



Odisee
DE CO-HOGESCHOOL

Test Driven Development - TDD



Jens Baetens

Wat is Test Driven Development?



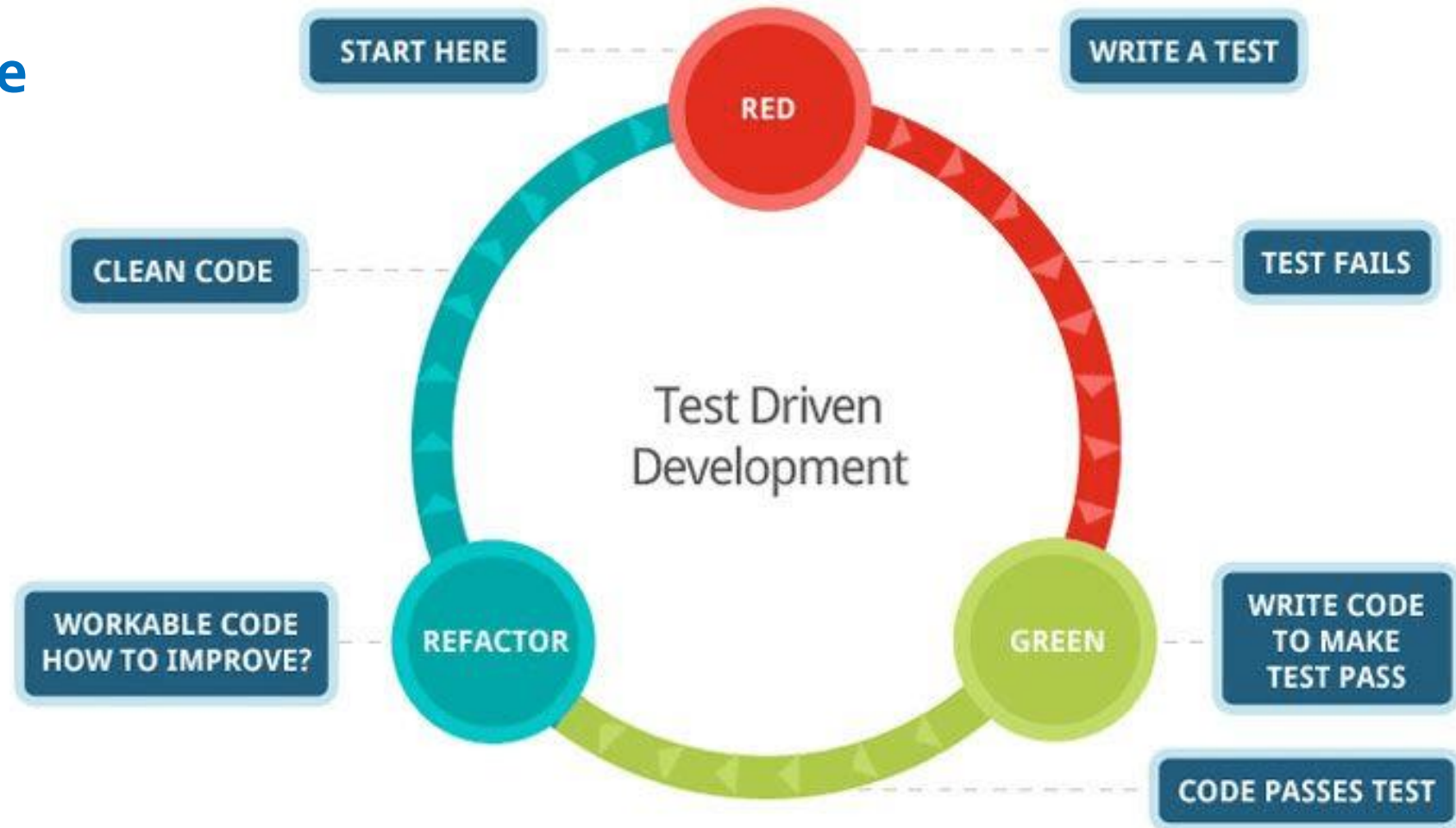
© <https://www.menti.com/eoejx6rapq>

Wat is Test Driven Development

- Een ontwikkelingsmethode waarbij eerst testen geschreven worden
 - ▬ Voor er functionele code geschreven wordt
 - ▬ Iteratief: Test en code schrijven wisselt af
- Applicatie groeit, gestuurd door tests
- Techniek werd in 2003 “heruitgevonden” door Kent Beck
- De techniek zet aan tot een eenvoudig design en vergroot het vertrouwen in de geschreven code

Wat is Test Driven Development

▣ Red => Green => Blue



Test Driven Development Cyclus

■ Schrijf test

- ▬ Op het moment dat test faalt, ga naar volgende stap
- ▬ Kan error/warning in IDE zijn, Compile error, Assertions die vallen

■ Schrijf code

- ▬ Minimale code om de test te doen slagen
- ▬ Test slaagt, ga naar volgende stap

■ Refactor (kan overgeslagen worden)

- ▬ Herschrijf de code voor leesbaarheid
- ▬ Geen nieuwe logica
- ▬ Herbegin

Waarom Test Driven Development?

