HO
GENT

OOSDII

Polymorfisme & Interfaces - Oefeningen

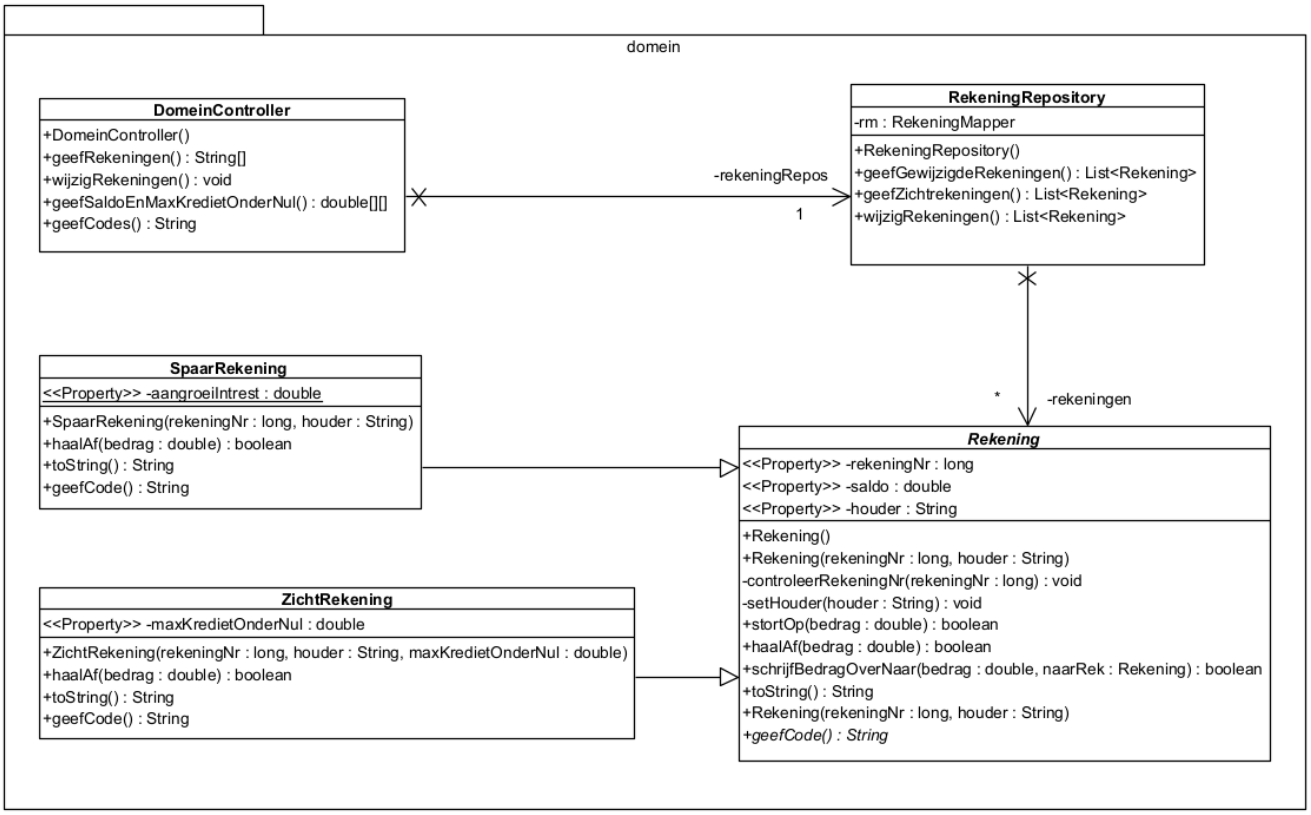
Table of Contents

1. Oefening 1

1.1. Gegeven: Rekening: Rode draad oefening

Gegeven een klasse Rekening (Versie oefeningen OOSDII - Overerving).

De eigenschappen, het gedrag en de relaties van en tussen klassen kan je aflezen uit onderstaand klassediagram:



