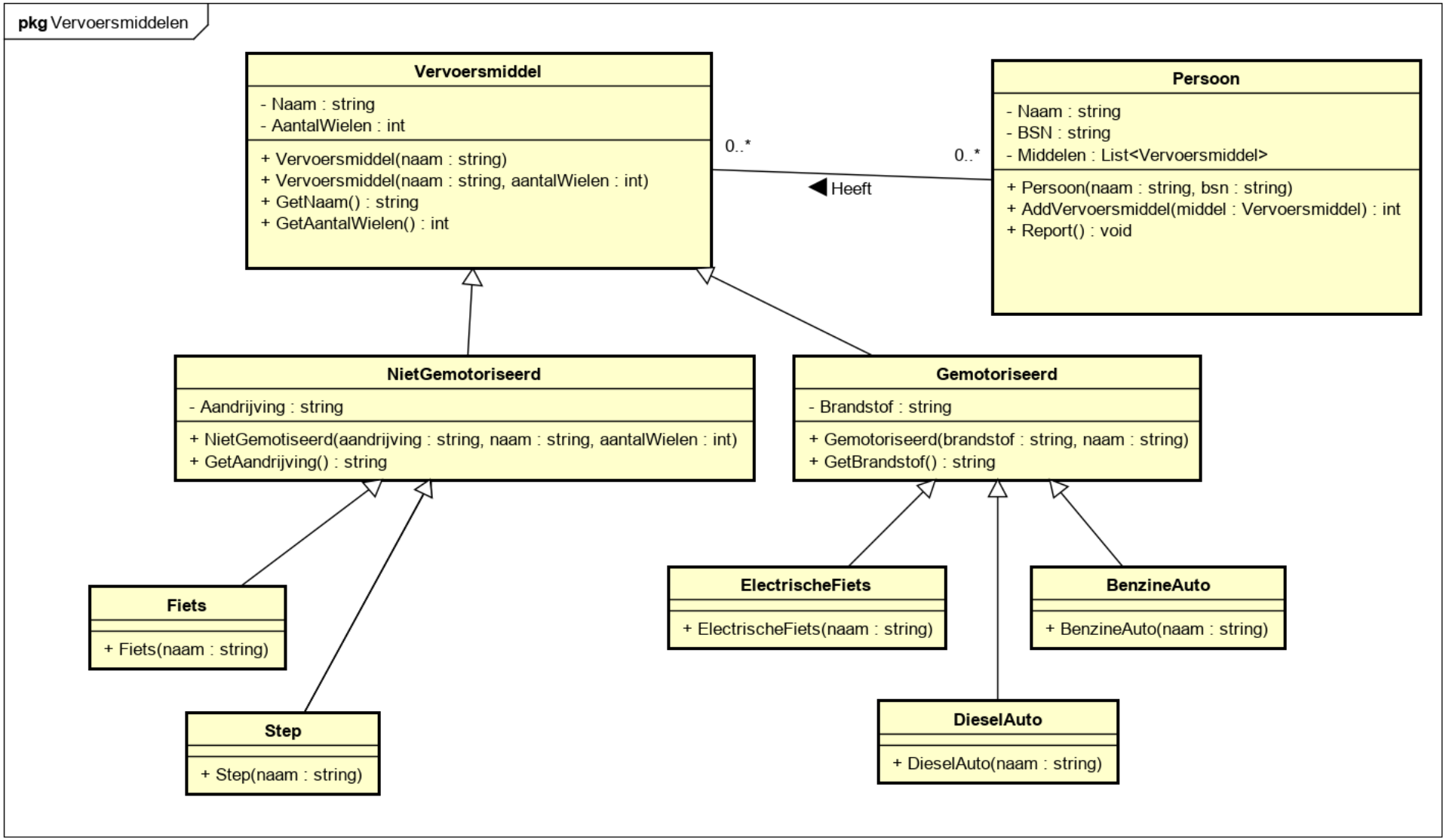
Opleiding	HBO ICT - Flex		
Titel Toetseenheid	Programmeren	**Toetsvorm**	Tentamen
Titel Toets	Programmeren	**Datum**	-
Module	Software Engineering 1 S2 2019/2020	**Examinator**	Martin Molema
		Tijdsduur	2,5 uur
Maximale score	90 punten	**Cijfer**	(90pt + 10pt) / 10
		Pagina's	7

Toegestane hulpmiddelen	Internet & Boeken

Vraag 1

Zie onderstaande klassendiagram



In bovenstaande klassendiagram worden een aantal klassen beschreven.