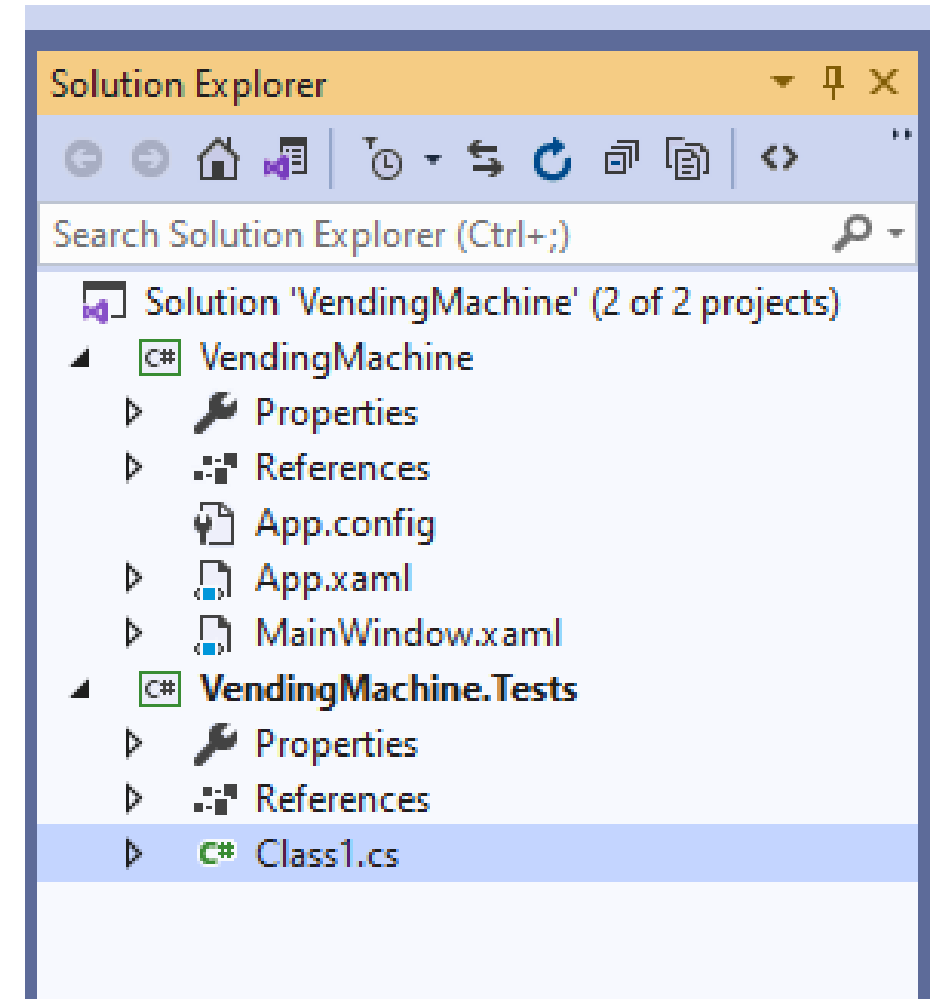
- ▣ Minder bugs tijdens schrijven van code
- ▣ Tests als documentatie
 - Tijdswinst
 - Later duidelijk wat er van de methode verwacht wordt
- ▣ Productiviteit
 - Focus op kleinere delen
 - Geen zorgen om afhankelijkheden
 - Denken over oplossingen tijdens schrijven van de test
- ▣ Niet meer tests schrijven dan nodig
- ▣ Clean code

Voorbeeld

- ▣ Schrijf een programma dat een drankautomaat simuleert
- ▣ Vereisten:
 - ▬ WPF-applicatie
 - ▬ Gebruiker kan geld in de machine werpen
 - ▬ Gebruiker kan ten allen tijde zijn/haar geld terugvragen
 - ▬ Machine heeft een aantal slots om drankjes te bewaren
 - ▬ Gebruiker kan een drankje selecteren door het nummer van het slot in te typen
 - Gebruiker heeft voldoende geld en drankje is voorradig: verlaag inventaris met 1
 - Wisselgeld blijft in budget zitten

Voorbeeld

- Maak een nieuw WPF project en testproject
- Voorzie alle dependencies en packages



Voorbeeld

■ Schrijf een eerste test

- Faalt omdat het Viewmodel niet bestaat

```
[TestFixture]
0 references
public class VendingMachineViewModelTests
{
    [SetUp]
    0 references
    public void Setup()
    {
    }

    [Test]
    0 references
    public void Add1EuroCommand_Adds1EuroToCredit()
    {
        VendingMachineViewModel sut = new VendingMachineViewModel();
    }
}
```

Voorbeeld

- Schrijf de klasse VendingMachineViewModel
 - Is de minimale code om de test te doen slagen
 - Refactor doen we nog niet

0 references

```
public class VendingMachineViewModel  
{  
}
```

Voorbeeld

- ▣ Vervolledig de test
 - Faalt omdat het Add1EuroCommand niet bestaat

```
[Test]
0 references
public void Add1EuroCommand__Adds1EuroToCredit()
{
    VendingMachineViewModel sut = new VendingMachineViewModel();

    sut.Add1EuroCommand.Execute(null);
}
```

Voorbeeld

- Voeg de property Add1EuroCommand toe
 - Test kan gestart worden maar faalt

2 references

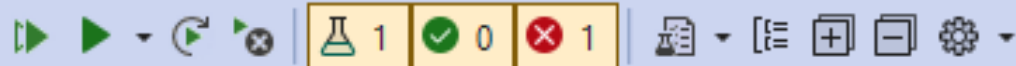




```
public class VendingMachineViewModel
```

```
{
```

1 reference

```
public ICommand Add1EuroCommand { get; set; }
```

```
}
```

Test Explorer			
			
Test	Duration	Traits	Error Message
▲  TDD.Tests (1)	92 ms		
▲  TDD.Tests (1)	92 ms		
▲  VendingMachineViewModelTests ...	92 ms		
 Add1EuroCommand_Adds1Eur...	92 ms		System.NullReferenceException : Object reference no...

