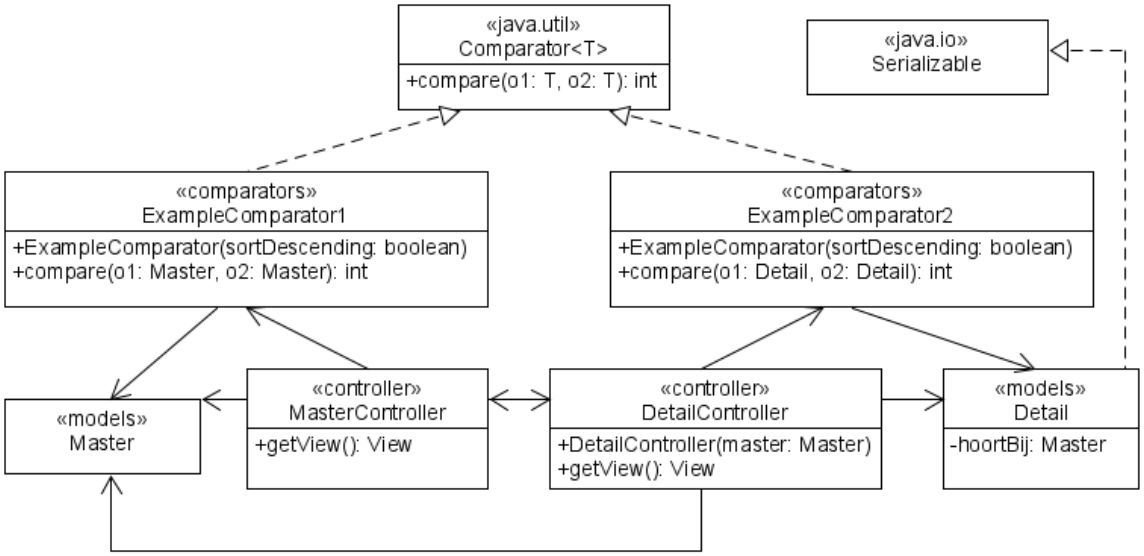
OOP2 – Practicumopdracht

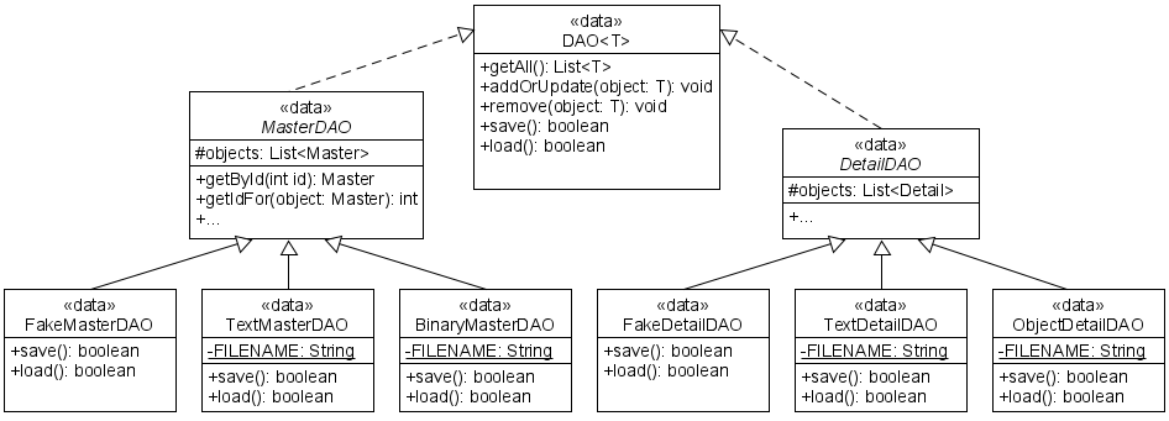
Week 6

Een heugelijk moment is aangebroken: je gaat deze week je applicatie afronden!

