



samen sterk voor werk

C# 2013

Programming Fundamentals Taken

Deze cursus is eigendom van VDAB Competentiecentra ©

Peoplesoftcode:

Wettelijk depot:

versie: 30/11/2014

INHOUD

1	Variabelen, constanten en bewerkingen	7
1.1	Conversie Celsius → Fahrenheit	7
1.2	Controle BTW-nummer	7
2	Selecties	8
2.1	Kortingsbon	8
2.2	Schrikkeljaar	8
3	Iteraties	9
3.1	Kleinste, grootste en gemiddelde	9
3.2	Priemgetal	9
3.3	IBAN rekeningnummer generator	9
3.4	Controle IBAN rekeningnummer	10
4	Arrays	11
4.1	Codeerprogramma	11
5	Classes, objects, objectvariables, constructors	12
5.1	Oefening Bank	12
5.1.1	De class Rekening	12
5.1.2	Het hoofdprogramma	12
5.2	Oefening Voertuigen	12
5.2.1	De class Voertuig	12
5.2.2	Het hoofdprogramma	12
6	Inheritance	13
6.1	Oefening Bank	13
6.1.1	De class Zichtrekening	13
6.1.2	De class SpaarRekening	13
6.1.3	Het hoofdprogramma	13
6.2	Oefening Voertuigen	13
6.2.1	De class Vrachtwagen	13

6.2.2	De class Personenwagen.....	13
6.2.3	Het hoofprogramma	13
7	Abstract classes, abstract members, static members	14
7.1	Oefening Bank.....	14
7.1.1	De abstracte class Rekening.....	14
7.1.2	Static member Intrest	14
7.2	Oefening Voertuigen.....	14
7.2.1	De abstracte class Voertuig.....	14
7.2.2	De method GetKyotoScore()	14
8	Inheritance polymorphisme	15
8.1	Oefening Bank.....	15
8.1.1	Rekeningen afbeelden	15
8.2	Oefening Voertuigen.....	15
8.2.1	Voertuigen afbeelden	15
9	Aggregation.....	16
9.1	Oefening Bank.....	16
9.1.1	De class Klant	16
9.1.2	De class Rekening.....	16
9.1.3	De class SpaarRekening.....	16
9.1.4	De class ZichtRekening.....	16
9.1.5	Het hoofdprogramma	16
10	Interfaces	17
10.1	Oefening Bank.....	17
10.1.1	De interface ISpaarmiddel	17
10.1.2	De class Rekening.....	17
10.1.3	De class Kasbon.....	17
10.1.4	Het hoofdprogramma.	17
10.2	Oefening Voertuigen.....	17
10.2.1	De interface IVervuiler	17
10.2.2	De class Stookketel	17
10.2.3	Het hoofdprogramma	18

10.2.4	De interfaces IPrivaat en IMilieu.....	18
11	Delegates en events	19
11.1	Oefening Bank	19
11.1.1	De class Rekening	19
11.1.2	De class BankBediende	19
11.1.3	Het hoofdprogramma	19
12	Exceptions	20
12.1	Oefening Bank	20
12.1.1	De class Rekening	20
12.1.2	De class ZichtRekening.....	20
12.1.3	De class SpaarRekening	20
12.1.4	De class Kasbon.....	20
13	Lambda expressies	21
14	LINQ	22
14.1	De class Plant.....	22
14.2	Het hoofdprogramma	22
15	Files and streams – I/O	24
15.1	De class Tweet	24
15.2	De class Tweets	24
15.3	De class Twitter	25
15.4	Het hoofdprogramma	25
16	Voorbeeldoplossing: Variabelen, constanten en bewerkingen	26
16.1	Conversie Celsius Fahrenheit	26
16.2	Controle BTW-nummer	26
18	Voorbeeldoplossing: selecties	27
18.1	Kortingsbon	27
18.2	Schrikkeljaar	27
19	Voorbeeldoplossing: iteraties.....	29

19.1	Kleinste, grootste en gemiddelde	29
19.2	Priemgetal	30
19.3	IBAN rekeningnummer generator	30
19.4	Controle IBAN rekeningnummer	31
20	Voorbeeldoplossing: arrays	33
20.1	Codeerprogramma	33
21	Voorbeeldoplossing: classes, objects, objectvariables, constructors ...	34
21.1	Oefening Bank	34
21.1.1	De class Rekening	34
21.1.2	Het hoofdprogramma	35
21.2	Oefening Voertuigen	36
21.2.1	De class Voertuig	36
21.2.2	Het hoofdprogramma	38
22	Voorbeeldoplossing: inheritance	39
22.1	Oefening Bank	39
22.1.1	De class Rekening	39
22.1.2	De class ZichtRekening	39
22.1.3	De class SpaarRekening	40
22.1.4	Het hoofdprogramma	40
22.2	Oefening Voertuigen	41
22.2.1	De class Voertuig	41
22.2.2	De class Vrachtwagen	41
22.2.3	De class Personenwagen	42
22.2.4	Het hoofdprogramma	43
23	Voorbeeldoplossing: abstract classes, abstract members, static members	44
23.1	Oefening Bank	44
23.1.1	De abstracte class Rekening	44
23.1.2	Static member Intrest in de class SpaarRekening	44
23.2	Oefening Voertuigen	45

23.2.1	De abstracte class Voertuig	45
23.2.2	De class Vrachtwagen	45
23.2.3	De class Personenwagen	45
24	Voorbeeldoplossing: inheritance polymorphisme	46
24.1	Oefening Bank	46
24.1.1	Rekeningen afbeelden	46
24.2	Oefening Voertuigen	46
24.2.1	Voertuigen afbeelden	46
25	Voorbeeldoplossing: aggregation	47
25.1	Oefening Bank	47
25.1.1	De class Klant	47
25.1.2	De class Rekening	48
25.1.3	De class SpaarRekening	49
25.1.4	De class ZichtRekening.....	49
25.1.5	Het hoofdprogramma	49
26	Voorbeeldoplossing: interfaces	51
26.1	Oefening Bank	51
26.1.1	De interface ISpaarmiddel	51
26.1.2	De class Rekening	51
26.1.3	De class Kasbon.....	51
26.1.4	Het hoofdprogramma	53
26.2	Oefening Voertuigen	54
26.2.1	De interface IVervuiler.....	54
26.2.2	de class Vrachtwagen	54
26.2.3	de class Personenwagen.....	54
26.2.4	De class Stookketel	54
26.2.5	Het hoofdprogramma	55
26.2.6	De interfaces IPrivaat en IMilieu.....	55
27	Voorbeeldoplossing: delegates en events	57
27.1	Oefening Bank	57
27.1.1	De class Rekening	57

27.1.2	De class BankBediende	58
27.1.3	Het hoofdprogramma	59
28	Voorbeeldoplossing: exceptions.....	61
28.1	Oefening Bank.....	61
28.1.1	De class Rekening.....	61
28.1.2	De class ZichtRekening	62
28.1.3	De class SpaarRekening.....	62
28.1.4	De class Kasbon.....	63
28.1.5	Het hoofdprogramma	64
29	Voorbeeldoplossing: lambda expressies.....	65
29.1	Oplossing met een expression lambda.....	65
29.2	Oplossing met een statement lambda	65
29.3	Oplossing met Func<>.....	66
29.4	Oplossing met Action<>	67
30	Voorbeeldoplossing: linq.....	69
30.1	De class Plant	69
30.2	Het hoofdprogramma	69
31	Voorbeeldoplossing: Files and streams – I/O.....	75
31.1	De class Tweet	75
31.2	De class Tweets.....	76
31.3	De class Twitter.....	76
31.4	Het hoofdprogramma	78
32	Colofon.....	80

1 Variabelen, constanten en bewerkingen

1.1 Conversie Celsius → Fahrenheit

Bereken de gemiddelde lichaamstemperatuur in graden Fahrenheit als je weet dat deze in Celsius 37° is. Om graden Celsius om te zetten naar Fahrenheit vermenigvuldig je deze met 9/5 en tel je er 32 bij op.

Gebruik een constante `GemLichTempCelsius` en geef deze de waarde 37. Toon het resultaat op het scherm.

1.2 Controle BTW-nummer

Controleer de geldigheid van het BTW-nummer 213.252.520.

Om geldig te zijn moet je het getal nemen gevormd door de eerste 7 cijfers en delen door 97. De rest moet afgetrokken worden van 97 en dit resultaat moet dan gelijk zijn aan de laatste 2 cijfers van het BTW-nummer.

Stop het BTW-nummer in een constante. Voer dan de nodige berekeningen uit en toon false indien het BTW-nummer ongeldig is en true als het wel geldig is. Gebruik nog geen if-instructie hiervoor.

Voorbeeld berekening:

Rest van 2132525 gedeeld door 97 is 77. Deze rest (77) aftrekken van 97 geeft 20.

2 Selecties

2.1 Kortingsbon

Een kledingzaak geeft een kortingsbon afhankelijk van je aankoopgedrag van het afgelopen jaar. Heb je tot 25 euro aangekocht, dan krijg je een bon ter waarde van 1% van je aankopen. Bij een aankoop van meer dan 25 euro tot 50 euro is dit 2%, tot 100 euro 3% en indien hoger dan 100 euro is het 5%. Voer een aankoopbedrag in, bereken en toon de waarde van de bijhorende korting.

2.2 Schrikkeljaar

Bereken of een ingevoerd jaartal een schrikkeljaar is of niet. Alle jaren die deelbaar zijn door 4 zijn schrikkeljaren behalve de jaren die deelbaar zijn door 100 en niet deelbaar zijn door 400.

3 Iteraties

3.1 Kleinste, grootste en gemiddelde

Voer een aantal positieve getallen in en bereken er de kleinste, grootste en gemiddelde waarde van. Beëindig de invoer van de getallen met -1.

3.2 Priemgetal

Bereken of een ingevoerd getal een priemgetal is of niet. Doe dit door het getal te delen door alle getallen die liggen tussen 1 en het getal zelf.

Toon de delers op het scherm. Besluit dan of het ingegeven getal een priemgetal is of niet.

3.3 IBAN rekeningnummer generator

Het IBAN (International Bank Account Number) bankrekeningnummer heeft een vaste lengte per land en bestaat in België uit 16 tekens.

O.w.v. de leesbaarheid wordt het voorgesteld in 4 groepen van 4 tekens:

BE73 0631 5475 6360

een landcode (2 letters, BE) + een controlegetal (2 cijfers) + het nationaal rekeningnummer (12cijfers).

Om het nationaal rekeningnummerformaat (063-1547563-60) om te zetten in een IBAN rekeningnummerformaat kan je de volgende werkwijze toepassen:

- Schrap alle niet-alfanumerieke tekens: 063-1547563-60 → 063154756360
- Voeg achteraan de landcode, gevolgd door "00" toe:
0631547563606360 → 063154756360BE00
- Vervang de letters door cijfers, meer bepaald door hun positie in het alfabet, vermeerderd met 9 (A=10, B=11, ... Z=35):
063154756360BE00 → 063154756360111400
- Deel dit getal door 97 en trek de rest van deze deling af van 98. Wanneer dit resultaat slechts één cijfer is, laat dit dan voorafgaan door een 0 (nul): 73
- Dit cijfer is het controlegetal dat volgt op de landcode in het IBAN rekeningnummer:
BE73 0631 5475 6360
- Schrijf een programma dat het IBAN bankrekeningnummer genereert op basis van een opgegeven Belgisch bankrekeningnummer.

