# Inleiding

In dit document vind je één of meerdere oefeningen die passen bij het onderwerp **Het klassediagram**.

## Oefening 1

In de sheets bij het onderwerp ‘Het klassediagram’, wordt een ontwerp van een Employee klasse vertaald naar C# code. Dit vertalen kan echter nog sneller door gebruik te maken van ‘automatic properties’ (zie HFC# H5 pag. 205.). Bij gebruik van automatic properties is het niet nodig datavelden in de code te declareren voor de opslag van de gegevens waartoe de property toegang biedt. Doordat het declareren van datavelden bij gebruik van automatic properties niet nodig is, is er minder kans op fouten in de code (hoe meer code, hoe groter de kans op fouten) en is je code overzichtelijker. Herschrijf onderstaande klasse mbv ‘automatic properties’. En vergelijk de lengte van de code.

class Employee

{

private string name;

private decimal salary;

public string Name

{

get { return name; }

set { name = value; }

}

public decimal Salary { get { return salary; } }

public Employee(string name, decimal salary)

{

this.name = name;

this.salary = salary;

}

public void RaiseSalary(int percentage)

{

salary = salary + (salary \* percentage / 100);

}

}

## Oefening 2

Vertaal het gegeven klassediagram naar code voor C# klassen. Maak daarna een C# Forms applicatie die de functionaliteit van de klassen in het klassediagram gebruikt en test.



## Oefening 3

Vertaal het gegeven klassediagram naar code voor C# klassen. Maak daarna een C# Forms applicatie die de functionaliteit van de klassen in het klassediagram gebruikt en test.



## Oefening 4

In deze oefening is de multipliciteit van de (navigatie) associatie aangepast.

Wat verandert er nu aan de benodigde operaties (constructoren / methoden)?

Bedenk zelf welke operaties je nodig hebt. Vul deze in, in het klassediagram. Vertaal vervolgens je klassediagram naar code voor C# klassen. Maak daarna een C# Forms applicatie die de functionaliteit van de klassen in het klassediagram gebruikt en test.

