**Workshop Unittest WPF**

Deze oefening laat je unittests creëren in WPF. De originele Walktrough kun je vinden op

<https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019>

Achter de bullets staan de handelingen die je moet uitvoeren.

Let op dat je .net framework gebruikt voor beide projecten.

* Maak een WPF Core solution aan geheten “unittestWPF”
* Noem het project Bank
* Voeg een klasse toe aan het project met de naam Bankrekening.cs
* Vul dit bestand met :

public class Bankrekening

{

public double Saldo { get; private set; } = 0.0;

public bool Bevroren { get; set; } = false;

public Bankrekening(double startbedrag)

{

Saldo = startbedrag;

}

public Bankrekening()

{

}

public void Opnemen(double bedrag) // verlaag saldo

{

if (Bevroren) return;

Saldo += bedrag;

}

public void Storten(double bedrag) // verhoog saldo

{

Saldo += bedrag;

}

}

Dit is de klasse voor een bankrekening. Het saldo wordt opgehoogd door storten en verlaagd door opnemen. Als de rekening geblokkeerd is (bevroren), dan mag het saldo niet verlaagd worden.

Maak nu al unittests voor deze klasse. Dat kan gewoon, ook al is er nog geen applicatie gebouwd.

* Voeg een unit test aan de klasse Bankrekening. Het project heet default BankTests.
* In BankrekeningTests.cs kun je unittesten maken. Voeg aan dit bestand toe:

[TestMethod()]

public void Storten\_Geldig\_Bedrag()

{

Bankrekening rekening = new Bankrekening(11.99); // Arrange

rekening.Storten(4.55); // Act

// Assert.AreEqual(expected, actual, precision, error message)

Assert.AreEqual(16.54, rekening.Saldo, 0.001, "Storten niet gelukt");

}

[TestMethod()]

public void Opnemen\_Geldig\_Bedrag()

{

Bankrekening rekening = new Bankrekening(11.99); // Arrange

rekening.Opnemen(4.55); // Act

// Assert.AreEqual(expected, actual, precision, error message)

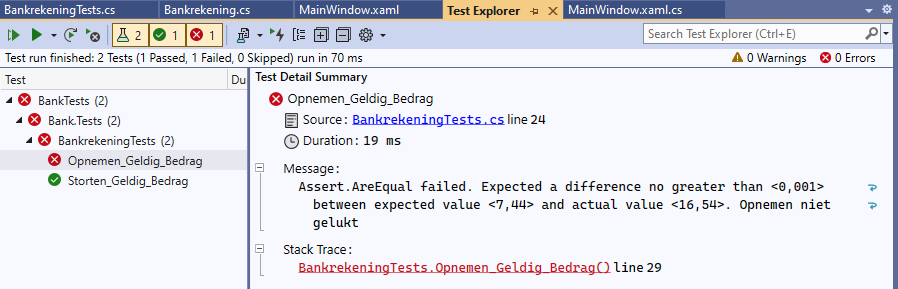
Assert.AreEqual(7.44, rekening.Saldo, 0.001, "Opnemen niet gelukt");

}

In de act-regel voer je de te testen methode uit, in dit geval een storting respectievelijk opname met een bedrag van 4,55. Deze methoden worden uitgevoerd op een object van de klasse Bankrekening. In de arrange-regel maak je het object aan met een startbedrag van 11,99. In de assert-regel controleer je de uitkomst. Het saldo moet uitkomen op 16,54 respectievelijk 4,55. De methode die je hier toepast is: Assert.AreEqual(expected, actual, precision, error message). De precisie geeft aan hoeveel het mag afwijken vanwege de afwijking die je kunt hebben wanneer je werkt met gebroken getallen.

* Voer de unttesten uit. Via menu test kun je de test explorer openen. Starten maar.

Het resultaat moet zijn dat er een test faalt met de mededeling “Opnemen niet gelukt”, omdat de verwachte waarde teveel (> 0,001) afwijkt van de werkelijke waarde.



* Blijkbaar is óf de unittest zelf fout óf de methode opnemen in de klasse Bankrekening. Verbeter deze fout en test opnieuw.
* Voeg 2 unittesten toe, namelijk: Storten\_Negatief en Opnemen\_negatief, waarin de methode Storten en Opnemen worden getest met een negatieve waarde.