Voorbeeld

- Schrijf code om test te doen slagen
 - ▬ Voeg het Add1EuroCommand toe

```
3 references
public class VendingMachineViewModel
{
    2 references | 0/1 passing
    public ICommand Add1EuroCommand { get; set; }

    1 reference | 0/1 passing
    public VendingMachineViewModel()
    {
        Add1EuroCommand = new RelayCommand(Add1Euro);
    }

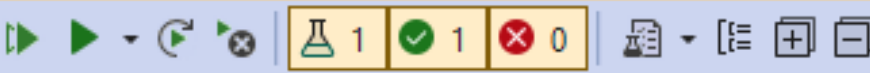
    1 reference
    private void Add1Euro()
    {
    }
}
```

```
1 reference
public class RelayCommand: ICommand
{
    public event EventHandler CanExecuteChanged;
    private Action action;

    1 reference
    public RelayCommand(Action action)
    {
        this.action = action;
    }

    0 references
    public bool CanExecute(object parameter)
    {
        return true;
    }

    0 references
    public void Execute(object parameter)
    {
        action();
    }
}
```

Test Explorer		
		
Test	Duration	T
▲ ✓ TDD.Tests (1)	25 ms	
▲ ✓ TDD.Tests (1)	25 ms	
▲ ✓ VendingMachineViewModelTests ...	25 ms	
✓ Add1EuroCommand_Adds1Eur...	25 ms	

Voorbeeld

- Vervolledig de test
 - Controleer budget
 - Faalt omdat het niet bestaat

```
[TestFixture]
0 references
public class VendingMachineViewModelTests
{
    [Test]
    0 references
    public void Add1EuroCommand__Adds1EuroToCredit()
    {
        //Arrange
        VendingMachineViewModel sut = new VendingMachineViewModel();

        //Act
        sut.Add1EuroCommand.Execute(null);

        //Assert
        Assert.That(sut.Budget, Is.EqualTo(1));
    }
}
```

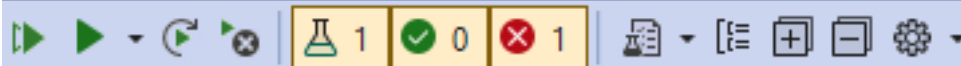
Voorbeeld

- Voeg de property Budget toe
 - Opnieuw enkel minimale code
 - Enkel definitie dus
 - Test faalt nog, budget is 0

```
3 references
public class VendingMachineViewModel
{
    1 reference | 1/1 passing
    public double Budget { get; set; }
    2 references | 1/1 passing
    public ICommand Add1EuroCommand { get; set; }

    1 reference | 1/1 passing
    public VendingMachineViewModel()
    {
        Add1EuroCommand = new RelayCommand(Add1Euro);
    }

    1 reference
    private void Add1Euro()
    {
    }
}
```

Test Explorer			
			
Test	Duration	Traits	Error Message
✖ TDD.Tests (1)	137 ms		
✖ TDD.Tests (1)	137 ms		
✖ VendingMachineViewModelTests ...	137 ms		
✖ Add1EuroCommand_Adds1Eur...	137 ms		Expected: 1 But was: 0.0d

Voorbeeld

- Schrijf logica om het budget aan te passen
 - ▬ Dit is minimale code
 - ▬ Dit voelt niet als de juiste oplossing
 - Toch doen we het als tussenstap
 - Test slaagt hiermee

3 references

```
public class VendingMachineViewModel
{
    1 reference | 0/1 passing
    public double Budget { get; set; }
    2 references | 0/1 passing
    public ICommand Add1EuroCommand { get; set; }

    1 reference | 0/1 passing
    public VendingMachineViewModel()
    {
        Add1EuroCommand = new RelayCommand(Add1Euro);
    }

    1 reference
    private void Add1Euro()
    {
        Budget = 1;
    }
}
```

Voorbeeld

- Een nieuwe test voor een 2 euro stuk
 - Faalt omdat Add2EuroCommand niet bestaat

```
[Test]
0 references
public void Add2EuroCommand__Adds2EuroToCredit()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();

    //Act
    sut.Add2EuroCommand.Execute(null);
}
```

Voorbeeld

- Voeg Add2EuroCommand toe
- Zorg ervoor dat het niet null is

```
5 references
public class VendingMachineViewModel
{
    2 references | 0/1 passing
    public double Budget { get; set; }
    2 references | 0/1 passing
    public ICommand Add1EuroCommand { get; set; }
    1 reference | 0/1 passing
    public ICommand Add2EuroCommand { get; set; }

    2 references | 0/2 passing
    public VendingMachineViewModel()
    {
        Add1EuroCommand = new RelayCommand(Add1Euro);
        Add2EuroCommand = new RelayCommand(Add2Euro);
    }

    1 reference
    private void Add1Euro()
    {
        Budget = 1;
    }

    1 reference
    private void Add2Euro()
    {
    }
}
```


Voorbeeld

■ Test vervolledigen

```
[Test]
0 references
public void Add2EuroCommand__Adds2EuroToCredit()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();

    //Act
    sut.Add2EuroCommand.Execute(null);

    //Assert
    Assert.That(sut.Budget, Is.EqualTo(2));
}
```

