OOP2 – Practicumopdracht

Week 2

Voor de practicumopdracht gaan we ons deze week focussen op de V van MVC, de *views*.

Views verzorgen de GUI die een gebruiker te zien krijgt bij het opstarten van de applicatie. Het is belangrijk om te onthouden dat je bezig bent om code in verschillende lagen op te bouwen. Dit betekend dat de views die je deze week gaat maken geen logica gaan bevatten en je dus alleen maar bezig zult zijn met hoe ze eruitzien.

Er zijn een aantal zaken die deze week in je practicum gebruikt moeten worden. Qua controls zijn dat:

- Label
- TextField
- TextArea
- DatePicker
- CheckBox
- ComboBox
- ListView
- Button

Qua *panes* zijn dat:

- HBox
- VBox
- GridPane

Overige controls en panes mogen uiteraard ook gebruikt worden, maar bovenstaande lijst moeten hoe dan ook toegepast worden.

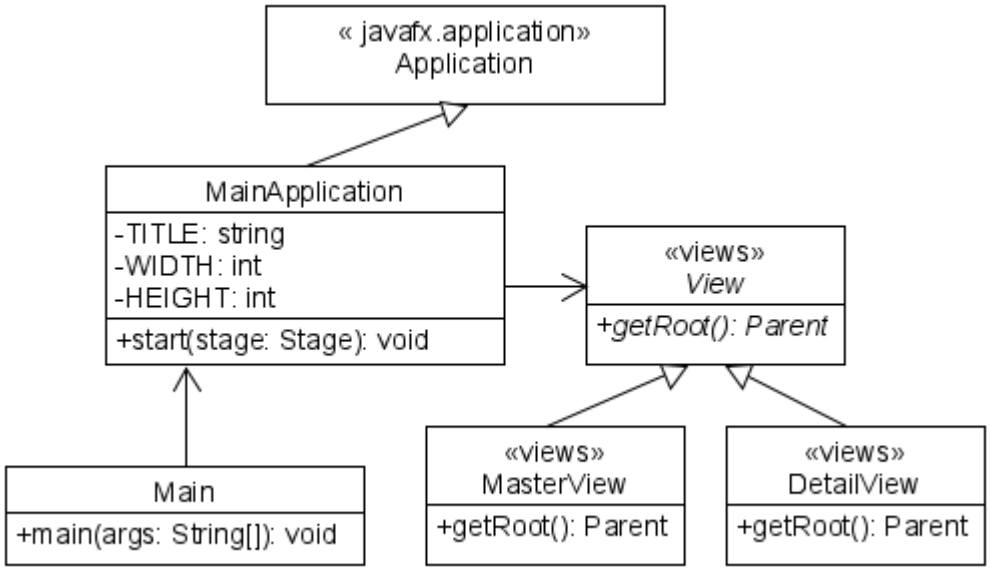
In relatie tot de attributen van de models die je eerder al hebt bedacht, moet je de volgende vertaling toepassen:

- Een String vertaald naar een TextField of TextArea.
- Een int of double vertaald ook naar een TextField.
- Een boolean vertaald naar een CheckBox.
- Een LocalDate vertaald naar een DatePicker.

Met de informatie hierboven ga je vervolgens voor elk van je models een view implementeren. Elke view heeft in ieder geval een ListView en een viertal Buttons nodig (Opslaan, Nieuw, Bewerken en een knop om te schakelen naar de andere view). De view voor je Detail-class bevat tevens nog een ComboBox voor de een-op-meer relatie.

De invulling en indeling van de views is geheel afhankelijk van de eerder bedachte attributen en de manier waarop je dat wilt tonen. Je kunt baseren op het voorbeeld verderop in het practicum voor deze week, maar je mag het dus ook je eigen invulling geven. Zorg er wel voor dat dit er een beetje verzorgt uitziet. Een leuke bonus is als de schermen op alle resoluties goed mee schalen.

Let op! Er zijn problemen geconstateerd met recente versies van IntelliJ in combinatie met JDK 13. Mocht je een fout krijgen met betrekking tot de “targets”-map, update dan je IntelliJ naar de laatste versie. Het is ook bekend dat *debugging* niet werkt, dat wordt na het volgen van het stappenplan hieronder vanzelf opgelost (in het specifiek stap 1).



Afbeelding: UML met de gewenste aanpassingen aan de structuur.

