

### **REKENMACHINE**

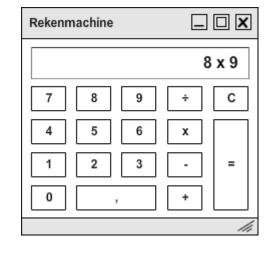
In deze oefening bouwen we de GUI van een rekenmachine. De rekenmachine ondersteunt enkel basisoperaties (+, -, x en ÷) en kan enkel een bewerking uitvoeren op exact twee operanden.

De voornaamste JavaFX klasse die we voor deze oefening nodig hebben is:

• javafx.scene.layout.GridPane

... die we gebruiken in combinatie met:

- javafx.geometry.Insets
- javafx.geometry.Pos
- (javafx.geometry.HPos)
- (javafx.geometry.VPos)
- (javafx.scene.layout.Priority)



Raadpleeg in eerste instantie de <u>JavaFX documentatie</u> als je ergens vast zit!

## 1 WIREFRAME

Bouw de gegeven wireframe met een tool naar keuze.

#### 2 HOOFDSCHERM AANMAKEN – MVP

De model klasse is Calculator. Aan deze klasse hoef je niets te wijzigen.

De *view* klasse is **CalculatorPane**. Deze erft al over van **GridPane**. We werken deze klasse uit in punt 3.

De *presenter* klasse is **Presenter**. In punt 4 werken we deze in detail uit. Op dit moment kan je alvast het volgende doen:

- Maak een attribuut voor het *model* en een attribuut voor de *view*.
- Schrijf een constructor die het model en de view als parameter binnenkrijgen.

De Main klasse is gegeven. Je hoeft enkel nog één lijn code uit commentaar te halen zodat de constructor die je zojuist geschreven hebt opgeroepen wordt.

# 3 UI OPBOUWEN

De klasse CalculatorPane:

Maak een attribuut voor het tekstveld en voor elke knop die je ziet op de wireframe.
Zorg ook voor getters met de juiste access level.



# 3.1 De methode initialiseNodes

- Initialiseer elke control
- Zorg er voor dat het tekstveld niet aangepast kan worden en dat het rechts uitgelijnd is

### 3.2 De methode layoutNodes

Plaats elke control op de **GridPane** (= **CalculatorPane**). Houd rekening met volgende beperkingen:

- Plaats elke control op de juiste plaats in de grid. Sommige componenten nemen meer dan één rij of kolom in beslag!
- Elke control heeft een minimumgrootte van 35 x 35 pixels.
- Elke control heeft een maximumgrootte van **Double.MAX\_VALUE** x **Double.MAX\_VALUE**Dit zorgt er voor dat de controls groeien tot de grootte van de cel waarin ze zitten.
- Zorg voor de nodig padding **binnen** de cellen. Dit doe je op grid-niveau.
- Zorg voor de nodig *gaps* (horizontaal en verticaal) **tussen** de cellen. Dit doe je eveneens op grid-niveau.

#### Extra:

- Zorg er voor dat elke control ook effectief zal groeien indien de grootte zijn cel verandert.
  - o Tip: Hiervoor heb je de klassemethode GridPane.setConstraints nodig.

### 4 AFHANDELEN EVENTS

De methode addEventHandlers (zie punt 4.1) moet in de constructor van de Presenter opgeroepen worden.

#### 4.1 De methode addEventHandlers

Deze methode zorgt voor het toevoegen van de event handlers aan de knoppen.

- Zorg voor een event handler voor elke knop. Afhankelijk van de knop moet de juiste methode van Calculator opgeroepen worden met de juiste parameter:
  - o handleInput (OperandCharacter)
  - o handleInput (Operator)
- Bij het indrukken van de "=" knop moet de calculate methode opgeroepen worden.
- Bij het indrukken van de "C" knop moet de clear methode opgeroepen worden.
- Beide handleInput methodes en de calculate methode kunnen een CalculatorException gooien indien de invoer niet ondersteund is.
- Bij het indrukken van eender welke knop dient de methode updateView (zie punt 4.2) opgeroepen te worden.

# 4.2 De methode updateView

• Zorg er voor dat het tekstveld van de view de juiste tekst bevat. Het model heeft de nodige informatie voorhanden (methode getDisplay).