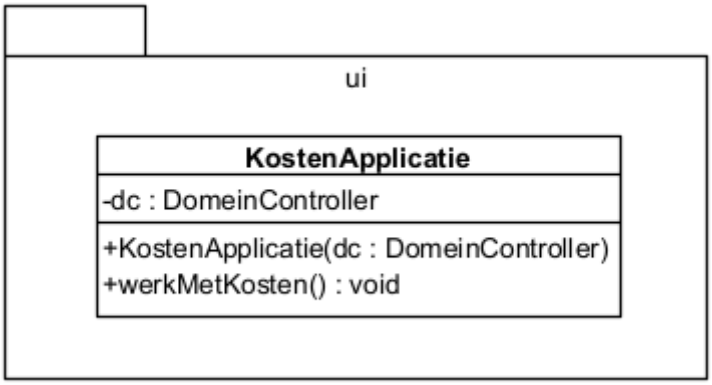
1.2. Gevraagd

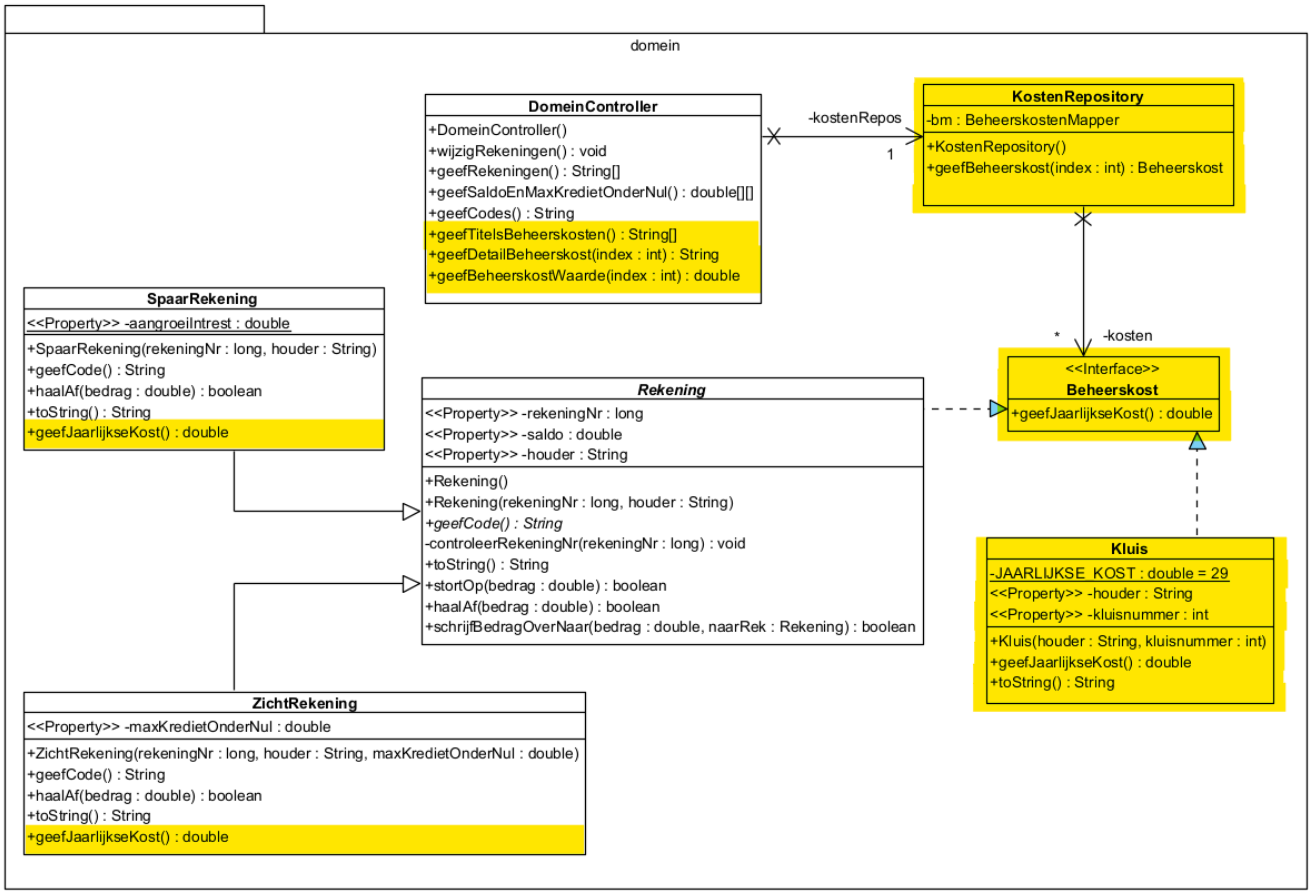
Uitbreidingen:

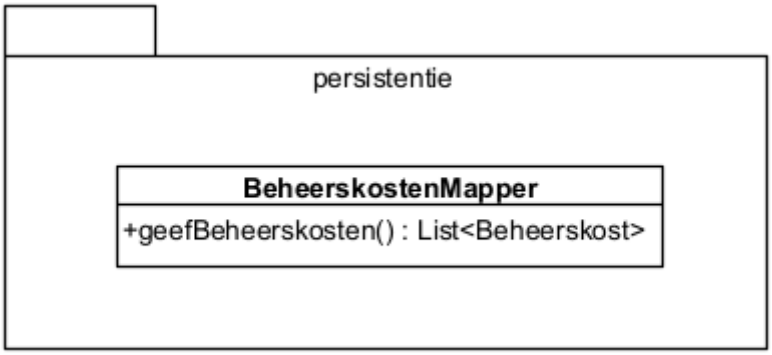
- Er zijn twee verschillende producten die een kost met zich meedragen: Rekening (zowel zicht als spaar) en het huren van een Kluis. Op het eerste zicht hebben Rekening en Kluis niets met elkaar te maken, toch willen we deze objecten op een zelfde manier behandelen.
- We voegen de BeheersKostenMapper toe.

