OOP2 – Practicumopdracht

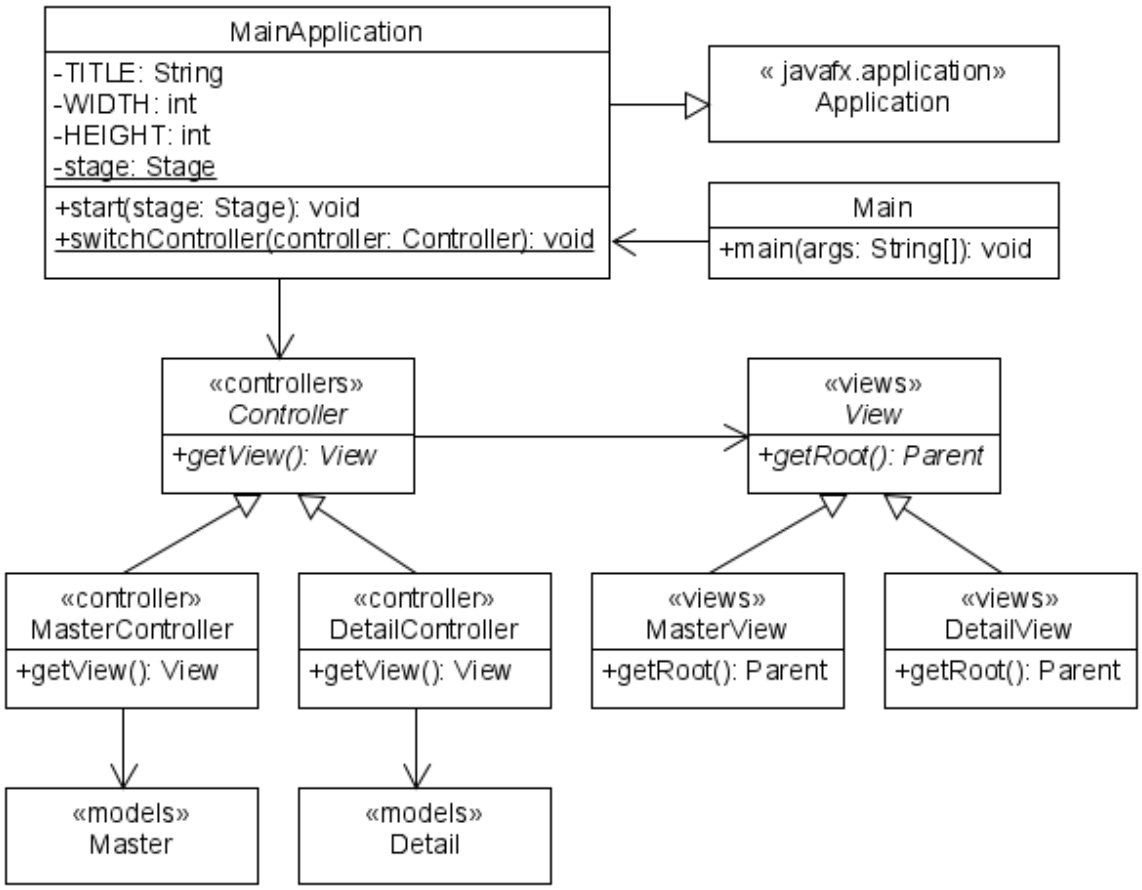
Week 3

Nu de M en de V zijn ontwikkeld in Week 1 en 2, is het tijd voor de verbindende factor tussen deze twee: de *Controllers*, oftewel de C in MVC!

Voor de vorm van MVC die wij bij deze practicumopdracht toepassen is het belangrijk dat we de volgende regels hanteren:

- Een View mag niet direct een Controller beïnvloeden, een Controller wel direct de View.
- Een View kan niet direct een Model beïnvloeden, dat doet de Controller voor de View.

De Controllers verzorgen dus de communicatie tussen de visuele laag (Views) en de achterliggende data-laag (Models). Als er op een knop gedrukt wordt, dan is het de taak van de bijbehorende Controller daar iets mee te doen.