Vraag 1a (theorie) (5+5 = 10 punten)

1) Bij de klasse Vervoersmiddel staan twee methodes met dezelfde naam. Hoe heet zo'n methode en wat is het doel? **(5 punten)**
2) Hoe noem je de relatie tussen de klasse "Fiets" en "Niet Gemotoriseerd"? **(5 punten)**

Vraag 1b (25 punten)

Bouw de basis van de klassen uit het bovenstaande klassendiagram met C# via een *Console Application*. Toelichting

- alle variabelen hebben de beperkte zichtbaarheid **'private**'.
- In de klasse "**`Persoon`**" staat "`Middelen[]`". Dit duidt op een lijst van vervoersmiddelen.

Je krijgt de volgende extra informatie om alle functies te implementeren:

- Elk voertuig krijgt een naam zodat je het voertuig terug kunt vinden in je garage of schuur. Bijvoorbeeld "Mercedes C63AMG", "Amslod e-bike", "Audi A4 diesel".
- De functie '`Persoon.AddVervoersmiddel(…)`' voegt een vervoersmiddel toe aan de lijst van vervoersmiddelen (variabele `Middelen`).
- Verder zie je een aantal functies die met 'Get' beginnen. Deze halen een attribuutwaarde op van een private variabele. Dus bijvoorbeeld: `Persoon.GetAdres()` ➔ haal de waarde van de variabele Adres op en geef deze terug.

Let op: De functie 'Report' wordt uitgelegd in vraag 2c.

Een aantal klassen hebben vaste eigenschappen. Deze staan hier onder weergegeven. Bouw deze met de juiste taalconstructie in de aangegeven klassen. **(5 punten per correcte klasse)**

- `Fiets`
  - Wielen: 2
  - Aandrijving: "Pedalen"
- `Step`
  - Wielen: 2
  - Aandrijving: "Steppen"
- `DieselAuto`
  - Brandstof: "Diesel"
  - Wielen: 4
- `BenzineAuto`
  - Brandstof: "Benzine"
  - Wielen: 4
- `ElectrischeFiets`
  - Brandstof: "Elektrisch"
  - Wielen: 2

Vraag 2 (20 punten)

In deze vraag gaan we met de gemaakte klassen aan de slag.

2a: Voertuigen aanmaken (5 punten)

Nu je het klassendiagram hebt geïmplementeerd ga je voertuigen maken. Maak van elke type voertuig minimaal 2 verschillende instanties:

- Step
- Fiets
- Electrische fiets
- Diesel Auto
- Benzine Auto

2b: personen aanmaken (5 punten)

Maak twee personen aan van de klasse 'Persoon'. Koppel per persoon minimaal 3 voertuigen. Gebruik hiervoor de voertuigen die in je 2a hebt aangemaakt.

2c: Wat staat er in mijn garage? (10 punten)

De eigenaren van vraag 2b hebben nu een aantal vervoersmiddelen in hun bezit. Tijdens verjaardagsfeestjes is het natuurlijk altijd leuk om op te scheppen wat je zoal in je garage hebt staan. Dus is het nuttig dat er een functie is die netjes een overzicht maakt van alle voertuigen in de garage van een persoon.

In de klasse '`Persoon`' staat een methode '`Report()`'. Deze gaat een overzicht maken van de vervoersmiddelen van de betreffende persoon. Dit overzicht moet er als volgt uitzien:

Een voorbeeld:
Vervoersmiddelen van : Martin Molema

```
---------------------------------------
1 : Union:Pedalen
2 : Gazelle:Pedalen
3 : Stella E-bike:Electrisch
4 : Amslod:Electrisch
5 : Audi 80:Diesel
6 : Mercedes C63 AMG Black Edition:Benzine
Totaal 6 vervoersmiddelen
---------------------------------------
```

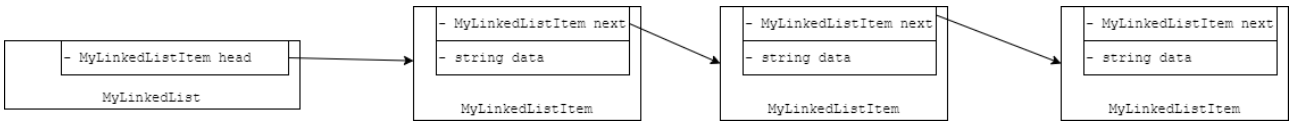
De opbouw van elke detail regel is als volgt:

```
<Positie> : <Naam van het voertuig>:<Aandrijving> of <Brandstof>
```

Vraag 3 (16 punten)

Voor deze vraag wordt een klasse '`MyLinkedList`' gebruikt die in de solution van Visual Studio zit. Je moet een aantal opdrachten uitvoeren die getest kunnen worden met NUnit. De solution is te vinden op BlackBoard. De klasse '`MyLinkedList`' is verborgen maar is wel bruikbaar.