1.2.1. Volledig UML


main
StartUp
+main(args : String[]) : void

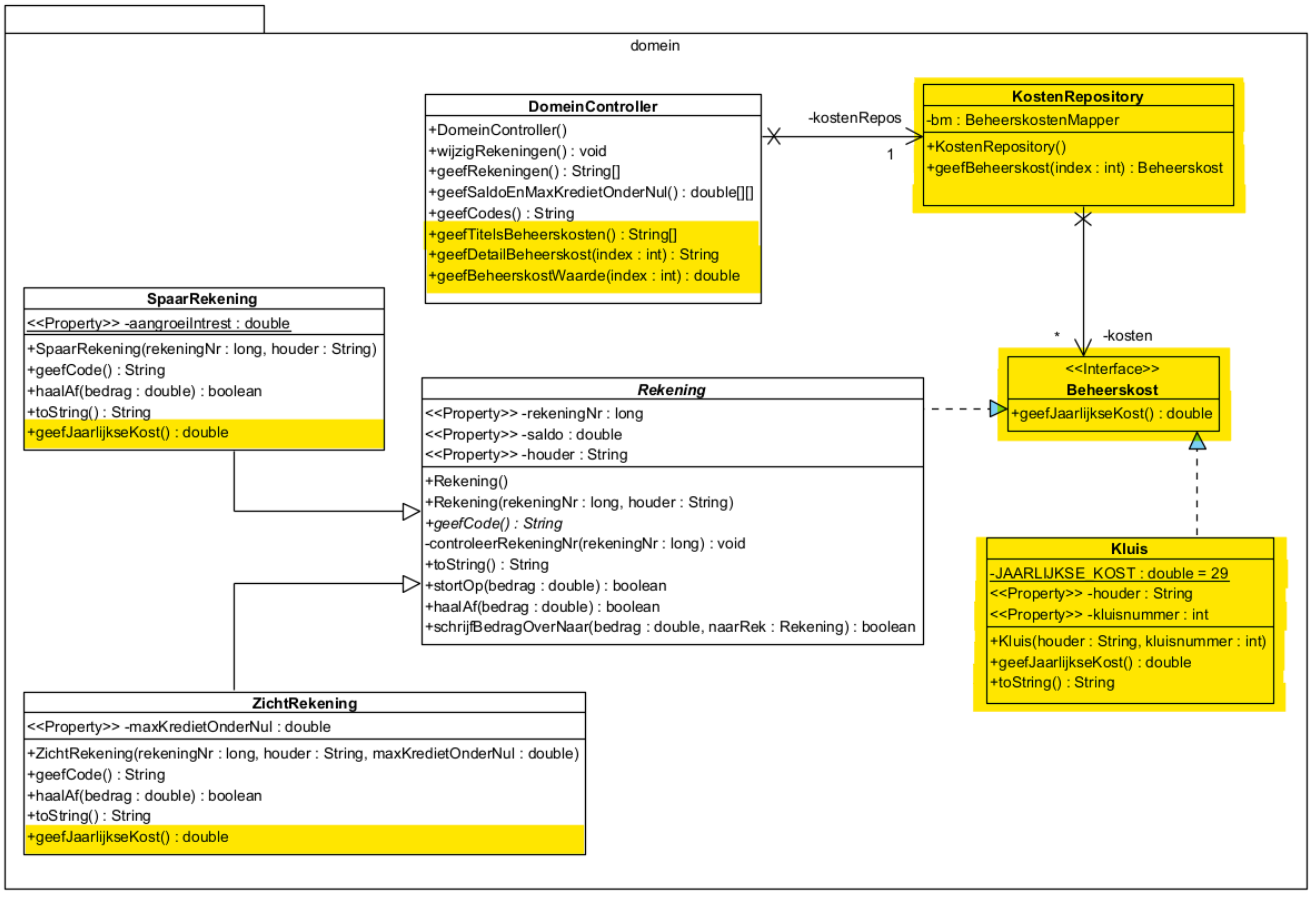

ui
KostenApplicatie
-dc : DomeinController
+KostenApplicatie(dc : DomeinController)
+werkMetKosten() : void


domein
DomeinController
+DomeinController()
+wijzigRekeningen() : void
+geefRekeningen() : String[]
+geefSaldoEnMaxKredietOnderNul() : double[][]
+geefCodes() : String
+geefTitelsBeheerskosten() : String[]
+geefDetailBeheerskost(index : int) : String
+geefBeheerskostWaarde(index : int) : double
-kostenRepos
1
KostenRepository
-bm : BeheerskostenMapper
+KostenRepository()
+geefBeheerskost(index : int) : Beheerskost
*
-kosten
<<Interface>>
Beheerskost
+geefJaarlijkseKost() : double
SpaarRekening
<<Property>> -aangroeiIntrest : double
+SpaarRekening(rekeningNr : long, houder : String)
+geefCode() : String
+haalAf(bedrag : double) : boolean
+toString() : String
+geefJaarlijkseKost() : double
Rekening
<<Property>> -rekeningNr : long
<<Property>> -saldo : double
<<Property>> -houder : String
+Rekening()
+Rekening(rekeningNr : long, houder : String)
+geefCode() : String
-controleerRekeningNr(rekeningNr : long) : void
+toString() : String
+stortOp(bedrag : double) : boolean
+haalAf(bedrag : double) : boolean
+schrijfBedragOverNaar(bedrag : double, naarRek : Rekening) : boolean
Kluis
-JAARLIJKSE_KOST : double = 29
<<Property>> -houder : String
<<Property>> -kluisnummer : int
+Kluis(houder : String, kluisnummer : int)
+geefJaarlijkseKost() : double
+toString() : String
ZichtRekening
<<Property>> -maxKredietOnderNul : double
+ZichtRekening(rekeningNr : long, houder : String, maxKredietOnderNul : double)
+geefCode() : String
+haalAf(bedrag : double) : boolean
+toString() : String
+geefJaarlijkseKost() : double



1.2.2. Domeinlaag

We starten met de domeinlaag:

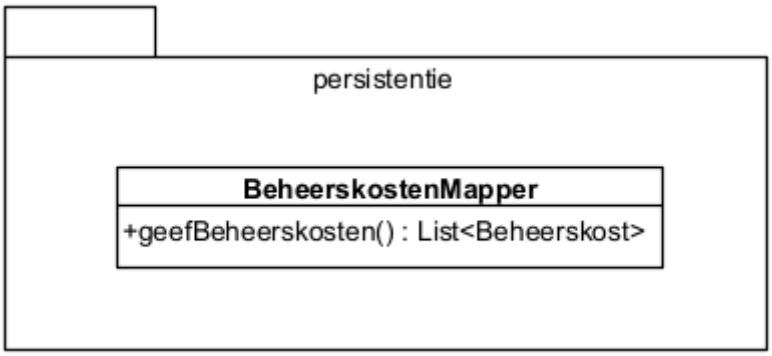


- Interface **Beheerskost**
  - geefJaarlijkseKost
- Klasse **Kluis** (implementeert interface BeheersKost):
  - final attributen: houder(String) en kluisnummer(int)
  - constructor: een kluis-object wordt aangemaakt en al zijn attributen worden opgevuld.
  - getters
  - toString: geeft het kluisnummer en houder als String terug
  - geefJaarlijkseKost: een kluis kost jaarlijks €29 (vaste prijs)

- De bestaande klasse **Rekening** implementeert ook de interface **Beheerskost**. Wat is het gevolg hiervan?
- De jaarlijkse kost van:
  - **Zichtrekening**: 1% van |maxKredietOnderNul|
  - **Spaarrekening**: indien saldo < €300, dan 5% van (€300 - saldo) anders 0
- De **KostenRepository** houdt een lijst van BeheersKost-objecten bij. De objecten worden gemaakt in de persistentielaag, meer bepaald in de BeheerskostenMapper.

Werk nu dus eerst verder aan de persistentielaag. Daarna kan dit stuk verder afgewerkt worden.

1.2.3. Persistentielaag



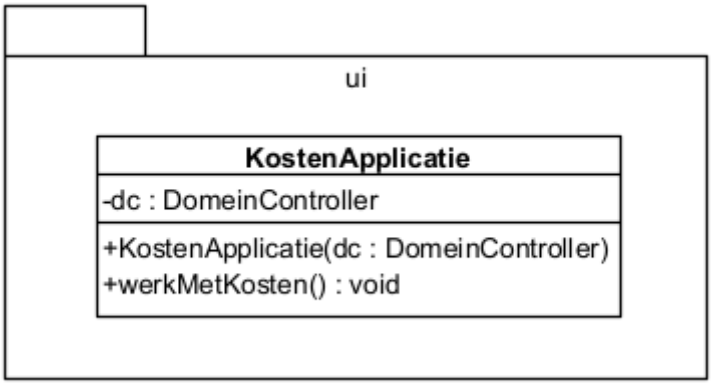
Klasse **BeheerskostenMapper**
* De methode geefBeheerskosten(): geeft een lijst terug gevuld met 2 kluizen, 1 zichtrekening en 1 spaarrekening aan de KostenRepository.
* De domeinlaag wil een lijst van **Beheerskosten** terugkrijgen en vraagt die aan de BeheerskostenMapper.