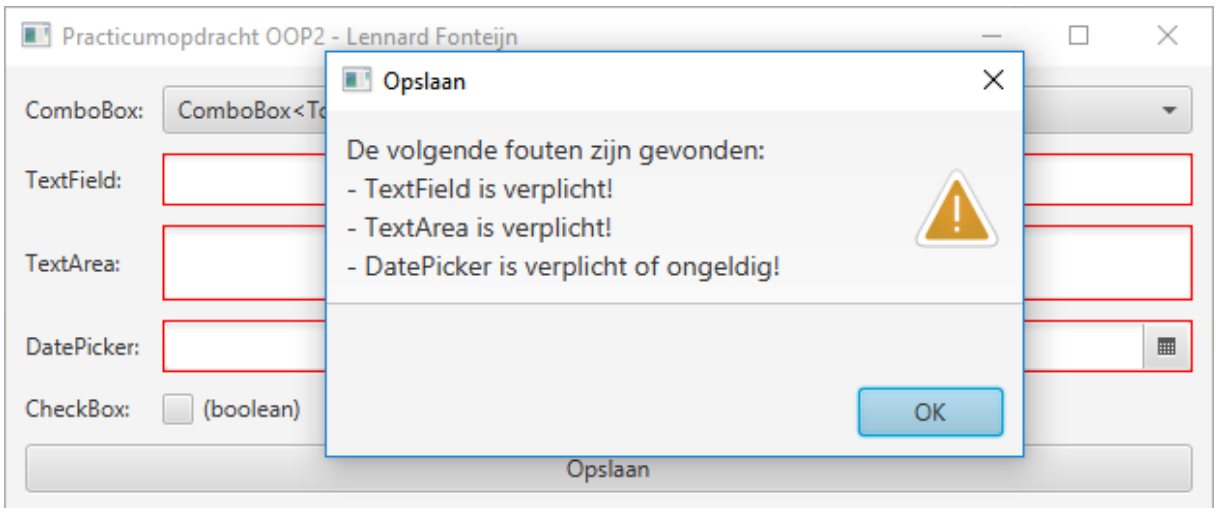
Afbeelding: UML met de gewenste aanpassingen aan de structuur.

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

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

   <NaamVanModel>Controller (bijv. *TodoLijstController* en *TodoRegelController*)

2. Maak de Controller-class uit de UML, erf de twee Controller-classes hiervan over en implementeer de *getView*-methode zodanig dat deze een instantie van de desbetreffende views teruggeeft.
3. Pas de MainApplication-class aan zoals weergegeven in de UML en gebruik de *switchController*-methode om bij het opstarten van de applicatie de view van je Master-model te tonen.
   - Let op! Ruim na de aanpassingen overbodige code direct op!
4. Maak voor alle knoppen in je views een *getter* en maak vervolgens voor alle knoppen een methode in de bijbehorende controllers. Zorg ervoor dat bij het indrukken van een knop de correcte methode wordt aangeroepen en een *Alert* toont met een tekst die aangeeft welke knop er is ingedrukt.
   - Let op! Dit hoef je (nog) niet te doen voor de “Opslaan”- en schakel-knoppen!
5. Zorg ervoor dat wanneer schakel-knop wordt ingedrukt het scherm omschakelt naar het andere scherm, doe dit voor beide schermen.
6. Zorg ervoor dat wanneer de “Opslaan”-knop wordt ingedrukt de invoervelden worden gevalideerd, doe dit voor beide schermen. Communiceer gemaakte fouten <u>duidelijk</u> naar de gebruiker. Denk bij validatie aan de volgende zaken:
   - Of verplichte velden gevuld zijn en niet enkel spaties bevatten
   - Of de invoer in een TextField wel een double / int is
   - Of de invoer in een DatePicker wel een geldige datum is
7. Zorg ervoor dat, indien alle invoer valide blijkt te zijn, de ingevoerde gegevens worden opgeslagen in een instantie van het bijbehorende model. Maak vervolgens gebruik van de *toString*-methode van de instantie om de actuele waardes van alle attributen aan de gebruiker te tonen met een *Alert*. Leeg als laatste alle invoervelden zodat de gebruiker direct weer opnieuw iets kan invullen.



Afbeelding: Voorbeeld van een manier waarop de gebruiker geïnformeerd kan worden.

Checklist

- ❑ Er is een *controllers*-package beschikbaar in het project met daarin twee controller-classes die overerven van *Controller*, met correcte benaming.
- ❑ Bij het opstarten van de applicatie wordt met gebruik van de *switchController*-methode het scherm van de Master-class getoond.
- ❑ Er is geen ongebruikte code aanwezig in de MainApplication-class.
- ❑ Alle knoppen in de views hebben een methode in de bijbehorende controller die ook wordt aangeroepen bij het indrukken.
    - ❑ Er wordt direct een *Alert* getoond bij het indrukken van “Bewerken”.
    - ❑ Er wordt direct een *Alert* getoond bij het indrukken van “Verwijderen”.
    - ❑ Er wordt geschakeld naar het andere scherm bij het indrukken van de schakel-knop.
- ❑ Bij het indrukken van “Opslaan” wordt invoer gevalideerd op de volgende manieren:
    - ❑ Minimaal één invoerveld is verplicht en accepteert daarbij geen spaties als geldige invoer.
    - ❑ Een TextField bedoeld voor een int geeft een geldige int-waarde.
    - ❑ Een TextField bedoeld voor een double geeft een geldige double-waarde.
    - ❑ Een DatePicker geeft een geldige LocalDate-waarde.
- ❑ Fouten die optreden tijdens de validatie worden duidelijk gecommuniceerd naar de gebruiker.
- ❑ Bij succesvolle validatie worden alle ingevoerde gegevens in een instantie van het bijbehorende model opgeslagen.
- ❑ De gegevens in het bijbehorende model worden met gebruik van de *toString*-methode en een *Alert* getoond aan de gebruiker.
- ❑ De invoervelden worden weer geleegd voor de gebruiker na opslaan.
- ❑ Het project bevat geen compile-errors.
- ❑ Het project is gecommit en gepushed naar GitLab.