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

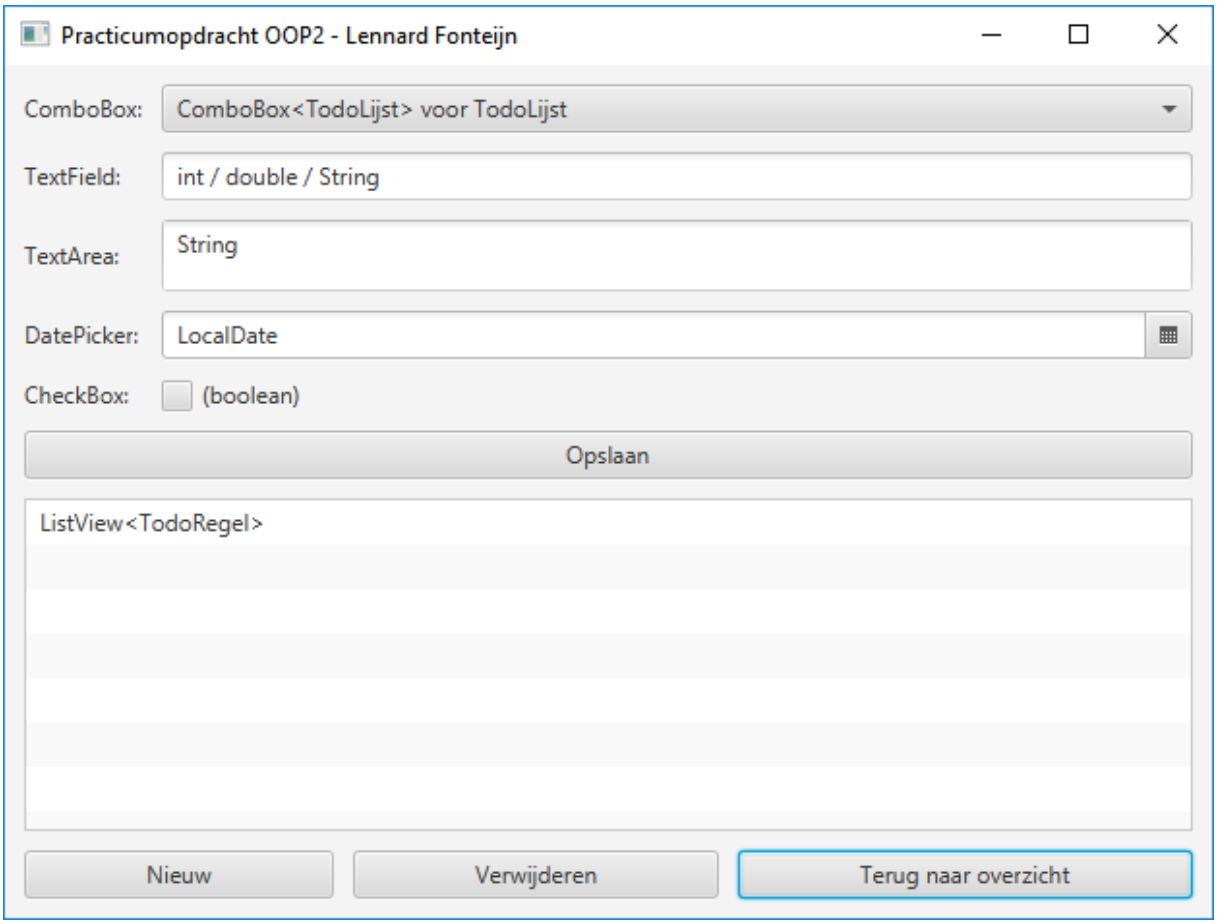
1. Verwijder je bestaande Main-class en voeg in plaats daarvan een nieuwe Main- en MainApplication-class toe volgens de UML. Zorg ervoor dat de Main-class bij het opstarten de volgende code aanroept:

   ```
   Application.launch(MainApplication.class, args);
   ```

2. Maak een package met de naam *views* in de *practicumopdracht*-package. Maak voor beide models in deze package een nieuwe class en geef deze een naam in het volgende formaat:

   <NaamVanModel>View (bijv. *TodoLijstView* en *TodoRegelView*)

3. Maak de View-class uit de UML, erf de twee View-classes hiervan over en implementeer de *getRoot*-methode zodanig dat deze de schermen teruggeeft zoals bedacht.
4. Zorg ervoor dat bij het opstarten van de applicatie één van de twee schermen getoond wordt, gebruik hiervoor de eerder geïmplementeerde *getRoot*-methode. Uiteraard moeten beide schermen wel functioneel zijn. Maak slim gebruik van *polymorphism* zodat je dit met één regel code kan omwisselen.



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

Checklist

- ❑ Er is een *views*-package beschikbaar in het project met daarin twee view-classes die overerven van *View*, met correcte benaming.
- ❑ Er is een *Main.java* en een *MainApplication.*java beschikbaar in het project die één van de views toont bij het opstarten van de applicatie, met gebruik van de *getRoot*-methode.
- ❑ Over beide views bekeken zijn de volgende *controls* gebruikt, met aansluiting op de bedachte attributen van de models:
    - ❑ Label
    - ❑ TextField (String, int en double)
    - ❑ TextArea (String)
    - ❑ DatePicker (LocalDate)
    - ❑ CheckBox (boolean)
- ❑ Over beide views bekeken zijn de volgende *panes* gebruikt:
    - ❑ HBox
    - ❑ VBox
    - ❑ GridPane
- ❑ De Detail-view heeft een ComboBox voor de één-op-meer relatie.
- ❑ Beide views hebben een ListView.
- ❑ Beide views hebben een viertal Buttons voor:
    - ❑ Opslaan
    - ❑ Nieuw
    - ❑ Verwijderen
    - ❑ Schakelen
- ❑ Beide views zijn functioneel, zien er verzorgd uit en kunnen getoond worden met een simpele code wijziging.
- ❑ Het project bevat geen compile-errors.
- ❑ Het project is gecommit en gepushed naar GitLab.