1.2.4. Afwerken domeinlaag

Implementeer nu de ontbrekende methodes in de DomeinController:

- Methode geefTitelsBeheersKosten
  - Geeft per BeheersKost een String terug:
  - Kluis: nummer en houder (hebben we daar iets voor?)
  - Rekening: type (spaar of zicht) en houder
- Methode geefDetailBeheersKost(index:int)
  - Geeft een String terug voor deze specifieke Beheerskost:
  - Kluis: nummer en houder
  - Rekening: alle details van de rekening opsommen!
- Methode geefBeheersKostWaarde(index:int)
  - Geeft een bedrag (double) terug. Dit is de jaarlijkse kost voor deze specifieke **Beheerskost**.

1.2.5. Presentatielaag



Maak een applicatie die per **BeheersKost** een titel, de detailinfo en de kost uitschrijft.

De applicatie zelf wordt opgestart via de StartUp-klasse.

```
zichtrekening, houder = Michiel
Details: ZichtRekening met rekeningnummer 000-0000004-04
staat op naam van Michiel
en bevat 400,00 euro. Max krediet onder nul = -2000,00
Kost: 20,00

kluisnr = 59, houder = Lander
Details: kluisnr = 59, houder = Lander
Kost: 29,00

zichtrekening, houder = Stef
Details: ZichtRekening met rekeningnummer 000-0000003-03
staat op naam van Stef
en bevat 300,00 euro. Max krediet onder nul = -1000,00
Kost: 10,00

spaarrekening, houder = Sebastien
Details: SpaarRekening met rekeningnummer 000-0000002-02
staat op naam van Sebastien
en bevat 200,00 euro. Aangroeiintrest = 2,00%
Kost: 5,00

spaarrekening, houder = Toon
Details: SpaarRekening met rekeningnummer 000-0000001-01
staat op naam van Toon
en bevat 100,00 euro. Aangroeiintrest = 2,00%
Kost: 10,00

kluisnr = 58, houder = Dylan
Details: kluisnr = 58, houder = Dylan
Kost: 29,00
```

2. Oefening 2

2.1. Gegeven Movie - Comparable interface

In het startproject 'H02_ComparableExample_Start' vind je de klassen Movie en EqualsApp terug.

Laat de klasse Movie de interface `Comparable` implementeren zodat je twee Movie objecten met elkaar (op natuurlijke wijze) kan sorteren.

Sorteer op basis van hun attributen `name` en `year`: sorteer eerst op basis van het attribuut `name`. Indien dit attribuut identiek is sorteer je op het attribuut `year`.

Let op: hoewel de compiler dit niet kan garanderen gaat men er wel van uit dat, indien twee objecten volgens hun `equals` methode gelijk zijn ze ook volgens de methode `compareTo` gelijk zijn.

Indien een lijst objecten bevat die de interface `Comparable` implementeren, dan kan deze lijst gesorteerd worden door de klasse methode `sort` in de klasse `Collections`. Deze methode is reeds aanwezig in de klasse ComparableApp maar staat in commentaar. Haal ze uit commentaar en beoordeel het resultaat.

```
Movies after sorting :
Empire Strikes Back 8,80 1980
Empire Strikes Back 8,80 2008
Force Awakens 8,30 2015
Return of the Jedi 8,40 1983
Star Wars 8,70 1977
```

3. Oefening 3

3.1. Gegeven Movie - Comparator interface

In het startproject 'H02_ComparatorExample_Start' vind je de klassen Movie en EqualsApp terug.

Implementeer een nieuwe klasse `RatingCompare` die de interface `Comparator` implementeert. Zorg dat de `compare` methode twee Movie objecten kan sorteren op basis van hun attribuut `rating`. Een hogere `rating` komt voor een lager.

Indien een lijst objecten bevat niet `Comparable` zijn, of je wil objecten op een andere manier sorteren dan op de natuurlijke wijze, dan kan deze lijst gesorteerd worden door de overloaded klasse methode `sort` in de klasse `Collections`. Deze methode verwacht als eerste parameter een lijst object, en als tweede parameter een `Comparator` object. Op basis van deze laatste zal de lijst gesorteerd worden.

Vul de klasse `ComparatorApp` aan zodat de gegeven lijst met Movie objecten gesorteerd wordt op basis van hun rating.

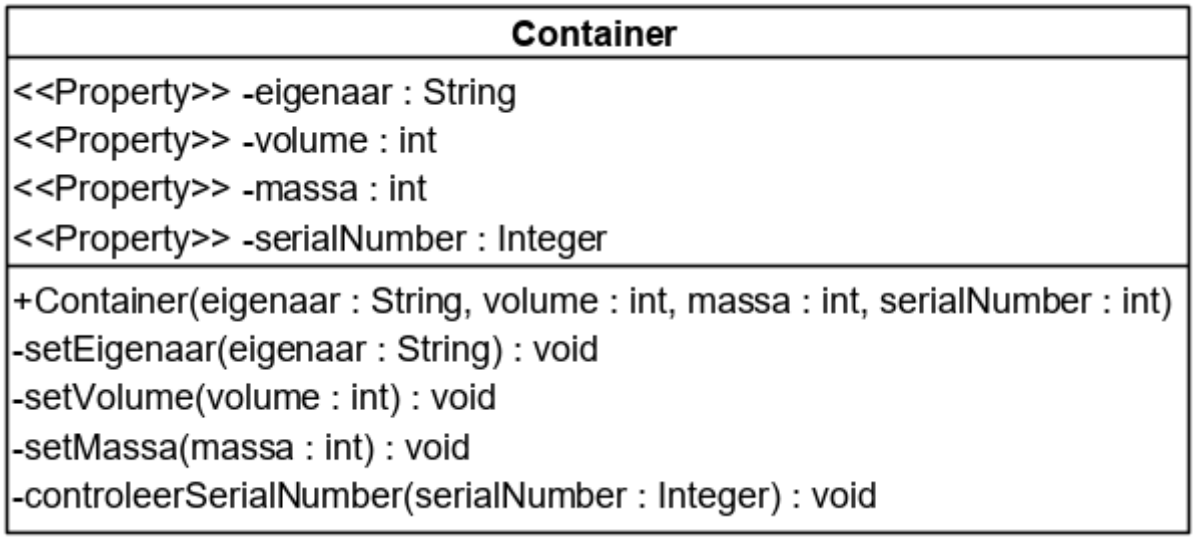
```
Sorted by rating
8.8 Empire Strikes Back 1980
8.7 Star Wars 1977
8.4 Return of the Jedi 1983
8.3 Force Awakens 2015
```

Aangezien de klasse `RatingCompare` hoort bij de klasse `Movie` kan je ze ook definiëren als een static geneste klasse binnen de klasse `Movie`. Op die manier is het duidelijker dat beide klassen bij elkaar horen.