Vorige week heb je op basis van Text IO ervoor gezorgd dat er daadwerkelijk data opgeslagen en ingeladen kan worden. Deze week komen daar de laatste twee varianten van IO bij: *Binary* en *Object Serialization*. Daarnaast gaan we het mogelijk maken om de twee *ListViews* in je applicatie op verschillende manieren te sorteren met gebruik van *Comparators*.



Afbeelding: Versimpelde UML van de applicatie met enkel de onderdelen en relaties die voor deze week relevant zijn (deel 1).



Afbeelding: UML met twee nieuwe DAO-classes, er zijn geen verandering in de structuur nodig (deel 2).

Werk nu het volgende stappenplan af met gebruik van de bovenstaande twee UML diagrammen:

1. Maak voor beide models twee nieuwe DAO-classes in de *data*-package en geef deze een naam in het volgende formaat:

   Master-model: Binary<NaamVanModel>DAO (bijv. *BinaryTodoLijstDAO*)
   Detail-model: Object<NaamVanModel>DAO (bijv. *ObjectTodoRegelDAO*)

2. Implementeer voor beide DAO's de *load*- en *save*-methodes. Bij de DAO voor je Master-model maak je gebruik van Binary IO. Bij de DAO voor je Detail-model maak je gebruik van Object Serialization. Je mag wederom zelf bepalen op wat voor manieren je dit doet, maar uiteraard moet wel alle data daadwerkelijk opgeslagen en weer ingeladen kunnen worden.

   - Zorg ervoor dat gebruikte bestanden worden vrijgegeven als je klaar bent.
   - Handel foutmeldingen die optreden af en print deze met *System.err* naar de console, zorg ervoor dat foutmeldingen specifiek aangeven wat er aan de hand is (bijv. "Bestand niet gevonden", "Bestand al in gebruik", enzovoorts).

3. Omdat er twee verschillende DAO's worden gebruikt in de applicatie, heeft het opslaan van de referentie naar het Master-model in je Detail-model geen enkel nut. Zorg ervoor dat deze referentie wordt uitgesloten van *Object Serialization* en dat je een andere manier verzint om toch de referentie naar het Master-object op te slaan.

   - **Let op!** Gebruik alles wat je in de afgelopen weken hebt toegevoegd aan de applicatie. Je hoeft dus geen nieuwe methodes aan de DAO's toe te voegen om dit voor elkaar te krijgen.

4. Pas de *MainApplication* zodanig aan dat de rest van de applicatie gebruik maakt van de nieuwe DAO's.

5. Verzin een nuttige manier om de *ListView* op Master-view te sorteren. Maak vervolgens een *comparators*-package aan in de *practicumopdracht*-package en voeg hier een class voor je Comparator aan toe met het volgende formaat: <TypeSortering>Comparator (bijv. *NameComparator*).

   - **Let op!** Zorg ervoor dat de Comparator zowel oplopend als aflopend gesorteerd kan worden, doe dit allemaal binnen één en dezelfde class zodat er optimaal code hergebruikt kan worden.

6. Zorg ervoor dat bij het opstarten van de Master-view deze Comparator direct wordt toegepast, zodat data altijd op een manier gesorteerd is.

7. Breid de menubalk van je Master-view uit met een "Sorteren" optie en geef deze twee andere opties: één om de ListView oplopend mee te sorteren en één om de ListView aflopend te sorteren (bijv. "Naam (A-Z)" en "Naam (Z-A)").