Voorbeelden:

	IBAN
001-9002000-88	BE56 0019 0020 0088
063-0255500-37	BE35 0630 2555 0037
310-1234567-37	BE35 3101 2345 6737
645-1000001-63	BE40 6451 0000 0163
679-0021820-92	BE12 6790 0218 2092
739-0102134-91	BE23 7390 1021 3491

3.4 Controle IBAN rekeningnummer

Schrijf een programma om te controleren of een IBAN rekeningnummer een geldig nummer is.

Hiervoor kan je als volgt tewerk gaan:

- Schrap alle spaties uit het IBAN nummer → BE73063154756360
- Verplaats de eerste 4 tekens naar uiterst rechts → 063154756360BE73
- Vervang de letters door cijfers, meer bepaald door hun positie in het alfabet, vermeerderd met 9 (A=10, B=11, ... Z=35) → 063154756360111473
- Deel dit getal door 97. Voor een geldig IBAN rekeningnummer moet de rest van deze deling gelijk zijn aan 1

4 Arrays

4.1 Codeerprogramma

Maak een codeerprogramma. Een ingevoerd stuk tekst moet omgezet worden gebruik makend van een sleutel. Deze sleutel bestaat uit de 26 letters van het alfabet maar niet gesorteerd. Komt er een A voor in het ingevoerde woord dan moet deze omgezet worden naar de eerste letter uit de sleutel. Een E moet omgezet worden naar de 5^e letter uit de sleutel, een K naar de 11^e letter uit de sleutel.

Schematisch :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Q	S	P	A	T	V	X	B	C	R	J	Y	E	D	U	O	H	Z	G	I	F	L	N	W	K	M

Stop de codeersleutel in een array van chars. Zet de ingevoerde tekst om naar hoofdletters vooraleer om te zetten naar code. Tekens die niet in de codeersleutel voorkomen (leestekens, spaties,...) moeten niet gecodeerd worden maar gewoon worden overgenomen.

De tekst *DIT IS GEHEIM* wordt op deze manier omgezet in *ACI CG XTBTCE*, de tekst *Amaai, mijn ratje!* wordt omgezet naar *QEQQC, ECRD ZQIRT!*.

5 Classes, objects, objectvariables, constructors

5.1 Oefening Bank

5.1.1 De class Rekening

Maak in het project CSharpPFOefenmap een class *Rekening* met de volgende eigenschappen:

Rekeningnummer

Saldo

Creatiedatum

Test het rekeningnummer op geldigheid in het set-gedeelte van de property *Rekeningnummer*. De *Creatiedatum* mag niet vóór het jaar 1900 zijn.

Voorzie in de class een method *Afbeelden()* die de eigenschappen op het scherm toont.

Voorzie in de class een method *Storten()* met een parameter *bedrag*. Tel het bedrag op bij het saldo.

Voorzie een geparametriseerde constructor.

5.1.2 Het hoofdprogramma

In het hoofdprogramma maak je een object aan van de class *Rekening*.

Je geeft het rekeningnummer, het saldo en de creatiedatum mee als parameter.

Toon de gegevens van de rekening op het scherm.

Stort 100 euro op de rekening en toon opnieuw de rekeninggegevens op het scherm.

5.2 Oefening Voertuigen

5.2.1 De class Voertuig

Ontwerp een class *Voertuig* met de volgende properties:

Polishouder (string)

Kostprijs (decimal)

Pk (int)

GemiddeldVerbruik (float)

Nummerplaat (string)

De default polishouder en nummerplaat zijn “onbepaald”. Voor de kostprijs, het aantal pk en het gemiddeld verbruik is de default waarde 0 (nul), een negatieve waarde is niet toegelaten.

Voorzie een default constructor en een geparametriseerde constructor.

Voorzie in de class een method *Afbeelden()* die de gegevens van een voertuig op het scherm toont.

5.2.2 Het hoofdprogramma

In het hoofdprogramma creëer je *Voertuig* objecten en toon je de gegevens van deze voertuigen op het scherm.

6 Inheritance

6.1 Oefening Bank

6.1.1 De class Zichtrekening

Maak een class *ZichtRekening*, gebaseerd op de class *Rekening*. Geef deze class een extra eigenschap *MaxKrediet*. Deze property geeft per zichtrekening aan hoe ver deze in het rood mag gaan. Deze property mag niet positief zijn. Overschrijf de method *Afbeelden()*.

6.1.2 De class SpaarRekening

Maak een class *SpaarRekening*, gebaseerd op de class *Rekening*. Maak een property *Intrest*. Deze property mag niet negatief zijn. Overschrijf ook hier de method *Afbeelden()*.

6.1.3 Het hoofdprogramma

Maak een spaarrekening aan.

Stort 1000 euro en beeld de gegevens af.

Maak vervolgens een zichtrekening aan.

Stort 125 euro en beeld de gegevens af.

6.2 Oefening Voertuigen

6.2.1 De class Vrachtwagen

Ontwerp een class *Vrachtwagen*, afgeleid van *Voertuig* en met een extra property *MaximumLading* (float). De default maximum lading is 10000 kg en mag niet negatief zijn.

Zorg ervoor dat de gegevens van de Vrachtwagen kunnen getoond worden op het scherm.

6.2.2 De class Personenwagen

Creëer een class *Personenwagen*, afgeleid van *Voertuig* en met de extra properties *AantalDeuren* (int, default 4) en *AantalPassagiers* (int, default 5).

Deze properties mogen niet negatief zijn.

Zorg ervoor dat de gegevens van de personenwagen kunnen getoond worden op het scherm.

6.2.3 Het hoofdprogramma

Creëer een *Vrachtwagen* object en een *Personenwagen* object en toon de gegevens van deze voertuigen op het scherm.

7 Abstract classes, abstract members, static members

7.1 Oefening Bank

7.1.1 De abstracte class Rekening

Maak van de class *Rekening* uit de vorige oefening een abstract class.

7.1.2 Static member Intrest

Alle spaarrekeningen krijgen een gelijke intrestvoet. Maak van de *Intrest* property van *SpaarRekening* een **static** property.

7.2 Oefening Voertuigen

7.2.1 De abstracte class Voertuig

Maak van de class *Voertuig* uit de vorige oefening een abstract class

7.2.2 De method `GetKyotoScore()`

Voeg aan de class *Voertuig* de abstracte method *GetKyotoScore()* (returntype *double*) toe.
Deze method geeft de Kyoto-score van een voertuig.

Voor een personenwagen is de Kyoto-score gelijk aan het gemiddeld verbruik vermenigvuldigd met het aantal pk, gedeeld door het aantal passagiers.

Voor een vrachtwagen is de Kyoto-score gelijk aan het gemiddeld verbruik vermenigvuldigd met het aantal pk, gedeeld door de lading in ton.

Werk deze method uit in de betreffende classes.

8 Inheritance polymorphisme

8.1 Oefening Bank

8.1.1 Rekeningen afbeelden

Creëer een array van twee rekeningen. Initialiseer de eerste rekening als een spaarrekening gebruik makend van een geparametriseerde constructor. De tweede rekening is een zichtrekening, eveneens te initialiseren met een geparametriseerde constructor. Overloop de array met een foreach opdracht en roep de method *Afbeelden()* op om het polymorphisme te demonstreren.

8.2 Oefening Voertuigen

8.2.1 Voertuigen afbeelden

Vul een array op met één personenwagen en één vrachtwagen. Toon de gegevens en de kyoto-score van beide voertuigen op het scherm.

9 Aggregation

9.1 Oefening Bank

9.1.1 De class Klant

Maak een class *Klant* met als properties *Voornaam* en *Familienaam*. Maak in deze class ook een geparametriseerde constructor (met een parameter *voornaam* en een parameter *familienaam*). Maak in deze class ook een method *Afbeelden()* die de properties van de klant afbeeldt.

9.1.2 De class Rekening

Breid de class *Rekening* uit met een property *Eigenaar*. Deze property heeft als type *Klant*. Met deze property houd je per rekening bij welke klant eigenaar is van die rekening.

Breid de constructor uit met een extra parameter voor de eigenaar.

Roep in de method *Afbeelden()* van de class *Rekening()* ook de method *Afbeelden()* van de eigenaar op.

9.1.3 De class SpaarRekening

Breid de constructor uit met een extra parameter voor de eigenaar.

9.1.4 De class ZichtRekening

Breid de constructor uit met een extra parameter voor de eigenaar.

9.1.5 Het hoofdprogramma

Maak een *Klant* object. Maak een *SpaarRekening* en een *ZichtRekening* object die deze *Klant* als eigenaar hebben. Beeld de rekeningen af.

10 Interfaces

10.1 Oefening Bank

10.1.1 De interface ISpaarmiddel

Maak een interface *ISpaarmiddel*. Beschrijf in deze interface de method *Afbeelden()*.

10.1.2 De class Rekening

Implementeer de interface *ISpaarmiddel* in de class *Rekening*.

10.1.3 De class Kasbon

Maak een class *Kasbon*. Deze heeft de volgende properties:

AankoopDatum (mag niet voor 1900 zijn)

Bedrag (moet groter dan nul zijn)

Looptijd (in jaren) (moet groter dan nul zijn)

Intrest (moet groter dan nul zijn)

Eigenaar (van het type klant).

Maak een geparametriseerde constructor. Implementeer de interface *ISpaarmiddel*.

10.1.4 Het hoofdprogramma.

Maak een *Klant* object. Maak daarna een array van het type *ISpaarmiddel* met 3 elementen. Het eerste element is een *ZichtRekening* met het *Klant* object als eigenaar. Het tweede element is een *SpaarRekening* met het *Klant* object als eigenaar. Het derde element is een *Kasbon* met het *Klant* object als eigenaar. Doorloop één voor één de elementen van de *ISpaarmiddel* array en pas voor ieder element de method *Afbeelden()* toe.

10.2 Oefening Voertuigen

10.2.1 De interface IVervuiler

Maak een interface *IVervuiler*. Beschrijf in deze interface de method *GeefVervuiling()* met als returntype double.

Implementeer deze interface in de classes *Vrachtwagen* en *Personenwagen*. Voor de personenwagen is de vervuiling gelijk aan de Kyoto-score maal 5, voor de vrachtwagen de Kyoto-score maal 20.

10.2.2 De class Stookketel

Ontwerp een class *Stookketel*. Deze class heeft een property *CONorm* (float) en implementeert de interface *IVervuiler*. De vervuiling is de *CONorm* maal 100.

10.2.3 Het hoofdprogramma

In het hoofdprogramma definieer je een array van het type *IVervuiler*. Plaats hierin een aantal objecten die de interface *IVervuiler* implementeren. Overloop de array en toon van elk object het resultaat van de method *GeefVervuiling()*.

10.2.4 De interfaces *IPrivaat* en *IMilieu*

Ontwerp twee interfaces: *IPrivaat* en *IMilieu*. In de interface *IPrivaat* voorzie je een method *GeefPrivateData()*, in de interface *IMilieu* een method *GeefMilieuData()*, beide met returntype string.

Werk deze interfaces uit in de class *Voertuig* door in de method *GeefPrivateData ()* enkel de polishouder en de nummerplaat weer te geven. Werk ook de method *GeefMilieuData()* uit, waarin je de properties pk, kostprijs en verbruik weergeeft.

In het hoofdprogramma definieer je een array van het type *IPrivaat*. Stop een aantal voertuigen (vrachtwagens en personenwagens) in deze array en laat alle private gegevens van deze voertuigen zien.

Definieer ook een array van het type *IMilieu* met eveneens een aantal voertuigen erin en ga na welke methods er nu beschikbaar zijn.