4. Oefening 4 - Container

Gebruik het startproject H02_Oef_Container_Start van op Chamilo

4.1. Gegeven



4.2. Opdracht

Vul de ontbrekende code aan zodat de applicatie correct kan worden uitgevoerd. Bij de start van de applicatie worden vier containerobjecten aangemaakt.

Onderaan vind je een voorbeeld van de uitvoer.

- Deel 1:
  - Breid de klasse Container uit zodat deze op een natuurlijke wijze kan gesorteerd worden. Een natuurlijke sortering vindt plaats op basis van het serienummer (naar analogie met de implementatie van de equals-methode).
  - Pas deze sortering toe in de applicatie en genereer de uitvoer.

```
Containers bij natuurlijk sorteren:
1234 - Antwerpen - 150kg - 60m³
2568 - Rotterdam - 110kg - 70m³
8564 - Brugge - 100kg - 70m³
8569 - Calais - 90kg - 80m³
```

- Deel 2:
  - Breid het domein uit zodat, naast de natuurlijke sortering, ook kan gesorteerd worden op massa. Doe dit met een gewone klasse.
  - Pas deze sortering toe in de applicatie en genereer de uitvoer.

```
Containers bij sorteren op massa:
90kg - Calais - 80m³
100kg - Brugge - 70m³
110kg - Rotterdam - 70m³
150kg - Antwerpen - 60m³
```

- Deel 3:
  - Breid het domein uit zodat, naast de natuurlijke sortering, ook kan gesorteerd worden op eigenaar. Let op dat je beschikbare code herbruikt: het type van het attribuut `eigenaar` is String. De String klasse implementeert de interface Comparable.
  - Pas deze sortering toe in de applicatie en genereer de uitvoer.

```
Containers bij sorteren op eigenaar:
Antwerpen - 60m³ - 150kg
Brugge - 70m³ - 100kg
Calais - 80m³ - 90kg
Rotterdam - 70m³ - 110kg
```

- Deel 4:
  - Breid het domein uit zodat, naast de natuurlijke sortering, ook kan gesorteerd worden op volume en bij gelijk volume op eigenaar. Hergebruik de code die je voor deel 3 hebt geschreven.
  - Pas deze sortering toe in de applicatie en genereer de uitvoer.

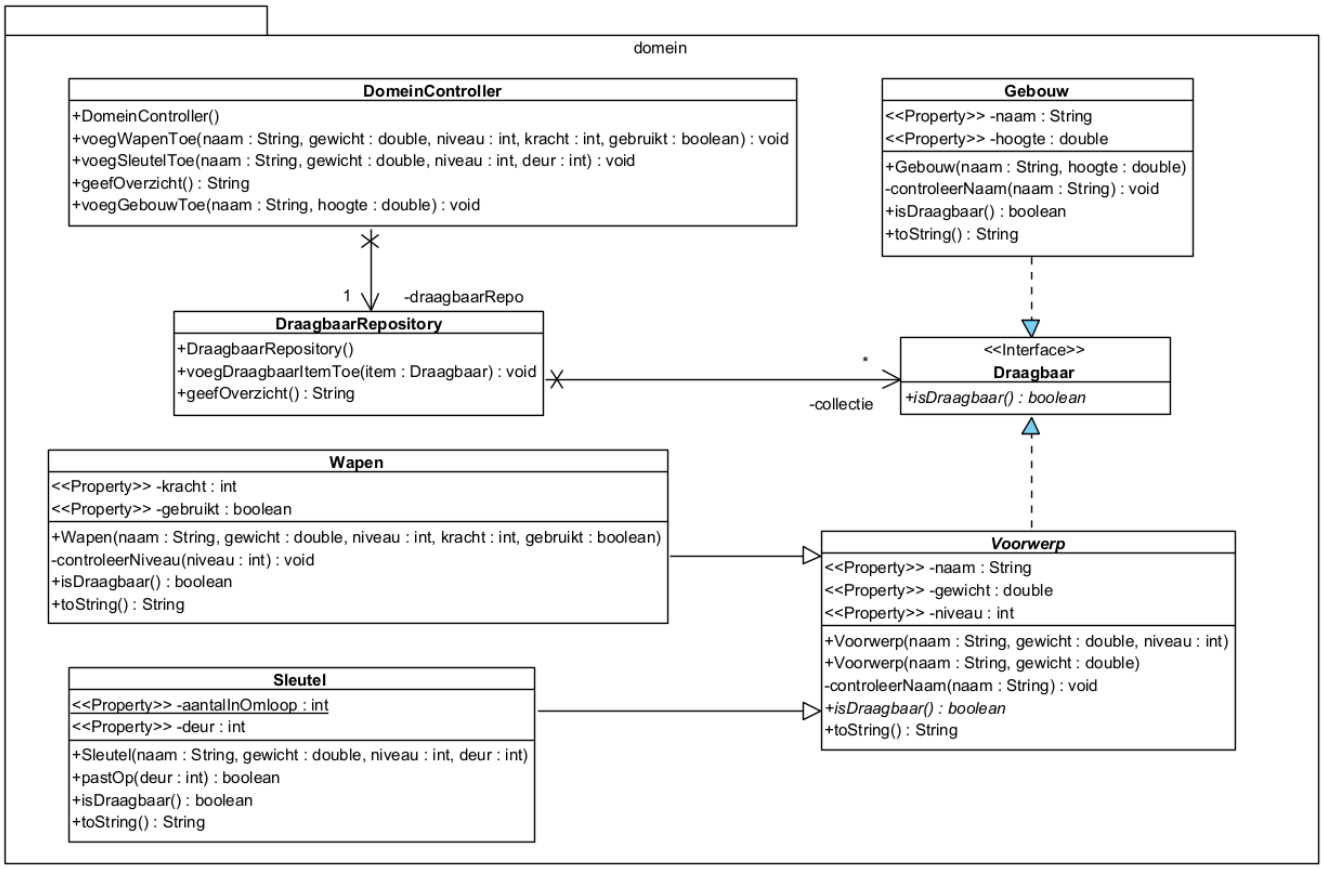
```
Containers bij natuurlijk sorteren:
60m³ - Antwerpen - 150kg
70m³ - Rotterdam - 110kg
70m³ - Brugge - 100kg
80m³ - Calais - 90kg
```

5. Oefening 5 - Voorwerp - Interface Draagbaar

Werk verder op de oefening Voorwerp uit OOSD1 - H7. Vertrek van het project 'H2_VoorwerpDraagbaar_Start'.