De enkelvoudige gelinkte lijst wordt gebruikt om een lijst van integers, strings of andere typen te kunnen opslaan. Hiervoor wordt een klasse gebruikt met de naam '`MyLinkedListItem`'. In onderstaande diagram wordt deze structuur uitgelegd waarbij `strings` opgeslagen worden.



Een lijst van het type '`MyLinkedList`' bevat een verwijzing naar een **lijst** met gegevens. De start van deze lijst is opgenomen in de variabele '`head`'. De lijst bestaat uit items van het type '`MyLinkedListItem`'. In een item van dat type zit een getal ('`Data`') en een verwijzing naar het volgende element ('`Next`'). Als er geen volgend item is, dan heeft '`Next`' de waarde NULL.

Met de trein

We gaan de LinkedList structuur gebruiken om een lijst met stations op te slaan. Het eerste station dat opgevoerd wordt is het vertrekstation, het laatste station in de lijst is het eindpunt. Dit kan er als volgt uitzien:

```
LinkedListStudent list = new LinkedListStudent();

list.AddItem("Meppel");
list.AddItem("Steenwijk");
list.AddItem("Wolvega");
list.AddItem("Heerenveen");
list.AddItem("Akkrum");
list.AddItem("Grou/Irnsum");
list.AddItem("Leeuwarden");
```

Vervolgens moeten twee **<u>recursieve</u>** functies geschreven worden die deze lijst met stations op het scherm plaatsen. De eerste ('`ListStations`') laat de heenreis zien, de tweede ('`ListRetourRoute`') de reis terug. Een voorbeeld:

```
list.ListStations();
Console.WriteLine("-------------");
list.ListRetourRoute();
```

De uitvoer is dan:

```
Meppel
Steenwijk
Wolvega
Heerenveen
Akkrum
Grou/Irnsum
Leeuwarden
-------------
Leeuwarden
Grou/Irnsum
Akkrum
Heerenveen
Wolvega
Steenwijk
Meppel
```

In de Solution op BlackBoard staat een bestand `Program.cs` met daarin de functies `ListStations` (**8 punten**) en `ListRetourRoute` (**8 punten**). Pas deze functies aan zodat ze hierboven beschreven uitvoer produceren.

Vraag 4 (19 punten)

In de Solution zit in de map 'RecursiveStudentAssignment2'. Daar zit in bronbestand 'UnitTest1' één functie (`OneOr42`) die je moet afmaken. Deze kun je testen met NUnit tests.

In deze opgave moet je een **recursieve** oplossing bedenken om onderstaande probleem op te lossen.

Stel je hebt het getal 999. We nemen van elk cijfer in het kwadraat en tellen dit op:
$9^2 + 9^2 + 9^2 = 81 + 81 + 81 = 243$.

Van deze cijfers nemen we weer de kwadraten, en zo verder:
$2^2 + 4^2 + 3^2 = 4 + 16 + 9 = 29$ ➔ $2^2 + 9^2 = 4 + 81 = 85$ ➔ $8^2 + 5^2 = 64 + 25 = 89$ ➔ $8^2 + 9^2 = 145$ ➔ $1^2 + 4^2 + 5^2 = 1 + 16 + 25 = 42$!

En daar stopt de berekening: als er uit de som van de kwadraten het getal 1 of 42 komt, stopt de procedure en geef je alleen het getal af dat er uitgekomen is (dus 1 of 42). Je kunt dat als volgt testen:

```
int result = OneOr42(42);
if (result == 42) Console.WriteLine("ok");
```

In de solution zitten een tweetal testprocedures (NUnit) voorzien van zeer veel testdata. Deze tests kun je gebruiken om je code te testen.

----------EINDE VAN HET TENTAMEN----------

Puntenverdeling en controle

Vraag	Onderdeel	Sub	Punten	Som
1	a	1	5	35
		2	5	
	b	1..5	25	
2	a	5		20
	b	5		
	c	10		
3		1	8	16
		2	8	
4			19	19
Totaal				**90**

TOTAAL 90 + 10 start punten = 100