Test Explorer			
			
Test	Duration	Traits	Error Message
✖ TDD.Tests (2)	123 ms		
✖ TDD.Tests (2)	123 ms		
✖ VendingMachineViewModelTests ...	123 ms		
✔ Add1EuroCommand__Adds1Eur...	75 ms		
✖ Add2EuroCommand__Adds2Eur...	48 ms		Expected: 2 But was: 0.0d

Voorbeeld

- Minimale code om test te doen slagen

5 references

```
public class VendingMachineViewModel
```

```
{
```

4 references | 1/2 passing

```
public double Budget { get; set; }
```

2 references | 1/1 passing

```
public ICommand Add1EuroCommand { get; set; }
```

2 references | 0/1 passing

```
public ICommand Add2EuroCommand { get; set; }
```

2 references | 1/2 passing

```
public VendingMachineViewModel()
```

```
{
```

```
    Add1EuroCommand = new RelayCommand(Add1Euro);
```

```
    Add2EuroCommand = new RelayCommand(Add2Euro);
```

```
}
```

1 reference

```
private void Add1Euro()
```

```
{
```

```
    Budget = 1;
```

```
}
```

1 reference

```
private void Add2Euro()
```

```
{
```

```
    Budget = 2;
```

```
}
```

```
}
```



Voorbeeld

- ▣ Doe hetzelfde voor het commando Add50Cent, Add10Cent, Add20Cent

Voorbeeld

- De oplossing is duidelijk nog niet volledig
 - Valideer wanneer er al een bedrag in het budget zit dat het optelt

```
[Test]
0 references
public void Add1EuroCommand_WithAlready1EuroInBudget__Adds1EuroToCredit()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();
    sut.Budget = 1;

    //Act
    sut.Add1EuroCommand.Execute(null);

    //Assert
    Assert.That(sut.Budget, Is.EqualTo(2));
}
```

▲	✖ TDD.Tests (3)	118 ms	
▲	✖ TDD.Tests (3)	118 ms	
▲	✖ VendingMachineViewModelTests ...	118 ms	
	✔ Add1EuroCommand__Adds1Eur...	70 ms	
	✖ Add1EuroCommand_WithAlrea...	48 ms	Expected: 2 But was: 1.0d
	✔ Add2EuroCommand__Adds2Eur...	< 1 ms	

Voorbeeld

- ▣ Code schrijven om de test te laten slagen
 - Tests slagen nu

```
1 reference  
private void Add1Euro()  
{  
    if(Budget == 1)  
    {  
        Budget = 2;  
    }  
    else  
    {  
        Budget = 1;  
    }  
}
```


Voorbeeld

▣ Nu doen we een refactor-stap

- ▬ We merken dat de code efficiënter geschreven kan worden dus doen we een refactor

```
1 reference  
private void Add1Euro()  
{  
    Budget += 1;  
}
```

▣ Deze code is duidelijk beter op langer termijn en de tests slagen

- ▬ Doe hetzelfde voor 2 euro, 50 cent, 20 cent en 10 cent

Voorbeeld

- Een nieuwe test die we willen is dat het PropertyChanged Event aangeroepen wordt
 - Test faalt niet aangezien het event nog niet aanwezig is

```
[Test]
public void Budget_Changed_OnPropertyChangedIsCalled()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();
    string changedProperty;
    sut.PropertyChanged += (e, args) =>
    {
        changedProperty = args.PropertyName;
    }
}
```

Voorbeeld

- Schrijf opnieuw de minimale code om de test te doen slagen
 - ▬ INotifyPropertyChanged Interface nodig

```
9 references
public class VendingMachineViewModel: INotifyPropertyChanged
{
    6 references | 2/3 passing
    public double Budget { get; set; }
    3 references | 1/2 passing
    public ICommand Add1EuroCommand { get; set; }
    2 references | 1/1 passing
    public ICommand Add2EuroCommand { get; set; }

    public event PropertyChangedEventHandler PropertyChanged;

    4 references | 2/4 passing
    public VendingMachineViewModel()
    {
        Add1EuroCommand = new RelayCommand(Add1Euro);
        Add2EuroCommand = new RelayCommand(Add2Euro);
    }
}
```

Voorbeeld

- ▣ Vervolledig de test
 - Test faalt nog, functie niet ok

```
[Test]
0 references
public void Budget_Changed_OnPropertyChangedIsCalled()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();
    string changedProperty = "";
    sut.PropertyChanged += (e, args) =>
    {
        changedProperty = args.PropertyName;
    };

    //Act
    sut.Budget = 5;

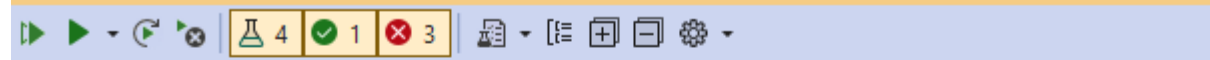
    //Assert
    Assert.That(changedProperty, Is.EqualTo("Budget"));
}
```

Voorbeeld

■ Schrijf minimale code

- Test slaagt maar de andere vallen
- Pas code verder aan

```
private double budget;  
7 references | 1/4 passing  
public double Budget  
{  
    get => budget;  
    set  
    {  
        budget= value;  
        PropertyChanged(this, new PropertyChangedEventArgs("Budget"));  
    }  
}
```

