClass]
  public class VeilingTest
  {
    [TestMethod]
    public void HetHoogsteBodVanEenNieuweVeilingStaatOpNul()
    {
      Assert.AreEqual(0m, new Veiling().HoogsteBod);
    }
    [TestMethod]
    public void NaEenEersteBodIsHetHoogsteBodGelijkAanHetBedragVanDitBod()
    {
      var veiling = new Veiling();
      veiling.DoeBod(100m);
      Assert.AreEqual(100m, veiling.HoogsteBod);
    }
    [TestMethod]
    public void NaMeerdereBiedingenIsHetHoogsteBodGelijkAanHetBedragVanDitBod()
    {
      var veiling = new Veiling();
      veiling.DoeBod(100m);
      veiling.DoeBod(200m);
      veiling.DoeBod(150m);
      Assert.AreEqual(200, veiling.HoogsteBod);
    }
  }
}
```

2.2.3 De class Veiling met echte code in zijn methods

```
namespace TDDCursusLibrary
{
  public class Veiling
  {
    private decimal hoogsteBod;
    public void DoeBod(decimal bedrag)
    {
      if (bedrag > hoogsteBod)
      {
        hoogsteBod = bedrag;
      }
    }
    public decimal HoogsteBod
    {
      get
      {
        return hoogsteBod;
      }
    }
  }
```

```
}
```

2.3 Test fixtures

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TDDCursusLibraryTest
{
  [TestClass]
  public class VeilingTest
  {
    private Veiling veiling;
    [TestInitialize]
    public void Initialize()
    {
      veiling = new Veiling();
    }
    [TestMethod]
    public void HetHoogsteBodVanEenNieuweVeilingStaatOpNul()
    {
      Assert.AreEqual(0m, veiling.HoogsteBod);
    }
    [TestMethod]
    public void NaEenEersteBodIsHetHoogsteBodGelijkAanHetBedragVanDitBod()
    {
      veiling.DoeBod(100m);
      Assert.AreEqual(100m, veiling.HoogsteBod);
    }
    [TestMethod]
    public void NaMeerdereodIsHetHoogsteBodGelijkAanHetBedragVanDitBod()
    {
      veiling.DoeBod(100m);
      veiling.DoeBod(200m);
      veiling.DoeBod(150m);
      Assert.AreEqual(200, veiling.HoogsteBod);
    }
  }
}
```

2.4 ISBN

2.4.1 De class Isbn zonder echte code in zijn methods

```
using System;

namespace TDDCursusLibrary
{
  public class Isbn
  {
    public Isbn(long nummer)
    {
      throw new NotImplementedException();
    }
    public override string ToString()
    {
       throw new NotImplementedException();
    }
  }
}
```

2.4.2 De unit test IsbnTest

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TDDCursusLibrary;

namespace TDDCursusLibraryTest
```

```
{
  [TestClass]
  public class IsbnTest
  {
    [TestMethod, ExpectedException(typeof(ArgumentException))]
    public void HetNummer0IsVerkeerd()
    {
      new Isbn(0);
    }
    [TestMethod, ExpectedException(typeof(ArgumentException))]
    public void EenNegatiefNummerIsVerkeerd()
    {
      new Isbn(-9789027439642L);
    }
    [TestMethod, ExpectedException(typeof(ArgumentException))]
    public void EenNummerMet12CijfersIsVerkeerd()
    {
      new Isbn(978902743964L);
    }
    [TestMethod, ExpectedException(typeof(ArgumentException))]
    public void EenNummerMet14CijfersIsVerkeerd()
    {
      new Isbn(97890274396421L);
    }
    [TestMethod, ExpectedException(typeof(ArgumentException))]
    public void EenNummerMet13CijfersMetVerkeerdControleGetal2()
    {
      new Isbn(8789027439642L);
    }
    [TestMethod]
    public void EenNummerMet13CijfersMetCorrectControleGetal2()
    {
      new Isbn(9789027439642L);
    }
    [TestMethod, ExpectedException(typeof(ArgumentException))]
    public void EenNummerMet13CijfersMetVerkeerdControleGetal0()
    {
      new Isbn(7789227439640L);
    }
    [TestMethod]
    public void EenNummerMet13CijfersMetCorrectControleGetal0()
    {
      new Isbn(9789227439640L);
    }
  }
}
```

2.4.3 De class Isbn met echte code in zijn methods

```
using System;

namespace TDDCursusLibrary
{
  public class Isbn
  {
    private const long GrootsteGetalMet13_Cijfers = 9999999999999L;
    private const long KleinsteGetalMet13_Cijfers = 1000000000000L;
    private long nummer;
    public Isbn(long nummer)
    {
      if (nummer < KleinsteGetalMet13_Cijfers || nummer > GrootsteGetalMet13_Cijfers)
      {
        throw new ArgumentException();
      }
      var somEvenCijfers = 0L;
      var somOnEvenCijfers = 0L;
```

