# STAMBOOM

*C2J Opdracht 2*

| | |
|---|---|
| Versie | 2015-01-16 Bas Michielsen (Template)<br>2014-02-01 Frank Peeters |

## Inleiding

In de vorige opdracht hebben we onderstaande taken geïmplementeerd, uiteraard dienen deze te blijven werken.

1. registratie persoon uit een al of niet bekend gezin;
2. registratie ongehuwd gezin;
3. registratie huwelijk;
4. registratie scheiding;
5. opvragen van gezinsgegevens;
6. zoeken en selecteren persoon op basis van de achternaam ongeacht hoofd- en kleine letters.

In deze opdracht gaan we aan de slag met de drie taken 7 t/m 9:

7. ophalen van de administratie uit een bestand;
8. bewaren van de administratie in een bestand;
9. opvragen van stamboomgegevens voor een persoon.

Allereerst richten we ons op het bewaren en ophalen van stamboomgegevens. Bestudeer daartoe onderstaand klassendiagram dat een beeld schept van het eindproduct.

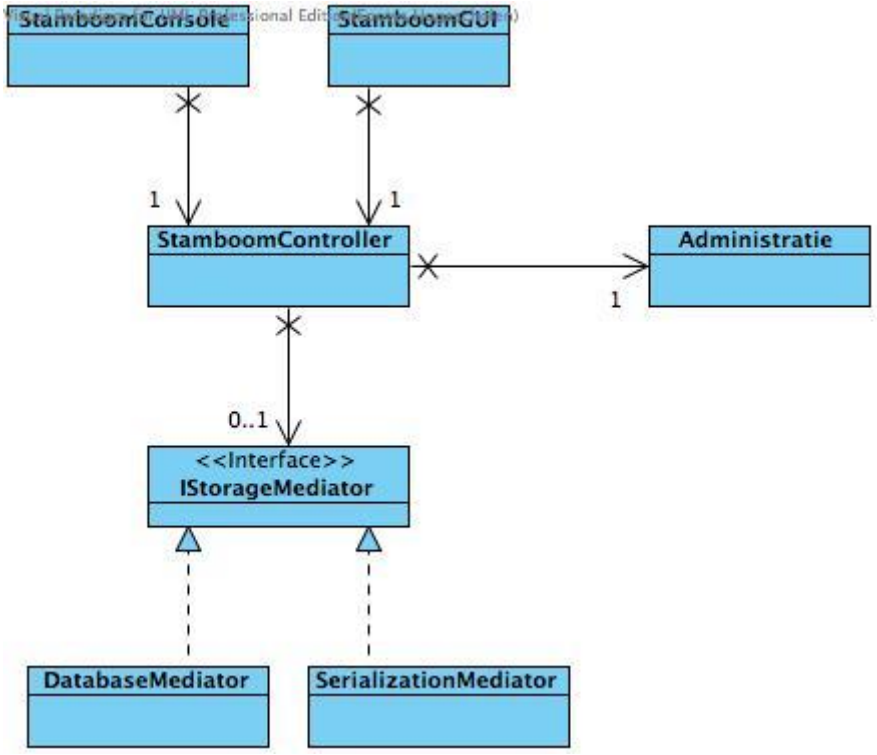


Figure 1: Overzicht van de diverse lagen in de stamboomapplicatie

Het `StamboomController`-object vormt het intermediair tussen alle lagen (Model, View, Opslag) in de stamboomapplicatie. De Modellaag wordt gevormd door `Administratie`, `Persoon` en `Gezin`. In opgave 1 en 2 wordt nieuwe stamboominformatie via de (primitieve) view van het `StamboomConsole`-object ingevoerd en actuele stamboominformatie getoond. In opgave 3 wordt zo straks de `StamboomConsole`-klasse vervangen door de grafische user interface van `StamboomGUI`. Binnen deze tweede opgave gaan we nog even door met de ouderwetse invoer en uitvoer via een console. In verband met de permanente opslag van stamboomgegevens kunnen de gegevens in een bestand of in een database worden opgeslagen. Het `StamboomController`-object zorgt ervoor dat de gegevens naar het verkozen opslagmedium worden doorgestuurd. In deze opgave gaan we aan de slag met het opslaan in een geserialiseerd bestand. In opgave 4 komt dan de opslag in een database aan bod. In de `IStorageMediator`-interface is in algemene termen vastgelegd hoe de gegevens kunnen worden opgeslagen en

opgehaald. Daarnaast wordt er in `IStorageMediator` beschreven hoe het onderliggende opslagmedium kan worden geconfigureerd:

```
public interface IStorageMediator {
  Administratie load() throws IOException;
  void save(Administratie admin) throws IOException;
  boolean configure(Properties props);
}
```

# Opdrachten

- Vul de broncode in SerializationMediator aan (zie // todo opgave 2). Maak gebruik van java.io.ObjectInputStream en java.io.ObjectOutputStream.
- Vul de broncode in `StamboomController` aan (zie // todo opgave 2).
- Test je code m.b.v. `SerialisatieTest`.
- Breid de `MenuItem`-klasse met de extra menu-items uit en voeg bijbehorende code aan de klasse `StamboomConsole` toe, zodat je taak 7 en 8 kunt testen.

