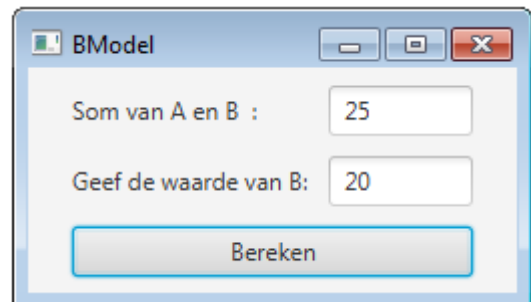
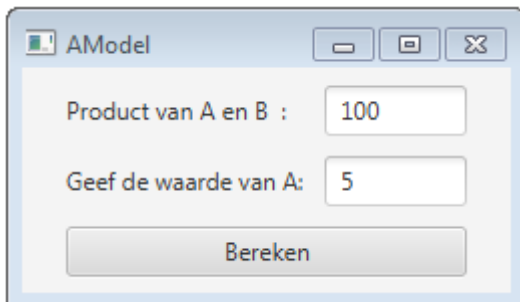


## W4 Observer - Twee views met Observer-pattern

---

1. Vertrek vanuit het gezipte IntelliJ-project. Het bevat een eenvoudige applicatie die is opgebouwd volgens het MVP-principe.
  - Bekijk de code van de 2 modelklassen, de 2 presenterklassen en van de 2 viewklassen.
  - Run het programma; merk op dat het nog niet werkt zoals het hoort. Er is namelijk geen communicatie tussen de modelklassen en de viewklassen.
2. Pas het **Observer**-pattern toe:
  - De klassen **AModel** en **BModel** maak je **Observable**.  
→ Zij moeten de gekoppelde views verwittigen bij elke wijziging.
  - De klassen **APresenter** en **BPresenter** maak je **Observer**.  
→ Zij worden door de modelklassen verwittigd en passen de view aan de gewijzigde data aan.
  - Vergeet niet dat de klassen **APresenter** en **BPresenter** zich moeten registreren als observers bij de beide modellen!
3. Test het resultaat uit:



4. Voeg een nieuwe klasse **ConsoleView** toe. Maak ook deze viewklasse **Observer** van het **AModel** én het **BModel**. Bij elke wijziging in één van de modellen druk je **enkel de laatst gewijzigde waarde** van a of b af in het consolevenster:

```
De waarde van a werd gewijzigd in: 5
De waarde van b werd gewijzigd in: 20
De waarde van b werd gewijzigd in: 99
De waarde van b werd gewijzigd in: 1000
De waarde van a werd gewijzigd in: 2
De waarde van b werd gewijzigd in: 1
```

## Uitbreiding / Aanpassing

---

In dit programma zit nogal wat *code duplicatie*. De bedoeling is om er zoveel mogelijk van weg te werken.

1. Maak van de beide modelklassen één klasse met de naam **Model**. Om beide modellen van mekaar te kunnen onderscheiden voeg je een extra **String** attribuut **naam** toe waarvan de waarde via de constructor wordt ingevuld ("a" of "b"). Dit attribuut zal je alleen nodig hebben bij de **ConsoleView**. Van deze klasse maak je dus 2 modelklassen. Mogelijk moet je nog een andere aanpassing doen.
2. Pas de **ConsoleView** aan, zorg ervoor dat het programma nog steeds de oorspronkelijke uitvoer oplevert. Vergeet niet de noodzakelijke aanpassingen in de **Main**-methode te doen.
3. Voor de doorzetters, de View klassen verschillen enkel in de namen van de operaties en modellen die gebruikt worden. Maak van beide view klassen één klasse, waar je deze beide namen aan de constructor doorgeeft.
4. Beide eventhandlers lijken sterk op mekaar. Maak een klasse **BerekenEventHandler**, waar je de model en view als een parameter in de constructor doorgeeft.
5. Vervang de klassen **APresenter** en **BPresenter** door de klasse **Presenter** waarvan er slechts één object bestaat, dit moet dus met beide modelklassen en beide views gekoppeld worden. Pas de nodige methoden aan.