Breid het vorig domein verder uit met de interface Draagbaar die de methode isDraagbaar bevat, de klasse Gebouw en de DraagbaarRepository (ipv VoorwerpRepository) → **zie UML** (De hoogte van een gebouw is minimum 3; een gebouw is NIET draagbaar)

De klasse Voorwerp én de klasse Gebouw implementeren de interface Draagbaar:



Pas de andere klassen aan waar nodig en zorg ervoor dat je in de collectie OOK een Gebouw-object kan stockeren!

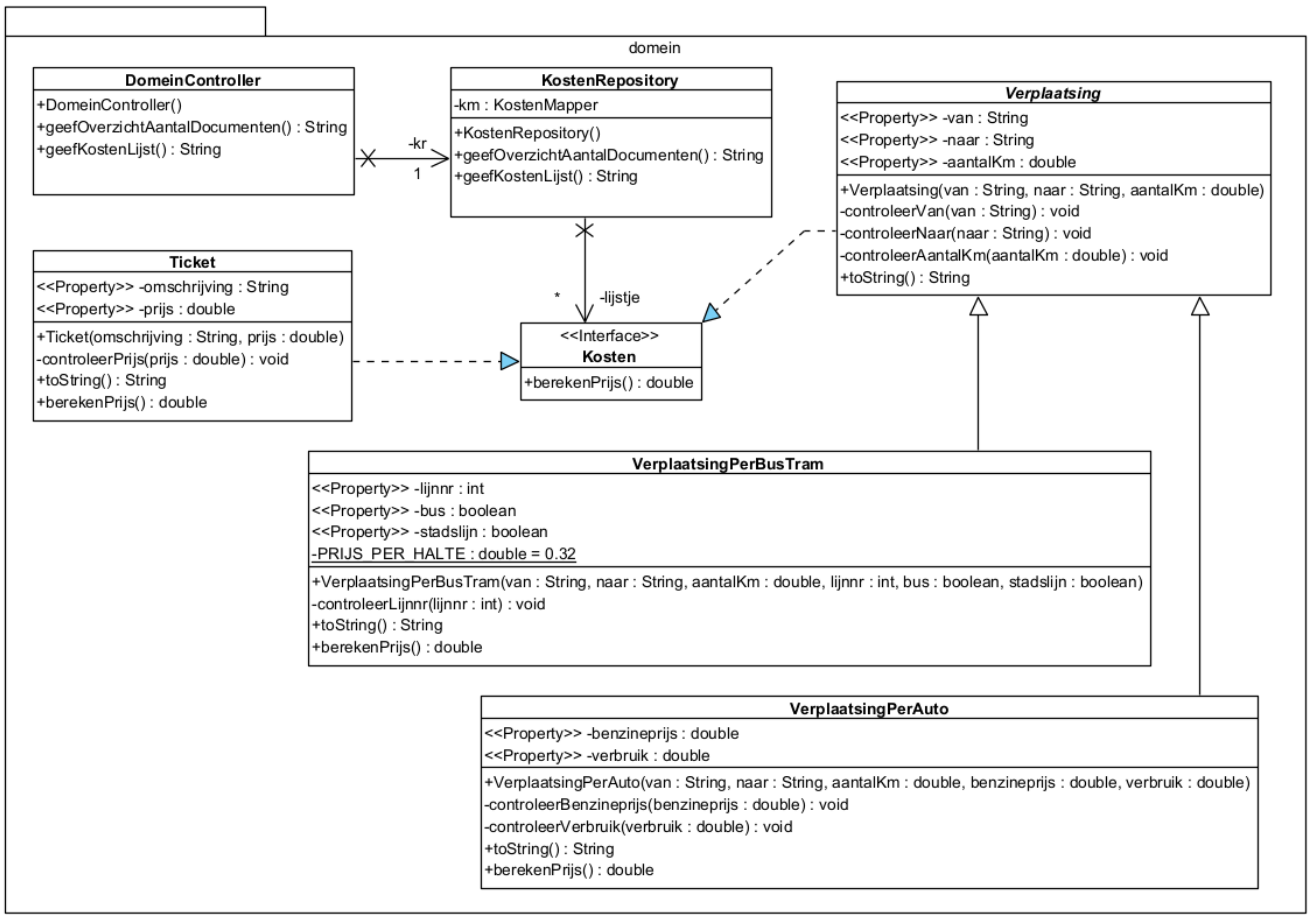
```
Wapen Colt met gewicht 1,500 kg uit niveau 3 en met kracht 6 nog niet gebruikt.
Sleutel Voordeur met gewicht 0,500 kg uit niveau 3 past op deur 1.
Er zijn 2 sleutel(s) in omloop.
Wapen Brown met gewicht 0,500 kg uit niveau 1 en met kracht 23 al gebruikt.
Sleutel Achterdeur met gewicht 0,500 kg uit niveau 1 past op deur 2.
Er zijn 2 sleutel(s) in omloop.
residentie Frankenstein met hoogte 4,5 is niet draagbaar.
```

6. Oefening 6 - Verplaatsing

Werk verder op de oefening Verplaatsing uit OOSD1 - H7. Vertrek van het project 'H02_Verplaatsing_Start'.

Breid het domein verder uit met de interface Kosten, de klasse Ticket, de KostenRepository en de DomeinController → zie UML

6.1. package domein



De interface Kosten:

- bevat de methode berekenPrijs.

De klasse Ticket (zie ook UML en testklasse):

- 2 attributen: omschrijving (verplicht, String) en prijs (double)
- constructor met 2 parameters
- getter voor elk attribuut
- setter voor de prijs:
  - prijs moet een strikt positief getal zijn, anders passende exception gooien met boodschap (via de methode controleerPrijs)
- De prijs van een ticket wordt berekend door de opgegeven prijs terug te geven.
- toString: voorzie 20 posities voor de omschrijving en een totale breedte van 10 posities voor de prijs (2 cijfers na de komma)

De klasse KostenRepository:

- 2 attributen: km (KostenMapper) en lijstje (List<Kosten>)
- constructor: lijst van kosten wordt hierin opgehaald uit de persistentielaag ahv het mapperobject
- methode geefOverzichtAantalDocumenten: geeft een String weer met de tekst "Volgende documenten werden ingediend: x ticket(s), y verplaatsing(en) per auto en z verplaatsing(en) per bus/tram" waarbij x, y en z in de methode bepaald worden adhv de gegevens uit de lijst
- methode geefKostenlijst geeft een String terug met daarin in tabelvorm (kolom 1: 70 posities breed, kolom 2: 15 posities breed) de gegevens van de kostenpost en de bijhorende prijs
  - voor een ticket worden als gegevens alleen de omschrijving getoond
  - voor een verplaatsing wordt 21% btw van de prijs afgetrokken alvorens de prijs getoond wordt In deze methode bepalen we ook de totale kosten en geven dit in de String al laatste door.