11 Delegates en events

11.1 Oefening Bank

Wanneer een klant geld stort op een rekening of geld afhaalt van een rekening, wordt er een rekeninguittreksel op het scherm getoond. Dit rekeninguittreksel toont:

- de rekeninggegevens
- het vorige saldo van de rekening
- het bedrag van de storting/afhaling
- het nieuwe saldo van de rekening

Als het saldo van de rekening bij afhaling echter ontoereikend is, wordt dit aan de klant signaleerd en gaat de transactie niet door. Er wordt dan geen rekeninguittreksel getoond, maar wel een boodschap met het maximum af te halen bedrag voor de betreffende rekening. Het maximum af te halen bedrag is het *Saldo* van de rekening (bij een zichtrekening hoef je geen rekening te houden met het *MaxKrediet*).

Voorzie deze events in de class *Rekening*.

11.1.1 De class *Rekening*

Voorzie een property *VorigSaldo* waarin het oude saldo van de rekening bijgehouden wordt.

Voeg een method *Afhalen()* toe.

Voeg de twee events *RekeningUittreksel* en *SaldoInHetRood* toe. Deze zijn gebaseerd op een delegate (bvb. met de naam *Transactie*) die je eveneens in deze class kan definiëren.

Zorg ervoor dat deze events veroorzaakt worden waar nodig. Bij het oproepen van de events wordt de betreffende rekening als parameter meegegeven.

11.1.2 De class *BankBediende*

Creëer een nieuwe class *BankBediende* met de properties *Voornaam* en *Naam*.

Een *BankBediende* object zal op de events *RekeningUittreksel* en *SaldoInHetRood* reageren.

Voorzie de twee methods om een rekeninguittreksel of een boodschap (bij ontoereikend saldo) op het scherm te tonen.

11.1.3 Het hoofdprogramma

Creëer een zichtrekening, een spaarrekening en een bankbediende object. Zorg ervoor dat de beide events optreden en de bankbediende op de events reageert.

12 Exceptions

12.1 Oefening Bank

Pas de volgende classes aan zodat er foutmeldingen getoond worden wanneer de gebruiker een verkeerde waarde toekent aan een property.

12.1.1 De class Rekening

Foutmelding bij een ongeldig rekeningnummer.

Foutmelding wanneer de creatiedatum vóór 1-1-1990 is.

12.1.2 De class ZichtRekening

Foutmelding bij een positieve waarde voor de property *MaxKrediet*.

12.1.3 De class SpaarRekening

Foutmelding bij een negatieve waarde voor de property *Intrest*.

12.1.4 De class Kasbon

Foutmeldingen bij een negatieve waarde voor de properties *Intrest*, *Looptijd* en *Bedrag*.

Foutmelding wanneer de aankoopdatum vóór 1-1-1990 is.

13 Lambda expressies

Schrijf een programma waarin de getallen van een array in een bepaalde kleur op het scherm getoond worden:

- de even getallen in het groen, de oneven getallen in het rood en vervolgens
- de positieve getallen in het wit, de negatieve getallen in het geel.

Maak hierbij gebruik van lambda expressies.

Tip:

Gebruik de property *ForegroundColor* van de class *Console* om de kleur van de tekst op het scherm in te stellen bvb. `Console.ForegroundColor = ConsoleColor.Green;`

14 LINQ

14.1 De class Plant

Creëer een nieuwe class Plant met de volgende properties:

- PlantId (int)
- Plantennaam (string)
- Kleur (string)
- Prijs (decimal)
- Soort (string)

14.2 Het hoofdprogramma

Creëer een verzameling Plant objecten (List<Plant>) met de volgende plantengegevens:

PlantID	Plantennaam	Kleur	Prijs	Soort
1	Tulp	rood	0,50	bol
2	Krokus	wit	0,20	bol
3	Narcis	geel	0,30	bol
4	Blauw druifje	blauw	0,20	bol
5	Azalea	rood	3,00	heester
6	Forsythia	geel	2,00	heester
7	Magnolia	wit	4,00	heester
8	Waterlelie	wit	2,00	water
9	Lisdodde	geel	3,00	water
10	Kalmoes	geel	2,50	water
11	Bieslook	paars	1,50	kruid
12	Rozemarijn	blauw	1,25	kruid
13	Munt	wit	1,10	kruid
14	Dragon	wit	1,30	kruid
15	Basilicum	wit	1,50	kruid

Voer de volgende LINQ queries uit op deze verzameling planten:

- Toon plantennaam, kleur en prijs van de witte planten, gesorteerd op prijs.
- Toon het aantal witte planten.
- Bereken de gemiddelde prijs van de heesters en toon deze op het scherm.
- Toon de gemiddelde prijs en de maximumprijs van de kruiden.
- Toon de plantennamen die met de letter "B" beginnen.
- Toon een lijst van de verschillende plantenkleuren op het scherm.
- Toon de plantennamen per kleur op het scherm.
- Toon per soort de maximum plantenprijs van die soort.
- Toon de soorten alfabetisch met per soort:
 - het aantal planten van deze soort
 - de namen van de planten van deze soort.
- Toon de namen van de planten gegroepeerd per soort en binnen de soort per kleur.

15 Files and streams – I/O

Schrijf een programma dat een eenvoudige versie van het sociaal medium “twitter” simuleert. Iedere gebruiker of twitteraar kan op elk moment van de dag in maximaal 140 tekens vertellen waar hij/zij mee bezig is, wat hij/zij van plan is of wat hem/haar bezighoudt. Een dergelijk bericht heet een “tweet”.

Deze tweets worden getoond op de gebruikerspagina van andere twittergebruikers – de zogenaamde “followers” – die hebben aangegeven deze te willen ontvangen.

15.1 De class Tweet

Creëer de class *Tweet* die een twitterbericht voorstelt. De class bevat de volgende properties:

- Naam (string): de naam van de twitteraar
- Bericht (string, max. 140 tekens): het bericht
- Tijdstip (DateTime): het tijdstip (datum + tijd) waarop het bericht geplaatst wordt

Overschrijf de method *ToString()*: naam, bericht en tijdstip worden getoond.

Het tijdstip wordt als volgt weergegeven:

- meer dan een dag geleden → de datum bvb. “31-10-2014”
- meer dan een uur maar minder dan een dag geleden → aantal uren geleden bvb. “3 uur geleden”
- minder dan een uur geleden → aantal minuten geleden bvb. “10 minuten geleden”
- minder dan een minuut geleden → het tijdstip bvb. “15:30”

15.2 De class Tweets

Creëer de class *Tweets* die alle berichten van alle gebruikers verzamelt in een property van het type *List<Tweet>*.

Aan deze verzameling kan enkel via de method *AddTweet(Tweet tweet)* een twitterbericht toegevoegd worden. Voorzie deze method.

Uiteraard kunnen de twitterberichten wel opgevraagd worden.

Tip: met de class *ReadOnlyCollection<T>* uit de namespace *System.Collections.ObjectModel* kan je een verzameling *List<T>* omvormen naar een readonly verzameling. Bekijk de documentatie ervan in de MSDN.

15.3 De class Twitter

De tweets worden opgeslagen in een bestand bvb. "twitter.obj".

Creëer hiervoor de class Twitter waarmee je dit bestand kan beheren:

- Een twitterbericht plaatsen: de gebruikersnaam, het bericht en het tijdstip worden geregistreerd.
- Een lijst van alle tweets tonen: de meest recente tweets worden bovenaan getoond.
- Een lijst van tweets van een opgegeven twitteraar tonen: de meest recente tweets worden bovenaan getoond.

15.4 Het hoofdprogramma

De gebruiker krijgt de volgende keuzemogelijkheden:

- Een twitterbericht plaatsen
Aan de gebruiker worden de gebruikersnaam en het bericht gevraagd. De datum van het bericht wordt automatisch ingevuld. De tweet wordt weggeschreven naar het bestand twitter.obj. gebruik hiervoor serialization.
- Alle twitterberichten lezen.
- De twitterberichten van een persoon die door de gebruiker gevolgd wordt, lezen. De naam van deze persoon wordt aan de gebruiker gevraagd.

16 Voorbeeldoplossing: Variabelen, constanten en bewerkingen

16.1 Conversie Celsius Fahrenheit

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        const float GemLichTempCelsius = 37.0F;
        static void Main(string[] args)
        {
            float gemLichTempFahrenheit =
                GemLichTempCelsius * 9.0F / 5.0F + 32.0F;
            Console.WriteLine("Gem. lichaamstemperatuur in graden " +
                "Celsius: {0}", GemLichTempCelsius);
            Console.WriteLine("Gem. lichaamstemperatuur in graden " +
                "Fahrenheit: {0}", gemLichTempFahrenheit);
        }
    }
}
```

Opmerking:

In deze oplossing worden zogenaamde placeholders {0} gebruikt om een samengestelde string af te beelden i.p.v. de +-operator. Uitleg hierover vind je in de cursus in het hoofdstuk HET TYPE STRING paragraaf EEN SAMENGESTELDE STRING AFBEELDEN.

16.2 Controle BTW-nummer

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        const ulong BtwNummer = 213252520UL;
        static void Main(string[] args)
        {
            ulong deeltal = BtwNummer / 100UL;
            int rest = (int)(deeltal % 97UL);
            int controle = (int)(BtwNummer % 100);
            Console.WriteLine(controle == 97 - rest);
        }
    }
}
```

18 Voorbeeldoplossing: selecties

18.1 Kortingsbon

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Voer het aankoopbedrag in: ");
            decimal aankoopBedrag = Convert.ToDecimal(Console.ReadLine());
            decimal kortingsPercentage;
            if (aankoopBedrag < 25m)
                kortingsPercentage = 0.01m;
            else
                if (aankoopBedrag < 50m)
                    kortingsPercentage = 0.02m;
                else
                    if (aankoopBedrag < 100m)
                        kortingsPercentage = 0.03m;
                    else
                        kortingsPercentage = 0.05m;
            Console.WriteLine("Je korting bedraagt: {0} euro"
                             , aankoopBedrag * kortingsPercentage);
        }
    }
}
```

18.2 Schrikkeljaar

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Voer jaartal in: ");
            int jaar = Convert.ToInt32(Console.ReadLine());
            string schrikkel;
            if (jaar % 4 == 0)
                if (jaar % 100 == 0 && jaar % 400 != 0)
                    schrikkel = "geen";
                else
                    schrikkel = "een";
            else
                schrikkel = "geen";
            Console.WriteLine("Het jaar {0} is {1} schrikkeljaar", jaar, schrikkel);
        }
    }
}
```

```
    }  
  }  
}
```

19 Voorbeeldoplossing: iteraties

19.1 Kleinste, grootste en gemiddelde

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            int eenGetal;
            //invoer eerste getal
            do
            {
                Console.Write("Geef een eerste positief geheel getal" +
                    " of -1 (= stoppen) in: ");
                eenGetal = Convert.ToInt32(Console.ReadLine());
            }
            while (eenGetal != -1 && eenGetal < 0);

            int kleinste = eenGetal, grootste = eenGetal, totaal = 0, aantal = 0;

            while (eenGetal != -1)
            {
                if (eenGetal >= 0)
                {
                    if (eenGetal < kleinste)
                        kleinste = eenGetal;
                    if (eenGetal > grootste)
                        grootste = eenGetal;
                    totaal += eenGetal;
                    aantal++;
                }
                else
                {
                    Console.WriteLine("Enkel positieve getallen " +
                        "zijn toegelaten,");
                    Console.WriteLine("eindigen met -1");
                }
                Console.Write("Geef een volgende positief geheel getal in " +
                    "(-1 = stoppen): ");
                eenGetal = Convert.ToInt32(Console.ReadLine());
            }

            if (aantal > 0)
            {
                Console.WriteLine("Het kleinste getal: {0}", kleinste);
                Console.WriteLine("Het grootste getal: {0}", grootste);
                Console.WriteLine("Het gemiddelde: {0}", (double)totaal / aantal);
            }
        }
    }
}
```