Test Explorer			
			
Test	Duration	Traits	Error Message
▲ [X] TDD.Tests (4)	102 ms		
▲ [X] TDD.Tests (4)	102 ms		
▲ [X] VendingMachineViewModelTests ...	102 ms		
[X] Add1EuroCommand_Adds1Eur...	92 ms		System.NullReferenceException : Ob...
[X] Add1EuroCommand_WithAlrea...	5 ms		System.NullReferenceException : Ob...
[X] Add2EuroCommand_Adds2Eur...	1 ms		System.NullReferenceException : Ob...
[✓] Budget_Changed_OnPropertyC...	4 ms		

Voorbeeld

■ Andere tests doen slagen

- ▬ Niet oude testen aanpassen, enkel code aanpassen in deze fase
- ▬ Tenzij functionaliteit gewijzigd is

```
private double budget;  
7 references | 0/4 passing  
public double Budget  
{  
    get => budget;  
    set  
    {  
        budget = value;  
        if (PropertyChanged != null)  
        {  
            PropertyChanged(this, new PropertyChangedEventArgs("Budget"));  
        }  
    }  
}
```

Er zou hier nog een refactor kunnen staan, zodat we het PropertyChanged event gaan uniformiseren zoals gezien in de les van MVVM. In dit geval heeft het eigenlijk geen voordeel.

Voorbeeld

- ▣ Volgende functionaliteit is terugvragen van geld
 - Test faalt omdat command niet bestaat
 - Voeg het toe

```
[Test]
0 | 0 references
public void RefundCommand_WithBudget5_BudgetBecomesZero()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();

    //Act
    sut.RefundCommand.Execute(null);
}
```

Voorbeeld

- Voeg de code toe om de compiler error op te lossen

```
2 references | 0/1 passing
public ICommand RefundCommand { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

5 references | 0/5 passing
public VendingMachineViewModel()
{
    Add1EuroCommand = new RelayCommand(Add1Euro);
    Add2EuroCommand = new RelayCommand(Add2Euro);
    RefundCommand = new RelayCommand(Refund);
}

1 reference
private void Refund()
{
}
```


Voorbeeld

▣ Vervolledig de test

```
[Test]
0 | 0 references
public void RefundCommand_WithBudget5_BudgetBecomesZero()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();

    //Act
    sut.RefundCommand.Execute(null);

    //Assert
    Assert.That(sut.Budget, Is.EqualTo(0));
}
```

Voorbeeld

- Schrijf de code om de test te doen slagen

```
1 reference  
private void Refund()  
{  
    Budget = 0;  
}
```

Voorbeeld

- ▣ Volgende functionaliteit: Aantal slots om drankjes te bewaren
 - Worden ingeladen bij opstarten
- ▣ Keuzes:
 - Voorzie een klasse slot met de nodige properties
 - Vanuit developer standpunt een logische oplossing
 - Werk verder in viewmodel
 - In een refactor fase dan de klasse slot aanmaken
 - Test Driven Development op de letter gevolgd
- ▣ Wij kiezen voor de eerste optie
 - TDD is een middel om goede tests te schrijven
 - Mag het nietodeloos complex maken

Voorbeeld

- Voorzie een test die de klasse slot definieert
 - In een nieuwe file
 - Test faalt door compiler problemen

```
[TestFixture]
0 references
public class SlotTests
{
    [Test]
    0 references
    public void Ctor_WithValidData_AllPropertiesFilled()
    {
        //Arrange
        double price = 2.5;
        string name = "cola";
        int slotNumber = 1;
        int quantity = 10;

        //Act
        Slot sut = new Slot(slotNumber, name, price, quantity);
    }
}
```

Voorbeeld

- ▣ Voorzie een klasse slot met constructor
 - Compiler problemen opgelost

```
3 references
public class Slot
{
    1 reference | 0/1 passing
    public Slot(int slotNumber, string name, double price, int quantity)
    {
    }
}
```

Voorbeeld

■ Vervolledig de test

- Faalt op compileerproblemen
- Faalt daarna op logische problemen

```
[Test]
0 | 0 references
public void Ctor_WithValidData_AllPropertiesFilled()
{
    //Arrange
    double price = 2.5;
    string name = "cola";
    int slotNumber = 1;
    int quantity = 10;

    //Act
    Slot sut = new Slot(slotNumber, name, price, quantity);

    //Assert
    Assert.AreEqual(2.5, sut.Price);
    Assert.AreEqual("cola", sut.Name);
    Assert.AreEqual(1, sut.Number);
    Assert.AreEqual(10, sut.Quantity);
}
```

Voorbeeld

- Schrijf de code om de test te doen slagen
 - Maar is dit wat we willen?

```
3 references
public class Slot
{
    1 reference | 0/1 passing
    public double Price { get => 2.5; }
    1 reference
    public string Name { get => "cola"; }
    1 reference
    public int Number { get => 1; }
    1 reference | 0/1 passing
    public int Quantity { get => 10; }

    1 reference | 0/1 passing
    public Slot(int slotNumber, string name, double price, int quantity)
    {
    }
}
```

Voorbeeld

- Schrijf een extra test voor de klasse slot
 - Faalt door logische fout

```
[Test]
0 references
public void Ctor_WithValidData_AllPropertiesFilled2()
{
    //Arrange
    double price = 1.5;
    string name = "water";
    int slotNumber = 2;
    int quantity = 20;

    //Act
    Slot sut = new Slot(slotNumber, name, price, quantity);

    //Assert
    Assert.AreEqual(1.5, sut.Price);
    Assert.AreEqual("water", sut.Name);
    Assert.AreEqual(2, sut.Number);
    Assert.AreEqual(20, sut.Quantity);
}
```


