



Programmeren 3

Scope: access modifiers

- Elke member field en methode in een class heeft een bepaalde 'toegankelijkheid'
- Deze toegankelijkheid is in te stellen met zogenaamde 'access modifiers'
 - public → field/methode is beschikbaar in eigen class, afgeleide classes, en buiten de class;
 - private → field/methode is alleen beschikbaar in eigen class;
 - protected → field/methode is alleen beschikbaar in eigen class en in afgeleide classes;

Public fields/methods

- Public fields/methods zijn overal te gebruiken

```
class Boek
{
    // member fields
    public string titel;
    public float prijs;
    public int aantal;

    // constructor
    public Boek(string titel, float prijs, int aantal)
    {
        this.titel = titel;
        this.prijs = prijs;
        this.aantal = aantal;
    }
}
```

```
static void Main(string[] args)
{
    Boek boek1 = new Boek("Joe speedboot", 10.00f, 5);
    boek1.aantal = 6;
    boek1.prijs = 8.99f;
}
```

Private fields/methods

- Private fields/methods zijn alleen in de class zelf te gebruiken (afscherming)

```
class Boek
{
    // member fields
    public string titel;
    public float prijs;
    private int aantal;

    // constructor
    public Boek(string titel,
    {
        this.titel = titel;
        this.prijs = prijs;
        this.aantal = aantal;
    }
```

```
public void WijzigVoorraad(int aantal)
{
    if (aantal >= 0)
        this.aantal = aantal;
}
```

```
public int GeefAantal()
static void Main(string[] args)
{
    Boek boek1 = new Boek("Joe speedboot", 10.00f, 5);
    boek1.prijs = 8.99f;
    boek1.aantal = 6;
}
```

int Boek.aantal

Error:

'BoekHandel.Boek.aantal' is inaccessible due to its protection level

Private fields/methods

```
class Weekblad : Boek
```

```
{  
    // member fields  
    public DayOfWeek uitgifteDag;  
  
    // constructor  
    public Weekblad(string titel, float prijs, int aantal, DayOfWeek uitgifteDag)  
        : base(titel, prijs, aantal)  
    {  
        this.uitgifteDag = uitgifteDag;  
    }  
  
    public override string ToString()  
    {  
        return "[Weekblad] \"" + titel + "\", " + prijs.ToString("0.00") + ", "  
            + aantal);  
    }  
}
```

int Boek.aantal

Error:

'BoekHandel.Boek.aantal' is inaccessible due to its protection level

Een afgeleide class heeft ook geen toegang tot private members in de base class.

Protected fields/methods

- Protected fields/methods zijn in de class zelf en in afgeleide classes te gebruiken

```
class Boek
{
    // member fields
    public string titel;
    public float prijs;
    protected int aantal;

    // constructor
    public Boek(string titel, float prijs, int aantal)
    {
        this.titel = titel;
        this.prijs = prijs;
        this.aantal = aantal;
    }
}
```

```
class Weekblad : Boek
{
    // member fields
    public DayOfWeek uitgifteDag;

    // ...

    public override string ToString()
    {
        return "[Weekblad] '" + titel + "', " +
            prijs.ToString("0.00") + ", " +
            aantal.ToString();
    }
}
```

Properties

- Properties zijn fields met 'toegangsregeling'
- Een property bestaat uit een set (write) en een get (read) methode / accessor
- Properties zonder set accessor zijn read-only
- Properties zonder get accessor zijn write-only
- Properties 'should be lightweight' (*geen langdurige operatie*)

Properties - voorbeeld 1

```
class Boek
{
    private string titel;

    public string Titel
    {
        get { return titel; }
        set { titel = value; }
    }
}
```

Extra voordeel van een property: je kunt nu ook een breakpoint plaatsen, als je bv wilt weten wanneer een field gewijzigd wordt.

```
static void Main(string[] args)
{
    Boek boek1 = new Boek();

    // bij het schrijven van de 'Titel' property wordt de 'set' accessor aangeroepen
    boek1.Titel = "Joe speedboot II";

    // bij het lezen van de 'Titel' property wordt de 'get' accessor aangeroepen
    Console.WriteLine("Titel van het boek is {0}.", boek1.Titel);
}
```


Properties - voorbeeld 2

```
class Boek
{
    private string titel;
    private int aantal;


    public int AantalExemplaren
    {
        get { return aantal; }
        set { aantal = value; }
    }

    public string Titel
    {
        get { return titel; }
        set { titel = value; }
    }
}
```

```
class Boek
{
    private string titel;
    private int aantal;

    public int AantalExemplaren
    {
        get { return aantal; }
        set
        {
            if (value >= 0)
                aantal = value;
        }
    }

    public string Titel
    {
        get { return titel; }
        set { titel = value; }
    }
}
```



Auto-implemented properties

- Auto-implemented properties: verkorte notatie, geen expliciete member fields (geen 'backing field')

```
class Boek
{
    // automatic properties
    public string Titel { get; set; }
    public float Prijs { get; set; }
    public int AantalExemplaren { get; set; }

    // constructor
    public Boek(string titel, float prijs, int aantal)
    {
        Titel = titel;
        Prijs = prijs;
        AantalExemplaren = aantal;
    }
}
```

*Latere interne
wijzigingen
veranderen de
interface niet!*

*Dus als bv
property Prijs
extra code krijgt
in de set, dan zal
dat voor de
buitenwereld geen
gevolgen hebben.*

Readonly properties

```
class Boek
{
    // backing field
    private float prijs;

    // readonly properties
    public string Titel { get; private set; }
    public float Prijs { get { return prijs; } }

    // read/write properties
    public int AantalExemplaren { get; set; }

    // constructor
    public Boek(string titel, float prijs, int aantal)
    {
        this.Titel = titel;
        this.prijs = prijs;
        this.AantalExemplaren = aantal;
    }
}
```

Property Titel is readonly (voor de buitenwereld) omdat de set alleen binnen de class te gebruiken is.

In bv de constructor kan de Titel wel ingesteld worden (omdat dit binnen de class is).

'Calculated' properties

```
class Boek
{
    public string Titel { get; set; }    // automatic property

    private int aantal;                 // backing field voor property Aantal
    public int Aantal                    // property
    {
        get { return aantal; }
        set
        {
            if (value >= 0)
                aantal = value;
        }
    }

    public float Prijs { get; private set; }    // read-only property

    public float TotalWaarde              // calculated property
    {
        get
        {
            return Prijs * Aantal;
        }
    }
}
```

calculated property:
Hier worden andere
properties gebruikt (om de
return waarde te bepalen).

Huiswerk voor volgende week

- Bestudeer de aangegeven paragrafen uit het 'Yellow Book' (zie Moodle)
- Week 3 opdrachten (zie Moodle)