De klasse Domeincontroller:

- 1 attribuut: kr (KostenRepository)
- constructor: de KostenRepository kr wordt gecreëerd en de domeincontroller kan hieraan vragen stellen om de informatie van de KostenRepository door te geven aan de GUI via de terugkeerwaarde van een methode.

6.2. Package persistentie

De klasse KostenMapper:

De methode geefKostenLijst geeft een lijstje terug met enkele kostenposten (verplaatsingen per auto en per bus/tram zowel als tickets door elkaar):

```
new VerplaatsingPerBusTram("Voskenslaan Gent", "Veldstraat Gent", 4.2, 1, false,
true));
new Ticket("Parkeerticket Flanders Expo", 5));
new VerplaatsingPerAuto("Gent", "Oudenaarde", 29.4, 1.591, 0.0538));
new Ticket("Toegangsticket beurs", 10));
new VerplaatsingPerBusTram("Laarne dorp", "Gent Sint-Pieters", 14.6, 34, true,
false));
new VerplaatsingPerAuto("Gent", "Brussel", 53.7, 1.488, 0.0692));
new VerplaatsingPerAuto("Gent", "Kortrijk", 47.1, 1.649, 0.084));//foutieve waarde!
```

6.3. Package main

In de klasse StartUp wordt de main-methode geschreven die de methode start uit de ui-klasse oproept. Vergeet niet een object van de DomeinController aan te maken.

6.4. Package ui

De start-methode in de klasse VerplaatsingApplicatie verzorgt onderstaande uitvoer. Zorg ervoor dat de overzichtstabel mooi uitgelijnd is zoals in onderstaand voorbeeld.

Mogelijke uitvoeren:

- Foutmelding in console door foutieve waarde in laatste object: -→ een Exception wordt gegooid.

```
Het verbruik moet tussen de 0.02 en de 0.07 liter per km liggen
```

- 0.084 wijzigen in 0.034 levert volgende output op:

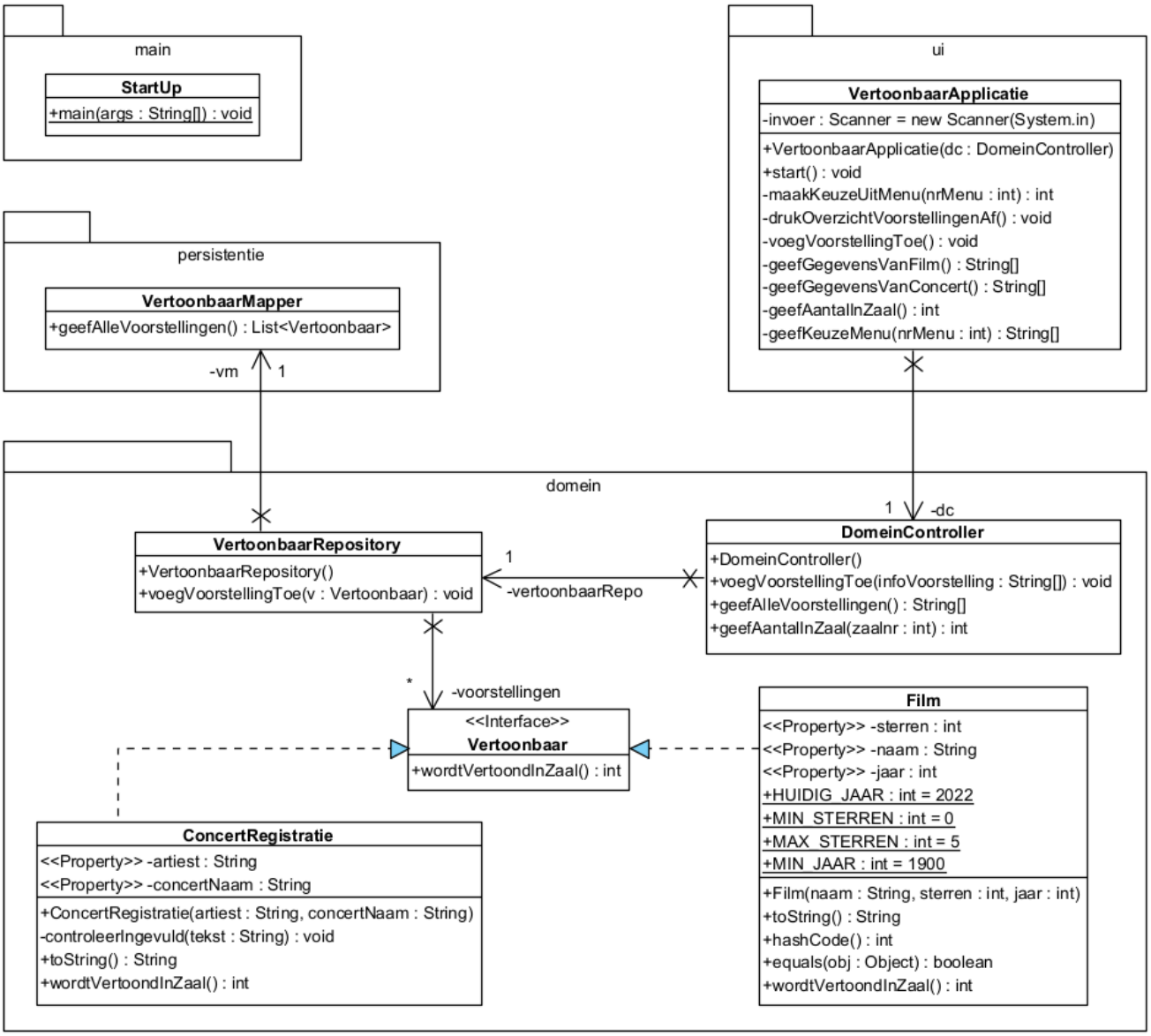
```
Volgende documenten werden ingediend:
2 ticket(s), 3 verplaatsing(en) per auto en 2 verplaatsing(en) per bus/tram.

Overzicht gemaakte kosten:

                                                          Kostenpost         Bedrag
verplaatsing van Voskenslaan Gent naar Veldstraat Gent met stadstram 1          2,12
                                                Parkeerticket Flanders Expo      5,00
                                verplaatsing van Gent naar Oudenaarde per auto      4,16
                                                Toegangsticket beurs     10,00
                    verplaatsing van Laarne dorp naar Gent Sint-Pieters met bus 34      7,93
                                  verplaatsing van Gent naar Brussel per auto      9,14
                                  verplaatsing van Gent naar Kortrijk per auto      4,36


Totaal gedeclareerde kosten =   42,71
```

7. Oefening 7 - Film

Werk verder op de oefening Film uit het vorige hoofdstuk over Overerving. Vertrek van het project 'H02_Film_Start'.