```

        else
            Console.WriteLine("Er werden geen getallen ingevoerd.");
    }
}

```

19.2 Priemgetal

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            int delers = 0;
            Console.Write("Voer een pos. geheel getal in groter dan nul: ");
            int getal = Convert.ToInt32(Console.ReadLine());
            if (getal > 0)
            {
                for (int teller = 2; teller <= getal - 1; teller++)
                    if (getal % teller == 0)
                    {
                        Console.WriteLine("Getal is deelbaar door {0}", teller);
                        delers++;
                    }
                if (delers > 0)
                    Console.WriteLine("Het getal is geen priemgetal.");
                else
                    Console.WriteLine("Het getal is een priemgetal.");
            }
            else
                Console.WriteLine("Verkeerde invoer.");
        }
    }
}

```

19.3 IBAN rekeningnummer generator

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        const string ALFABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        static void Main(string[] args)
        {
            string belgischRekeningNr, ibanRekeningNr,
                ibanRekeningNrControle, controlegetal, controleNr;

```



```

        ulong rest97;

        belgischRekeningNr = "739-0102134-91";
        ibanRekeningNrControle = belgischRekeningNr.Replace("-", "") + "BE00";

        //vervang letters door cijfers: A=10, B=11, ...
        controleNr = VervangLetters(ibanRekeningNrControle);

        //IBAN rekeningnummer samenstellen
        rest97 = ulong.Parse(controleNr) % 97ul;
        controlegetal = (98 - rest97).ToString();

        ibanRekeningNr = "BE" +
            (controlegetal.Length == 2 ? controlegetal : "0" + controlegetal) +
            ibanRekeningNrControle.Substring(0, banRekeningNrControle.Length - 4);

        ibanRekeningNr = ibanRekeningNr.Insert(12, " ")
            .Insert(8, " ").Insert(4, " ");

        Console.WriteLine("rekeningnummer: {0} als IBAN rekeningnummer: {1}",
            belgischRekeningNr, ibanRekeningNr);
    }

    private static string VervangLetters(string nummer)
    {
        char teken;
        string nr = string.Empty;
        for (int teller = 0; teller < nummer.Length; teller++)
        {
            teken = nummer[teller];
            if (teken >= 'A' && teken <= 'Z')
            {
                nr += ALFABET.IndexOf(teken) + 10;
            }
            else
            {
                nr += teken;
            }
        }
        return nr;
    }
}

```

19.4 Controle IBAN rekeningnummer

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        const string ALFABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        static void Main(string[] args)
        {
            string ibanRekeningNr, controleNr;

```

```
ibanRekeningNr = "BE23 7390 1021 3491";

controleNr = ibanRekeningNr.Replace(" ", "");
controleNr = controleNr.Substring(4) +
               controleNr.Substring(0, 4);
controleNr = VervangLetters(controleNr);                                     (1)

Console.WriteLine(ulong.Parse(controleNr) % 97ul == 1 ?
    "geldig rekeningnummer" : "geen geldig rekeningnummer");
    }
}
```

(1) zie method *VervangLetters()* van de vorige oefening

20 Voorbeeldoplossing: arrays

20.1 Codeerprogramma

```
using System;
namespace CSharpPF0efenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            char[] sleutel ={'Q','S','P','A','T',
                           'V','X','B','C','R',
                           'J','Y','E','D','U',
                           'O','H','Z','G','I',
                           'F','L','N','W','K','M'};

            Console.Write("Voer je tekst in:");
            string tekst = Console.ReadLine().ToUpper();
            string gecodeerd = string.Empty;
            for (int teller = 0; teller < tekst.Length; teller++)
            {
                if (tekst[teller] >= 'A' && tekst[teller] <= 'Z')
                    gecodeerd += sleutel[(int)tekst[teller] - (int)'A'];
                else
                    gecodeerd += tekst[teller];
            }
            Console.WriteLine("In code: {0}", gecodeerd);
        }
    }
}
```

21 Voorbeeldoplossing: classes, objects, objectvariables, constructors

21.1 Oefening Bank

21.1.1 De class Rekening

```
using System;
namespace CSharpPFOefenmap
{
    public class Rekening
    {
        private readonly DateTime EersteCreatie = new DateTime(1900, 1, 1);
        private ulong nummerValue;
        private decimal saldoValue;
        private DateTime creatieDatumValue;
        public ulong Nummer
        {
            get
            {
                return nummerValue;
            }
            set
            {
                ulong eerste10 = value / 100ul;
                int laatste2 = (int)(value % 100ul);
                if ((int)(eerste10 % 97ul) == laatste2)
                    nummerValue = value;
            }
        }
        public decimal Saldo
        {
            get
            {
                return saldoValue;
            }
            set
            {
                saldoValue = value;
            }
        }
        public DateTime CreatieDatum
        {
            get
            {
                return creatieDatumValue;
            }
            set
            {
                if (value >= EersteCreatie)
```

```

        creatieDatumValue = value;
    }
}

public Rekening(ulong nummer, decimal saldo, DateTime creatieDatum)
{
    Nummer = nummer;
    Saldo = saldo;
    CreatieDatum = creatieDatum;
}
public void Afbeelden()
{
    Console.WriteLine("Rekeningnummer: {0:000-0000000-00}", Nummer);
    Console.WriteLine("Saldo: {0}", Saldo);
    Console.WriteLine("Creatiedatum: {0:dd-MM-yyyy}", CreatieDatum);
}
public void Storten(decimal bedrag)
{
    Saldo += bedrag;
}
}
}

```

21.1.2 Het hoofdprogramma

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Rekening mijnRekening =
                new Rekening(747524091936ul, 0m, DateTime.Today);
            mijnRekening.Afbeelden();
            mijnRekening.Storten(100m);
            mijnRekening.Afbeelden();
        }
    }
}

```

Opmerking:

Wanneer er bij de creatie van het *Rekening* object geen geldig rekeningnummer opgegeven wordt, gebeurt hier niets: de property *Nummer* (dus de interne variabele *nummerValue*) behoudt zijn default waarde (nul). Verder in de cursus vangen we dit op met een foutmelding (zie hoofdstuk EXCEPTIONS).

21.2 Oefening Voertuigen

21.2.1 De class Voertuig

```
using System;
namespace CSharpPFOefenmap
{
    public class Voertuig
    {
        //constructors
        public Voertuig()
            : this("onbepaald", 0m, 0, 0f, "onbepaald")
        {
        }

        public Voertuig(string polishouder, decimal kostprijs,
            int pk, float gemiddeldVerbruik, string nummerplaat)
        {
            this.Polishouder = polishouder;
            this.Kostprijs = kostprijs;
            this.Pk = pk;
            this.GemiddeldVerbruik = gemiddeldVerbruik;
            this.Nummerplaat = nummerplaat;
        }

        //properties
        private string polishouderValue;
        public string Polishouder
        {
            get
            {
                return polishouderValue;
            }
            set
            {
                polishouderValue = value;
            }
        }

        private decimal kostprijsValue;
        public decimal Kostprijs
        {
            get
            {
                return kostprijsValue;
            }
            set
            {
                if (value > 0)
                {
                    kostprijsValue = value;
                }
            }
        }

        private int pkValue;
        public int Pk
        {
            get
            {
                return pkValue;
            }
            set
            {
                pkValue = value;
            }
        }
    }
}
```

```
{
    return pkValue;
}
set
{
    if (value >= 0)
    {
        pkValue = value;
    }
}

private float gemiddeldVerbruikValue;
public float GemiddeldVerbruik
{
    get
    {
        return gemiddeldVerbruikValue;
    }
    set
    {
        if (value >= 0)
        {
            gemiddeldVerbruikValue = value;
        }
    }
}

private string nummerplaatValue;
public string Nummerplaat
{
    get
    {
        return nummerplaatValue;
    }
    set
    {
        nummerplaatValue = value;
    }
}

public void Afbeelden()
{
    Console.WriteLine("Polishouder: {0}", Polishouder);
    Console.WriteLine("Kostprijs: {0}", Kostprijs);
    Console.WriteLine("Aantal pk: {0}", Pk);
    Console.WriteLine("Gemiddeld verbruik: {0}", GemiddeldVerbruik);
    Console.WriteLine("Nummerplaat: {0}", Nummerplaat);
}
}
```

21.2.2 Het hoofdprogramma

```
using System;

namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Voertuig v1 = new Voertuig();
            Voertuig v2 = new Voertuig("Jan", 10000m, 75, 6.0f, "1-ABC-123");
            Voertuig v3 = new Voertuig("Piet", 15000m, 80, -7.5f, "1-DEF-456");
            v1.Afbeelden();
            Console.WriteLine();
            v2.Afbeelden();
            Console.WriteLine();
            v3.Afbeelden();
        }
    }
}
```


22 Voorbeeldoplossing: inheritance

22.1 Oefening Bank

22.1.1 De class Rekening

```
using System;
namespace CSharpPFOefenmap
{
    public class Rekening
    {
        ...
        public virtual void Afbeelden()
        {
            Console.WriteLine("Rekeningnummer: {0:000-0000000-00}", Nummer);
            Console.WriteLine("Saldo: {0}", Saldo);
            Console.WriteLine("Creatiedatum: {0:dd-MM-yyyy}", CreatieDatum);
        }
        ...
    }
}
```

22.1.2 De class ZichtRekening

```
using System;
namespace CSharpPFOefenmap
{
    public class ZichtRekening : Rekening
    {
        private decimal maxKredietValue;
        public decimal MaxKrediet
        {
            get
            {
                return maxKredietValue;
            }
            set
            {
                if (value <= 0m)
                    maxKredietValue = value;
            }
        }

        public ZichtRekening(ulong nummer, decimal saldo,
            DateTime creatieDatum, decimal maxKrediet)
            : base(nummer, saldo, creatieDatum)
        {
            MaxKrediet = maxKrediet;
        }
    }
}
```

```
        public override void Afbeelden()
        {
            base.Afbeelden();
            Console.WriteLine("Max.krediet: {0}", MaxKrediet);
        }
    }
}
```

22.1.3 De class SpaarRekening

```
using System;
namespace CSharpPFOefenmap
{
    public class SpaarRekening : Rekening
    {
        private decimal intrestValue;
        public decimal Intrest
        {
            get
            {
                return intrestValue;
            }
            set
            {
                if (value >= 0m)
                    intrestValue = value;
            }
        }

        public SpaarRekening(ulong nummer, decimal saldo,
            DateTime creatieDatum, decimal intrest)
            : base(nummer, saldo, creatieDatum)
        {
            Intrest = intrest;
        }

        public override void Afbeelden()
        {
            base.Afbeelden();
            Console.WriteLine("Intrest: {0}", Intrest);
        }
    }
}
```

22.1.4 Het hoofdprogramma

```
using System;
namespace CSharpPFOefenmap
{
```

```

class Program
{
    static void Main(string[] args)
    {
        ZichtRekening mijnZichtRekening =
            new ZichtRekening(63154756360ul, 0m, DateTime.Today, -1000m);
        mijnZichtRekening.Storten(1000m);
        mijnZichtRekening.Afbeelden();

        SpaarRekening mijnSpaarRekening
            = new SpaarRekening(83179377453ul, 0m, DateTime.Today, 4.5m);
        mijnSpaarRekening.Storten(125m);
        mijnSpaarRekening.Afbeelden();
    }
}

```

22.2 Oefening Voertuigen

22.2.1 De class Voertuig

```

namespace CSharpPFOefenmap
{
    public class Vrachtwagen:Voertuig
    {
        ...
        public virtual void Afbeelden()
        {
            Console.WriteLine("Polishouder: {0}", Polishouder);
            Console.WriteLine("Kostprijs: {0}", Kostprijs);
            Console.WriteLine("Aantal pk: {0}", Pk);
            Console.WriteLine("Gemiddeld verbruik: {0}", GemiddeldVerbruik);
            Console.WriteLine("Nummerplaat: {0}", Nummerplaat);
        }
    }
}

```

22.2.2 De class Vrachtwagen