Voorbeeld

▣ Code dat de tweede test doet slagen

```
5 references
public class Slot
{
    3 references | 0/2 passing
    public double Price { get; set; }
    3 references | 0/2 passing
    public string Name { get; set; }
    3 references | 0/2 passing
    public int Number { get; set; }
    3 references | 0/2 passing
    public int Quantity { get; set; }

    2 references | 0/2 passing
    public Slot(int slotNumber, string name, double price, int quantity)
    {
        Number = slotNumber;
        Name = name;
        Price = price;
        Quantity = quantity;
    }
}
```

Voorbeeld

- ▣ Refactor van test omdat er te veel herhaling is
 - Gebruik testcases

```
[TestFixture]
0 references
public class SlotTests
{
    [Test]
    [TestCase(1, "Cola", 2.5, 10)]
    [TestCase(2, "Water", 1.5, 20)]
    0 references
    public void Ctor_WithValidData_AllPropertiesFilled(int slotNumber, string name, double price, int quantity)
    {
        //Act
        Slot sut = new Slot(slotNumber, name, price, quantity);

        //Assert
        Assert.AreEqual(2.5, sut.Price);
        Assert.AreEqual("cola", sut.Name);
        Assert.AreEqual(1, sut.Number);
        Assert.AreEqual(10, sut.Quantity);
    }
}
```

Voorbeeld

- Maak nu een test dat controleert dat bij opstart, de slots leeg zijn
 - In het viewmodel
 - Property slots bestaat niet

```
[Test]
| 0 references
public void Ctor_WithNoItems_CreatesEmptyList()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel();

    //Assert
    Assert.AreEqual(new List<Slot>(), sut.Slots);
}
```

Voorbeeld

- Schrijf de code om de test te doen slagen

```
-  
2 references | 0/1 passing  
public List<Slot> Slots { get; set; }  
  
3 references | 0/2 passing  
public ICommand Add1EuroCommand { get; set; }  
2 references | 0/1 passing  
public ICommand Add2EuroCommand { get; set; }  
2 references | 0/1 passing  
public ICommand RefundCommand { get; set; }  
  
public event PropertyChangedEventHandler PropertyChanged;  
  
6 references | 0/6 passing  
public VendingMachineViewModel()  
{  
    Add1EuroCommand = new RelayCommand(Add1Euro);  
    Add2EuroCommand = new RelayCommand(Add2Euro);  
    RefundCommand = new RelayCommand(Refund);  
    Slots = new List<Slot>();  
}
```

Voorbeeld

- Schrijf een test die ervoor zorgt dat de slots data ingelezen wordt uit de database
 - Gebruik een substitute object hiervoor
 - Faalt omdat interface niet bestaat

```
[Test]
0 references
public void Ctor_WithDataFromDatabase_AddDataToList()
{
    ISlotDataRepository dataRepository = Substitute.For<ISlotDataRepository>();
}
```

Voorbeeld

- ▣ Voorzie de interface ISlotDataRepository

```
2 references  
public interface ISlotDataRepository  
{  
}
```

Voorbeeld

- Vervolledig de test
 - Enkele stappen samen genomen

```
[Test]
0 references
public void Ctor_WithDataFromDatabase_AddDataToList()
{
    //Arrange
    ISlotDataRepository dataRepository = Substitute.For<ISlotDataRepository>();
    List<Slot> slots = new List<Slot>()
    {
        new Slot(1, "Cola", 2.5, 10),
        new Slot(2, "Water", 1.5, 20)
    };
    dataRepository.LoadData().Returns(slots);

    //Act
    VendingMachineViewModel sut = new VendingMachineViewModel(dataRepository);

    //Assert
    Assert.AreEqual(slots, sut.Slots);
}
```

Voorbeeld

■ Vervolledig de code

- Vul interface aan
- Constructor met parameter

```
2 references  
public interface ISlotDataRepository  
{  
    1 reference | 0/1 passing  
    List<Slot> LoadData();  
}
```

```
1 reference | 0/1 passing  
public VendingMachineViewModel(ISlotDataRepository dataRepository)  
{  
    Add1EuroCommand = new RelayCommand(Add1Euro);  
    Add2EuroCommand = new RelayCommand(Add2Euro);  
    RefundCommand = new RelayCommand(Refund);  
    Slots = dataRepository.LoadData();  
}
```


Voorbeeld

■ Laatste requirement: Drankjes kopen

- Slot kiezen
- Koop Commando
- Controleer het budget

```
[Test]
public void BuyCommand_WithEnoughMoneyAndItemSelected_RemovesOneItemFromListAndReloadsView()
{
    //Arrange
    ISlotDataRepository dataRepository = Substitute.For<ISlotDataRepository>();
    List<Slot> beforeSlots = new List<Slot>()
    {
        new Slot(1, "Cola", 1.5, 10)
    };

    List<Slot> afterSlots = new List<Slot>()
    {
        new Slot(1, "Cola", 1.5, 10)
    };
    dataRepository.LoadData().Returns(beforeSlots, afterSlots);
    VendingMachineViewModel sut = new VendingMachineViewModel(dataRepository);
    sut.Budget = 1.5;
    sut.SelectedSlot = "1";

    //Act
    sut.BuyCommand.Execute(null);

    //Assert
    dataRepository.Received(1).RemoveItemFromSlot(1);
    Assert.That(sut.Slots, Is.EqualTo(afterSlots));
    Assert.IsEmpty(sut.SelectedSlot);
    Assert.That(sut.Budget, Is.EqualTo(0));
}
```

Voorbeeld

▣ Code om test te doen slagen

```
7 references | 0/7 passing
public VendingMachineViewModel(ISlotDataRepository dataRepository)
{
    Add1EuroCommand = new RelayCommand(Add1Euro);
    Add2EuroCommand = new RelayCommand(Add2Euro);
    RefundCommand = new RelayCommand(Refund);
    BuyCommand = new RelayCommand(Buy);
    slotRepository = dataRepository;

    Slots = dataRepository.LoadData();
    if(Slots == null)
    {
        Slots = new List<Slot>();
    }
}

1 reference:
private void Buy()
{
    slotRepository.RemoveItemFromSlot(1);
    Slots = slotRepository.LoadData();
    SelectedSlot = String.Empty;
    Budget = 0;
}
```

Voorbeeld

- Valideren dat wanneer er geen item geselecteerd is, dat er een boodschap getoond wordt en geen aankoop

```
[Test]
0 references
public void BuyCommand_WithNoItemSelected_SetsPropperErrorMessageAndDoesntExecutePayment()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel(dataRepository);
    sut.Budget = 1.5;

    //Act
    sut.BuyCommand.Execute(null);

    //Assert
    Assert.That(sut.Message, Is.EqualTo("Invalid Slot"));
    dataRepository.DidNotReceive().RemoveItemFromSlot(Arg.Any<int>());
    Assert.That(sut.Budget, Is.EqualTo(1.5));
}
```

Voorbeeld

▣ Code om test te doen slagen

```
1 reference  
private void Buy()  
{  
    Message = "Invalid slot";  
    if (!string.IsNullOrEmpty(SelectedSlot))  
    {  
        slotRepository.RemoveItemFromSlot(1);  
        Budget = 0;  
    }  
    Slots = slotRepository.LoadData();  
    SelectedSlot = string.Empty;  
}
```

Voorbeeld

- Bericht wordt nu altijd getoond, willen we niet als het goed loopt
 - Wijzig de vorige test om dit te controleren

```
[Test]
public void BuyCommand_WithEnoughMoneyAndItemSelected_RemovesOneItemFromListAndReloadsView()
{
    //Arrange
    ISlotDataRepository dataRepository = Substitute.For<ISlotDataRepository>();
    List<Slot> beforeSlots = new List<Slot>()
    {
        new Slot(1, "Cola", 1.5, 10)
    };

    List<Slot> afterSlots = new List<Slot>()
    {
        new Slot(1, "Cola", 1.5, 10)
    };
    dataRepository.LoadData().Returns(beforeSlots, afterSlots);
    VendingMachineViewModel sut = new VendingMachineViewModel(dataRepository);
    sut.Budget = 1.5;
    sut.SelectedSlot = "1";

    //Act
    sut.BuyCommand.Execute(null);

    //Assert
    dataRepository.Received(1).RemoveItemFromSlot(1);
    Assert.That(sut.Slots, Is.EqualTo(afterSlots));
    Assert.IsEmpty(sut.SelectedSlot);
    Assert.That(sut.Budget, Is.EqualTo(0));
    Assert.IsEmpty(sut.Message);
}
```

Voorbeeld

▣ Code om test te doen slagen

```
1 reference  
private void Buy()  
{  
    Message = "Invalid slot";  
    if (!string.IsNullOrEmpty(SelectedSlot))  
    {  
        slotRepository.RemoveItemFromSlot(1);  
        Budget = 0;  
        Message = string.Empty;  
    }  
    Slots = slotRepository.LoadData();  
    SelectedSlot = string.Empty;  
}
```

Voorbeeld

■ Test of het selectedslot een getal is

```
[Test]
0 references
public void BuyCommand_WithSelectedSlotNotANumber_SetsPropperErrorMessageAndDoesntExecutePayment()
{
    //Arrange
    VendingMachineViewModel sut = new VendingMachineViewModel(dataRepository);
    sut.Budget = 1.5;
    sut.SelectedSlot = "vijf";

    //Act
    sut.BuyCommand.Execute(null);

    //Assert
    Assert.That(sut.Message, Is.EqualTo("Invalid Slot"));
    dataRepository.DidNotReceive().RemoveItemFromSlot(Arg.Any<int>());
    Assert.That(sut.Budget, Is.EqualTo(1.5));
}
```

Voorbeeld

▣ Code wijziging

```
1 reference
private void Buy()
{
    Message = "Invalid slot";
    int selectedSlot;
    if (!string.IsNullOrEmpty(SelectedSlot) && int.TryParse(SelectedSlot, out selectedSlot))
    {
        slotRepository.RemoveItemFromSlot(1);
        Budget = 0;
        Message = string.Empty;
    }
    Slots = slotRepository.LoadData();
    SelectedSlot = string.Empty;
}
```


Voorbeeld

▣ Wat als slot number niet in de lijst?

```
[Test]
• | 0 references
public void BuyCommand_WithSelectedSlotNotAnExistingSlot_SetsPropperErrorMessageAndDoesntExecutePayment()
{
    //Arrange
    List<Slot> beforeSlots = new List<Slot>()
    {
        new Slot(1, "Cola", 1.5, 10)
    };
    dataRepository.LoadData().Returns(beforeSlots);
    VendingMachineViewModel sut = new VendingMachineViewModel(dataRepository);
    sut.Budget = 1.5;
    sut.SelectedSlot = "2";

    //Act
    sut.BuyCommand.Execute(null);

    //Assert
    Assert.That(sut.Message, Is.EqualTo("Invalid Slot"));
    dataRepository.DidNotReceive().RemoveItemFromSlot(Arg.Any<int>());
    Assert.That(sut.Budget, Is.EqualTo(1.5));
}
```

Voorbeeld

▣ Code om test te laten slagen

```
1 reference
private void Buy()
{
    Message = "Invalid slot";
    int selectedSlot;
    if (!string.IsNullOrEmpty(SelectedSlot) && int.TryParse(SelectedSlot, out selectedSlot))
    {
        if (Slots.Exists(slot => slot.Number == selectedSlot)) {
            slotRepository.RemoveItemFromSlot(1);
            Budget = 0;
            Message = string.Empty;
        }
    }
    Slots = slotRepository.LoadData();
    SelectedSlot = string.Empty;
}
```

Voorbeeld

- Test die valideert dat er een errormessage verschijnt als budget te laag is

```
[Test]
public void BuyCommand_WithNotEnoughBudget_SetsPropperErrorMessageAndDoesntExecutePayment()
{
    //Arrange
    List<Slot> beforeSlots = new List<Slot>()
    {
        new Slot(1, "Cola", 1.5, 10)
    };
    dataRepository.LoadData().Returns(beforeSlots);
    VendingMachineViewModel sut = new VendingMachineViewModel(dataRepository);
    sut.Budget = 1;
    sut.SelectedSlot = "1";

    //Act
    sut.BuyCommand.Execute(null);

    //Assert
    Assert.That(sut.Message, Is.EqualTo("Not enough budget"));
    dataRepository.DidNotReceive().RemoveItemFromSlot(Arg.Any<int>());
    Assert.That(sut.Budget, Is.EqualTo(1.5));
}
```

Voorbeeld

▣ Code om test te doen slagen

```
1 reference
private void Buy()
{
    Message = "Invalid slot";
    int selectedSlot;
    if (!string.IsNullOrEmpty(SelectedSlot) && int.TryParse(SelectedSlot, out selectedSlot))
    {
        if (Slots.Exists(slot => slot.Number == selectedSlot)) {
            if (Budget >= Slots.Find(slot => slot.Number == selectedSlot).Price){
                slotRepository.RemoveItemFromSlot(1);
                Budget = 0;
                Message = string.Empty;
            } else
            {
                Message = "Not enough budget";
            }
        }
    }
    Slots = slotRepository.LoadData();
    SelectedSlot = string.Empty;
}
```



Voorbeeld

- ▣ Het is duidelijk dat nog niet alles voldoende getest is.
 - Voeg een aantal testcases toe voor de test waar het aankopen lukt en kijk of alle cases slagen



Voorbeeld

- ▣ Schrijf nu de code om deze tests te doen slagen

Voorbeeld

▣ Momenteel enkel een viewmodel met enkele rules

▬ Nog Toe te voegen:

- Geen controle op voorraad
- PropertyChangedEvent voor SelectedSlot, Message, Slots
- DataRepository die de data operaties effectief afhandeld (nieuwe file, dus nieuwe testfile)
- Gui om de applicatie te gebruiken
 - ▼ Kunnen we niet testen



Voordelen

- ▣ Helpt om modulaire en testbare code te schrijven
- ▣ Veilige manier om te refactoren en te onderhouden
 - Bugs na wijzigingen gedetecteerd in de testresultaten
- ▣ Verhoogt productiviteit
- ▣ Helpt om bugs te vermijden
 - Heel snel gedetecteerd
- ▣ Verplicht de ontwikkelaar om na te denken over de requirements voor te coderen





Nadelen

- ▣ Tests moeten onderhouden worden
- ▣ Niet altijd eenvoudig om tests te schrijven
- ▣ Moeilijk om toe te passen op bestaande code
- ▣ Refactoring van code die getest wordt vergt ook vaak refactoring van testcode
- ▣ Lijkt trager te gaan als je begint

Tips – Orde van testen

▣ Degenerate case

- Start met tests voor null waarden, lege waarden, 0, etc
- Zorgt voor de basis van je klassen

▣ Happy Paths

- Enkele simple cases die de standard gebruiker uitvoert
- Helpt om de core functionaliteit van de code te ontwikkelen

Tips – Orde van testen

- ▣ Tests met nieuwe informatie/kennis
 - Nieuwe cases verzinnen die kunnen voorvallen
 - Leren welke zaken er nog verwacht kunnen worden
- ▣ Error handling en negatieve testen
 - Testen dat de juiste zaken gebeuren als er foutieve data wordt doorgestuurd
 - Code robuuster tegen afwijkingen

Tips - Red to green bar strategieën

▣ Faking

- Simpelste manier om test te doen slagen
- Geef terug wat de test verwacht

▣ Kiss – meest voor de hand liggende oplossing

- Keep it simple, stupid
- Niet overengineren

▣ Triangulatie

- Vanuit een paar specifieke scenario's en implementaties een generalisatie maken
- Meerdere tests met andere parameters en output leiden naar algemenere oplossing