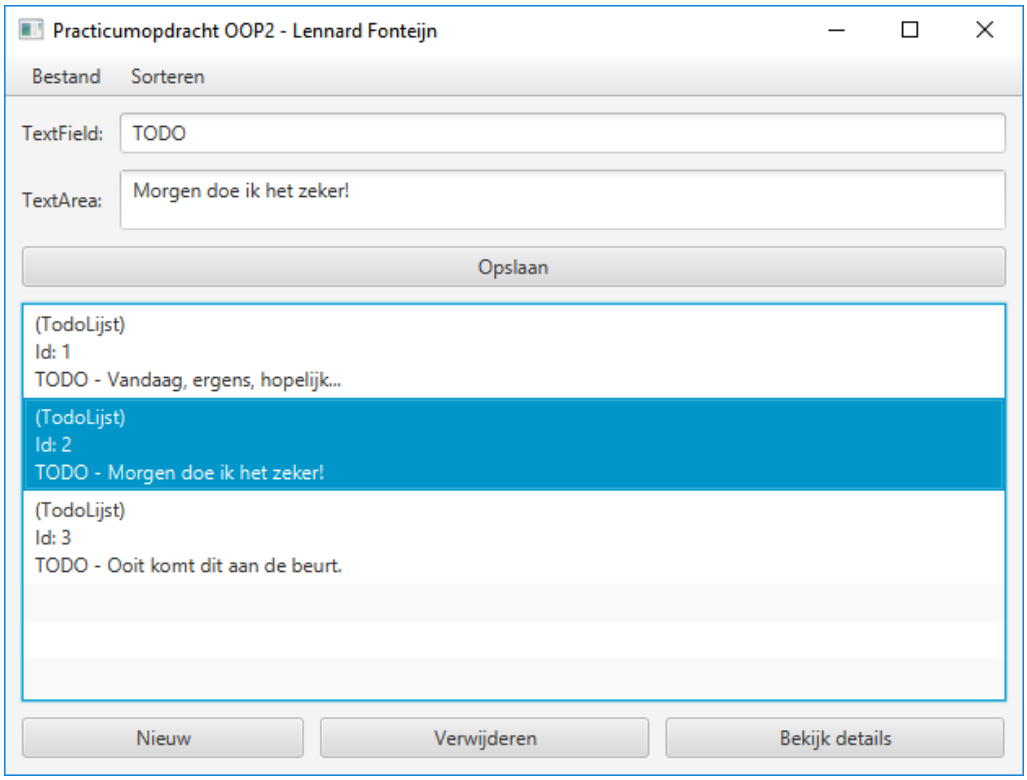
8. Zorg ervoor dat bij het aanklikken van de opties in het menu de Comparator wordt toegepast op de ListView en deze direct de nieuwe sortering toont.

9. Verzin twee nuttige manieren om de *ListView* op de Detail-view te sorteren met toepassing van dezelfde regels als bij de eerste Comparator (naamgeving, oplopend/aflopend). Zorg ervoor dat één van de twee Comparators een dubbele sortering gebruikt, hiermee bedoelen we: Indien twee waardes overeen komen, wordt er op een tweede manier gesorteerd (bijv. Eerst op naam, vervolgens op datum).

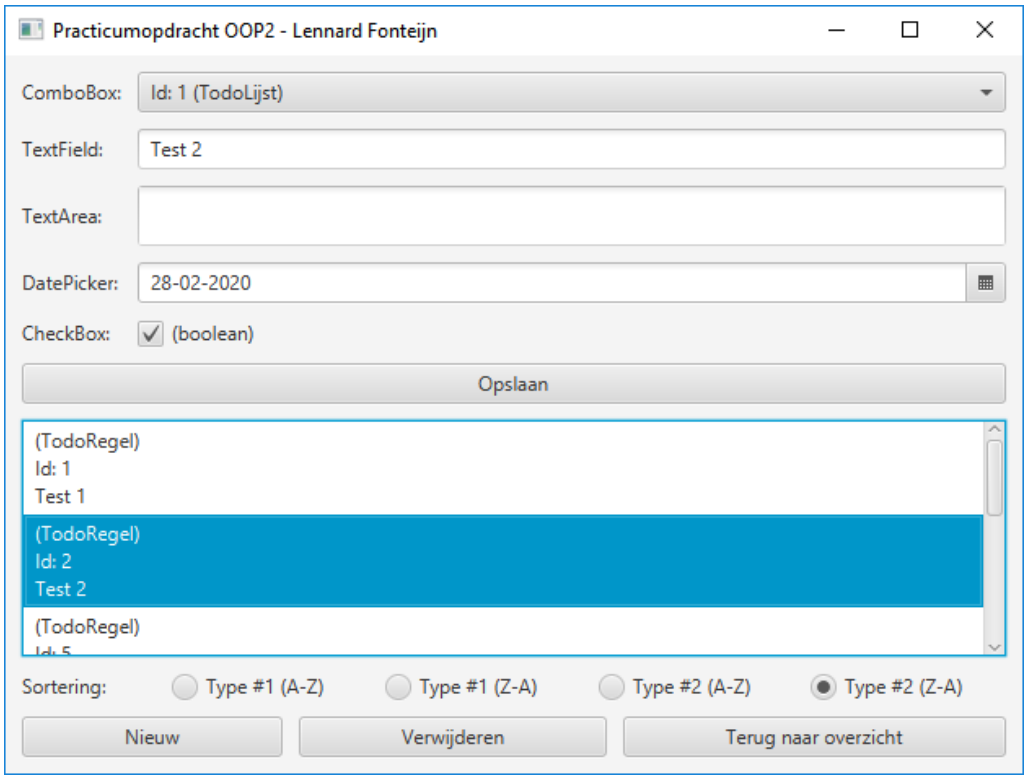
10. Zorg ervoor dat bij het opstarten van de Detail-view één van de twee Comparators direct wordt toegepast, zodat data altijd op een manier gesorteerd is.