```
      var teVerwerkenCijfers = nummer / 10;
      for (int teller = 0; teller != 6; teller++)
      {
        somEvenCijfers += teVerwerkenCijfers % 10;
        teVerwerkenCijfers /= 10;
        somOnEvenCijfers += teVerwerkenCijfers % 10;
        teVerwerkenCijfers /= 10;
      }
      var controleGetal = somEvenCijfers * 3 + somOnEvenCijfers;
      var naastGelegenHoger10Tal = controleGetal - controleGetal % 10 + 10;
      var verschil = naastGelegenHoger10Tal - controleGetal;
      var laatsteCijfer = nummer % 10;
      if (verschil == 10)
      {
        if (laatsteCijfer != 0)
        {
          throw new ArgumentException();
        }
      }
      else
      {
        if (laatsteCijfer != verschil)
        {
          throw new ArgumentException();
        }
      }
      this.nummer = nummer;
    }
    public override string ToString()
    {
      return nummer.ToString();
    }
  }
}
```

2.5 Stub

2.5.1 IOpbrengstDAO

```
namespace TDDCursusLibrary
{
  public interface IOpbrengstDAO
  {
    decimal TotaleOpbrengst();
  }
}
```

2.5.2 IKostDAO

```
namespace TDDCursusLibrary
{
  public interface IKostDAO
  {
    decimal TotaleKost();
  }
}
```

2.5.3 WinstService

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TDDCursusLibrary
{
  public class WinstService
  {
    private readonly IOpbrengstDAO opbrengstDAO;
```

```
    private readonly IKostDAO kostDAO;
    public WinstService(IOpbrengstDAO opbrengstDAO, IKostDAO kostDAO)
    {
      this.opbrengstDAO = opbrengstDAO;
      this.kostDAO = kostDAO;
    }
    public Decimal Winst
    {
      get
      {
        throw new NotImplementedException();
      }
    }
  }
}
```

2.5.4 OpbrengstDAOStub

```
using TDDCursusLibrary;

namespace TDDCursusLibraryTest
{
  class OpbrengstDAOStub:IOpbrengstDAO
  {
    public decimal TotaleOpbrengst()
    {
      return 200m;
    }
  }
}
```

2.5.5 KostDAOStub

```
using TDDCursusLibrary;

namespace TDDCursusLibraryTest
{
  class KostDAOStub:IKostDAO
  {
    public decimal TotaleKost()
    {
      return 169m;
    }
  }
}
```

2.5.6 WinstServiceTest

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TDDCursusLibrary;

namespace TDDCursusLibraryTest
{
  [TestClass]
  public class WinstServiceTest
  {
    private WinstService winstService;
    private IKostDAO kostDAO;
    private IOpbrengstDAO opbrengstDAO;
    [TestInitialize]
    public void Initialize()
    {
      kostDAO = new KostDAOStub();
      opbrengstDAO = new OpbrengstDAOStub();
      winstService = new WinstService(opbrengstDAO, kostDAO);
    }
    [TestMethod]
    public void WinstIsOpbrengstMinKost()
```

```
    {
      Assert.AreEqual(31m, winstService.Winst);
    }
  }
}
```

2.5.7 Echte code in de property Winst van WinstService

```
public Decimal Winst
{
  get
  {
    return opbrengstDAO.TotaleOpbrengst() - kostDAO.TotaleKost();
  }
}
```

2.6 Mock

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TDDCursusLibrary;
using Moq;

namespace TDDCursusLibraryTest
{
  [TestClass]
  public class WinstServiceTest
  {
    private WinstService winstService;
    private IKostDAO kostDAO;
    private IOpbrengstDAO opbrengstDAO;
    private Mock<IKostDAO> mockKostDAO;
    private Mock<IOpbrengstDAO> mockOpbrengstDAO;
    [TestInitialize]
    public void Initialize()
    {
      mockKostDAO = new Mock<IKostDAO>();
      mockOpbrengstDAO = new Mock<IOpbrengstDAO>();
      kostDAO = mockKostDAO.Object;
      opbrengstDAO = mockOpbrengstDAO.Object;
      mockOpbrengstDAO.Setup(
        eenOpbrengstDAO => eenOpbrengstDAO.TotaleOpbrengst()).Returns(200m);
      mockKostDAO.Setup(eenKostDAO => eenKostDAO.TotaleKost()).Returns(169m);
      winstService = new WinstService(opbrengstDAO, kostDAO);
    }
    [TestMethod]
    public void WinstIsOpbrengstMinKost()
    {
      Assert.AreEqual(31m, winstService.Winst);
      mockKostDAO.Verify(eenKostDAO=>eenkostDAO.TotaleKost());
      mockOpbrengstDAO.Verify(
        eenOpbrengstDAO=>eenOpbrengstDAO.TotaleOpbrengst());
    }
  }
}
```

3 COLOFON

Domeinexpertisemanager: Rita Van Damme

Moduleverantwoordelijke: Hans Desmet

Medewerkers: Hans Desmet
Veerle Smet

Versie: 25/3/2016

Nummer dotatielijst: