Hand-out

Programmeren

Periode 2

03. Klassen & Objecten I

1. **Inleiding**

   Voor dat we met WPF echt iets moois kunnen maken is het van belang dat we iets te weten komen over klassen en objecten. Dit komt door dat het hele .NET framework bestaat uit klassen en objecten en bijvoorbeeld alle controls hun eigen klassen hebben en je bij het gebruik ervan daar instanties van opmaakt wat een object oplevert.

   In de vorige periode hebben we alleen gewerkt met de console, en de console is een statische klasse, waardoor we methodes en members ook static moesten definiëren. Static betekent, er hoeft geen instantie gemaakt te worden van de desbetreffende klasse en daarom werken we ook niet met objecten. In deze les gaan we leren wat een object is en hoe we een klasse maken.

2. **Een class**

   Een class is een beschrijving van een object, deze beschrijving word meestal in een apart bestand opgemaakt onder de zelfde naam. Van deze beschrijving kan vervolgens een instantie worden gemaakt en op die manier heb je toegang tot een object.

   Een class kan een object beschrijven dat bestaat uit deel objecten. Als we bijvoorbeeld een auto class zouden maken beschrijft deze class auto in zijn algemeen maar een auto bestaat uit enorm veel onderdelen. Al deze onderdelen zijn ook objecten en hebben dus ook elk een class die dit beschrijft.

   Een wiel is bijvoorbeeld onderdeel van een auto en heeft dus ook een eigen class. Maar een auto heeft vier wielen en dus maken we vier instanties van de zelfde class zodat we vier verschillende objecten krijgen. Als we nu ooit ons wiel aan willen passen doen we dit dus nog maar op een plek, in de class wiel. En niet te bedenken dat een fiets ook twee wielen heeft!

   Een voorbeeld :

```
2   public class Car : Vehicle
3   {
4       public string brand;
5       public string model;
6       public string type;
7
8       public void Car(string brand, string model, string type)
9       {
10          this.brand = brand;
11          this.model = model;
12          this.type  = type;
13      }
14  }
15
16  Car myFirstCar  = new Car("mercedes", "benz", "X1");
17  Car mySecondCar = new Car("toyota", "aigo", "303");
18  Car myThirdCar  = new Car("BMW", "3", "M");
```

3. **Instances**

   Een instantie of instance maak je van een class. Dit doe je door het type van je class te definiëren als variabele en er met het "new" keyword een object van te maken zoals in het voorbeeld hier boven in de onderste drie regels.

   De hele bedoeling van een class is dat we er verschillende instanties van aan kunnen maken zodat we verschillende objecten tot onze beschikking hebben die een enkele code-base delen. Zo zorgen we er voor dat we maar op een plek aanpassingen hoeven te doen en dat vervolgens op alle objecten effect heeft.

   Dit geld dus ook voor je formulier, we hebben ten eerste de Application class in de App.xaml.cs en vervolgens de MainWindow class in de MainWindow.xaml.cs.
   Dit zijn classes waar .NET op de achtergrond instanties van aanmaakt zodat je formulier kan worden uitgevoerd.

4. **Scope**

   Scope is iets wat we al eerder hebben behandeld in de eerste periode toen we het hadden over variabelen en datatypen. In dit hoofdstuk geld scope ook voor de class. Een klasse kan public, protected of private zijn net als haar members en methodes.

   public wil zeggen, deze klasse is beschikbaar voor alle andere klassen binnen de zelfde namespace.
   protected betekent, deze klasse is beschikbaar voor alle klassen die hier van overerven binnen de zelfde namespace.
   private houdt in, deze klasse is niet beschikbaar voor andere klassen.

   Voor methodes en members (klassen variabelen) geld de zelfde regel!

5. **Demo**

   In dit stuk gaan we klassikaal een demo maken. Het is de bedoeling dat je mee kijkt en zo stapsgewijs houvast krijgt op het onderwerp, maak daar waar nodig is aantekeningen in je schrift of in je kladblok. Als je klaar bent ga je zelf aan de slag dus let goed op!

6. **Opdracht**

   In deze opdracht ga je verder met het Boter Kaas en Eieren spel. Het is de bedoeling dat de Graphical User Interface nu gekoppeld gaat worden aan de logica, dit doe je in de code-behind.