Taak 9 gaat over het tonen van een stamboom van alle geregistreerde voorouders van een persoon. Zo'n stamboom zou er voor M.G. Pieterse als volgt uit kunnen zien.

```
M.G. Pieterse (VROUW) 5-5-2004
  L. van Maestricht (MAN) 27-6-1953
    A.G. von Bisterfeld (VROUW) 13-4-1911
      I.T.M.A. Goldsmid (VROUW) 22-12-1876
      F.A.I. von Bisterfeld (MAN) 27-6-1874
    H.C. van Maestricht (MAN) 17-2-1909
  J.A. Pieterse (MAN) 23-6-1964
    M.A.C. Hagel (VROUW) 12-0-1943
    J.A. Pieterse (MAN) 4-8-1923
```

Van elke persoon in de stamboom worden de `standaardgegevens()` getoond. Elke voorgaande generatie springt twee spaties in. De ouders van M.G. Pieterse zijn

dus L. van Maestricht en J.A. Pieterse. De ouders van L.van Maestricht zijn A. G. von Bisterfeld en H.C. van Maestricht etc.

De tekst van zo'n stamboom valt met de gangbare middelen van een for- of while-statement moeilijk samen te stellen. Dat heeft te maken met het feit dat een stamboom een zogenaamde recursieve datastructuur heeft. We gaan daarom dit soort problemen doorgaans met een recursieve aanpak te lijf. Nadat de docent de recursieve aanpak klassikaal heeft geïnstrueerd, ga je eerst oefenen met een eenvoudiger probleem:

- Schrijf in de klasse `Persoon` een recursieve oplossing voor de methode

```
/**
  * @return het aantal personen in de stamboom van deze persoon
  * (ouders, grootouders etc); de persoon zelf telt ook mee
  */
public int afmetingStamboom()
```

Om de gewenste stamboomoutput te genereren worden de gegevens eerst verzameld in een `ArrayList`. Deze `ArrayList` bevat objecten van de klasse `PersoonMetGeneratie`. Een beschrijving van deze klasse staat hieronder.

```
public class PersoonUitGeneratie {
	private String tekst;
	private int generatie;

	public PersoonUitGeneratie(String tekst, int generatie) {
		this.tekst = tekst;
		this.generatie = generatie;
	}

	public int getGeneratie() {
		return generatie;
	}

	public String getTekst() {
		return tekst;
	}
}
```

De property `generatie` verwijst naar de positionering van een persoon in de stamboom van een gegeven persoon. Als bijvoorbeeld Marie de grootmoeder is van Piet, dan zal het `PersoonUitGeneratie`–object uit de stamboom van Piet, dat betrekking heeft op Marie een generatienummer 2 hebben. Piet zelf heeft generatienummer 0 en Marie is twee generaties ‘ouder’ dan Piet.

De `tekst`-property van dit `PersoonUitGeneratie`-object bevat de standaardgegevens van Marie, hetgeen neerkomt op een vermelding van haar naam, geslacht en geboortedatum.

In de klasse `Persoon` zit een methode om de stamboomgegevens van de betreffende persoon aan een bestaande `lijst` toe te voegen. De betreffende persoon hoort bij generatie `g`:

```
void voegJouwStamboomToe(
  ArrayList<PersoonMetGeneratie> lijst, int g)
```

De `lijst`-parameter vervult hierin een bijzondere rol. Hij wordt namelijk zowel als invoer- ook als uitvoerwaarde gebruikt. Dat laatste wil zeggen dat deze parameter binnen de aanroep wordt gewijzigd. Er worden immers de stamboomgegevens van de betreffende persoon aan toegevoegd. Je zou eigenlijk verwachten dat deze lijst ook als returnwaarde is gedefinieerd. Maar dat is dus kennelijk niet per se nodig.

In de klasse `Persoon` is ook nog een String-methode waarmee de stamboom via de console-klasse zichtbaar kan worden gemaakt:

```
public String stamboomAlsString()
```

- Schrijf de recursieve methode `voegJouwStamboomToe` en de niet-recursieve methode `stamboomAlsString`.
- Test je code met stamboom.domain.StamboomTest.
- Breid `MenuItem` en `StamboomConsole` uit, zodat taak 9 uitvoerbaar is.