```

using System;
namespace CSharpPFOefenmap
{
    public class Vrachtwagen : Voertuig
    {
        public Vrachtwagen()
            : base()
        {
            MaximumLading = 10000f;
        }

        public Vrachtwagen(string polishouder, decimal kostprijs, int pk,
            float gemiddeldVerbruik, string nummerplaat, float maximumLading)
            : base(polishouder, kostprijs, pk, gemiddeldVerbruik, nummerplaat)
        {
        }
    }
}

```

```

    {
        MaximumLading = maximumLading;
    }

    private float maximumLadingValue;
    public float MaximumLading
    {
        get
        {
            return maximumLadingValue;
        }
        set
        {
            if (value >= 0f)
                maximumLadingValue = value;
        }
    }

    public override void Afbeelden()
    {
        Console.WriteLine("Vrachtwagen");
        base.Afbeelden();
        Console.WriteLine("Maximum lading: {0}", MaximumLading);
    }
}

```

22.2.3 De class Personenwagen

```

using System;
namespace CSharpPFOefenmap
{
    public class Personenwagen : Voertuig
    {
        public Personenwagen()
            : base()
        {
            AantalDeuren = 4;
            AantalPassagiers = 5;
        }

        public Personenwagen(string polishouder, decimal kostprijs, int pk,
                               float gemiddeldVerbruik, string nummerplaat,
                               int aantalDeuren, int aantalPassagiers)
            : base(polishouder, kostprijs, pk, gemiddeldVerbruik, nummerplaat)
        {
            AantalDeuren = aantalDeuren;
            AantalPassagiers = aantalPassagiers;
        }

        private int aantalDeurenValue;
        public int AantalDeuren
        {
            get
            {
                return aantalDeurenValue;
            }
            set

```

```

        {
            if (value > 0)
            {
                aantalDeurenValue = value;
            }
        }
    }

    private int aantalPassagiersValue;
    public int AantalPassagiers
    {
        get
        {
            return aantalPassagiersValue;
        }
        set
        {
            if (value >= 0)
            {
                aantalPassagiersValue = value;
            }
        }
    }

    public override void Afbeelden()
    {
        Console.WriteLine("Personenwagen");
        base.Afbeelden();
        Console.WriteLine("Aantal Deuren: {0}", AantalDeuren);
        Console.WriteLine("Aantal passagiers: {0}", AantalPassagiers);
    }
}
}

```

22.2.4 Het hoofdprogramma

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Vrachtwagen vw = new Vrachtwagen("Jan", 40000m, 500, 30,
                                              "1-ABC-123", 10000);
            Personenwagen pw = new Personenwagen("Piet", 15000m, 8, 6.5f,
                                                  "1-DEF-4156", 5, 5);

            vw.Afbeelden();
            Console.WriteLine();
            pw.Afbeelden();
        }
    }
}

```

23 Voorbeeldoplossing: abstract classes, abstract members, static members

23.1 Oefening Bank

23.1.1 De abstracte class Rekening

```
using System;
namespace CSharpPFOefenmap
{
    public abstract class Rekening
    {
        ...
    }
}
```

23.1.2 Static member Intrest in de class SpaarRekening

```
using System;
namespace CSharpPFOefenmap
{
    public class SpaarRekening : Rekening
    {
        private static decimal intrestValue;
        public static decimal Intrest
        {
            get
            {
                return intrestValue;
            }
            set
            {
                if (value >= 0m)
                    intrestValue = value;
            }
        }

        //De parameter Intrest is verwijderd
        public SpaarRekening(ulong nummer, decimal saldo,
            DateTime creatieDatum)
            : base(nummer, saldo, creatieDatum)
        {
        }

        public override void Afbeelden()
        {
            base.Afbeelden();
            Console.WriteLine("Intrest: {0}", Intrest);
        }
    }
}
```

```
    }  
  }  
}
```

23.2 Oefening Voertuigen

23.2.1 De abstracte class Voertuig

```
using System;  
namespace CSharpPFOefenmap  
{  
    public abstract class Voertuig  
    {  
        ...  
        public abstract double GetKyotoScore();  
    }  
}
```

23.2.2 De class Vrachtwagen

```
using System;  
namespace CSharpPFOefenmap  
{  
    public class Vrachtwagen : Voertuig  
    {  
        ...  
        public override double GetKyotoScore()  
        {  
            double kyotoScore = 0.0;  
            if (MaximumLading != 0)  
            {  
                kyotoScore = (GemiddeldVerbruik * Pk) / (MaximumLading/1000.0);  
            }  
            return kyotoScore;  
        }  
    }  
}
```

23.2.3 De class Personenwagen

```
using System;  
namespace CSharpPFOefenmap  
{  
    public class Personenwagen : Voertuig  
    {  
        ...  
        public override double GetKyotoScore()  
        {  
            double kyotoScore = 0.0;  
            if (AantalPassagiers != 0)  
            {  
                kyotoScore = (GemiddeldVerbruik * Pk) / AantalPassagiers;  
            }  
            return kyotoScore;  
        }  
    }  
}
```

24 Voorbeeldoplossing: inheritance polymorphisme

24.1 Oefening Bank

24.1.1 Rekeningen afbeelden

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            SpaarRekening.Intrest = 3m;
            Rekening[] rekeningen = new Rekening[2];
            rekeningen[0] =
                new ZichtRekening(747524091936ul, 14.51m, DateTime.Today, -500m);
            rekeningen[1] =
                new SpaarRekening(737524091952ul, 1000m, DateTime.Today);
            foreach (Rekening rekening in rekeningen)
                rekening.Afbeelden();
        }
    }
}
```

24.2 Oefening Voertuigen

24.2.1 Voertuigen afbeelden

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Vrachtwagen vw = new Vrachtwagen("Jan", 40000m, 500, 30,
                "1-ABC-123", 10000);
            Personenwagen pw = new Personenwagen("Piet", 15000m, 8, 6.5f,
                "1-DEF-4156", 5, 5);
            Voertuig[] voertuigen = new Voertuig[2];
            voertuigen[0] = vw;
            voertuigen[1] = pw;
            foreach (Voertuig voertuig in voertuigen)
            {
                voertuig.Afbeelden();
                Console.WriteLine(voertuig.GetKyotoScore());
                Console.WriteLine();
            }
        }
    }
}
```


25 Voorbeeldoplossing: aggregation

25.1 Oefening Bank

25.1.1 De class Klant

```
using System;
namespace CSharpPFOefenmap
{
    public class Klant
    {
        private string voornaamValue;
        private string familienaamValue;
        public string Voornaam
        {
            get
            {
                return voornaamValue;
            }
            set
            {
                voornaamValue = value;
            }
        }
        public string Familienaam
        {
            get
            {
                return familienaamValue;
            }
            set
            {
                familienaamValue = value;
            }
        }

        public Klant(string voornaam, string familienaam)
        {
            Voornaam = voornaam;
            Familienaam = familienaam;
        }

        public void Afbeelden()
        {
            Console.WriteLine("Voornaam: {0}", Voornaam);
            Console.WriteLine("Familienaam: {0}", Familienaam);
        }
    }
}
```

25.1.2 De class Rekening

```

using System;
namespace CSharpPFOefenmap
{
    public abstract class Rekening
    {
        private readonly DateTime EersteCreatie = new DateTime(1900, 1, 1);
        private ulong nummerValue;
        private decimal saldoValue;
        private DateTime creatieDatumValue;
        private Klant eigenaarValue;
        ...
        public Klant Eigenaar
        {
            get
            {
                return eigenaarValue;
            }
            set
            {
                eigenaarValue = value;
            }
        }
        ...

        public Rekening(ulong nummer, decimal saldo,
            DateTime creatieDatum, Klant eigenaar)
        {
            Nummer = nummer;
            Saldo = saldo;
            CreatieDatum = creatieDatum;
            Eigenaar = eigenaar;
        }

        public virtual void Afbeelden()
        {
            if (Eigenaar != null)
            {
                Console.WriteLine("Eigenaar: ");
                Eigenaar.Afbeelden();
            }

            Console.WriteLine("Rekeningnummer: {0:000-00000000-00}", Nummer);
            Console.WriteLine("Saldo: {0}", Saldo);
            Console.WriteLine("Creatiedatum: {0:dd-MM-yyyy}", CreatieDatum);
        }
        ...
    }
}

```

25.1.3 De class SpaarRekening

```
using System;
namespace CSharpPFOefenmap
{
    public class SpaarRekening : Rekening
    {
        ...
        public SpaarRekening(ulong nummer, decimal saldo,
            DateTime creatieDatum, Klant eigenaar)
            : base(nummer, saldo, creatieDatum, eigenaar)
        {
        }
        ...
    }
}
```

25.1.4 De class ZichtRekening

```
using System;
namespace CSharpPFOefenmap
{
    public class ZichtRekening : Rekening
    {
        ...
        public ZichtRekening(ulong nummer, decimal saldo,
            DateTime creatieDatum, decimal maxKrediet, Klant eigenaar)
            : base(nummer, saldo, creatieDatum, eigenaar)
        {
            MaxKrediet = maxKrediet;
        }
        ...
    }
}
```

25.1.5 Het hoofdprogramma

```
using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            SpaarRekening.Intrest = 3m;
            Klant ik = new Klant("Eddy", "Wally");
            ZichtRekening mijnZichtRekening =
                new ZichtRekening(747524091936ul, 14.51m, DateTime.Today, -500m, ik);
            SpaarRekening mijnSpaarRekening =
```

```
        new SpaarRekening(737524091952u1, 1000m, DateTime.Today, ik);
        mijnZichtRekening.Afbeelden();
        mijnSpaarRekening.Afbeelden();
    }
}
```

26 Voorbeeldoplossing: interfaces

26.1 Oefening Bank

26.1.1 De interface ISpaarmiddel

```
using System;
namespace CSharpPFOefenmap
{
    interface ISpaarmiddel
    {
        void Afbeelden();
    }
}
```

26.1.2 De class Rekening

```
using System;
namespace CSharpPFOefenmap
{
    public abstract class Rekening : ISpaarmiddel
    {
        ...
    }
}
```

26.1.3 De class Kasbon

```
using System;
namespace CSharpPFOefenmap
{
    public class Kasbon : ISpaarmiddel
    {
        private readonly DateTime EersteAankoop = new DateTime(1900, 1, 1);
        private DateTime aankoopDatumValue;
        private decimal bedragValue;
        private int looptijdValue;
        private decimal intrestValue;
        private Klant eigenaarValue;

        public DateTime AankoopDatum
        {
            get
            {
                return aankoopDatumValue;
            }
            set
            {
                if (value >= EersteAankoop)
```

```
        aankoopDatumValue = value;
    }
}

public decimal Bedrag
{
    get
    {
        return bedragValue;
    }
    set
    {
        if (value > 0m)
            bedragValue = value;
    }
}

public int Looptijd
{
    get
    {
        return looptijdValue;
    }
    set
    {
        if (value > 0)
            looptijdValue = value;
    }
}

public decimal Intrest
{
    get
    {
        return intrestValue;
    }
    set
    {
        if (value > 0m)
            intrestValue = value;
    }
}

public Klant Eigenaar
{
    get
    {
        return eigenaarValue;
    }
    set
    {
        eigenaarValue = value;
    }
}
```

```

public Kasbon(DateTime aankoopDatum, decimal bedrag,
               int looptijd, decimal intrest, Klant eigenaar)
{
    AankoopDatum = aankoopDatum;
    Bedrag = bedrag;
    Looptijd = looptijd;
    Intrest = intrest;
    Eigenaar = eigenaar;
}

public void Afbeelden()
{
    if (Eigenaar != null)
    {
        Console.WriteLine("Eigenaar");
        Eigenaar.Afbeelden();
    }
    Console.WriteLine("Aankoopdatum: {0:dd-MM-yyyy}", AankoopDatum);
    Console.WriteLine("Bedrag: {0}", Bedrag);
    Console.WriteLine("Looptijd: {0}", Looptijd);
    Console.WriteLine("Intrest: {0}", Intrest);
}
}
}

```

26.1.4 Het hoofdprogramma

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            SpaarRekening.Intrest = 3m;
            Klant ik = new Klant("Eddy", "Wally");
            ISpaarmiddel[] spaarmiddelen = new ISpaarmiddel[3];
            spaarmiddelen[0] = new ZichtRekening(747524091936ul, 14.51m,
                DateTime.Today, -500m, ik);
            spaarmiddelen[1] = new SpaarRekening(737524091952ul, 1000m,
                DateTime.Today, ik);
            spaarmiddelen[2] = new Kasbon(DateTime.Today, 1000m, 5, 3.5m, ik);
            foreach (ISpaarmiddel spaarmiddel in spaarmiddelen)
                spaarmiddel.Afbeelden();
        }
    }
}

```

26.2 Oefening Voertuigen

26.2.1 De interface IVervuiler

```
namespace CSharpPFOefenmap
{
    public interface IVervuiler
    {
        double GeefVervuiling();
    }
}
```

26.2.2 de class Vrachtwagen

```
using System;
namespace CSharpPFOefenmap
{
    public class Vrachtwagen : Voertuig, IVervuiler
    {
        ...
        public double GeefVervuiling()
        {
            return GetKyotoScore() * 20;
        }
    }
}
```

26.2.3 de class Personenwagen

```
using System;
namespace CSharpPFOefenmap
{
    public class Personenwagen : Voertuig, IVervuiler
    {
        ...
        public double GeefVervuiling()
        {
            return GetKyotoScore() * 5;
        }
    }
}
```

26.2.4 De class Stookketel

```
using System;
namespace CSharpPFOefenmap
{
    public class Stookketel : IVervuiler
    {
        public Stookketel(float cONorm)
        {
            this.CONorm = cONorm;
        }

        private float CONormValue;
        public float CONorm
    }
}
```



```

    {
        get
        {
            return CONormValue;
        }
        set
        {
            if (value > 0)
            {
                CONormValue = value;
            }
        }
    }

    public double GeefVervuiling()
    {
        return CONorm * 100;
    }
}

```

26.2.5 Het hoofdprogramma

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Vrachtwagen vw = new Vrachtwagen("Jan", 40000m, 500, 30,
                                             "1-ABC-123", 10000);
            Personenwagen pw = new Personenwagen("Piet", 15000m, 8, 6.5f,
                                                  "1-DEF-4156", 5, 5);

            IVervuiler[] vervuilers = new IVervuiler[3];
            vervuilers[0] = vw;
            vervuilers[1] = pw;
            vervuilers[2] = new Stookketel(7.5f);
            foreach (IVervuiler vervuiler in vervuilers)
            {
                Console.WriteLine("Vervuiling: {0}", vervuiler.GeefVervuiling());
            }
        }
    }
}

```

26.2.6 De interfaces IPrivaat en IMilieu

```

namespace CSharpPFOefenmap
{
    public interface IPrivaat
    {
        string GeefPrivateData();
    }
}

```

```

namespace CSharpPFOefenmap
{
    public interface IMilieu
    {
        string GeefMilieuData();
    }
}

using System;
namespace CSharpPFOefenmap
{
    public abstract class Voertuig: IPrivaat,IMilieu
    {
        public string GeefPrivateData()
        {
            return string.Format("Polishouder: {0} - Nummerplaat: {1}",
                                Polishouder, Nummerplaat);
        }
        public string GeefMilieuData()
        {
            return string.Format("PK: {0} - Kostprijs: {1} - Gemiddeld verbruik: {2}",
                                Pk, Kostprijs,GemiddeldVerbruik);
        }
    }
}

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Vrachtwagen vw = new Vrachtwagen("Jan", 40000m, 500, 30,
                                             "1-ABC-123", 10000);
            Personenwagen pw = new Personenwagen("Piet", 15000m, 8, 6.5f,
                                                  "1-DEF-4156", 5, 5);

            Console.WriteLine("Private gegevens:");
            IPrivaat[] privategegevens = new IPrivaat[2];
            privategegevens[0] = vw;
            privategegevens[1] = pw;
            foreach (IPrivaat voertuig in privategegevens)
            {
                Console.WriteLine(voertuig.GeefPrivateData());
            }
            Console.WriteLine("Milieugegevens:");
            IMilieu[] milieugegevens = new IMilieu[2];
            milieugegevens[0] = vw;
            milieugegevens[1] = pw;
            foreach (IMilieu voertuig in milieugegevens)
            {
                Console.WriteLine(voertuig.GeefMilieuData());
            }
        }
    }
}

```

27 Voorbeeldoplossing: delegates en events

27.1 Oefening Bank

27.1.1 De class Rekening

```
using System;
namespace CSharpPFOefenmap
{
    public abstract class Rekening : ISpaarmiddel
    {
        ...
        //Definitie van de delegate
        public delegate void Transactie(Rekening rekening);

        //Declaratie van de events
        public Transactie RekeningUittreksel;
        public Transactie SaldoInHetRood;

        ...
        private decimal vorigSaldoValue;
        public decimal VorigSaldo
        {
            get
            {
                return vorigSaldoValue;
            }
            set
            {
                vorigSaldoValue = value;
            }
        }
        ...
        public void Storten(decimal bedrag)
        {
            VorigSaldo = Saldo;
            Saldo += bedrag;
            if (RekeningUittreksel != null)
                RekeningUittreksel(this);
        }

        public void Afhalen(decimal bedrag)
        {
            VorigSaldo = Saldo;
            if (bedrag <= Saldo)
            {
                Saldo -= bedrag;
                if (RekeningUittreksel != null)
                    RekeningUittreksel(this);
            }
        }
    }
}
```

```
        else
        {
            if (SaldoInHetRood != null)
                SaldoInHetRood(this);
        }
    }
}
```

27.1.2 De class BankBediende

```
using System;
namespace CSharpPFOefenmap
{
    public class BankBediende
    {
        private string voornaamValue;
        private string familienaamValue;
        public string Voornaam
        {
            get
            {
                return voornaamValue;
            }
            set
            {
                voornaamValue = value;
            }
        }
        public string Familienaam
        {
            get
            {
                return familienaamValue;
            }
            set
            {
                familienaamValue = value;
            }
        }

        public BankBediende(string voornaam, string familienaam)
        {
            Voornaam = voornaam;
            Familienaam = familienaam;
        }

        public override string ToString()
        {
            return "Bankbediende " + Voornaam + ' ' + Familienaam;
        }
    }
}
```

```

    }

    public void RekeningUittrekselTonen(Rekening rekening)
    {
        Console.WriteLine("Datum: {0:dd-MM-yyyy}", DateTime.Today);
        Console.WriteLine("Rekeninguittreksel van " +
            "rekening: {0:000-0000000-00}", rekening.Nummer);
        Console.WriteLine("Vorig saldo: {0} euro", rekening.VorigSaldo);
        if (rekening.Saldo > rekening.VorigSaldo)
        {
            Console.WriteLine("Storting van {0} euro.",
                rekening.Saldo - rekening.VorigSaldo);
        }
        else
        {
            Console.WriteLine("Afhaling van {0} euro.",
                rekening.VorigSaldo - rekening.Saldo);
        }
        Console.WriteLine("Nieuw saldo: {0} euro",
            rekening.Saldo);
    }

    public void RekeningInHetRoodMelden(Rekening rekening)
    {
        Console.WriteLine("Afhaling niet mogelijk, saldo ontoereikend!");
        Console.WriteLine("Maximum af te halen bedrag: {0} euro",
            rekening.Saldo);
    }
}
}

```

27.1.3 Het hoofdprogramma

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            BankBediende deBankbediende = new BankBediende("Bas", "Paris");
            Klant ik = new Klant("Eddy", "Wally");

            ZichtRekening mijnZichtrekening =
                new ZichtRekening(747524091936ul, 100m,
                    new DateTime(2008, 1, 1), -10, ik);
            Console.WriteLine("Zichtrekening: ");
            mijnZichtrekening.Afbeelden();
            Console.WriteLine();
        }
    }
}

```

```
mijnZichtrekening.RekeningUittreksel +=
    deBankbediende.RekeningUittrekselTonen;
mijnZichtrekening.SaldoInHetRood +=
    deBankbediende.RekeningInHetRoodMelden;

//rekeninguittreksel storten
mijnZichtrekening.Storten(50m);
Console.WriteLine();

//rekeninguittreksel afhalen
mijnZichtrekening.Afhalen(100m);
Console.WriteLine();

//SaldoInHetRood
mijnZichtrekening.Afhalen(100m);
Console.WriteLine();

SpaarRekening.Intrest = 0.01m;
SpaarRekening mijnSpaarrekening =
    new SpaarRekening(737524091952u1, 1000m, DateTime.Today, ik);

mijnSpaarrekening.RekeningUittreksel +=
    deBankbediende.RekeningUittrekselTonen;
mijnSpaarrekening.SaldoInHetRood +=
    deBankbediende.RekeningInHetRoodMelden;

Console.WriteLine("Spaarrekening: ");
mijnSpaarrekening.Afbeelden();
Console.WriteLine();

//rekeninguittreksel storten
mijnSpaarrekening.Storten(2000m);
Console.WriteLine();

//rekeninguittreksel afhalen
mijnSpaarrekening.Afhalen(1000m);
Console.WriteLine();

//SaldoInHetRood
mijnSpaarrekening.Afhalen(4000m);
    }
}
}
```

28 Voorbeeldoplossing: exceptions

28.1 Oefening Bank

28.1.1 De class Rekening

```
using System;
namespace CSharpPFOefenmap
{
    public abstract class Rekening : ISpaarmiddel
    {
        private readonly DateTime EersteCreatie = new DateTime(1900, 1, 1);
        private ulong nummerValue;
        private DateTime creatieDatumValue;
        ...

        public ulong Nummer
        {
            get
            {
                return nummerValue;
            }
            set
            {
                ulong eerste10 = value / 100ul;
                int laatste2 = (int)(value % 100ul);

                if ((int)(eerste10 % 97ul) != laatste2)
                    throw new Exception("Ongeldig rekeningnummer!");
                nummerValue = value;
            }
        }
        ...
        public DateTime CreatieDatum
        {
            get
            {
                return creatieDatumValue;
            }
            set
            {
                if (value < EersteCreatie)
                    throw new Exception("De creatiedatum mag " +
                        "niet voor 1-1-1990 zijn!");
                creatieDatumValue = value;
            }
        }
        ...
    }
}
```

28.1.2 De class ZichtRekening

```
using System;
namespace CSharpPFOefenmap
{
    public class ZichtRekening : Rekening
    {
        private decimal maxKredietValue;
        public decimal MaxKrediet
        {
            get
            {
                return maxKredietValue;
            }
            set
            {
                if (value > 0m)
                    throw new Exception("De waarde van MaxKrediet " +
                        "mag niet positief zijn!");
                maxKredietValue = value;
            }
        }
        ...
    }
}
```

28.1.3 De class SpaarRekening