Gegeven: de domeinklasse Film en de bijhorende testklasse, de klasse StartUp, de ui-klasse en een mapperklasse die een voorbeeldlijst met gegevens kan ophalen. Enkel aan de klasse Film dient iets gewijzigd te worden (zie verder).

Gevraagd: breid het domein verder uit met de interface Vertoonbaar, de klasse ConcertRegistratie, de VertoonbaarRepository en de DomeinController. Voorzie ook een testklasse voor de klasse ConcertRegistratie.



7.1. package domein

De interface Vertoonbaar:

- bevat de methode wordtVertoondInZaal die het zaalnummer teruggeeft waarin de voorstelling zal vertoond worden.

De klasse Film ondertekent deze interface

- een film wordt vertoond in zaal 1 als hij 4 of 5 sterren heeft, in zaal 2 als hij 3 sterren heeft en in zaal 3 als hij minder dan 3 sterren gekregen heeft

De klasse ConcertRegistratie ondertekent eveneens de interface Vertoonbaar

- schrijf **eerst** een test voor deze klasse, waarna je de bijhorende code schrijft die hieraan voldoet
- 2 attributen: artiest en concertnaam (beide verplicht, na creatie niet wijzigbaar, String)
- constructor met 2 parameters
- getter voor elk attribuut
- methode controleerIngevuld
  - kan gebruikt worden om te controleren of de artiest ingevuld is en eveneens om te checken of de concertnaam ingevuld is
  - de waarde die men wenst te controleren wordt als parameter aan deze methode meegegeven
  - de methode gooit een exception als de parameterwaarde een nullreferentie bevat, een lege String bevat of enkel spaties
- toString: toont de naam van de runtime klasse waartoe het object behoort, gevolgd door de concertnaam en de artiest
- een concert wordt altijd in zaal 2 vertoond

De klasse VertoonbaarRepository:

- 2 attributen: vm (VertoonbaarMapper) en voorstellingen (List van Vertoonbaar)
- constructor: haalt de lijst van voorstellingen op uit de mapper
- voegVoorstellingToe: als de voorstelling uit de parameter nog niet in het lijstje zit, mag ze toegevoegd worden
- getter voor voorstellingen
- geefAantalInZaal: bepaalt het aantal voorstellingen dat wordt vertoond in de zaal die opgegeven is als parameter

De klasse Domeincontroller:

- 1 attribuut: vertoonbaarRepo (VertoonbaarRepository)
- constructor: de repository wordt gecreëerd zodat deze kan gebruikt worden in de methodes om hier dingen aan te vragen
- voegVoorstellingToe:
  - op basis van de lengte van de array die als parameter meegegeven werd, kan bepaald worden of het om een Film of een concertRegistratie gaat
  - bij een Film moeten het 2e en 3e element uit de meegegeven array (respectievelijk het aantal sterren en het jaar) eerst omgezet worden naar een geheel getal (Integer.parseInt), waarna hiermee een Film-object kan gemaakt worden
  - bij een ConcertRegistratie kan met het 1e en 2e element uit de array een object geregistreerd worden
  - het gemaakte object wordt doorgegeven aan de methode voegVoorstellingToe van de repository

- geefAlleVoorstellingen
  - zet de lijst van voorstellingen uit de repository om naar een array van Strings
  - elke voorstelling wordt omgezet naar één element van deze array die bestaat uit de tekstuele weergave, aangevuld met "in zaal x", waarbij x het nummer van de zaal voorstelt
- geefAantalInZaal: vraagt het aantal voorstellingen die in een bepaalde zaal wordt vertoond op aan de repository

Voorbeeld uitvoer:

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 1
Overzicht voorstellingen:
Film film A - 5 - 2020 in zaal 1
ConcertRegistratie concert 1 van artiest 1 in zaal 2
ConcertRegistratie concert 2 van artiest 2 in zaal 2
Film film B - 3 - 2018 in zaal 2
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 2
1. Voeg een film toe
2. Voeg een concertregistratie toe
3. Keer terug naar hoofdmenu
Geef je keuze: 1
Geef de naam van de film: film C
Geef het jaar waarin de film uitkwam: 2021
Hoeveel sterren verdient deze film: 4
```

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 2
1. Voeg een film toe
2. Voeg een concertregistratie toe
3. Keer terug naar hoofdmenu
Geef je keuze: 2
Geef de naam van de artiest: artiest 3
Geef de benaming van het concert: concert 3
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 1
Overzicht voorstellingen:
Film film A - 5 - 2020 in zaal 1
ConcertRegistratie concert 1 van artiest 1 in zaal 2
ConcertRegistratie concert 2 van artiest 2 in zaal 2
Film film B - 3 - 2018 in zaal 2
Film film C - 4 - 2021 in zaal 1
ConcertRegistratie concert 3 van artiest 3 in zaal 2
```

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 3
Geef het zaalnr:
1. Zaal 1
2. Zaal 2
3. Zaal 3
4. Keer terug naar hoofdmenu
Geef je keuze: 1
Het aantal voorstellingen bedraagt 2
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 3
Geef het zaalnr:
1. Zaal 1
2. Zaal 2
3. Zaal 3
4. Keer terug naar hoofdmenu
Geef je keuze: 2
Het aantal voorstellingen bedraagt 4
```

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 3
Geef het zaalnr:
1. Zaal 1
2. Zaal 2
3. Zaal 3
4. Keer terug naar hoofdmenu
Geef je keuze: 3
Het aantal voorstellingen bedraagt 0
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 4
```