11. Breid de Detail-view uit met een viertal *RadioButtons* waarmee de ListView gesorteerd kan worden. Zorg ervoor dat er maar één tegelijk aangeklikt kan zijn.

12. Zorg ervoor dat bij het aanklikken van de *RadioButtons* de juiste Comparator wordt toegepast op de ListView en deze direct de nieuwe sortering toont.



Afbeelding: Voorbeeld van een TodoLijstView na het afronden van alle stappen.



Afbeelding: Voorbeeld van een TodoRegelView na het afronden van alle stappen.

Checklist

- ❑ De DAO's zijn geïmplementeerd zoals aangegeven in de UML.
  - ❑ Er is een BinaryMasterDAO- en ObjectDetailDAO-class aanwezig.
  - ❑ Beide classes zitten in een *data*-package.
- ❑ De BinaryMasterDAO vertoont het gedrag zoals beschreven in het stappenplan:
  - ❑ Er is gebruik gemaakt van *Binary IO*.
  - ❑ Beide *save*- en *load*-methodes geeft het binaire bestand na gebruik vrij en handelt foutmeldingen af met een specifieke rede zichtbaar in de console.
  - ❑ Alle data wordt daadwerkelijk opgeslagen en weer ingeladen.
- ❑ De ObjectDetailDAO vertoont het gedrag zoals beschreven in het stappenplan:
  - ❑ Er is gebruik gemaakt van *Object Serialization*.
  - ❑ Beide *save*- en *load*-methodes geeft het bestand na gebruik vrij en handelt foutmeldingen af met een specifieke rede zichtbaar in de console.
  - ❑ Alle data wordt daadwerkelijk opgeslagen, behalve de Master-referentie in het Detail-model.
  - ❑ Alle data wordt daadwerkelijk geladen, de referentie naar de bijbehorende Master-objecten worden hersteld.
- ❑ De applicatie maakt gebruik van de BinaryMasterDAO en ObjectDetailDAO.
- ❑ Er is een *comparators*-package beschikbaar in het project met daarin drie verschillende comparator-classes.
  - ❑ Alle comparators kunnen zowel oplopend als aflopend sorteren.
  - ❑ Eén van de comparators bedoeld voor de Detail-view heeft een dubbele sortering.
- ❑ De menubalk van de Master-view bevat een optie "Sorteren", met daarin twee opties om oplopend en aflopend te sorteren. Bij het aanklikken van een optie wordt een comparator toegepast en ververst direct de *ListView*.
- ❑ De Detail-view is uitgebreid met vier *RadioButtons*, waarvan er maar één tegelijk geselecteerd kan worden. Bij het selecteren van een *RadioButton* wordt een comparator toegepast en ververst direct de *ListView*.
- ❑ Bij het tonen van een scherm wordt direct een comparator toegepast op een *ListView*.
- ❑ Het project bevat geen compile-errors.
- ❑ Het project is gecommit en gepushed naar GitLab.