```
using System;
namespace CSharpPFOefenmap
{
    public class SpaarRekening : Rekening
    {
        private static decimal intrestValue;
        public static decimal Intrest
        {
            get
            {
                return intrestValue;
            }
            set
            {
                if (value <= 0m)
                    throw new Exception("Intrest moet positief zijn!");
                intrestValue = value;
            }
        }
        ...
    }
}
```


28.1.4 De class Kasbon

```
using System.Text;
namespace CSharpPFOefenmap
{
    public class Kasbon : ISpaarmiddel
    {
        private readonly DateTime EersteAankoop = new DateTime(1900, 1, 1);
        private DateTime aankoopDatumValue;
        private decimal bedragValue;
        private int looptijdValue;
        private decimal intrestValue;
        private Klant eigenaarValue;

        public DateTime AankoopDatum
        {
            get
            {
                return aankoopDatumValue;
            }
            set
            {
                if (value < EersteAankoop)
                    throw new Exception("De aankoopdatum mag " +
                        "niet voor 1-1-1990 zijn!");
                aankoopDatumValue = value;
            }
        }

        public decimal Bedrag
        {
            get
            {
                return bedragValue;
            }
            set
            {
                if (value <= 0m)
                    throw new Exception("Het bedrag moet positief zijn!");
                bedragValue = value;
            }
        }

        public int Looptijd
        {
            get
            {
                return looptijdValue;
            }
            set
            {
                if (value <= 0)

```

```

        throw new Exception("De looptijd moet positief zijn!");
        looptijdValue = value;
    }
}

public decimal Intrest
{
    get
    {
        return intrestValue;
    }
    set
    {
        if (value <= 0m)
            throw new Exception("Intrest moet positief zijn!");
        intrestValue = value;
    }
}
...
}
}

```

28.1.5 Het hoofdprogramma

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                SpaarRekening.Intrest = 3m;
                Klant ik = new Klant("Eddy", "Wally");
                ISpaarmiddel[] spaarmiddelen = new ISpaarmiddel[3];
                spaarmiddelen[0] = new ZichtRekening(747524091936ul, 14.51m,
                    DateTime.Today, -500m, ik);
                spaarmiddelen[1] = new SpaarRekening(737524091952ul, 1000m,
                    DateTime.Today, ik);
                spaarmiddelen[2] = new Kasbon(DateTime.Today, 1000m, 5, 3.5m, ik);
                foreach (ISpaarmiddel spaarmiddel in spaarmiddelen)
                    spaarmiddel.Afbeelden();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}

```

29 Voorbeeldoplossing: lambda expressies

Deze oefening kan je op verschillende manieren oplossen.

29.1 Oplossing met een expression lambda

```
using System;
using System.Text;
namespace CSharpPFOefenmap
{
    class Program
    {
        delegate ConsoleColor KleurGetal(int getal); //functie

        static void Main(string[] args)
        {
            KleurGetal kleurEvenOneven = getal =>
                getal % 2 == 0 ? ConsoleColor.Green : ConsoleColor.Red;

            KleurGetal kleurNegatiefPositief = getal =>
                getal < 0 ? ConsoleColor.Yellow : ConsoleColor.White;

            ToonGetallen(kleurEvenOneven);
            ToonGetallen(kleurNegatiefPositief);
        }

        private static void ToonGetallen(KleurGetal kleurGetal)
        {
            var getallen = new[] { 0, -1, 2, -3, 4, -5, 6, -7, 8, -9 };
            foreach (var getal in getallen)
            {
                Console.ForegroundColor = kleurGetal(getal);
                Console.WriteLine(getal);
            }
            Console.WriteLine();
        }
    }
}
```

29.2 Oplossing met een statement lambda

```
using System;
using System.Text;
namespace CSharpPFOefenmap
{
    class Program
    {
        delegate void KleurGetal(int getal); //sub

        static void Main(string[] args)
```

```

{
    KleurGetal kleurEvenOneven =
        getal =>
        {
            Console.ForegroundColor =
                getal % 2 == 0 ? ConsoleColor.Green : ConsoleColor.Red;
            Console.WriteLine(getal);
        };

    KleurGetal kleurNegatiefPositief =
        getal =>
        {
            Console.ForegroundColor =
                getal < 0 ? ConsoleColor.Yellow : ConsoleColor.White;
            Console.WriteLine(getal);
        };

    ToonGetallen(kleurEvenOneven);
    ToonGetallen(kleurNegatiefPositief);
}

private static void ToonGetallen(KleurGetal kleurGetal)
{
    var getallen = new[] { 0, -1, 2, -3, 4, -5, 6, -7, 8, -9 };
    foreach (var getal in getallen)
    {
        kleurGetal(getal);
    }
    Console.WriteLine();
}
}
}

```

29.3 Oplossing met Func<>

```

using System;
using System.Text;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Func<int, ConsoleColor> kleurEvenOneven = getal =>
                getal % 2 == 0 ? ConsoleColor.Green : ConsoleColor.Red;

            Func<int, ConsoleColor> kleurPositiefNegatief = getal =>
                getal < 0 ? ConsoleColor.Yellow : ConsoleColor.White;

            ToonGetallen(kleurEvenOneven);
            ToonGetallen(kleurPositiefNegatief);
        }
    }
}

```

```

    }

    private static void ToonGetallen(Func<int, ConsoleColor> kleurGetal)
    {
        var getallen = new[] { 0, -1, 2, -3, 4, -5, 6, -7, 8, -9 };
        foreach (var getal in getallen)
        {
            Console.ForegroundColor = kleurGetal(getal);
            Console.WriteLine(getal);
        }
        Console.WriteLine();
    }
}

```

29.4 Oplossing met Action<>

```

using System;
using System.Text;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Action<int> kleurEvenOneven =
                getal =>
                {
                    Console.ForegroundColor =
                        getal % 2 == 0 ? ConsoleColor.Green : ConsoleColor.Red;
                    Console.WriteLine(getal);
                };

            Action<int> kleurPositiefNegatief =
                getal =>
                {
                    Console.ForegroundColor =
                        getal < 0 ? ConsoleColor.Yellow : ConsoleColor.White;
                    Console.WriteLine(getal);
                };

            ToonGetallen(kleurEvenOneven);
            ToonGetallen(kleurPositiefNegatief);
        }

        private static void ToonGetallen(Action<int> kleurGetal)
        {
            var getallen = new[] { 0, -1, 2, -3, 4, -5, 6, -7, 8, -9 };
            foreach (var getal in getallen)
                kleurGetal(getal);
        }
    }
}

```

```
        Console.WriteLine();  
    }  
}  
}
```

30 Voorbeeldoplossing: linq

30.1 De class Plant

```
using System;
namespace CSharpPFOefenmap
{
    public class Plant
    {
        public int PlantID { get; set; }
        public string Plantennaam { get; set; }
        public string Kleur { get; set; }
        public decimal Prijs { get; set; }
        public string Soort { get; set; }
    }
}
```

30.2 Het hoofdprogramma

```
using System;
using System.Text;
using System.Collections.Generic;
using System.Linq;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            var planten = new List<Plant> {
                new Plant {PlantID=1,Plantennaam="Tulp", Kleur="rood",
                    Prijs=0.5m, Soort="bol"},
                new Plant {PlantID=2, Plantennaam="Krokus", Kleur="wit",
                    Prijs=0.2m, Soort="bol"},

                new Plant {PlantID=3, Plantennaam="Narcis", Kleur="geel",
                    Prijs=0.3m, Soort="bol"},
                new Plant {PlantID=4, Plantennaam="Blauw druifje", Kleur="blauw",
                    Prijs=0.2m, Soort="bol"},
                new Plant {PlantID=5, Plantennaam="Azalea",Kleur="rood",
                    Prijs=3m, Soort="heester"},
                new Plant {PlantID=6, Plantennaam="Forsythia", Kleur="geel",
                    Prijs=2m, Soort="heester"},
                new Plant {PlantID=7, Plantennaam="Magnolia", Kleur="wit",
                    Prijs=4m, Soort="heester"},
                new Plant {PlantID=8, Plantennaam="Waterlelie", Kleur="wit",
                    Prijs=2m,Soort="water"},
                new Plant {PlantID=9, Plantennaam="Lisdodde", Kleur="geel",
                    Prijs=3m,Soort="water"},
                new Plant {PlantID=10,Plantennaam="Kalmoes", Kleur="geel",
```

```

        Prijs=2.5m, Soort="water"},
    new Plant {PlantID=11,Plantennaam="Bieslook", Kleur="paars",
        Prijs=1.5m,Soort="kruid"},
    new Plant {PlantID=12,Plantennaam="Rozemarijn", Kleur="blauw",
        Prijs=1.25m, Soort="kruid"},
    new Plant {PlantID=13,Plantennaam="Munt", Kleur="wit",
        Prijs=1.1m, Soort="kruid"},
    new Plant {PlantID=14,Plantennaam="Dragon", Kleur="wit",
        Prijs=1.3m, Soort="kruid"},
    new Plant {PlantID=15,Plantennaam="Basilicum", Kleur="wit",
        Prijs=1.5m, Soort="kruid"}};

//Plantengegevens van alle witte planten, gesorteerd op prijs
//Oplossing 1
Console.WriteLine("Witte planten");
var wittePlanten =
    from plant in planten
    where plant.Kleur == "wit"
    orderby plant.Prijs
    select new { plant.Plantennaam, plant.Kleur, plant.Prijs };
foreach (var plant in wittePlanten)
    Console.WriteLine("{0} ({1}): {2} euro",
        plant.Plantennaam,plant.Kleur, plant.Prijs);
Console.WriteLine();

//Oplossing 2
Console.WriteLine("Witte planten");
var wittePlanten2 = planten
    .Where(plant => plant.Kleur == "wit")
    .OrderBy(plant => plant.Prijs);
foreach (var plant in wittePlanten2)
    Console.WriteLine("{0} ({1}): {2} euro",
        plant.Plantennaam,plant.Kleur, plant.Prijs);
Console.WriteLine();

//Aantal witte planten
//Oplossing 1
var aantalWittePlanten = (from plant in planten
    where plant.Kleur == "wit"
    select plant).Count();
Console.WriteLine("Aantal witte planten: {0}", aantalWittePlanten);

//Oplossing 2
Console.WriteLine("Aantal witte planten: {0}", planten
    .Where(plant => plant.Kleur == "wit")
    .Count());

//Oplossing 3
var aantalWittePlanten2 = planten