[TestMethod()]

public void Storten\_Negatief()

{

Bankrekening rekening = new Bankrekening(11.99); // Arrange

rekening.Storten(-4.55); // Act

Assert.AreEqual(11.99, rekening.Saldo, 0.001,

"Negatieve waarde in storting"); // Assert

}

[TestMethod()]

public void Opnemen\_Negatief()

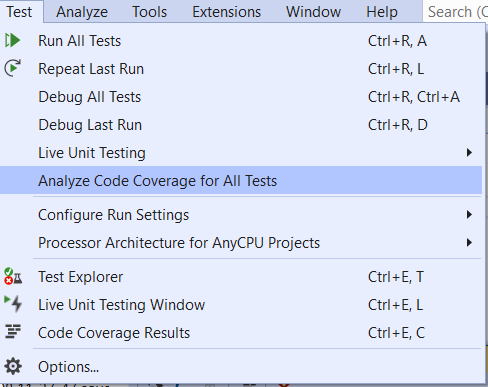
{

Bankrekening rekening = new Bankrekening(11.99); // Arrange

rekening.Opnemen(-4.55); // Act

Assert.AreEqual(11.99, rekening.Saldo, 0.001, "Negatieve waarde in opname");

}

* Deze beide testen falen doordat de methoden in de klasse Bankrekening niet controleren op negatieve bedragen. Verbeter deze fout in de klasse Bankrekening.
* Maak nog 2 unittesten erbij, namelijk:  
  - Storten\_Bevroren\_Rekening  
  - Opnemen\_Bevroren\_Rekening  
  Waarin je Bevroren in het arrange-gedeelte op true zet en de storting en opname met correcte waarden test. In het geval van opname zou het saldo niet aangepast mogen zijn in en het geval van storting zou het saldo correct opgehoogd moeten zijn.
* Controleer in hoeverre de code is gedekt door unittests. Daarvoor start je de analyse via menu test. Daarna zie je in “Code coverage results” hoe hoog de dekking is.

Als het goed is, dan heb je een groot deel van de code afgedekt met unittesten.

(PS: voor coverage heb je enterprise edition nodig. Community heeft deze niet.)

* Als laatste kun je nog de applicatie bouwen. Met een geteste klasse Bankaccount zou dat moeten lukken. Je kunt de volgende xaml en c# gebruiken, maar je kunt het zelf vast veel beter.

<Window x:Class="WalkThrough.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Bank"

mc:Ignorable="d"

Title="Bankrekening" Height="450" Width="800">

<StackPanel Width="200" Margin="0,20,0,0" >

<Label Content="Bankrekening" FontSize="16" FontWeight="Bold" />

<StackPanel Orientation="Horizontal" >

<Label Content="Saldo: " />

<Label x:Name="Saldo" Content="0.00" />

</StackPanel>

<StackPanel Orientation="Horizontal" >

<Label Content="Bevroren: " />

<CheckBox x:Name="Bevroren" IsChecked="False"

Click="Bevroren\_Click" VerticalAlignment="Center" />

</StackPanel>

<StackPanel Margin="0,40,0,0">

<Label Content="Bank Transfer" Margin="0, 10" FontSize="16" FontWeight="Bold" />

<Label Content="Bedrag:" />

<TextBox x:Name="Bedrag" ></TextBox>

<Button Content="Opnemen" Margin="0, 10" Click="Opnemen\_Clicked"/>

<Button Content="Storten" Margin="0, 5" Click="Storten\_Clicked"/>

</StackPanel>

</StackPanel>

</Window>

public partial class MainWindow : Window

{

Bankrekening rekening = new Bankrekening();

public MainWindow()

{

InitializeComponent();

}

private double stringToDouble(string b)

{

double bedrag;

Double.TryParse(b, out bedrag);

return bedrag;

}

private void Opnemen\_Clicked(object sender, RoutedEventArgs e)

{

rekening.Opnemen(stringToDouble(Bedrag.Text));

Saldo.Content = rekening.Saldo.ToString("0.00");

}

private void Storten\_Clicked(object sender, RoutedEventArgs e)

{

rekening.Storten(stringToDouble(Bedrag.Text));

Saldo.Content = rekening.Saldo.ToString("0.00");

}

private void Bevroren\_Click(object sender, RoutedEventArgs e)

{

rekening.Bevroren = (bool)Bevroren.IsChecked;

}

}