```



```
.Where(plant => plant.Kleur == "wit")
.Count();
Console.WriteLine("Aantal witte planten: {0}", aantalWittePlanten2);
Console.WriteLine();

//Gemiddelde prijs van de heesters
//Oplossing 1
var gemiddeldePrijsHeesters = (from plant in planten
                               where plant.Soort == "heester"
                               select plant.Prijs).Average();
Console.WriteLine("Gemiddelde prijs van de heesters: {0} euro",
                  gemiddeldePrijsHeesters);

//Oplossing 2
var gemiddeldePrijsHeesters2 = planten
    .Where(plant => plant.Soort=="heester")
    .Average(plant => plant.Prijs);
Console.WriteLine("Gemiddelde prijs van de heesters: {0} euro",
                  gemiddeldePrijsHeesters2);

//Oplossing 3
var gemiddeldePrijsHeesters3 = (from plant in planten
                               where plant.Soort == "heester"
                               select plant).Average(plant =>
                                                       plant.Prijs);
Console.WriteLine("Gemiddelde prijs van de heesters: {0} euro",
                  gemiddeldePrijsHeesters3);
Console.WriteLine();

//Gemiddelde en maximumprijs van de kruiden
var prijzenKruiden = from plant in planten
                     where plant.Soort == "kruid"
                     select plant.Prijs;
Console.WriteLine("Gemiddelde prijs kruiden: {0} euro" +
                  System.Environment.NewLine +
                  "Maximumprijs kruiden: {1} euro",
                  prijzenKruiden.Average(), prijzenKruiden.Max());
Console.WriteLine();

//Planten waarvan de naam met B begint
Console.WriteLine("Planten waarvan de naam begint met B:");
var plantenB = from plant in planten
               where plant.Plantennaam.ToUpper().StartsWith("B")
               select plant;
foreach (var plant in plantenB)
    Console.WriteLine(plant.Plantennaam);
Console.WriteLine();
```

```
//Overzicht van de verschillende kleuren
Console.WriteLine("De verschillende plantenkleuren:");
var verschillendeKleuren = (from plant in planten
                             select plant.Kleur).Distinct();
foreach (var kleur in verschillendeKleuren)
    Console.WriteLine(kleur);
Console.WriteLine();

//Plantgegevens per kleur
Console.WriteLine("Plantennamen per kleur:");
var plantenPerKleur = from plant in planten
                       group plant by plant.Kleur
                       into kleurgroep
                       select kleurgroep;
foreach (var groep in plantenPerKleur)
{
    Console.WriteLine("Kleur {0}", groep.Key);
    foreach (var plant in groep)
    {
        Console.WriteLine("\t{0}", plant.Planten-naam);
    }
}
Console.WriteLine();

//Maximum plantenprijs per soort
Console.WriteLine("Maximum plantenprijs per soort:");
var groepenPerSoort = from plant in planten
                      group plant by plant.Soort
                      into soortgroep
                      select new { Soort = soortgroep.Key,
                                   MaxPrijs = soortgroep.Max(
                                       plant => plant.Prijs) };
foreach (var groep in groepenPerSoort)
{
    Console.WriteLine("{0}: max. prijs = {1} euro",
        groep.Soort, groep.MaxPrijs);
}
Console.WriteLine();

//Per soort (alfabetisch op soort):
//het aantal planten
//de namen van de planten van de soort

//Oplossing 1
Console.WriteLine("Aantal planten + plantennamen " +
    "per soort (alfabetisch):");
var soortenMetPlanten = from plant in planten
                        group plant by plant.Soort
```

```
        into soortgroep
        orderby soortgroep.Key
        select new { Soort = soortgroep.Key,
                      AantalPlanten =
                        soortgroep.Count(),
                      Planten = soortgroep };

foreach (var groep in soortenMetPlanten)
{
    Console.WriteLine("Soort {0}: {1} planten",
        groep.Soort, groep.AantalPlanten);
    foreach (var plant in groep.Planten)
        Console.WriteLine("\t{0}", plant.Plantennaam);
}
Console.WriteLine();

//Oplossing 2
Console.WriteLine("Aantal planten + plantennamen " +
    "per soort (alfabetisch:");
var soortenMetPlanten2 = planten
    .OrderBy(plant => plant.Soort)
    .GroupBy(plant => plant.Soort);
foreach (var groep in soortenMetPlanten2)
{
    Console.WriteLine("Soort {0}: {1} planten",
        groep.Key, groep.Count());
    foreach (var plant in groep)
        Console.WriteLine("\t{0}", plant.Plantennaam);
}
Console.WriteLine();

//Oplossing 3
Console.WriteLine("Aantal planten + plantennamen " +
    "per soort (alfabetisch:");
var soortenMetPlanten3 = planten
    .GroupBy(plant => plant.Soort)
    .OrderBy(groep => groep.Key);
foreach (var groep in soortenMetPlanten3)
{
    Console.WriteLine("Soort {0}: {1} planten",
        groep.Key, groep.Count());
    foreach (var plant in groep)
        Console.WriteLine("\t{0}", plant.Plantennaam);
}
Console.WriteLine();

//Planten gegroepeerd per soort en binnen de soort per kleur
Console.WriteLine("Plantennamen gegroepeerd per soort " +
    "en binnen de soort per kleur:");
var groepenPerSoortKleur
```

```

        = from plant in planten
            group plant by plant.Soort
            into soortgroep
            select new
            {
                Soort = soortgroep.Key,
                Groepkleur =
                    from plant in soortgroep
                    group plant by plant.Kleur
                    into kleurgroep
                    select new
                    {
                        Kleur = kleurgroep.Key,
                        Planten = kleurgroep
                    }
            };
    foreach (var soort in groepenPerSoortKleur)
    {
        Console.WriteLine("Soort: {0}", soort.Soort);
        foreach (var kleur in soort.Groepkleur)
        {
            Console.WriteLine("  Kleur: {0}", kleur.Kleur);
            foreach (var plant in kleur.Planten)
            {
                Console.WriteLine("    {0}", plant.Plantennaam);
            }
        }
    }
}

```

31 Voorbeeldoplossing: Files and streams – I/O

31.1 De class Tweet

```
using System;
using System.Text;
namespace CSharpPFOefenmap
{
    [Serializable]
    public class Tweet
    {
        public Tweet()
        {
            this.Tijdstip = DateTime.Now;
        }

        public string Naam { get; set; }
        public DateTime Tijdstip { get; private set; }

        private string berichtValue;
        public string Bericht
        {
            get
            {
                return berichtValue;
            }
            set
            {
                berichtValue = value.Length <= 140 ? value : value.Substring(0, 140);
            }
        }

        public override string ToString()
        {
            //naam + bericht + tijdstip
            StringBuilder tweet = new StringBuilder(this.Naam
                + ": " + this.Bericht + ": ");
            TimeSpan verschil = DateTime.Now - this.Tijdstip;
            if (verschil.Days > 0)
                tweet.Append(this.Tijdstip.ToShortDateString());
            else if (verschil.Hours > 0)
                tweet.Append(verschil.Hours + " uur geleden");
            else if (verschil.Minutes > 0)
                tweet.Append(verschil.Minutes +
                    (verschil.Minutes == 1 ? " minuut" : " minuten") + " geleden");
            else
                tweet.Append(this.Tijdstip.ToShortTimeString());
            return tweet.ToString();
        }
    }
}
```

31.2 De class Tweets

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
namespace CSharpPFOefenmap
{
    [Serializable]
    public class Tweets
    {
        private List<Tweet> alleTweetsvalue;

        public ReadOnlyCollection<Tweet> AlleTweets()
        {
            return new ReadOnlyCollection<Tweet>(alleTweetsvalue);
        }

        //een tweet toevoegen
        public void AddTweet(Tweet tweet)
        {
            if (alleTweetsvalue == null)
                alleTweetsvalue = new List<Tweet>();
            alleTweetsvalue.Add(tweet);
        }
    }
}
```

31.3 De class Twitter

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization;
namespace CSharpPFOefenmap
{
    public class Twitter
    {
        const string twitterbestand = @"C:\Data\Twitter.obj";

        //alle tweets in omgekeerde chronologische volgorde
        public List<Tweet> AlleTweets()
        {
            if (File.Exists(twitterbestand))
            {
                var tweets = LeesTweets();
                return tweets.AlleTweets().OrderByDescending(
                    t => t.Tijdstip).ToList();
            }
            else
            {

```

```
        throw new Exception("Het bestand " + twitterbestand +  
                             " is niet gevonden!");  
    }  
}  
  
//Alle Tweets van één twitteraar  
public List<Tweet> TweetsVan(string naam)  
{  
    return AlleTweets().Where(  
        t => t.Naam.ToUpper() == naam.ToUpper()).ToList();  
}  
  
//Een tweet toevoegen  
public void SchrijfTweet(Tweet tweet)  
{  
    Tweets tweets;  
    if (File.Exists(twitterbestand))  
    {  
        //als het bestand bestaat,  
        //eerst de verzameling van bestaande tweets inlezen  
        tweets = LeesTweets();  
    }  
    else  
    {  
        tweets = new Tweets();  
    }  
  
    tweets.AddTweet(tweet);  
    //de verzameling tweets wegschrijven  
    SchrijfTweets(tweets);  
}  
  
//verzameling tweets inlezen uit bestand  
private Tweets LeesTweets()  
{  
    try  
    {  
        using (var bestand = File.Open(twitterbestand,  
                                         FileMode.Open, FileAccess.Read))  
        {  
            var lezer = new BinaryFormatter();  
            return ((Tweets)lezer.Deserialize(bestand));  
        }  
    }  
    catch (IOException)  
    {  
        throw new Exception("Fout bij het openen van het bestand!");  
    }  
    catch (SerializationException)  
    {  
        throw new Exception("Fout bij het deserialiseren, " +  
                              "het twitterbestand kan niet meer geopend worden");  
    }  
    catch (Exception ex)  
    {  
    }
```

```

        throw new Exception(ex.Message);
    }
}

//verzameling tweets wegschrijven naar bestand
private void SchrijfTweets(Tweets tweets)
{
    try
    {
        using (var bestand = File.Open(twitterbestand,
                                       FileMode.OpenOrCreate))
        {
            var schrijver = new BinaryFormatter();
            schrijver.Serialize(bestand, tweets);
        }
    }
    catch (IOException)
    {
        throw new Exception("Fout bij het openen van het bestand!");
    }
    catch (SerializationException)
    {
        throw new Exception("Fout bij het serialiseren");
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
}
}
}

```

31.4 Het hoofdprogramma

```

using System;
namespace CSharpPFOefenmap
{
    class Program
    {
        static void Main(string[] args)
        {
            Twitter twitter = new Twitter();
            int keuze = MaakKeuze();
            while (keuze != 4)
            {
                string naam, bericht;
                try
                {
                    switch (keuze)
                    {
                        case 1:
                            Console.Write("Naam? ");
                            naam = Console.ReadLine();
                            Console.Write("Bericht? ");
                            bericht = Console.ReadLine();
                            Tweet tweet = new Tweet()
                            {

```



```

        Naam = naam,
        Bericht = bericht
    };
    twitter.SchrijfTweet(tweet);
    break;
case 2:
    var tweets = twitter.AlleTweets();
    foreach (var eenTweet in tweets)
    {
        Console.WriteLine(eenTweet);
    }
    break;
case 3:
    Console.Write("Wie wil je volgen? ");
    naam = Console.ReadLine();
    var tweetsVan = twitter.TweetsVan(naam);
    if (tweetsVan.Count != 0)
    {
        foreach (var eenTweet in tweetsVan)
            Console.WriteLine(eenTweet);
    }
    else
    {
        Console.WriteLine("Geen tweets van {0}", naam);
    }
    break;
}
Console.WriteLine("-----");
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
keuze = MaakKeuze();
}

private static int MaakKeuze()
{
    int keuze;
    Console.WriteLine("Maak een keuze:");
    Console.WriteLine("1 --> een twitterbericht plaatsen");
    Console.WriteLine("2 --> alle twitterberichten tonen");
    Console.WriteLine("3 --> twitterberichten van één persoon tonen");
    Console.WriteLine("4 --> stoppen");
    Console.Write("Keuze? ");

    while (!int.TryParse(Console.ReadLine(), out keuze)
        || (keuze != 1 && keuze != 2 && keuze != 3 && keuze != 4))
    {
        Console.WriteLine("Verkeerde keuze, geef een getal (1, 2, 3 of 4): ");
    }
    return keuze;
}
}
}

```

32 Colofon

Domeinexpertisemanager	Jean Smits
Moduleverantwoordelijke	Mariëlla Cleuren
Auteurs	Hans Desmet Mariëlla Cleuren
Versie	30/11/2014

Omschrijving module-inhoud

Abstract	Doelgroep	.NET ontwikkelaar met C#
	Aanpak	Begeleide zelfstudie
	Doelstelling	De basissyntax van C# leren kennen om objectgeoriënteerde console-applicaties te ontwerpen
Trefwoorden		C#
Bronnen/meer info		http://msdn.microsoft.com